Contents

Sections and articles in this PDF.

User documentation

where to start.mdx

Getting Started ..... where-to-start.mdx

FAQ.mdx

FAQ ..... FAQ.mdx

glossary.mdx

Altcraft glossary ..... glossary.mdx

archive.mdx

Documentation archive ..... archive.mdx

profiles and databases

Profiles and databases ..... profiles-and-databases/index.mdx

Subscription resources profiles-and-databases/66427.mdx

Managing databases profiles-and-databases/66420.mdx

Subscriber profile profiles-and-databases/9405665.mdx

Profiles import and data update ..... profiles-and-databases/profiles-import-and-data-update.mdx

Scheduled customer data import profiles-and-databases/9404661.mdx

Automatic data collection profiles-and-databases/66433.mdx

Bulk customers profiles update profiles-and-databases/9406694.mdx

Double opt-in subscription profiles-and-databases/9404703.mdx

Suppression lists profiles-and-databases/66534.mdx

Profile relations profiles-and-databases/20480122.mdx

Profile history export profiles-and-databases/49710319.mdx

Profile export profiles-and-databases/76808636.mdx

Creating a static segment based on import results profiles-and-databases/37847201.mdx

How to open a CSV file ..... profiles-and-databases/csv-in-excel.mdx

Matching ..... profiles-and-databases/matching.mdx

Types of fields in the database ..... profiles-and-databases/types-of-db-fields.mdx

Global control groups ..... profiles-and-databases/gcg.mdx

Subscription Manager ..... profiles-and-databases/subscription-manager.mdx

communication channels

Communication channels ..... communication-channels/index.mdx

WhatsApp ..... communication-channels/whatsapp.mdx

Viber*™ ..... communication-channels/viber.mdx

Notify ..... communication-channels/notify.mdx

Communication Channels Workflow ..... communication-channels/scheme.mdx

Email: sending domain configuration communication-channels/email/9404444.mdx

Email: setting up and using postmasters ..... communication-channels/email/postmasters.mdx

Quick Start ..... communication-channels/email/first-mailing/quick-start.mdx

Email ..... communication-channels/email/first-mailing/email.mdx

Push channel ..... communication-channels/push/index.mdx

Mobile Push ..... communication-channels/push/mobile-push/index.mdx

Settings & implementation communication-channels/push/mobile-push/22708437.mdx

First Mobile Push Mailing ..... communication-channels/push/mobile-push/mobile-push-first-mailing.mdx

Integrate your app with Altcraft ..... communication-channels/push/mobile-push/mobile-push-app-integration/index.mdx

Providers: push message structure communication-channels/push/mobile-push/mobile-push-app-integration/49709434.mdx

Processing and adding a subscription communication-channels/push/mobile-push/mobile-push-app-integration/49709437.mdx

Event registration communication-channels/push/mobile-push/mobile-push-app-integration/49709440.mdx

Web push ..... communication-channels/push/web-push/index.mdx

Preliminary Settings ..... communication-channels/push/web-push/preliminary-settings.mdx

Connecting Web Push to a Website ..... communication-channels/push/web-push/setting-up.mdx

Transferring Data to the Platform ..... communication-channels/push/web-push/data-import.mdx

Web Push SDK Methods ..... communication-channels/push/web-push/methods.mdx

First Web Push Mailing ..... communication-channels/push/web-push/web-push-first-mailing.mdx

Web browser push configuration ..... communication-channels/push/web-push/providers/index.mdx

Firebase Cloud messaging ..... communication-channels/push/web-push/providers/firebase-cloud-messaging.mdx

Apple Safari ..... communication-channels/push/web-push/providers/Apple-Safari_37847599.mdx
Mozilla Services ..... communication-channels/push/web-push/providers/Mozilla-Services_37847581.mdx
Migrating push subscriptions from third-party services communication-channels/push/web-push/migration/37847852.mdx
How to transfer push subscriptions configured for Safari? communication-channels/push/web-push/migration/37847870.mdx
Migration from OneSignal communication-channels/push/web-push/migration/37847889.mdx
SMS channel ..... communication-channels/sms/index.mdx
SMS ..... communication-channels/sms/sms-first-mailing.mdx
Telegram ..... communication-channels/telegram/index.mdx
Telegram Bot ..... communication-channels/telegram/telegram-bot.mdx
Telegram Group ..... communication-channels/telegram/telegram-group.mdx
Max ..... communication-channels/max/index.mdx
MAX Bot ..... communication-channels/max/max-bot.mdx
MAX Group ..... communication-channels/max/max-group.mdx

segmentation

Segmentation ..... segmentation/index.mdx
Static Segments segmentation/9407279.mdx
Dynamic Segments ..... segmentation/dynamic-segments.mdx
Updatable Segments ..... segmentation/updatable-segments.mdx
Best Send Time (BST) segmentation/89325640.mdx
Logical operators "AND" and "OR" ..... segmentation/operators-and-or.mdx
Recommendations for working with segments ..... segmentation/segmentation-recommendations.mdx
Segmentation Conditions ..... segmentation/conditions/index.mdx
Segmentation by Profile data ..... segmentation/conditions/profile-data.mdx
Segmentation by Interactions with Entities ..... segmentation/conditions/interactions.mdx
Segmentation by Activity of the channel ..... segmentation/conditions/activity.mdx
Segmentation by external data segmentation/conditions/66468.mdx
Segmentation by external SQL tables segmentation/conditions/8650757.mdx

message templates

Message templates ..... message-templates/index.mdx
Managing message templates message-templates/66476.mdx
Template fragments message-templates/66494.mdx
Image gallery message-templates/66507.mdx
Content personalization message-templates/66490.mdx
Creating tables based on array elements message-templates/37847460.mdx
Block editor for email template ..... message-templates/block-editor.mdx
Web version of an email ..... message-templates/webversion.mdx
AMP Email message-templates/89326164.mdx
Importing and exporting a message template message-templates/56258129.mdx
Importing a template from a third-party service message-templates/76808233.mdx
Exporting a template from Pixcraft ..... message-templates/export-from-pixcraft.mdx
Working with message templates ..... message-templates/managing-templates/index.mdx
Working in the editor ..... message-templates/managing-templates/template-editor.mdx
Email-template ..... message-templates/managing-templates/email.mdx
SMS template ..... message-templates/managing-templates/sms.mdx
Push template ..... message-templates/managing-templates/push.mdx
MAX template ..... message-templates/managing-templates/max.mdx
Telegram template ..... message-templates/managing-templates/telegram.mdx
WhatsApp template ..... message-templates/managing-templates/whatsapp.mdx
Viber™ template ..... message-templates/managing-templates/viber.mdx
Notify template ..... message-templates/managing-templates/notify.mdx
Visual editor for email-template ..... message-templates/visual-editor/index.mdx
Visual editor interface message-templates/visual-editor/37848561.mdx
Adding blocks message-templates/visual-editor/37848658.mdx
Elements and their settings message-templates/visual-editor/37848574.mdx
Custom blocks message-templates/visual-editor/37849457.mdx
Style manager message-templates/visual-editor/37848592.mdx
Layer manager message-templates/visual-editor/37849442.mdx
Altcraft Variables and Functions ..... message-templates/altcraft-variables-and-functions/index.mdx
Logical expressions in messages message-templates/altcraft-variables-and-functions/66511.mdx
Loops in messages message-templates/altcraft-variables-and-functions/2196195.mdx

Market variables in templates message-templates/altcraft-variables-and-functions/56257541.mdx
Using the JSONPath functionality ..... message-templates/altcraft-variables-and-functions/jsonpath.mdx
Dynamic HTML content message-templates/dynamic-content/66496.mdx
Dynamic JSON content message-templates/dynamic-content/66498.mdx
Content from SQL database in templates message-templates/dynamic-content/66501.mdx
Dynamic API content message-templates/dynamic-content/66503.mdx

campaigns

Mailings ..... campaigns/index.mdx
Mailings calendar campaigns/66540.mdx
Broadcast mailings campaigns/66526.mdx
Regular mailing campaigns/9405823.mdx
Trigger mailing campaigns/66530.mdx
Multivariate testing campaigns/9405875.mdx
Mailing testing campaigns/66542.mdx
Mailing schedule campaigns/66538.mdx
Placement mailing campaigns/9406025.mdx

super campaigns

Campaigns ..... super-campaigns/index.mdx
Working with Campaigns ..... super-campaigns/where-to-start.mdx
Local control groups (LCG) ..... super-campaigns/lcg.mdx
Stratification Violation Error ..... super-campaigns/stratificcation_cases.mdx
Audience expansion ..... super-campaigns/audience-expansion.mdx
Audience building ..... super-campaigns/audience-distribution.mdx

automation scenarios

Automation scenarios ..... automation-scenarios/index.mdx
Managing scenarios automation-scenarios/9406861.mdx
Nodes of the scenario automation-scenarios/9407058.mdx
Classic marketing scenarios automation-scenarios/22708542.mdx
Step-by-step welcome scenario guide automation-scenarios/25985320.mdx
Scenario for automatic notification of the manager automation-scenarios/28934313.mdx
Abandoned cart scenario automation-scenarios/56254697.mdx

market

Market ..... market/index.mdx
Market settings market/49710292.mdx
Orders market/49710295.mdx
Market variables in message template market/56252235.mdx
Guide: how to send an order confirmation email market/56257397.mdx
Products ..... market/products/index.mdx
How to create a product manually market/products/56249576.mdx
How to import a product from a file market/products/56249588.mdx
Scheduled product import ..... market/products/feed-import.mdx
Product and SKU Segments ..... market/products/segments.mdx
Preparing the YML file ..... market/products/yml.mdx

loyalty programs

Loyalty programs ..... loyalty-programs/index.mdx
Loyalty programs ..... loyalty-programs/loyalties.mdx
Loyalty integration with external systems ..... loyalty-programs/loyalty-market-integraton.mdx
Basic loyalty program use cases ..... loyalty-programs/loyalty-cases.mdx
Order Segments ..... loyalty-programs/order-segments.mdx
Promotion codes loyalty-programs/9406715.mdx

reports and analytics

Reports and analytics ..... reports-and-analytics/index.mdx
Channel report ..... reports-and-analytics/channels-report.mdx
Traffic report ..... reports-and-analytics/traffic-report.mdx
Cohorts report ..... reports-and-analytics/cohorts-report.mdx
Lifetime report ..... reports-and-analytics/lifetime-report.mdx
Funnels report ..... reports-and-analytics/funnels-report.mdx
Goals report ..... reports-and-analytics/goals-report.mdx
Audience growth report ..... reports-and-analytics/audience-growth-report.mdx

Click map report ..... reports-and-analytics/click-map-report.mdx
Loyalty programs report ..... reports-and-analytics/loyalty.mdx
Bounces report ..... reports-and-analytics/bounces.mdx
Undeliveries report ..... reports-and-analytics/undeliveries.mdx
Report on global control groups ..... reports-and-analytics/gcg.mdx
Summary report ..... reports-and-analytics/summary-report/index.mdx
Summary report metrics reports-and-analytics/summary-report/33063594.mdx

integrations

Facebook Ads Manager™ ..... integrations/facebook-ads-manager.mdx
Google Ads Audiences ..... integrations/google-audience.mdx
MAX ..... integrations/max.mdx
Yandex.Audience™ ..... integrations/yandex-audience.mdx
VK Ads™ ..... integrations/vk.mdx
Static segment synchronization integrations/9407286.mdx
Yandex AppMetrica™ ..... integrations/yandex-app-metrica.mdx
Tilda™ integrations/2195939.mdx
Lpgenerator™ integrations/2195949.mdx
WhatsApp ..... integrations/whatsapp.mdx
Viber*™ integration ..... integrations/viber.mdx
Integration scope integrations/45514832.mdx
Data Transmitted During Synchronization ..... integrations/transmitted-data.mdx
Notify ..... integrations/notify.mdx
Altcraft Action Hooks integrations/action-hooks/37849833.mdx
Action hooks event types integrations/action-hooks/37849805.mdx
Action Hook Message Structure integrations/action-hooks/37849819.mdx
JSON batch request (HTTP POST action hook) integrations/action-hooks/37849807.mdx
Message to RabbitMQ broker integrations/action-hooks/37849816.mdx
Message to RabbitMQ exchanger integrations/action-hooks/37849810.mdx
Message to Kafka broker integrations/action-hooks/37849813.mdx
Test event integrations/action-hooks/37849821.mdx
Connecting Altcraft to Albato integrations/albato-integrations/76808402.mdx
Launching the welcome scenario using Albato integrations/albato-integrations/768084024.mdx
Transmitting event data integrations/albato-integrations/768084025.mdx
Setting app a trigger mailing integrations/albato-integrations/768084026.mdx
Event registration integrations/albato-integrations/768084027.mdx

weblayers

Data analytics weblayers/forms/37848282.mdx
Binding data channel and forms weblayers/forms/37849222.mdx
NPS testing ..... weblayers/forms/nps.mdx
Create a form ..... weblayers/forms/form-creating/index.mdx
General settings ..... weblayers/forms/form-creating/general-settings.mdx
Form constructor weblayers/forms/form-creating/37848206.mdx
Appearance weblayers/forms/form-creating/37848222.mdx
Actions and form publication weblayers/forms/form-creating/37848225.mdx
Conditional logic in forms and surveys weblayers/forms/form-creating/37849748.mdx
Pixels ..... weblayers/pixels/index.mdx
Goal customer actions and scoring weblayers/pixels/9406180.mdx
Pop-ups ..... weblayers/popups/index.mdx
Creating and publishing a pop-up weblayers/popups/76809121.mdx
Setting up a popup in the code editor weblayers/popups/109903904.mdx
Managing pop-ups manually via script weblayers/popups/109903905.mdx
Popup analytics weblayers/popups/109903934.mdx
Guide: pop-up for push subscriptions ..... weblayers/popups/popup-to-push.mdx
Case: Creating a pop-up with the "Wheel of Fortune" widget ..... weblayers/popups/wheel-of-fortune.mdx
Basic cases of placing a popup via the Tag Manager ..... weblayers/popups/popup-with-tagmanager.mdx
Tag Manager ..... weblayers/tag-manager/index.mdx
Configuring and installing Tag Manager weblayers/tag-manager/95715878.mdx
Trigger types weblayers/tag-manager/95715843.mdx
Variables types weblayers/tag-manager/95715946.mdx
Linking a pixel and the Tag manager weblayers/tag-manager/109903922.mdx

settings

Settings ..... settings/index.mdx
Account settings settings/9407614.mdx
Custom links settings/9407616.mdx
Virtual senders settings/9405938.mdx
Sending policies settings/9405554.mdx
Audit journal settings/3113292.mdx
Tags FAQ settings/89326254.mdx
Attribute settings ..... settings/attributes.mdx
Users, groups and access ..... settings/users-groups-access/index.mdx
Two-Factor Authentication (2FA) settings/users-groups-access/56255904.mdx
Connections ..... settings/connections/index.mdx
Connection to Facebook Ads Manager ..... settings/connections/fb-connection.mdx
Connection to Google Ads ..... settings/connections/google-ads.mdx
Connecting to Yandex.Audience™ ..... settings/connections/ya-audiences.mdx
Connection to 360dialog settings/connections/360.mdx
Connection to Edna ..... settings/connections/edna.mdx
Connection to Devino Telecom ..... settings/connections/devino.mdx
Connection to SMSTraffic ..... settings/connections/smstraffic.mdx
Connection to VK Ads™ ..... settings/connections/vk-connection.mdx
Connection to MTS OmniChannel ..... settings/connections/mts.mdx
Custom Authentication Connection ..... settings/connections/custom.mdx
OAuth2 connection settings/connections/oauth20.mdx
Basic Authentication connection ..... settings/connections/basic.mdx
Token Authentication connection ..... settings/connections/token.mdx
Connection to Rapporto ..... settings/connections/rapporto.mdx
MAX connection ..... settings/connections/max.mdx
Connection to Notify ..... settings/connections/notify.mdx

api

API requests: where to start ..... api/index.mdx
Import or update a profile ..... api/api-import.mdx
Trigger mailing launch ..... api/api-trigger.mdx
Engage profile in scenario ..... api/api-scenario.mdx

changelog

Changelog ..... changelog/index.mdx
v2026.2.77 ..... changelog/v2026277.mdx
v2026.1.76 ..... changelog/v2026176.mdx
v2025.4.75 ..... changelog/v2025475.mdx
v2025.4.74 ..... changelog/v2025474.mdx
v2025.3.73 ..... changelog/v2025373.mdx
v2025.2.72 ..... changelog/v2025272.mdx
v2025.1.71 ..... changelog/v2025171.mdx
v2024.4.70 ..... changelog/v2024470.mdx
v2024.3.69 ..... changelog/v2024369.mdx
v2024.2.68.2 ..... changelog/v20242682.mdx
v2024.1.68 ..... changelog/v2024168.mdx

API / Developer Guide

API interaction ..... api-interaction.mdx
Matching 37848489.mdx
Importing the API collection in Postman ..... api-collection.mdx
List of API endpoints ..... api-endpoints-list.mdx
Import profile ..... profiles/import.mdx
Update profile ..... profiles/update.mdx
Import multiple profiles profiles/2195504.mdx
Update multiple profiles profiles/2195501.mdx
Add multiple profiles profiles/2195496.mdx
Add profile to database profiles/2195477.mdx

Quick profile import profiles/3113182.mdx
Import profile to RabbitMQ profiles/2195492.mdx
Get profile data profiles/2195482.mdx
Uploading profiles to a file profiles/3115796.mdx
Suspend all database profiles profiles/2195553.mdx
Unsuspend all profiles in a database profiles/2195556.mdx
Delete profile profiles/28936545.mdx
Subscription fields functional update profiles/3113721.mdx
Database fields functional update profiles/3113720.mdx
Merging multiple profiles profiles/41484440.mdx
Unsubscribe profile from resource profiles/41484441.mdx
Profile splitting profiles/41484445.mdx
Get data for multiple profiles profiles/37847989.mdx
Add or edit subscription profiles/subscriptions/2195519.mdx
Get all profile subscriptions profiles/subscriptions/2195522.mdx
Get all subscriptions from multiple profiles profiles/subscriptions/2195524.mdx
Get profile subscription profiles/subscriptions/2195527.mdx
Delete profile subscription profiles/subscriptions/2195529.mdx
Restore deleted subscription profiles/subscriptions/2195531.mdx
Suspend all subscriptions profiles/subscriptions/2195533.mdx
Unsuspend all suspended subscriptions profiles/subscriptions/2195539.mdx
Get profile action history profiles/action-history/2195546.mdx
Get multiple profiles action history profiles/action-history/2195548.mdx
Profile relations ..... profiles/profile-relations/index.mdx
Attach relation profiles/profile-relations/9406148.mdx
Detach relation profiles/profile-relations/9406152.mdx
Modify relation properties profiles/profile-relations/9406155.mdx
Overwrite relation properties profiles/profile-relations/9406157.mdx
Get profile relations info profiles/profile-relations/9406159.mdx
Get profile relations info ..... profiles/profile-relations/get-relations-list.mdx
Get database statistics ..... databases/get-counters.mdx
Update statistics on database ..... databases/refresh-counters.mdx
Get database list databases/2195563.mdx
Get database information databases/2195565.mdx
Get database fields databases/2195567.mdx
Database wipe databases/2195815.mdx
Get resource statistics ..... resources/get-counters.mdx
Update statistics on resource ..... resources/refresh-counters.mdx
Get resources list resources/2195587.mdx
Get resource information resources/2195590.mdx
Get resource subscription fields resources/2195591.mdx
Get statistics on resource ..... segments/get-counters.mdx
Update statistics on segment ..... segments/refresh-counters.mdx
Add or remove profile segments/9407390.mdx
Get profile data in a static segment segments/9407394.mdx
Create segment segments/2195594.mdx
Update segment segments/2195596.mdx
Get segment information segments/2195598.mdx

Get segments list segments/2195600.mdx
Delete segment segments/2195602.mdx
Create suppression list suppression-lists/2195613.mdx
Update suppression list suppression-lists/2195615.mdx
Get suppression list info suppression-lists/2195617.mdx
Get the list of suppression lists suppression-lists/2195619.mdx
Delete suppression list suppression-lists/2195621.mdx
Upload suppression list data to file suppression-lists/76809125.mdx
Suppression list actions ..... suppression-lists/suppression-list-actions/index.mdx
Check if email is suppressed suppression-lists/suppression-list-actions/2195633.mdx
Add email to suppression list suppression-lists/suppression-list-actions/2195625.mdx
Add multiple emails to suppression list suppression-lists/suppression-list-actions/2195627.mdx
Remove email from suppression list suppression-lists/suppression-list-actions/2195637.mdx
Remove all emails from suppression list suppression-lists/suppression-list-actions/2195641.mdx
Check if domain is suppressed suppression-lists/suppression-list-actions/2195635.mdx
Add domain to suppression list suppression-lists/suppression-list-actions/2195629.mdx
Add multiple domains to suppression list suppression-lists/suppression-list-actions/2195631.mdx
Remove domain from suppression list suppression-lists/suppression-list-actions/2195639.mdx
Remove all domains from suppression list suppression-lists/suppression-list-actions/2195643.mdx
Check if phone number is suppressed suppression-lists/suppression-list-actions/76809084.mdx
Add phone number to suppression list suppression-lists/suppression-list-actions/76809087.mdx
Add multiple phones to suppression list suppression-lists/suppression-list-actions/76809091.mdx
Remove phone number from suppression list suppression-lists/suppression-list-actions/76809096.mdx
Remove all phone numbers from suppression list suppression-lists/suppression-list-actions/76809100.mdx
Get templates list templates/2195606.mdx
Get template info templates/2195608.mdx
Delete template templates/56255090.mdx
Add template templates/56255095.mdx
Update template templates/56255277.mdx
Channel object ..... templates/Channel-object_56255164.mdx
Get campaign information ..... super-campaigns/get-scampaigns-info.mdx
Get campaign list ..... super-campaigns/get-scampaigns-list.mdx
Activate campaign ..... super-campaigns/scampaigns-activation.mdx
Complete campaign ..... super-campaigns/scampaigns-complete.mdx
Deactivate campaign ..... super-campaigns/scampaigns-deactivation.mdx
Get campaign status ..... super-campaigns/scampaigns-status.mdx
Activate mailing campaigns/2195660.mdx
Deactivate mailing campaigns/2195662.mdx
Get mailing list campaigns/2195648.mdx
Get mailing information campaigns/2195650.mdx
Get mailing log campaigns/2195652.mdx
Clone mailing campaigns/2195654.mdx
Delete mailing campaigns/2195656.mdx
Get mailing status campaigns/2195664.mdx
Get broadcasts list campaigns/broadcast-campaigns/2195885.mdx
Get broadcast information campaigns/broadcast-campaigns/2195887.mdx
Create broadcast mailing campaigns/broadcast-campaigns/2195891.mdx
Update broadcast mailing campaigns/broadcast-campaigns/2195895.mdx

Launch a broadcast mailing campaigns/broadcast-campaigns/3116196.mdx

Get regular mailings list campaigns/regular-campaigns/2195899.mdx

Get regular mailing information campaigns/regular-campaigns/2195901.mdx

Create regular mailing campaigns/regular-campaigns/2195903.mdx

Update regular mailing campaigns/regular-campaigns/2195905.mdx

Launch a regular mailing campaigns/regular-campaigns/3116201.mdx

Get trigger mailings list campaigns/trigger-campaigns/2195915.mdx

Get trigger mailing information campaigns/trigger-campaigns/2195917.mdx

Create trigger mailing campaigns/trigger-campaigns/2195922.mdx

Update trigger mailing campaigns/trigger-campaigns/2195925.mdx

Trigger launch (API call) campaigns/trigger-campaigns/2196177.mdx

Profile import + trigger mailing launch campaigns/trigger-campaigns/22708362.mdx

Task for bulk trigger launch campaigns/trigger-campaigns/33063451.mdx

Task for bulk profiles import + trigger launch campaigns/trigger-campaigns/33063474.mdx

Bulk trigger launch campaigns/trigger-campaigns/37847471.mdx

Bulk profiles import + trigger mailing launch campaigns/trigger-campaigns/37847473.mdx

Clone a trigger mailing campaigns/trigger-campaigns/56258725.mdx

Data array ..... campaigns/trigger-campaigns/Data-array_41484531.mdx

Engage profile in scenario automation-scenarios/2195668.mdx

Import and engage profile in scenario automation-scenarios/28934227.mdx

Batch import and engage profiles in a scenario automation-scenarios/33063492.mdx

Task for batch import and engaging profiles in the scenario automation-scenarios/33063493.mdx

Get scenarios list automation-scenarios/95715617.mdx

Activate scenario ..... automation-scenarios/activate-workflow.mdx

Deactivate scenario ..... automation-scenarios/deactivate-workflow.mdx

Get profile tier in a loyalty program ..... loyalty/tiers.mdx

Export points transactions ..... loyalty/transactions-export.mdx

Expiring points for a period ..... loyalty/expiring-points.mdx

Get profile account transactions ..... loyalty/transactions-list.mdx

Get trigger promotions list ..... loyalty/trigger-promotions.mdx

Accrue points to a member ..... loyalty/accrue-points.mdx

Redeem member points ..... loyalty/redeem-points.mdx

Commit temporary transaction ..... loyalty/commit-transaction.mdx

Preliminary Order Calculation ..... loyalty/preorder.mdx

Order Confirmation ..... loyalty/preorder-commit.mdx

Roll back temporary transaction ..... loyalty/rollback-transaction.mdx

Cancel points transaction ..... loyalty/cancel-transation.mdx

Get points account balance ..... loyalty/balance.mdx

Register member in a loyalty program ..... loyalty/register-member.mdx

Batch adding participants to loyalty program ..... loyalty/register-member-batch.mdx

Task for batch add participants to loyalty program ..... loyalty/register-member-batch-task.mdx

Remove member from loyalty program ..... loyalty/delete-member.mdx

Get form information ..... forms/get.mdx

Get form list ..... forms/list.mdx

Get form fills ..... forms/fill.mdx

Publish form ..... forms/activate.mdx

Unpublish form ..... forms/deactivate.mdx

Delete form ..... forms/delete.mdx

Import promo codes promocodes/2195573.mdx
Get promo code information promocodes/2195575.mdx
Activate promo code promocodes/2195577.mdx
Update promo code promocodes/2195579.mdx
Attach promo code promocodes/2195581.mdx
Detach promo code promocodes/2195583.mdx
Get all promo codes promocodes/3113099.mdx
Goals and goal values registration goals/2195843.mdx
Processing and adding a subscription application-push-notifications/2196103.mdx
Add app push events application-push-notifications/2196102.mdx
Order data object market/market-objects/56249763.mdx
Product data object market/market-objects/Product-data-object_56249759.mdx
SKU data object market/market-objects/SKU-data-object_56249769.mdx
Categories array market/market-objects/Categories-array_56249780.mdx
Custom fields array market/market-objects/Custom-fields-array_56249775.mdx
Import order and item statuses market/orders/56249746.mdx
Get orders list market/orders/56249749.mdx
Delete order market/orders/76808865.mdx
Get order status market/orders/get-status.mdx
Update order line status market/orders/update-order-line-status.mdx
Import products, SKUs and categories market/products-and-sku/56249718.mdx
Import SKUs and categories market/products-and-sku/56249733.mdx
Get products list market/products-and-sku/56249723.mdx
Get SKUs list market/products-and-sku/56249724.mdx
Delete products market/products-and-sku/delete-product.mdx
Delete SKU market/products-and-sku/delete-sku.mdx
Get summary report analytic-reports/2195743.mdx
Get soft bounces report analytic-reports/2195748.mdx
Get undeliveries report analytic-reports/2195750.mdx
Get senders list senders/2195754.mdx
Get virtual senders list senders/virtual-senders/2195762.mdx
Get virtual sender information senders/virtual-senders/2195765.mdx
Clone virtual sender senders/virtual-senders/2195767.mdx
Create virtual sender senders/virtual-senders/2195769.mdx
Update virtual sender senders/virtual-senders/2195771.mdx
Delete virtual sender senders/virtual-senders/2195773.mdx
Add segmentation query datatables/query_segments/add.mdx
Update segmentation query datatables/query_segments/update.mdx
Get segmentation query information datatables/query_segments/get.mdx
Get segmentation queries list datatables/query_segments/list.mdx
Delete segmentation query datatables/query_segments/delete.mdx
Add template query datatables/query_templates/add.mdx
Update template query datatables/query_templates/update.mdx
Get template query information datatables/query_templates/get.mdx
Get template queries list datatables/query_templates/list.mdx
Delete template query datatables/query_templates/delete.mdx
AKMTA object objects/AKMTA-object_2195811.mdx
Content object objects/Content-object_2195781.mdx

Email rule object ..... objects/Email-rule-object_2195791.mdx
File object ..... objects/File-object_2195839.mdx
Profile data object ..... objects/Profile-data-object_3116008.mdx
Sender object ..... objects/Sender-object_2195784.mdx
Sender types ..... objects/Sender-types_2195789.mdx
SMS rule object ..... objects/SMS-rule-object_2195796.mdx
Start schedule object ..... objects/Start-schedule-object_2195802.mdx
Subscription object ..... objects/Subscription-status_2195807.mdx
Trigger types ..... objects/Trigger-types_2195809.mdx
Upload file miscellaneous/2195777.mdx
Get message web version miscellaneous/2196089.mdx
Push providers miscellaneous/37848490.mdx
Deduplication of requests miscellaneous/41484312.mdx
How to send API request with RabbitMQ miscellaneous/44662785.mdx
List of gender identifications miscellaneous/9405775.mdx
Obtain valid values for fields: browsers, devices, tz, oses, languages ..... miscellaneous/dictionary.mdx
mSDK ..... sdk/mobile/index.mdx
Managing JWT and Role Token ..... sdk/mobile/jwt-r-token.mdx
Quick Start ..... sdk/mobile/android/quick-start.mdx
SDK Functionality ..... sdk/mobile/android/functionality.mdx
SDK Configuration ..... sdk/mobile/android/setup.mdx
Public SDK API ..... sdk/mobile/android/api.mdx
Firebase Cloud Messaging ..... sdk/mobile/android/providers/fcm.mdx
Huawei Mobile Services ..... sdk/mobile/android/providers/hms.mdx
RuStore ..... sdk/mobile/android/providers/rustore.mdx
Quick Start ..... sdk/mobile/ios/quick-start.mdx
SDK Functionality ..... sdk/mobile/ios/functionality.mdx
SDK Configuration ..... sdk/mobile/ios/setup.mdx
Public SDK API ..... sdk/mobile/ios/api.mdx
Apple Push Notification Service ..... sdk/mobile/ios/providers/apns.mdx
Firebase Cloud Messaging ..... sdk/mobile/ios/providers/fcm.mdx
Huawei Mobile Services ..... sdk/mobile/ios/providers/hms.mdx
Quick Start ..... sdk/mobile/react-native/quick-start.mdx
SDK Configuration ..... sdk/mobile/react-native/setup.mdx
SDK Functionality ..... sdk/mobile/react-native/functionality.mdx
Public SDK API ..... sdk/mobile/react-native/api.mdx
Provider setup ..... sdk/mobile/react-native/providers.mdx

Administrator documentation

Getting Started ..... how-to-begin.mdx
Functional characteristics 33062926.mdx
System requirements 2196309.mdx
Processing HTTP/HTTPS traffic 56259196.mdx
Documentation Archive ..... archive.mdx
Architecture Overview technology-description/3115143.mdx
Component Description technology-description/3112973.mdx
Basic scheme technology-description/deployment-schemes/2196292.mdx
Fail-safe scheme technology-description/deployment-schemes/2196296.mdx

Typical Placement in Infrastructure technology-description/deployment-schemes/3113190.mdx
Nodes ..... admin-panel/nodes.mdx
Tariffs ..... admin-panel/account-area/tariffs.mdx
External data configuration ..... admin-panel/account-area/external-data-configuration.mdx
LDAP ..... admin-panel/account-area/ldap.mdx
Tasks ..... admin-panel/account-area/tasks.mdx
Schedule Jobs ..... admin-panel/account-area/schedule-jobs.mdx
Global Stop Lists ..... admin-panel/account-area/global-stop-lists.mdx
Webversion Store Policies ..... admin-panel/account-area/webversion-store-policies.mdx
Accounts ..... admin-panel/account-area/accounts/index.mdx
Account Users ..... admin-panel/account-area/accounts/users.mdx
Account Virtual Senders ..... admin-panel/account-area/accounts/virtual-senders.mdx
Account Database Indexes ..... admin-panel/account-area/accounts/account-db.mdx
Databases ..... admin-panel/settings/databases.mdx
Notifiers ..... admin-panel/settings/notifiers.mdx
Admins ..... admin-panel/settings/access/admins.mdx
API tokens ..... admin-panel/settings/access/api-tokens.mdx
MTA ..... admin-panel/settings/mta/index.mdx
Default rules ..... admin-panel/settings/mta/default-rules.mdx
Retry rules ..... admin-panel/settings/mta/retry-rules.mdx
Lock rules ..... admin-panel/settings/mta/lock-rules.mdx
Bounce patterns ..... admin-panel/settings/mta/bounce-patterns.mdx
Strategies ..... admin-panel/settings/mta/strategies.mdx
Keys ..... admin-panel/settings/mta/keys.mdx
ISPs ..... admin-panel/settings/mta/isps.mdx
Pools ..... admin-panel/settings/mta/pools.mdx
Email ..... admin-panel/senders/email.mdx
SMS ..... admin-panel/senders/sms.mdx
Event generator ..... admin-panel/senders/event-generator.mdx
Integration with Sendsay ..... admin-panel/senders/sendsay.mdx
Audit Journal ..... admin-panel/reports/audit-journal.mdx
Data Usage ..... admin-panel/reports/data-usage.mdx
ARF decoder ..... admin-panel/tools/arf-decoder.mdx
URL decoder ..... admin-panel/tools/url-decoder.mdx
SMID decoder ..... admin-panel/tools/smid-decoder.mdx
License ..... admin-panel/tools/license.mdx
Manual installation platform-installation/9406483.mdx
Running the platform in a Docker container platform-installation/28934367.mdx
Configuration file platform-configuration/22709423.mdx
Domain settings platform-configuration/9404422.mdx
LDAP access configuration platform-configuration/3116037.mdx
Sending Email via SMTP relay platform-configuration/3115304.mdx
Pixel and push domain configuration platform-configuration/3113280.mdx
Cluster and Replication Setup platform-configuration/89325722.mdx
System notifications configuration platform-configuration/3115246.mdx
Processes UNIX sockets configuration platform-configuration/3113071.mdx
HTTPS Configuration platform-configuration/3113225.mdx
Migrating from MongoDB Community Edition to Percona Server for MongoDB ..... platform-configuration/mongodb-to-percona.mdx

Adding sender IP addresses platform-configuration/3114051.mdx

Deduplication request settings platform-configuration/41484334.mdx

PostgreSQL database for account data ..... platform-configuration/PostgreSQL-database-for-account-data_25985138.mdx

Proxy server settings ..... platform-configuration/proxy.mdx

Keycloak Integration with Altcraft ..... platform-configuration/keycloak-integration.mdx

Getting HTTP service statuses ..... platform-configuration/status-handlers.mdx

Configuration MongoDB logs rotation ..... platform-configuration/log-rotation.mdx

Configuration of system constants and directories ..... platform-configuration/json-consts/index.mdx

Filtering bot actions ..... platform-configuration/json-consts/bots.mdx

Directory of gender markers ..... platform-configuration/json-consts/genders.mdx

Personnel requirements platform-maintenance/33063284.mdx

Platform maintenance processes platform-maintenance/33063294.mdx

Platform updating platform-maintenance/2196347.mdx

Platform service monitoring platform-maintenance/9406592.mdx

RabbitMQ migration platform-maintenance/3113563.mdx

Backup and recovery platform-maintenance/49709540.mdx

Transferring the platform to a new server platform-maintenance/56252916.mdx

Creating, deleting, and populating tables for statistics in ClickHouse platform-maintenance/95715733.mdx

Usage of the aktool utility ..... platform-maintenance/aktool.mdx

Users and directories engaged by the platform ..... platform-maintenance/users-directories.mdx

Overview communication-channels-guide/9405614.mdx

Creating and configuring a channel communication-channels-guide/9405608.mdx

Entity field configuration communication-channels-guide/33063507.mdx

Template language communication-channels-guide/9405611.mdx

Entities ..... communication-channels-guide/Entities_33063331.mdx

Pipe: Basic Concepts communication-channels-guide/pipes/33063348.mdx

Result ..... communication-channels-guide/pipes/Result_33063345.mdx

Errors ..... communication-channels-guide/pipes/Error_33063361.mdx

Log ..... communication-channels-guide/pipes/Log_33063367.mdx

Pack ..... communication-channels-guide/pipes/Pack_33063390.mdx

Unpack ..... communication-channels-guide/pipes/Unpack_33063369.mdx

HTTP Request ..... communication-channels-guide/pipes/HTTP-Request_33063371.mdx

Store Set ..... communication-channels-guide/pipes/Store-Set_33063405.mdx

Store Get ..... communication-channels-guide/pipes/Store-Get_33063420.mdx

Selector ..... communication-channels-guide/pipes/Selector_33063425.mdx

SQL ..... communication-channels-guide/pipes/SQL_33063429.mdx

Eventer ..... communication-channels-guide/pipes/Eventer_33063531.mdx

Scheduler ..... communication-channels-guide/pipes/Scheduler_33063535.mdx

Pipeline: Basic Concepts communication-channels-guide/Pipelines/33063464.mdx

Message ..... communication-channels-guide/Pipelines/Message_33063439.mdx

Schedule ..... communication-channels-guide/Pipelines/Schedule_33063442.mdx

Listener ..... communication-channels-guide/Pipelines/Listener_33063447.mdx

Moderate ..... communication-channels-guide/Pipelines/Moderate_33063460.mdx

Stop ..... communication-channels-guide/Pipelines/Stop_33063455.mdx

System page customization extra/37847908.mdx

Send Message ID extra/56257642.mdx

Instructions for migrating history to ClickHouse extra/56257641.mdx

Utility for importing push subscriptions to Firebase project extra/56254360.mdx

Utility for importing push subscriptions to Firebase project extra/37847945.mdx
Administrator API ..... admin-api/index.mdx
Get accounts list admin-api/accounts/67728.mdx
Add a new account ..... admin-api/accounts/add-account.mdx
Delete the account ..... admin-api/accounts/delete-account.mdx
Update an Existing Account ..... admin-api/users/update-user.mdx
Add a new account ..... admin-api/users/add-user.mdx
Delete a user ..... admin-api/users/delete-user.mdx
Get a list of users ..... admin-api/users/get-users-list.mdx
Sending a Welcome Email ..... admin-api/users/welcome-message.mdx
Synchronize node MTA configuration admin-api/nodes/67761.mdx
Get nodes list admin-api/nodes/67751.mdx
Get node MTA status admin-api/nodes/67753.mdx
Activate node MTA admin-api/nodes/67755.mdx
Deactivate node MTA admin-api/nodes/67759.mdx
Create or update AKMTA sender admin-api/senders/67769.mdx
Get AKMTA sender information admin-api/senders/67767.mdx
Assign account to sender admin-api/senders/67771.mdx
Get senders list admin-api/senders/67765.mdx
Delete sender admin-api/senders/67773.mdx
Restore sender admin-api/senders/67777.mdx
Get sender queue information admin-api/senders/sender-queue/67781.mdx
Hold sender queue admin-api/senders/sender-queue/67783.mdx
Release sender queue admin-api/senders/sender-queue/67786.mdx
Clear sender queue admin-api/senders/sender-queue/67789.mdx
Get virtual senders list admin-api/virtual-senders/67735.mdx
Get virtual sender information admin-api/virtual-senders/67737.mdx
Create virtual sender admin-api/virtual-senders/67739.mdx
Update virtual sender admin-api/virtual-senders/67741.mdx
Clone virtual sender admin-api/virtual-senders/67745.mdx
Delete virtual sender admin-api/virtual-senders/67747.mdx

Getting Started

Altcraft Platform is an omnichannel CDP platform designed for storing customer data and managing communications across multiple channels such as email, SMS, push notifications, messengers, and more.

The platform includes a wide range of features and tools, so during the first login it may not be immediately clear where to start and how different interface modules are connected. This article helps you understand the core logic of Altcraft Platform and points you to the documentation sections worth exploring next.

If you are just starting to work with the platform, we also recommend reviewing the FAQ and Glossary sections.

If you prefer learning through practical examples, this article also includes videos demonstrating basic operations: preparing a profile database and launching the first email mailing.

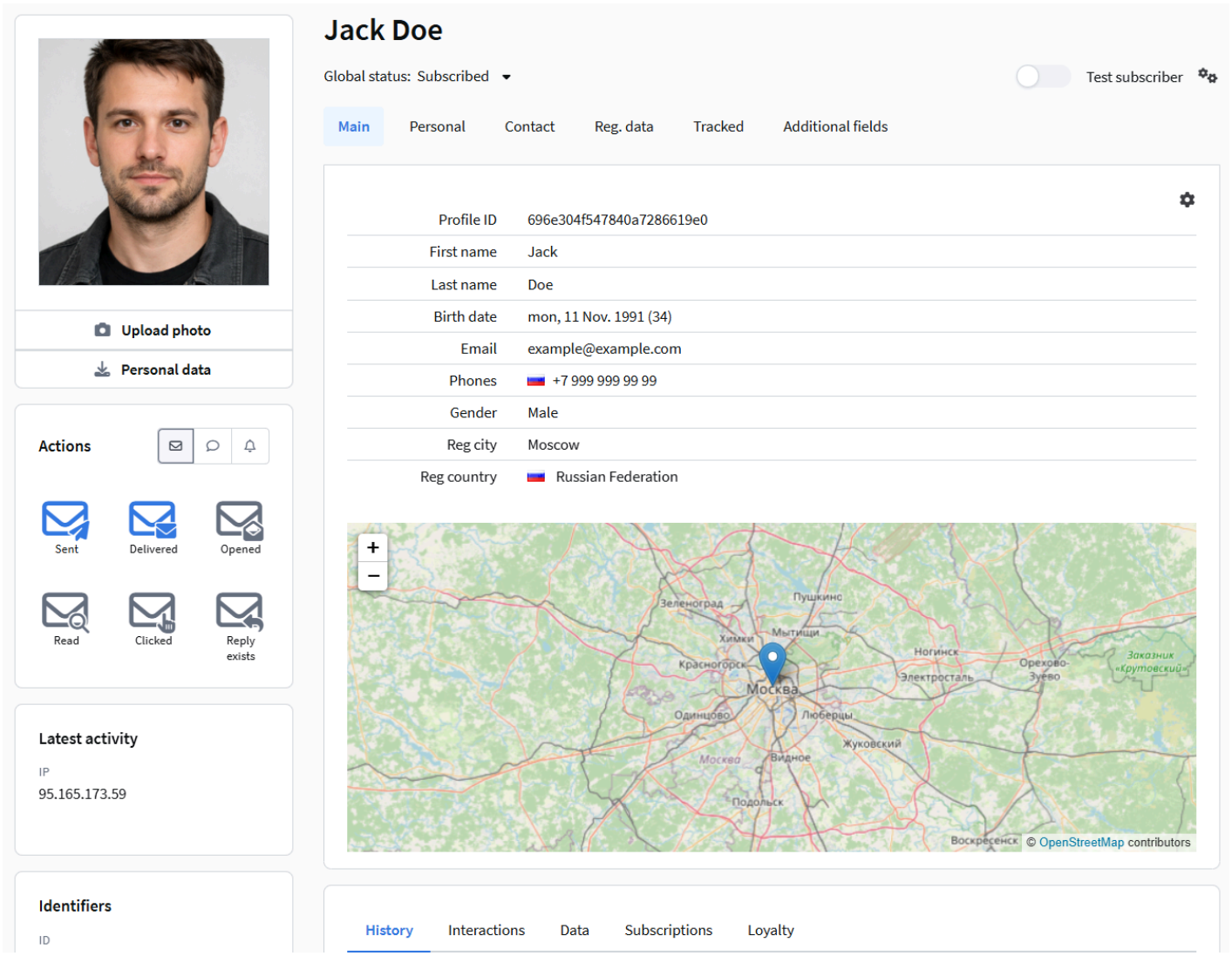
Profile Databases and Subscriber Profile

Altcraft Platform is built around data. All communications, automation, and analytics are based on customer profiles and their actions. All customer data in Altcraft is stored in **profile databases**.

A **profile database** defines the data structure: which fields are used, what data types are allowed, and which resources (subscription sources) can be associated with profiles. In most cases, a single profile database corresponds to a specific product, service, or business unit. More details on data structure configuration are available in this article.

A **subscriber profile** is a unified digital customer profile. It combines contact details, personal information, action history, channel subscriptions, and participation in marketing activities. In the documentation, you may encounter the terms “customer”, “subscriber”, or “user”, but in all cases they refer to the same object — the subscriber profile.

For example, a profile may initially contain only an email address, later include an SMS subscription, and then record activity from a mobile application. All this information is stored within a single subscriber profile.



A detailed description of the profile structure and its tabs is available in this article.

Resources and Subscriptions

Altcraft uses **subscription resources** to manage communications.

A **resource** is a logical content source within which a customer receives messages. It can be viewed as a context or interaction point: a website, mobile application, service, brand, or a specific company project.

Resources allow you to separate customer subscriptions and consents. The same profile can be subscribed to multiple resources and have different subscription statuses in each of them.

A profile subscription links a customer to a specific resource and communication channel. It is the subscription that determines whether messages can be sent to particular contact details.

More information about configuring resources and subscriptions is available in this article.

How profile databases, resources, and subscriptions are created

Below is a basic process of preparing data for communications: creating a profile database, creating a resource, and adding profiles with subscriptions. These steps form the foundation for further segmentation, mailings, and automation scenarios.

Data and Action History

Altcraft Platform tracks all key customer actions and stores them in the subscriber profile history. This includes message sends and opens, link clicks, form submissions, website or app events, as well as order and promo code data.

Action history is used for segmentation, automation, and analytics. For example, you can send a message only to customers who opened emails in the last seven days or made a purchase.

Audience Segmentation

Segmentation allows you to group profiles based on defined conditions.

Altcraft supports static, dynamic, and updatable segments. These segment types differ in how the audience is formed and how data is refreshed.

Static segments are used for working with fixed lists of profiles. Dynamic segments are recalculated at the time of message sending and always use up-to-date data. Updatable segments are recalculated on a defined schedule and store the result until the next update.

For example, you can create a segment of customers who placed an order in the last week or a segment of active subscribers who opened emails during the last month.

More details about segment types and selection conditions are available in this article.

Communications

Communications in Altcraft Platform refer to sending messages to subscriber profiles through various communication channels. The platform supports email, SMS, Web Push, Mobile Push, messengers, and other channels that can be used individually or as part of omnichannel scenarios.

Each channel has its own configuration and delivery specifics, but the overall communication logic in the platform is unified. Messages are sent based on profile data and always take into account contact details, subscriptions, and consents.

Altcraft supports several main types of mailings. Broadcast mailings are used for one-time sends, such as news or promotions. Regular mailings are suitable for repeated communications based on a schedule. Trigger mailings automatically send messages in response to customer actions or specific events.

Communications can be launched for the entire profile database or for selected audience segments. Mailings are often part of more complex processes — automation scenarios or campaigns.

How to launch the first email mailing

> **INFO**
>
> To send email messages, a configured “DKIM” is required for the sender domain. It is not issued automatically. Please contact technical support via chat to set it up.

Below is an example of the full process of creating an email mailing: from preparing a template to launching the send. The video shows how working with profiles, subscriptions, and templates comes together in a real communication process.

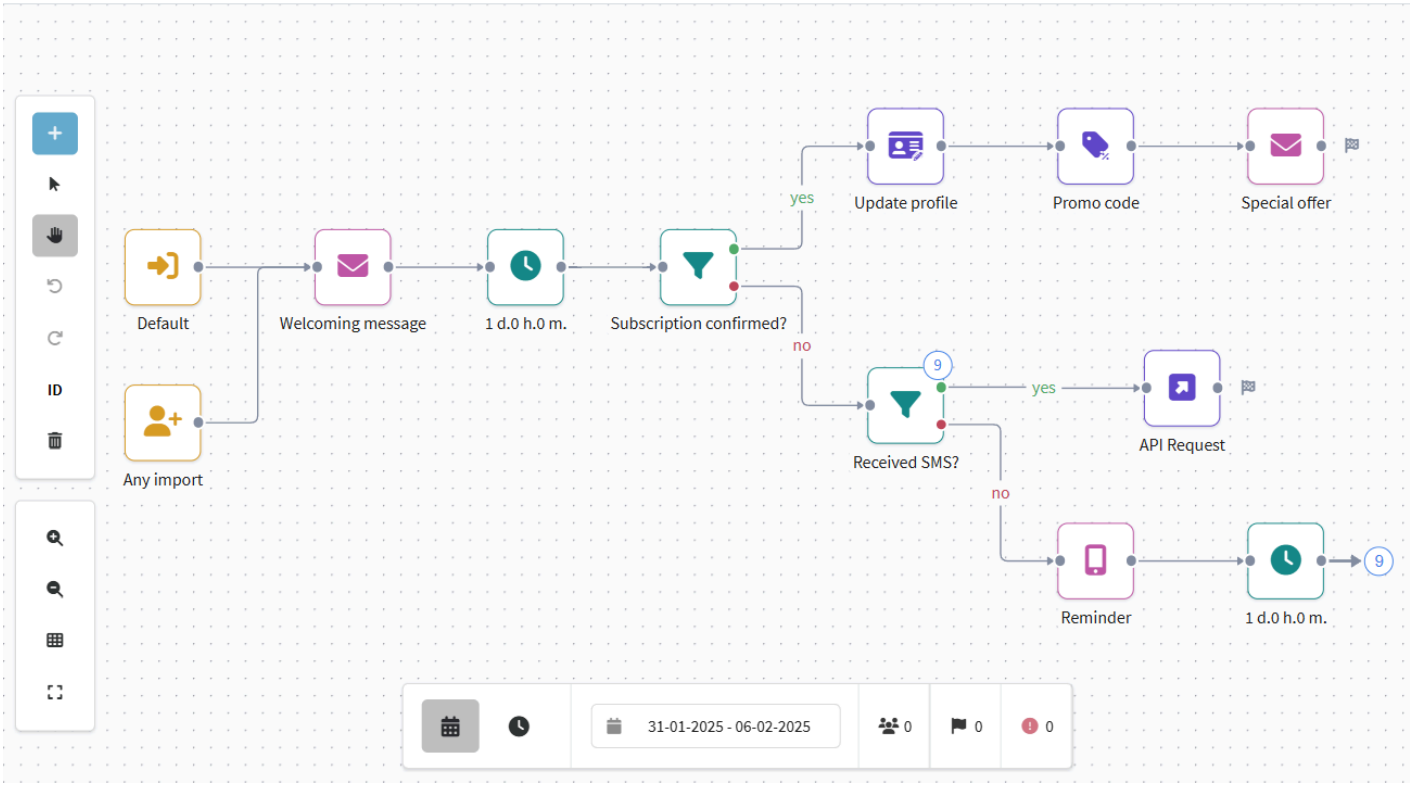
> **INFO: Notice:**
>
> In the interface shown in the video, the term “campaign” may be used to refer to a mailing.
> **In the current version of the platform, the terminology has changed:**
>
> **Mailings** are a separate tool for sending messages (Broadcast, Regular, Triggered).
>
> **Campaigns** are a different tool designed for building complex marketing processes and combining communications, scenarios, and customer interaction logic.

For your first hands-on experience with message sending, we recommend starting with step-by-step channel guides, for example Your First Email mailing. A full list of tutorials is available in the mailing Creation from Scratch section.

Automation Scenarios

Automation scenarios allow you to build communication flows based on customer actions, events, and conditions.

Scenarios are used for onboarding, welcome journeys, reactivation, reminders, and other common marketing use cases. Profiles can enter scenarios either from segments or directly from the database if they meet the entry conditions.



An overview of the automation tool is available in the Automation Scenarios section, and examples of common scenarios can be found in Classic Marketing Automation Scenarios.

Other Platform Capabilities

Altcraft Platform is not limited to data storage, segmentation, and communications. The platform includes additional modules that support comprehensive customer lifecycle management.

Marketing activities can be managed using mailings. Learn more in the mailings section.

Working with product and order data is described in the Market section.

The loyalty module allows you to create reward programs and use promo codes. More details are available in the Loyalty section.

The web layer is used for data collection and subscription management. Learn more in the Web Layer section.

Analytics is covered in the Reports and Analytics section.

Integrations are described in the Integrations section, and technical capabilities are covered in the Developer Documentation and API: Getting Started.

FAQ

Managing profiles subscriptions

Are people who unsubscribe not by the link in the letter, but in the mail service interface, marked as unsubscribing?

Yes, if mail service supports special header, which is transferred into the letter.
Global unsubscription is performed, if resource was not chosen before mailing.
Resource unsubscription is performed, if resource was chosen before mailing.

Why the profile status is changed when unsubscription from non-resource mailing is performed?

If the mailing is performed without selecting resource, global profile status is used.

If the user will subscribe again, will he restore from the suppression list?

No, if the mailing was performed without resource, or with the same resource. Use API to restore him.
In future, we recommend to use several resource subscriptions.

What is resource identifier?

Resource is the mailing source by one or several channels. For example, one website - one resource.
Resource identifier is a number, which is displayed next to its name. It is used in API queries.

What is the difference between suppression list and unsubscription?

Suppression list and unsubscription are different things. Suppression list is commonly used for temporary exception from the mailings, like the situation when content will have negative reaction from the subscribers.

How can we manage subscription confirmation for users, who can see only text version?

They can also have the link. You can set its name like `<name="Name">`. Part in angle brackets will not be displayed while mailing.

What essences can be marked with tag in the platform?

You can mark databases, segments, channels, templates, scheduled imports, suppression lists, mailings, campaigns, workflows, promocodes, summary reports.

Why does the number of unsubscribe link clicks not match the number of unsubscribes?

A user can unsubscribe not only via the link in the email, but also through the service interface (e.g., by clicking a button). These actions are not always tracked as link clicks. As a result, the total number of clicks may differ from the number of actual unsubscribes.

Managing databases

How can developer configure data exchange with the website?

Developer must be provided with API token and link to documentation page — see User API.

Can we upload database without email addresses? For example, for SMS mailing.

Yes, you can. Select coincidence by profile email address. This field will be filled with data like “ID@localhost”.
Profile identifier will be generated by the platform.

How can we spot subscriber territory or country?

The platform can spot geolocation by IP address.

What field type can be used for currency?

Unfortunately, there are no fractional fields in the database. Use integer type.

How can we transfer resource subscriptions during profile import?

You can include subscriptions array into the `data` object during subscriptions import like:

```
subscriptions": [
      {
        "resource_id": 46,
        "channel": "email",
        "email_data": {
          "email": "example_email@example.org",
        },
        "priority": 0,
        "status": "subscriber"
      }, ...
```

Or simple option:

```
subscriptions": [
      {
        "resource_id": 46,
        "channel": "email",
        "email": "example_email@example.org",
        "priority": 0,
        "status": "subscriber"
      },
```

Can we transfer several profiles in one API query for profile import?

For several profiles import, use Massive profiles import.

How can we clear the database?

Use API query to clear the database.

Also, you can clear the database using query to MongoDB.

Can we spot devices, OS and browsers used by customers?

Information is collected in the User Agent field in customer profile.

How can we create the field, which displays customer's interests in different categories?

You can create "Tags" type field in the database. Tags can be added and removed manually, using APIs and workflows.

How can we clone the database with all its profiles and subscriptions?

When cloning the database, only its structure is copied. For copying profiles, press "Copy profiles" button, and after that you will be suggested to select the segment.

If i upload customers from the database, will email addresses be uploaded from the profile or from the profile subscriptions?

You can export only encrypted subscriptions. By default, all profile subscriptions will be exported. In the export menu, you can configure, what subscriptions will be considered.

Why does the report metric not match the number of profiles in the segment?

Reports are based on events, not individual profiles. A single profile can generate multiple identical events — for example, clicking a link twice or opening an email multiple times. In such cases, the event count exceeds the number of unique profiles in the segment.

Why doesn't the static segment recalculate automatically?

A static segment does not update automatically — it is a fixed list of profiles formed based on the specified conditions at the time of creation. If the data in the database has changed, the static segment must be recalculated manually. To do this, click the **"Recount** button at the top of the segment editor.

When recounting, only the current filters will be used. Profiles added or removed manually will not be preserved.

Managing mailings

What is mailings queue used for?

It displays present mailing status for certain mailing.

How can we send mailing to subscribers, who did not receive it?

You can create the segment and select customers, who did not receive the message.

Is there a global limit of mailing amount to one customer in a day or a week? Can it be ignored for some messages? How can we configure it?

Yes, it is called Sending policy. It can be set for a resource or for the account. If necessary, it can be changed in some mailings.

How can we send the mailing partially?

You can set limits for sending in the mailing.

I open the letter and put it into spam, why the complain is not displayed in the report?

Perhaps, FBL/CFL was not configured. Contact your system administrator.

Can we have an example of API query for trigger sending to several subscribers?

```
{
  "token": "e00c3da0a53e4e54a795da58701f4d70",
  "id": 5511,
  "custom_request": {
    "Custom_ID": {
      "$in": [
        "IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiWFItJhr"
      ]
    }
  }
}
```

Can we use blind carbon copy during sending mailings?

Yes, Altcraft MP supports BCC in messages. To use it, see mailing Smart settings and enter address for BCC in **Send hidden copy to these addresses** block.

Managing message templates

How can we use data tables?

See Using content from SQL database in the messages.

Managing scenarios

How does the profile move through the scenario if the mailing element failed to send a message?

If sending a message ended in an error (e.g. due to a technical reason), the profile automatically moves to the next element according to the scenario.

How does the profile move through the scenario if there is an error in the element of an API call, profile update or condition?

If an error occurs in the element and the scenario does not provide a separate exit for error handling, the profile will be excluded from the scenario. This is done to prevent incorrect execution.

The HTTP element supports error handling — if a request fails, it can retry or send the profile along an alternative path.

What happens if you delete the element where the profiles reside?

If an element is deleted during scenario execution and it contains active profiles, such profiles will be immediately excluded from the scenario.

How to see the apicontent of a profile?

To view the apicontent of a profile, hover the cursor over the desired element in the scenario. An icon with a number above it will be displayed - this is the number of profiles in the element. Click on this icon to open the list of profiles. In the opened window you can see the data of each profile, including the apicontent block, if API requests were used in the scenario.

Recommendations for managing scenarios

- Try to avoid conditions with negation (e.g. “is not equal”) - it complicates the logic and reduces the visibility of the scenario.
- It is recommended to filter out most of the profiles already in the start element to minimize unnecessary load on the scenario.
- Use static segments.

Other

Not all functions are available in the interface.

Contact platform administrator for turning Smart settings on.

What coding is used for files import?

UTF-8 coding is used.

How can we download the report?

Select Tasks in the main menu.

Is it possible to automatically update profile fields during event transfer into the platform?

Yes, you can use workflows for this.

Why are there no complains in the report?

FBL must be configured for complaints display in the report.

> **NOTE**
>
> Complaints are processed with automatic subscription change.

What is pixel? What is it used for?

Pixel is a script, that send query to the tracking server, when subscriber performs any actions on the website or in the application.

Pixel can transfer information related to this action, like website address, which is subscriber source, achieved goal and others. This information is written into every pixel code and it is transferred in profile history. Achieved goals and action price can be used for segmentation, including automatic segmentation in the workflows.

Also, pixel information is used in “Pixels” and “Goals” reports.

Is it possible to make segment by User-Agent parameter?

Yes, it is divided by Browser/Platform in the segments.

Promocodes import and generation buttons are not available.

You need to save loyalty program first.

The file name in the attachment in emails has incorrect name.

The problem can be resolved by the following methods:

1. Reinstall Outlook from the Standard package.
2. Configure the Outlook client: use **UTF-8** for the **Outgoing message encoding parameter**.

- In Outlook select **File** -> **Option** -> **Advanced**.
- In the **International options** section, set the **Preferred encoding for outgoing messages** option to **UTF-8**.

3. Avoid using Russian characters in the file name.
4. Consider using a different version of Outlook.

How are email opens and reads tracked?

At the end of each email, several 1×1 pixel images are automatically added.

When the first pixel is loaded, an open is recorded. If the second pixel is loaded with a short delay (up to 8 seconds), a read is recorded.

How is a profile’s geolocation tracked?

A profile’s geolocation is determined by its IP address at the moment the email is opened or an action is performed.

Altcraft glossary

Email Marketing Terms

API (Application Programming Interface) — the language of communication of one computer program with another. By delivering requests and returning responses, two computer programs talk to each other in order to execute some common task together.

BCC (Blind Carbon Copy) — hidden recipients of the letter, whose addresses are not shown to other recipients.

Caller ID— the identifier of the subscriber-sender of SMS messages. It can contain Latin letters and numbers without spaces.

Double Opt-In — the user indicated his email at the moment of subscribing and confirmed it by clicking on the special link in the confirmation letter. Advantage: a more interested and active audience gets into the database. Disadvantage: conversion reduces, because a part of the audience does not confirm the subscription.

DKIM (Domain Keys Identified Mail) — an email authentication technique that adds a digital signature to the email. It allows the receiver to check that an email was indeed sent and authorized by the owner of that domain.

OR (Open Rate) — the percentage of subscribers who open a specific email out of your total number of subscribers.

CTR (Click-Through Rate) — the ratio of users who click on a specific link to total nubmer of delivered messages.

CTOR (Click to Open Rate) — the ratio of users who click on a specific link to the number of total users who open the letter.

SB (Soft Bounce) — the letter returns because the email server is temporarily unable to deliver your message. There are two possible reasons: the recipient's email box is full or recipient's email server is temporarily inaccessible.

SPF (Sender Policy Framework) — an email-authentication technique that allows you to check if the sender's domain is forged. It is used to prevent spammers from sending messages on behalf of your domain. You need to add a special TXT record to the sending domain (sender domain, from domain), which determines the policy of allowing sending this domain from various hosts.

SQL (Structured Query Language) — a special language used for managing databases. With the help of SQL, you can set the structure of the database, organize the selection of data from it according to some specific criteria, delete data.

HB (Hard Bounce) — email addresses that do not accept emails due to invalid emails or an unexpected error during sending.

Pop-up — a small window that appears on top of the main content of a web page.

SMS gateway — an infrastructure component that enables the delivery of SMS messages from the Altcraft platform to mobile network operators (or intermediaries). It handles routing, formatting, message queues, delivery reports, and error processing.

Event capture — a mechanism for recording and transmitting user events (e.g., page views, cart additions, link clicks) to the Altcraft platform. These events are used for segmentation, personalization, trigger-based communications, and analytics.

Web version of email — an online page that displays the content of an email (HTML) identical to the sent message, but accessible via a web browser.

Altcraft Marketing Platform internal terms

Subscriber profile — contains information about mailing receivers - customers, partners.

Profiles database — defines structure of profile data fields and subscriptions.

Static segment — fixed profiles list, formed by several conditions, or by other ways (manually, by scenario or API). Static segment can be counted during selection from the database, but it will not be recounted automatically.

Dynamic segment — set of rules, that will be used for selection in the moment of using the segment — like mailing launch.

Profile subscription — a record that combines contact data for a specific communication channel and information about registration.

Resource — objectfor organizing customer subscriptions to certain content via certain communication channels.

Mailing — creates message for certain resource subscribers, based on selected template, and adds customer information from the database, dynamic content, parts and fragments.

Broadcast mailing — tool used when you need to communicate to a specific audience one single time.

Regular mailing — a tool for organizing **massive multiple** mailings to your subscribers.

Trigger mailing — tool for organizing transactional messages.

Multivariate testing — mailing that can have from 2 to 8 alternatives with different content and settings. You can compare efficiency of this messages and select the best one.

Mailing log — tool for tracking all stages and errors during mailing sending.

Message template — a set of content that you send to your customers within one or more marketing mailings.

Scenario — sequences of actions used for marketing workflow automation.

Channels report — a report that provides data on all events tracked in mailings: sends, deliveries, opens, clicks, and more.

Traffic report — a report that provides data on profile traffic from sources such as channels, forms, pixels, and pop-ups.

Cohorts report — report by groups of subscriber profiles from specific databases imported at the same time period.

Lifetime report — a tool to analyze customer cohorts lifetime.

Conversion funnel report — you can analyze your marketing strategy conversion or count it for certain action using this report.

Goals report — this report shows analytics about website visits and performing target actions like: income form deal, partner commission etc.

Audience growth report — report that shows subscriber profiles quantity changes.

Click map report — report that allows you to compare different message versions and different links values

User panel — interface for user to work with Altcraft MP.

Administrator panel — service interface used by Altcraft MP administrator.

Tag — mark for databases, segments, message templates, database imports, forms, suppression lists, products and orders, used for navigation in platform; used to navigate and improve the search for these entities in the platform.

Connection — a mechanism type which let you authorize in external systems to expand Altcraft Marketing capabilities with the functionality of these systems. You usually need an account in an external system with API access rights to create such a connection.

Sending policy — settings which regulate number of messages per client. Such policy can be applied to an account, a resource or a mailing.

Suppression list — ɜa list of email addresses, email domains and phone numbers, which you do not want to send messages to. Suppression list filters an email address or a phone number at moment of mailing launch.

Tracking domain — a domain for tracking subscriber activity. It allows you to find out which links the recipient of the message has clicked on.

Pixel — HTML or JavaScript code placed on website, used for customers behavior tracking.

Watchers database — profiles database, that can be used to control mailings sending process and result. These profiles will receive same messages as your customers.

Market — functionality that allows you to import product and order data from various marketplaces into the platform and use this data to communicate with customers.

Form — a service that allows you to receive feedback from your customers. So you can find out the missing information directly by creating questionnaires or surveys.

Tag manager — a tool that allows you to manage HTML tags and JavaScript code without any need for developers. It is opportunity for you to create, add, and update code snippets on your site to track user activity and feed data into the platform.

mSDK — a mobile Software Development Kit (SDK) for integrating push notifications into mobile applications. It helps developers easily connect the Altcraft notification system to their Android and iOS apps via Firebase and APNs platforms.

Best Send Time Module — a feature that automatically calculates the optimal time to send a message to each user based on their behavior and historical data (activity, opens, clicks). It helps increase the effectiveness of mailings by minimizing idle time and improving engagement.

Subscription manager — a page that allows subscribers to manage their profile data and subscriptions to a resource: pause, resume, or suspend them for a specified period (14, 30, 60, or 90 days). If your company already uses its own subscription management system, it can be integrated with the Altcraft Platform.

Documentation archive

With the release of new versions of Altcraft Platform, the documentation presented on the website is updated in accordance with platform changes. For user convenience, documentation for outdated versions is stored for six months after the release of the next version. Below are links to the administrator documentation, user documentation, and API reference combined into a single PDF file.

- **Version 74** — PDF
- **Version 74** — PDF
- **Version 73** — PDF
- **Version 72** — PDF
- **Version 71** — PDF

INFO: Important

The documentation versions listed above may contain factual inaccuracies. No changes are made to archived versions; they remain in the state in which they were archived.

Profiles and databases

How to start

Create a database

	Start from creating a database, where you will store digital profiles of your customers. Define the structure for your base, using standard data fields and custom data fields to implement **Creating a database is your first step in using Altcraft MP.**

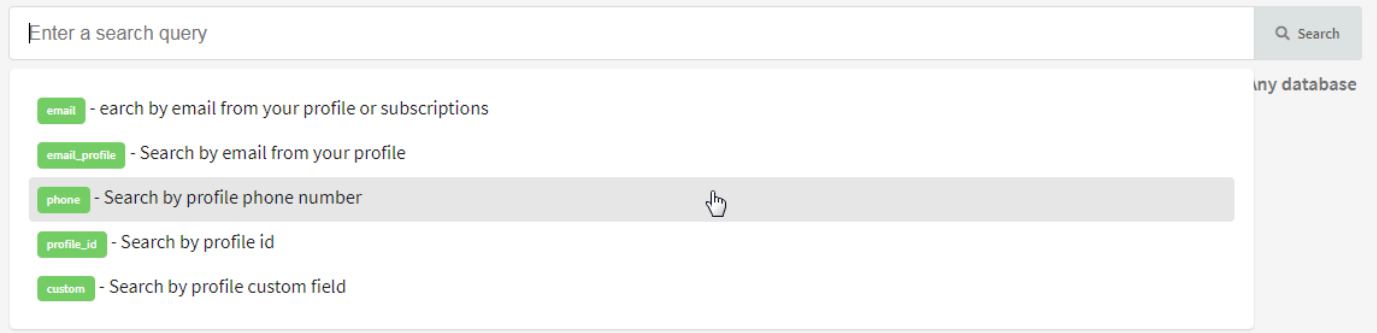
Create subscriptions resource

	Communications with customers in Altcraft is designed for interaction with several communication channels, including different contacts in one channel. Subscriptions of profiles for different types of mailing (advertising, inbound, service, partners) are grouped into Resources. A resource is a content publisher for which you can separately configure sender data, tracking, sending policies, and also communications channels used. **Create at least one resource to start communication with your customers**

TIP

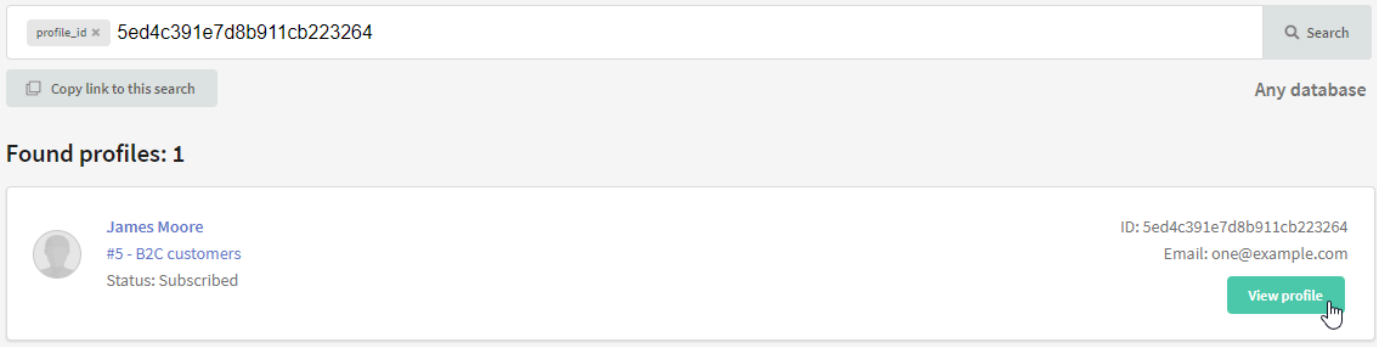
To **find** a specific customer profile in your databases use the **quick search** tool (**Data —> Search**). Just start entering - or paste - your customer's data.

For email you can specify to search for subscriptions, or profile data only:



If you need to search for a profile by an **additional (custom) field**, use the following format: `field_name:value`.

You can copy link to search query and view the profiles:



Contents

Subscription resources

Resource in Altcraft is a tool for organizing customer subscriptions to certain content via certain communication channels. Subscriptions to different resources are not connected to one another. Unsubscribing to one resource does not affect other subscriptions and **global profile status**, thus a subscriber can choose which content to receive - and you get more opportunities to retain your customer.

One or several communication channels can be configured in a resource for forming subscriptions. A resource can be:

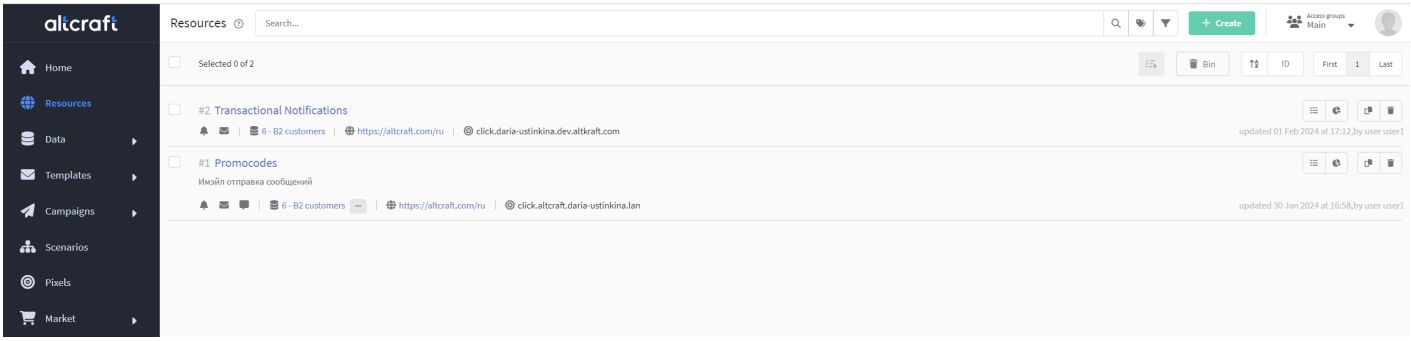
- Website or landing page
- Mobile or desktop application
- Registration form on the partner's website
- Certain type of mailing
- Special offer or product

For example, if a subscriber is not satisfied with advertisement mailings, but he is interested in information mailings, using two resources, the chance of saving potential customers and making a deal is much higher.

Different resources can use different tracking and sender domains - and different **From names** as well. In channel reports, general statistics intersections are available for resources, which makes effective end-to-end analysis of different marketing strategies possible.

Managing resources

Choose **Resources** in the Main menu to start managing resources.



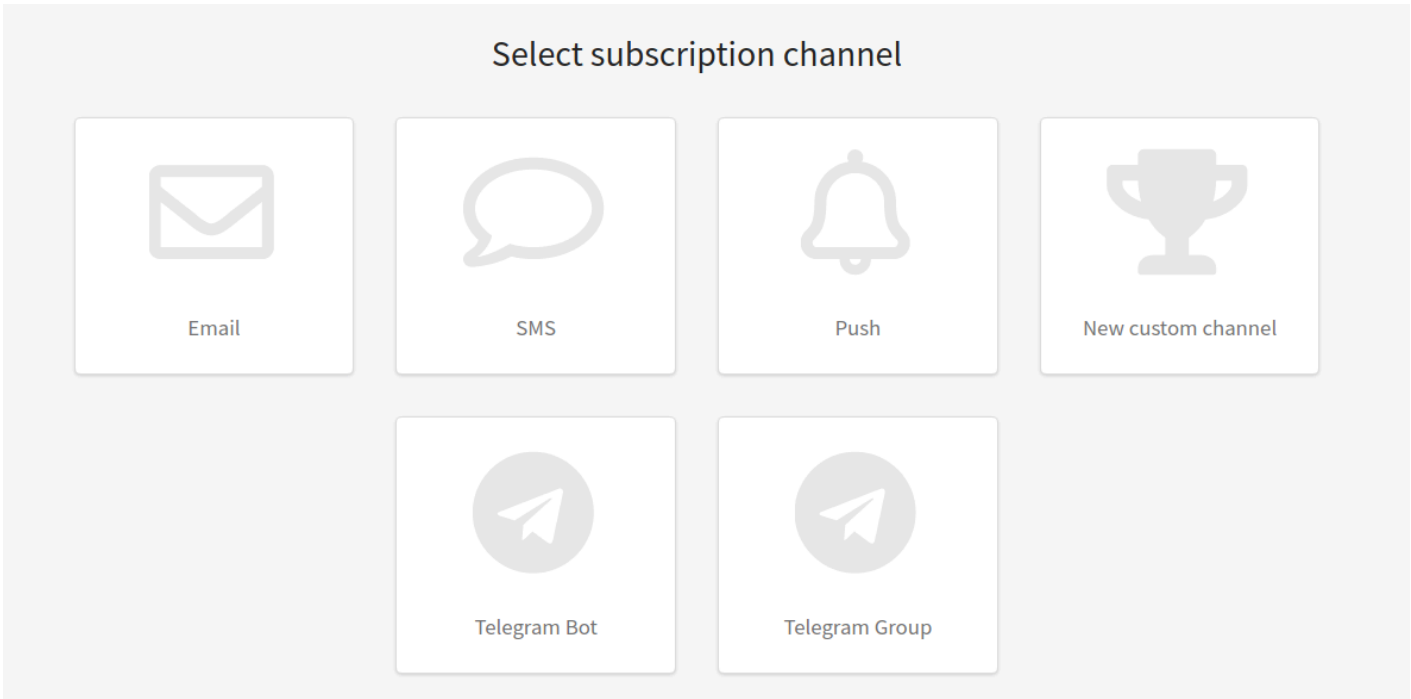
Click on the resource name to edit it. Use the action menu on the right side to view the resource's subscribers

, clone the resource

, or delete it

. To add a new resource, click **+ Create** at the top of the page.

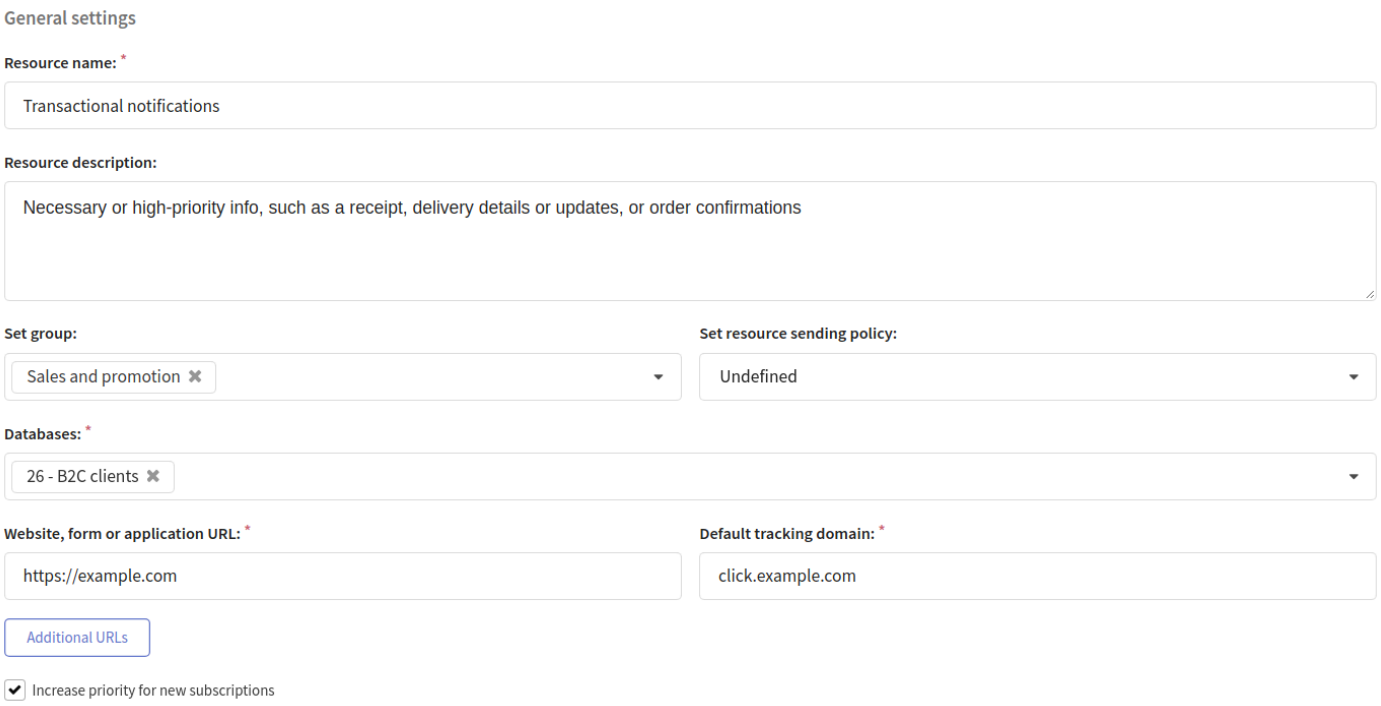
Next, select a primary subscription channel. Note that you can add another one at any time:



General settings

For a new resource, enter a **Name** and **Description** (optionally). For every resource you can configure:

- **Access group** — place resource in account **objects group**.
- **Resource sending policy** — limitcommunication with one subscriber within the resource.
- **Databases** — selectdatabases whose profiles can subscribe to the resource.
- **Website, form or application URL** — URL of your resource or its opt-in page. Additional URLs can be added.
- **Default tracking domain** — domain for tracking subscriber actions within the sent messages. In mailing **Smart settings** you can override or add tracking domains.



TIP: Increase priority for new subscriptions

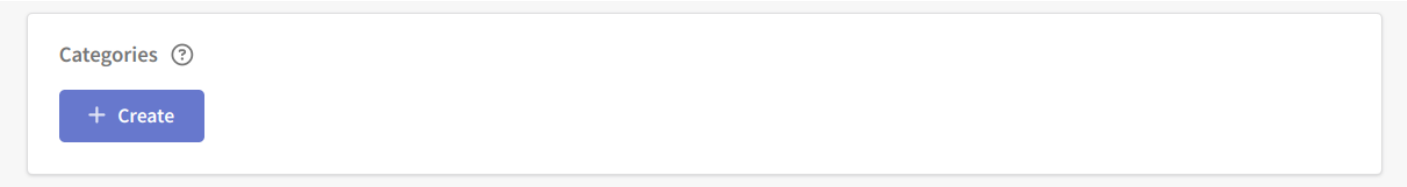
A user can subscribe to one resource several times by entering different contact details (email or phone number). If you activate the "**Increase priority for new subscriptions**" option, the contact information that the user provided during the last subscription will become a priority. It is to this email or number that messages will be sent in subsequent mailings.

Please note that the "Increase priority for new subscriptions" option applies only if the priority is not explicitly specified when adding a new subscription. If the priority of the new subscription is explicitly specified (for example, in an API request), then the priorities of existing subscriptions remain unchanged.

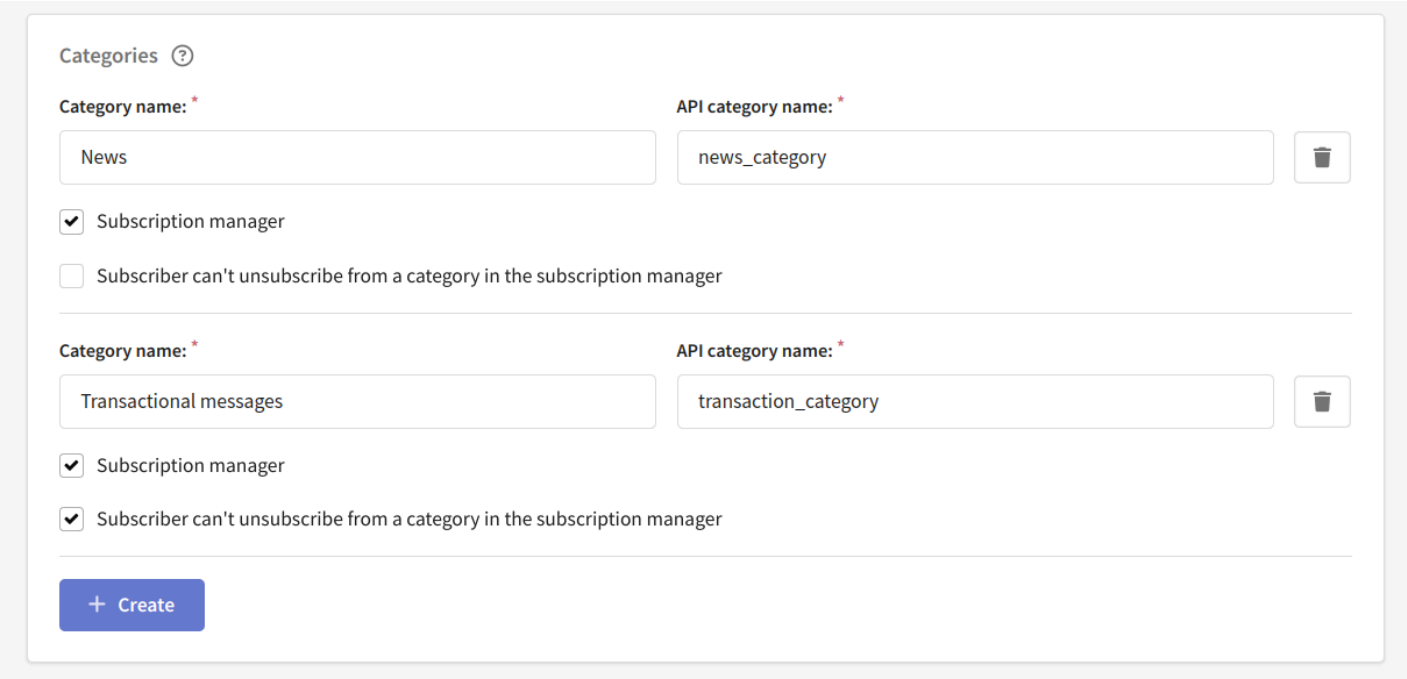
Resource categories

Categories enable you to specify the type of content a profile is subscribed to. For instance, a resource like "Mobile App" can include categories such as "News", "Promotional Offers", and "Transactional messages". If a client unsubscribes from one category within a resource, they will still receive messages from other categories. Unlike subscribing to a resource, subscribing to a category has just two statuses: "Subscribed" and "Unsubscribed". You can manually modify this status in customer profiles or via API requests.

To add a new category to a resource, click the **+ Create** button:



Specify a custom **category name** and **name** that will be used in **API requests**:



Additionally, you have the option to add a category to the subscription manager, This way, you allow subscribers to manage their category subscriptions by their own. If receiving messages within this category is required, activate the option “Subscriber can’t unsubscribe from a category in the subscription manager”.

The number of categories is not limited. To delete a category, click the button

.

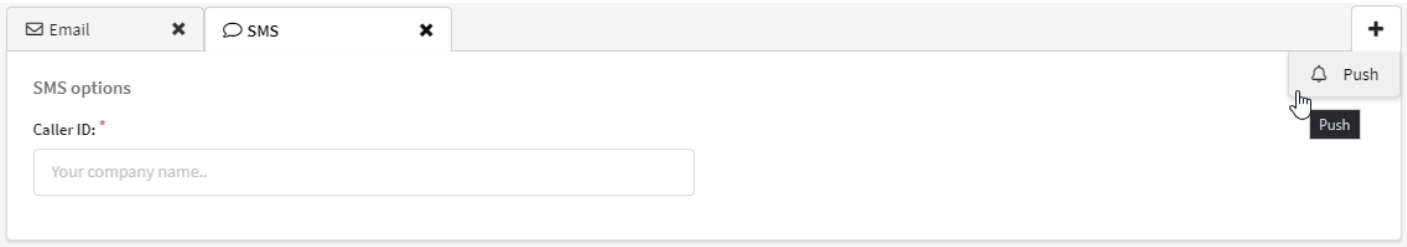
Please note that after saving the resource, you will be able to edit the category name, but not the API name.

Communication channels settings

INFO

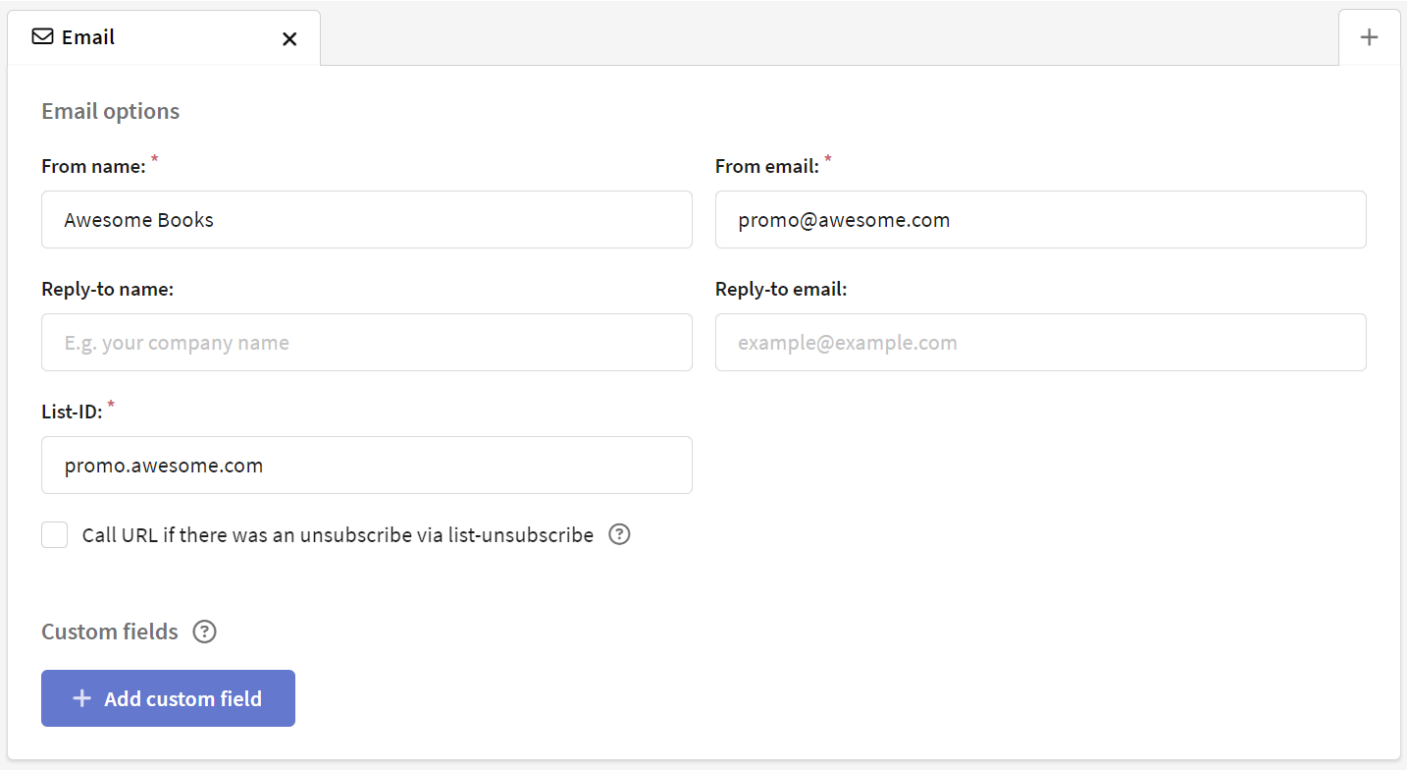
You can add **no more than 8 communication channels** for a single resource!

Each resource can form subscriptions via one or several communication channels. 7 channels are available out of the box: Email, SMS, Push, Telegram, WhatsApp*, MAX, Viber and Notify. Custom channels are preconfigured by platform administrators. To add a channel configuration tab, press **plus** on the right side:



Email

To communicate with customers by email, enter **From name**, **From email**, **Reply-to name** and **Reply-to email** - in case you will gather replies to your mailings:



You also need to fill in the **List-ID** field. List-ID is an additional header included in a message that transmits a unique mailing list identifier. Email clients use information from the header for filtering and sorting incoming messages. Details of the List-ID standard can be found here.

Enter the List-Id in the format `[list-label].[domain]`. Examples:

- `list-header.nisto.com`
- `commonspace-users.list-id.within.com`
- `0Jks9449.list-id.cmu.edu`

List-ID is specified without `<` and `>` symbols. The length of the identifier should not exceed 255 characters. The domain should match the sender's domain.

INFO

It is recommended to use different List-IDs for each resource.

Unsubscribe settings in the Email channel

The platform provides two mechanisms for unsubscribing from the Email channel in the resource:

1. **Unsubscribe link in the List-Unsubscribe** is transmitted in the header of the letter for use by mail services.
2. **Unsubscribe link in the body of the letter** is added to the template using the variable `{unsubscribe}`.

The following settings are available to manage unsubscribe via **List-unsubscribe**:

- **Don't use list-unsubscribe in email** — unsubscribe list-unsubscribe link is not added to the header parameter.
- **Call URL if there was an unsubscribe via list-unsubscribe** — the required field will be displayed **Called URL**. It should specify the URL that the user will go to after unsubscribing via the unsubscribe link of the mail service. If the URL is omitted, the user will be redirected to the system unsubscribe page.

CAUTION

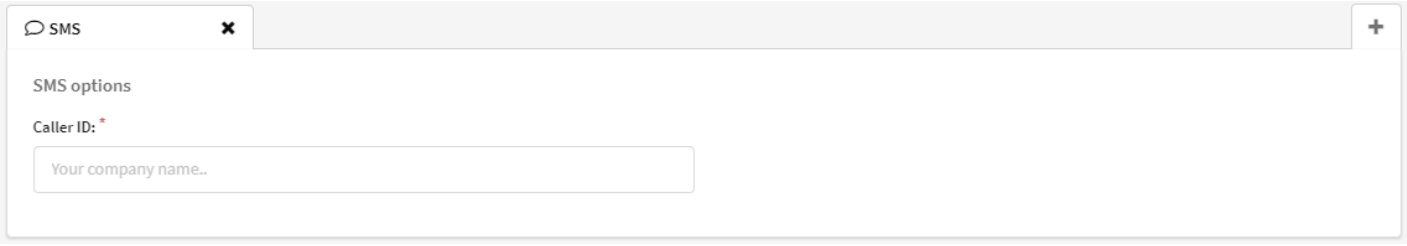
Unsubscribe option **Don't use list-unsubscribe in email** is available only for users of the On-Premise solution. To activate this option, add the `unconditional_subscriber` parameter to the `EXTRAS` property in the configuration file `main.json`.

TIP

To communicate with subscribers via the email channel, you need to use email addresses formatted as john@example.com. Email addresses must comply with RFC standards. You can find the main requirements at this link.

SMS

To communicate with short telecom messages, activate the **SMS channel** in the resource and enter **operator name (Caller ID)**. Generally,it needs to be agreed on with your SMS provider, unless you use your own telecom gateway.



TIP

Communication with subscribers through the SMS channel is done using their phone numbers. The number must be provided in international format, without spaces, hyphens, or other separators, and it should start with a plus sign. For example: `+79123456789`. Learn more.

Push

CAUTION

If a custom unsubscribe link is specified in the email template, then it will be a priority. Read more about user links in the article.

Push channel combines settings for all supported platforms:

Web browsers: Google Chrome, Mozilla Firefox, Apple Safari and Opera browser

Mobile applications: Android and iOS.

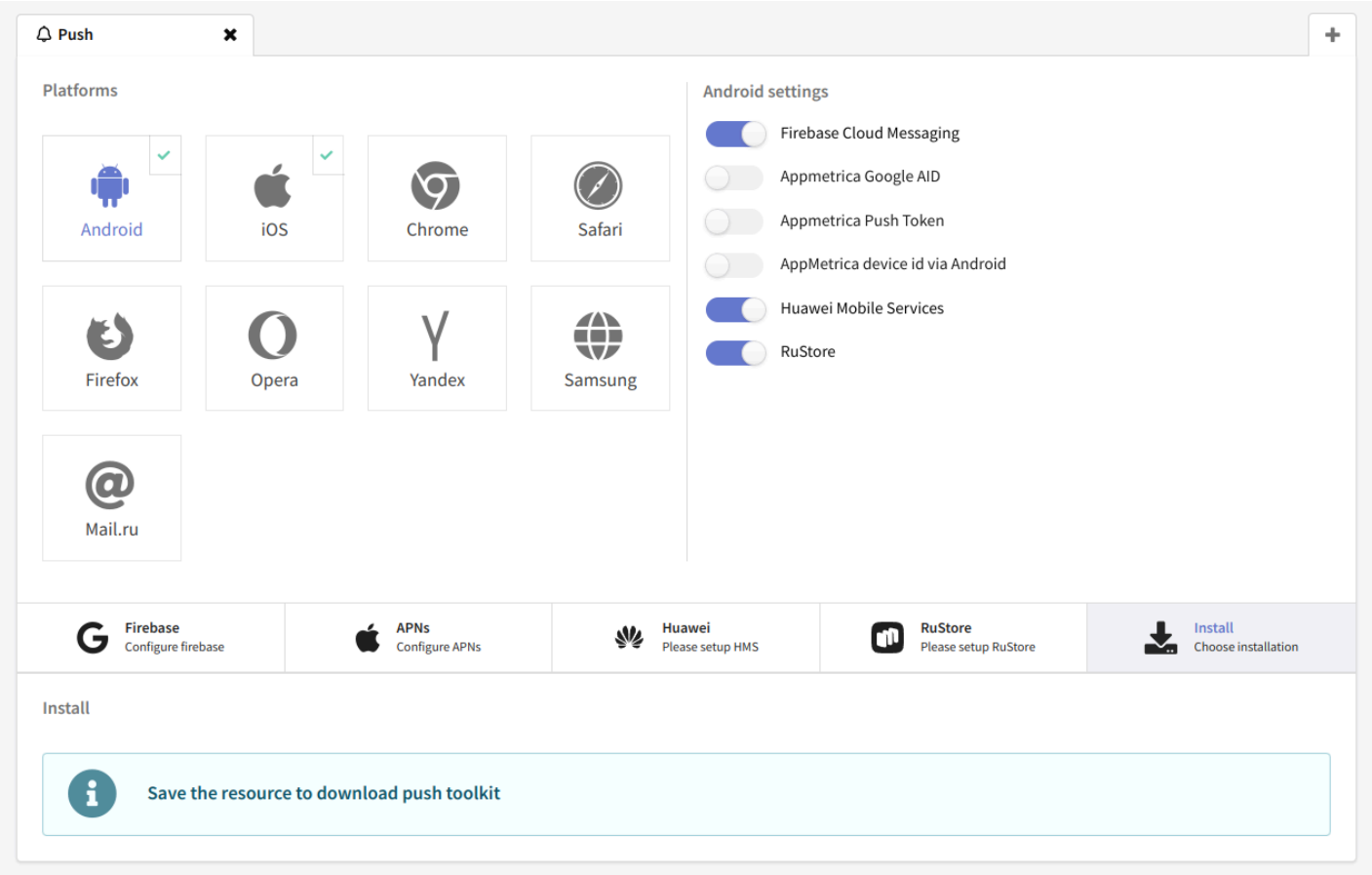
The following providers are available out of the box:

- **Google Firebase Cloud Messaging** — all platforms, except Safari
- **Apple Push Notification Service** — Safari & iOS
- **Mozilla Services** — Mozilla Firefox
- **Yandex AppMetrica** (AppMetrica integration) — Android & iOS with Google FCM SDK.
- **Huawei Mobile Services** — Android & iOS
- **RuStore** — only Android

To work with push notifications, activate the **Push channel** in resource and configure the necessary platforms and providers.

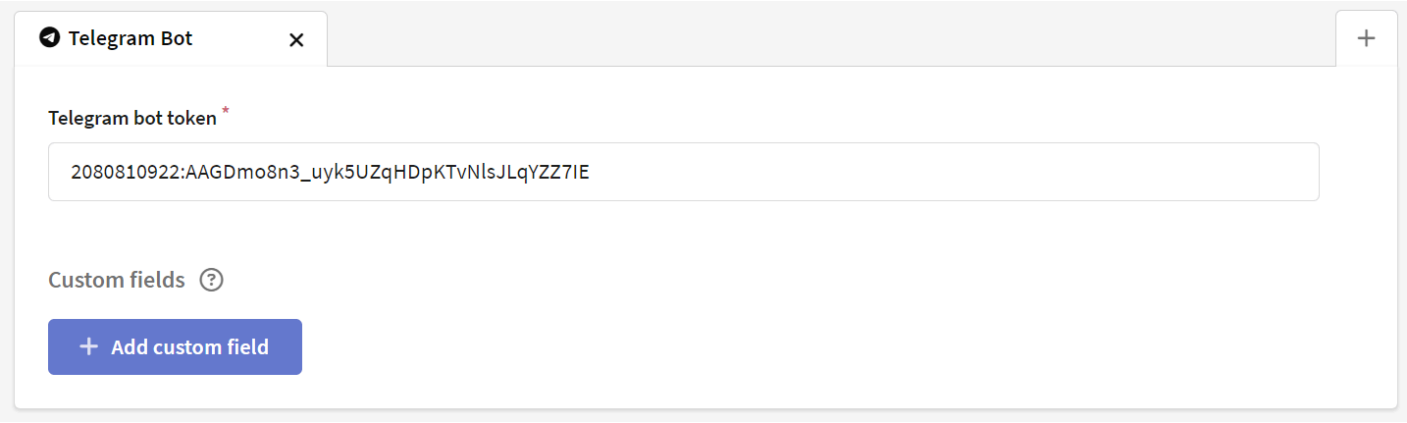
TIP

Configuration options for the supported platforms are described in detail in this article.



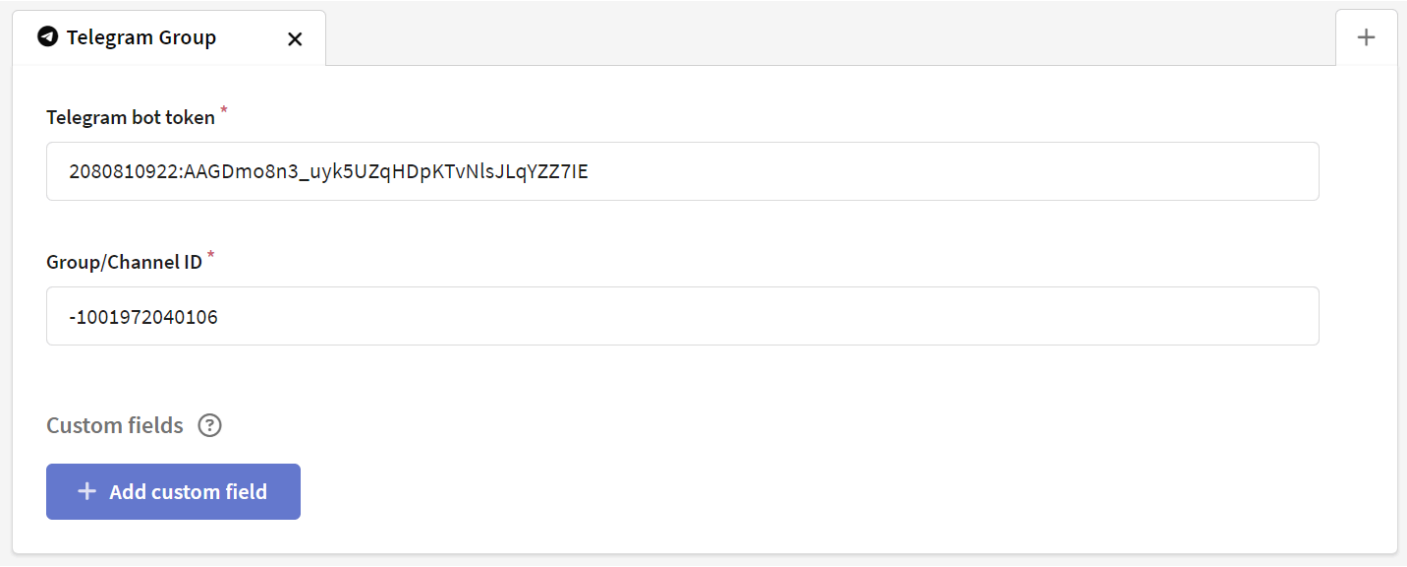
Telegram Bot

Use this channel to send messages to subscribers via the Telegram bot. You need to enter the bot token. Learn how to get a token in this guide.



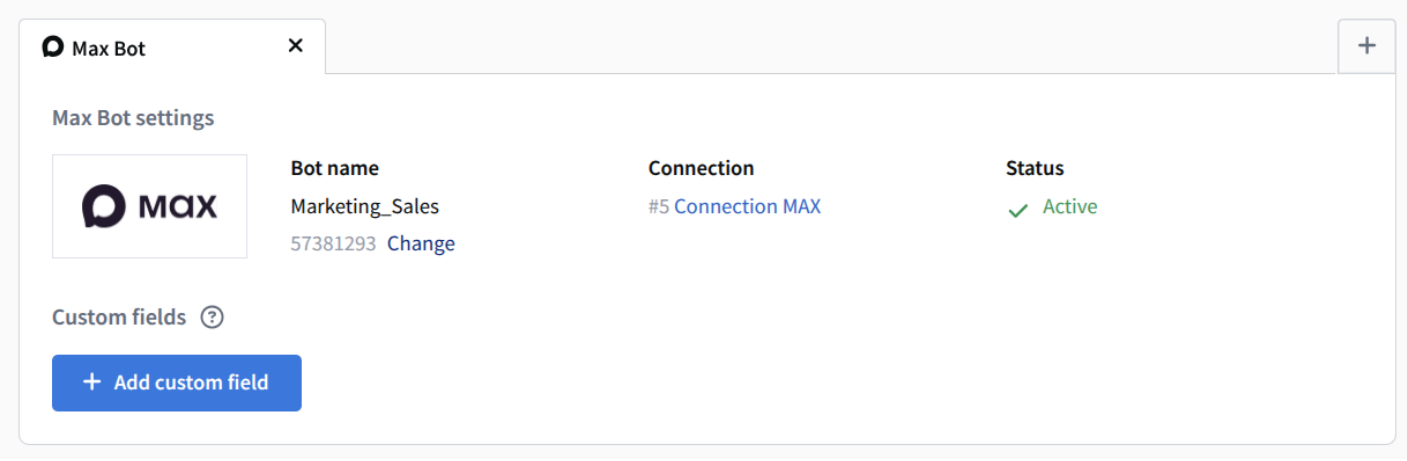
Telegram Group

You can send messages to this channel using placement mailings. Specify the bot token and group or channel ID. More information about channel setup can be found right there.



MAX Bot channel

Use this channel to send personalized messages to subscribers via a bot in the MAX messenger. To create Campaigns in this channel, you need to create a bot connection and configure the MAX integration. In the resource settings, add the MAX Bot channel and select the bot linked to the integration:

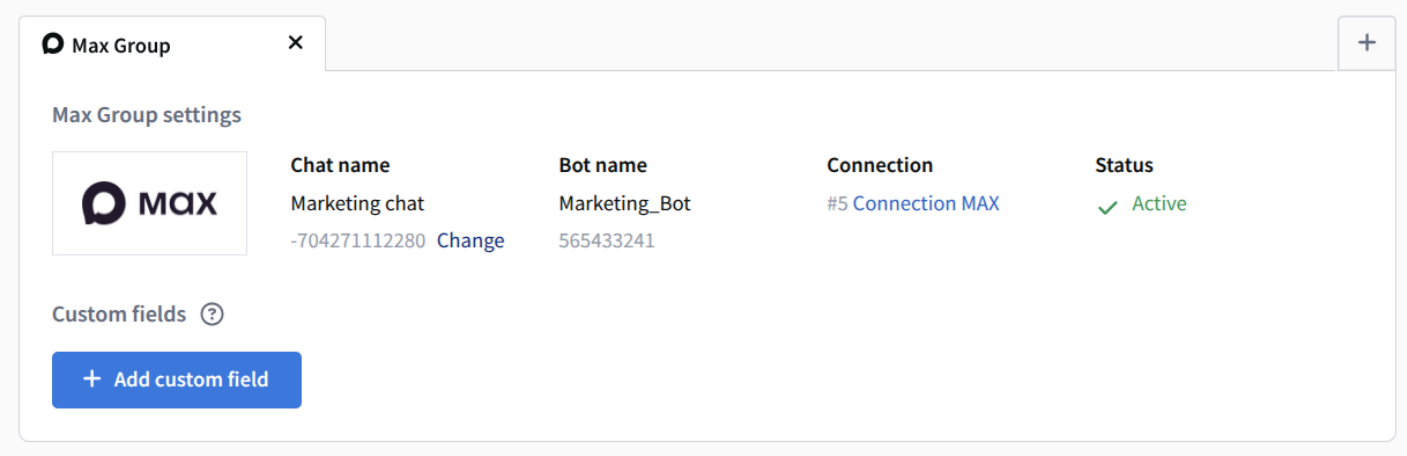


A full step-by-step guide is available on this page.

MAX Group channel

Use this channel to send messages to MAX chats using placements. This is an **indirect communication channel**, meaning Campaigns in this channel cannot be personalized using profile data variables.

To create Campaigns in this channel, you need to create a bot connection and configure the MAX integration. In the resource settings, add the MAX Group channel and select the chat linked to the integration:

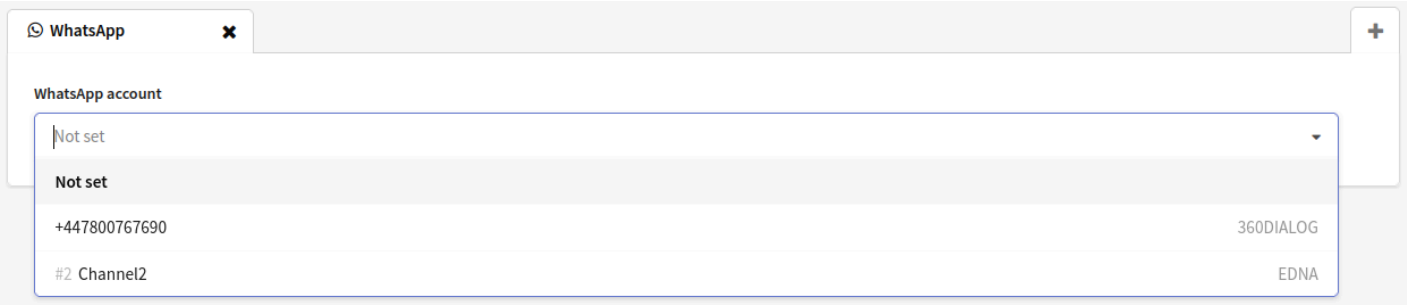


A full step-by-step guide is available on this page.

WhatsApp

To send messages to subscribers in the messaging app, you need to set up integration with the WhatsApp* Business API. Integration can be done through such providers as edna or 360dialog.

After setting up the integration, select the connected account (360dialog) or channel (Edna) in the drop-down list:



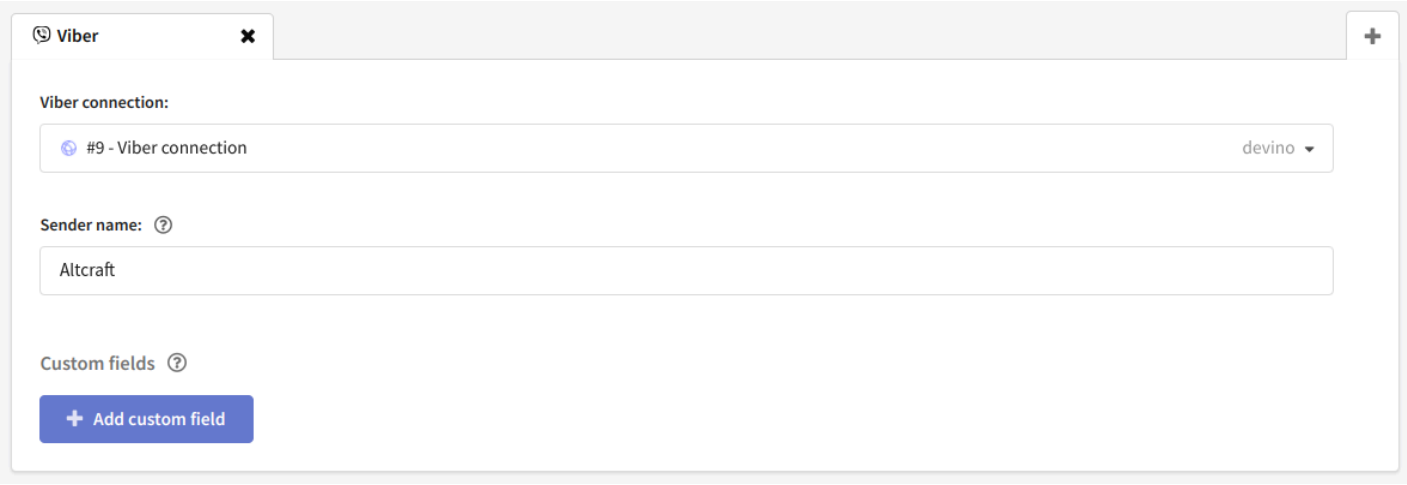
TIP

Communication with subscribers through the WhatsApp* channel is done using their phone numbers. The number must be provided in international format, without spaces, hyphens, or other separators, and it should start with a plus sign. For example: `+79123456789`.

Viber

To send messages on Viber, you need to set up an integration through the provider, such as Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto.

In the channel settings, select the Viber connection with Devino Telecom, SMSTraffic, MTS OmniChannel or Rapporto you created earlier and specify the **sender's name**. This name will be visible to subscribers when they receive a message. You can find the sender's name in your provider dashboard, and it should be verified by your provider.



INFO

To set up callbacks with message statuses, please contact your provider's account manager or technical support (Devino Telecom tech support, SMStraffic tech support, MTS OmniChannel, Rapporto).

TIP

Communication with subscribers through the Viber channel is done using their phone numbers. The number must be provided in international format, without spaces, hyphens, or other separators, and it should start with a plus sign. For example: `+79123456789`.

Notify

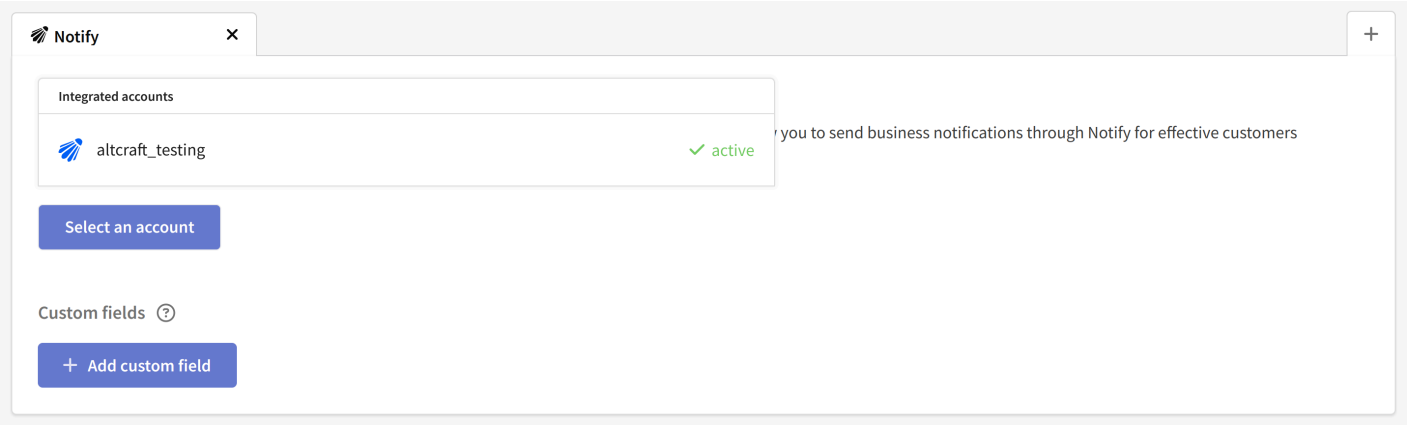
INFO

According to the service policy, new access to the Notify service personal account is currently **not provided**.

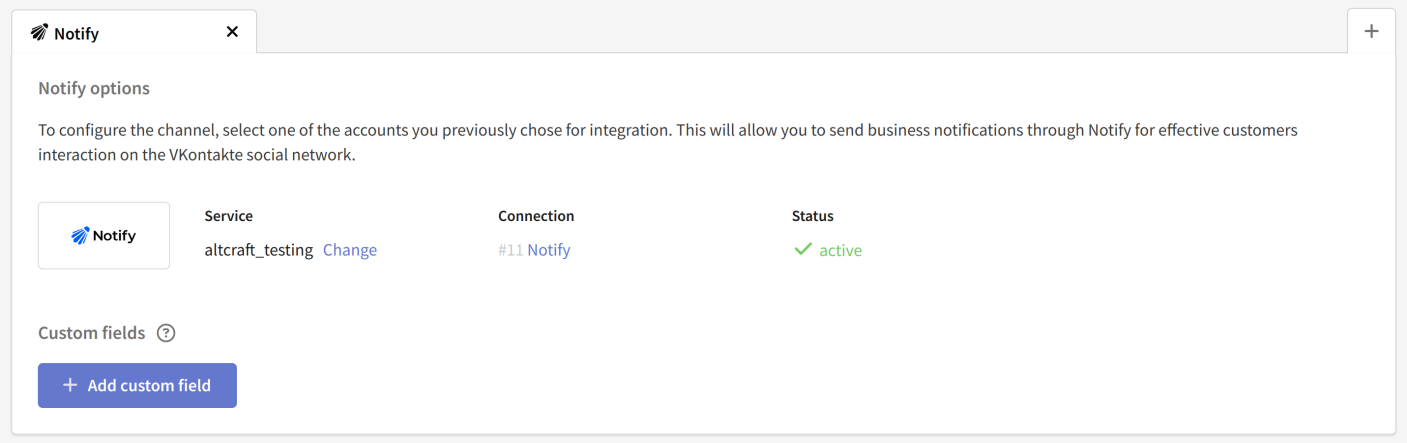
Support for current users is provided by MAX Business tech support.

Notify service allows sending business notifications via VK and OK for effective communication with customers. To use Notify in the platform, you need to pre-configure connection and integration of the service.

In the channel settings, select the previously integrated Notify account:



Check if the account is active and save the resource:

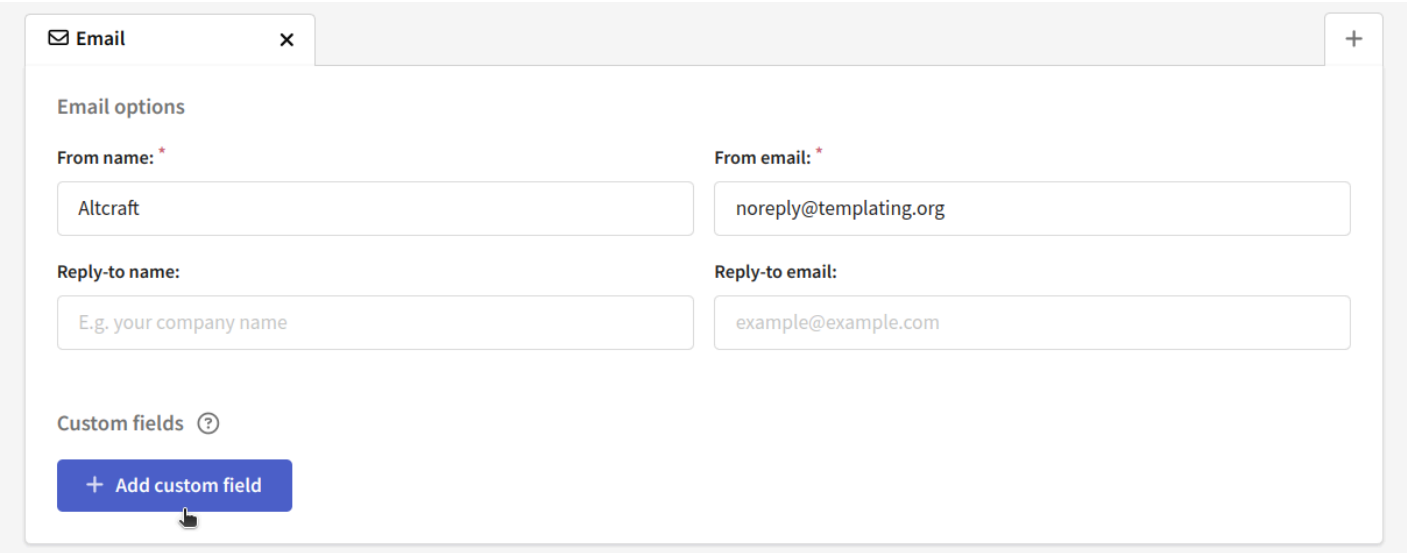


Custom fields in subscription channels

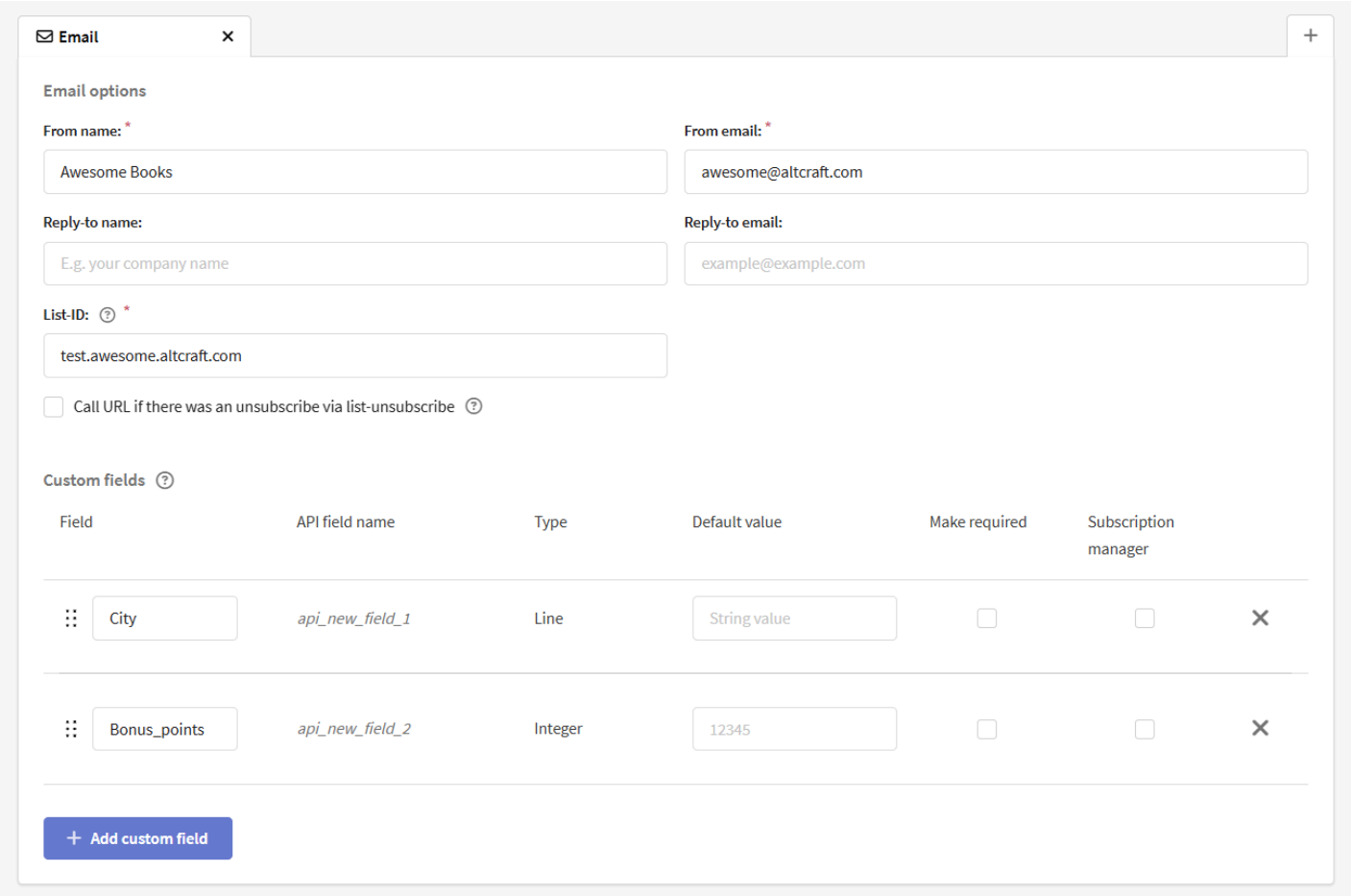
You can add custom fields to any channel resource. They are similar to additional fields in the profile database. Use them to store extra information related to the subscription to the channel.

To create a field, click the **Add custom field** button and select its type:

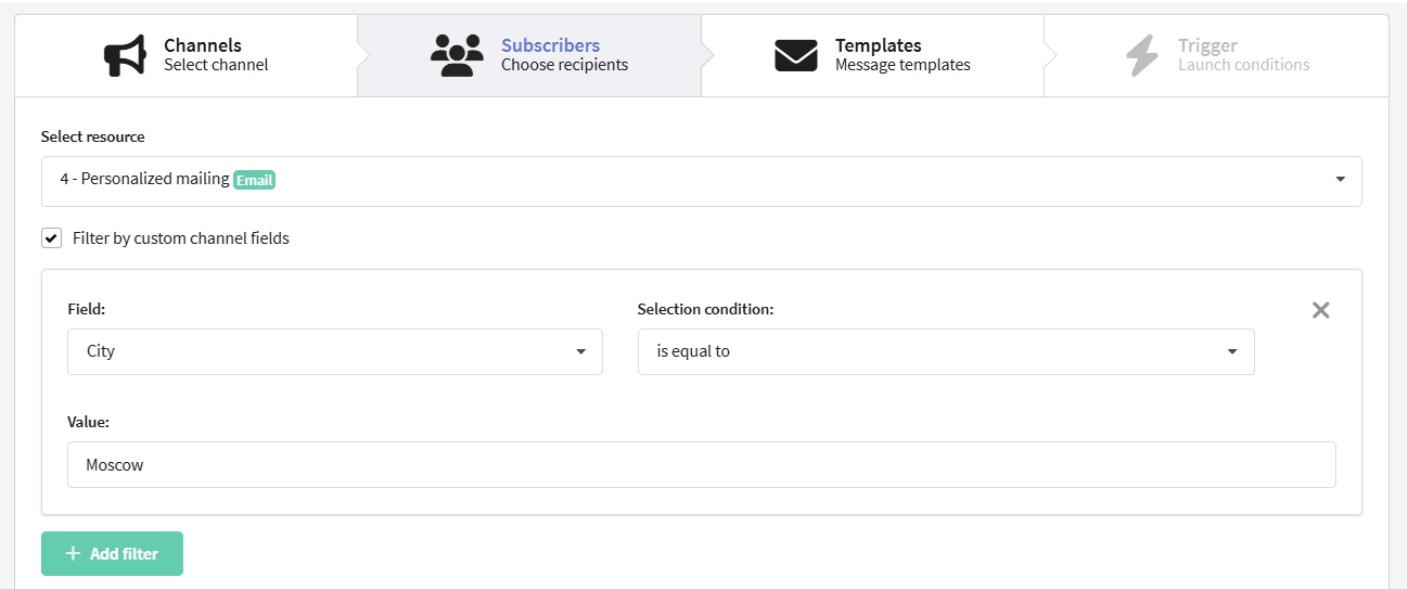
- **Integer** — positive and negative whole numbers.
- **Float** — positive and negative floating-point numbers.
- **Line** — text in any language, including numbers, special characters, and spaces. It uses the Unicode standard.
- **Boolean** — a field that provides a choice between two options: True or False (Yes or No).



Next, specify the **field name** and the **name** to be used in **API requests**. The field name can include any characters, while the API name should only contain Latin characters, numbers, and underscores. It also must begin with a letter.



After creating custom fields you can apply them to your trigger mailing. This will make it more personalized.



Additional settings:

- **Default value** — used when no data for this field is provided during profile import.
- **Make required** — data for this field will be required during import.
- **Subscription manager** — the field will be added on the Subscription Manager page.

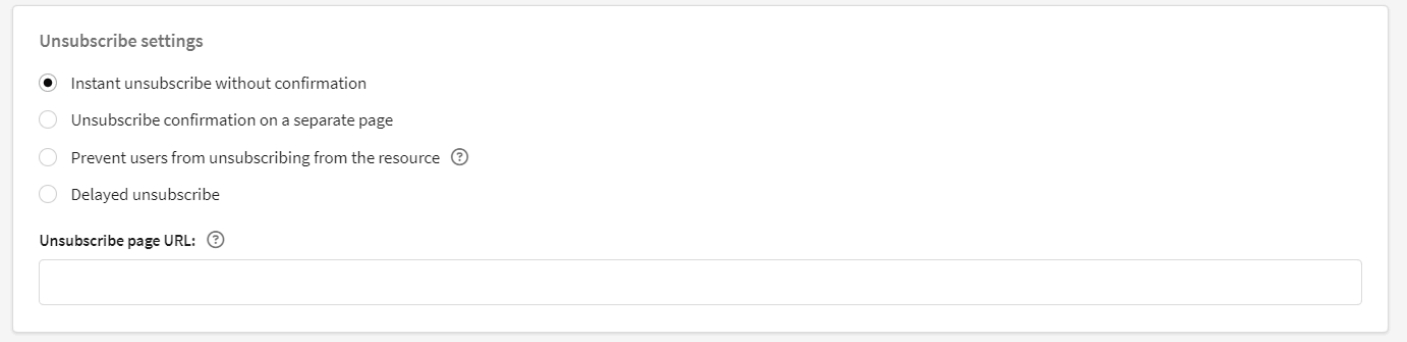
To delete a field, click the cross. The number of additional fields is not limited. After saving the resource, you can edit the field name and its additional settings, but changing the API name is not possible.

Unsubscribe settings

You can set up a separate unsubscribe mechanism for each resource. The unsubscribe page URL specified in the resource will override the system unsubscribe page. However, if a custom unsubscribe link is specified in the email template, then it will take priority. Learn more about user links.

Instant unsubscribe without confirmation

When clicking on the unsubscribe link, the profile goes to the unsubscribe system page and its status immediately changes to **Unsubscribed**. This type of unsubscription is used by default.



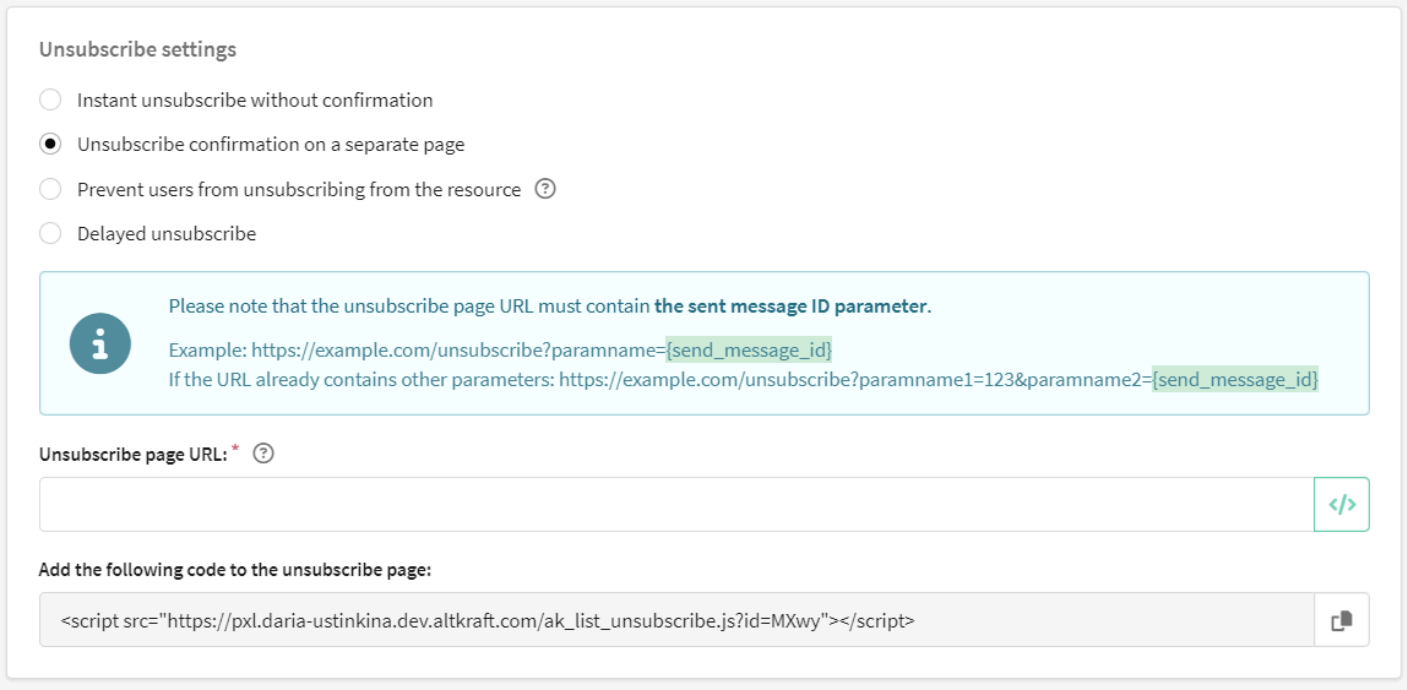
You can redirect the profile to your own unsubscribe page by specifying a different URL in the **Unsubscribe page URL** field.

Unsubscribe confirmation on a separate page

CAUTION

This unsubscribe setting is only available for users of the On-Premise solution. To activate these settings, add the `unconditional_subscriber` parameter to the `EXTRAS` property in the 'main' configuration file.json`.

When you click on the unsubscribe link in the message, the profile will go to the user's unsubscribe page. Its status will change to **Unsubscribed** only after he confirms the unsubscription. Such a page can be used to clarify the reason for unsubscribing.



Setting up fields:

Specify **Unsubscribe page URL** — this URL will override the URL of the system unsubscribe page.

In the URL of the unsubscribe page, add the parameter **ID of the sent message** `{send_message_id}`. This is a required parameter that will allow you to identify the resource and profile for which you want to unsubscribe.

Select the unsubscribe mechanism:

Unsubscribe mechanism using the Altcraft SDK

CAUTION

This mechanism is implemented **on the client side** (in the browser). It is safe to use on the unsubscribe page accessible to the user. All operations are performed via the SDK, without exposing tokens or other sensitive data.

Paste the code provided in the resource to your unsubscribe page to connect the Altcraft SDK. Next, initialize the `AkUnsub` class.

Example:

```
<!-- Connecting the Altcraft SDK -->
 <script src="https://pxl.altkraft.com/ak_list_unsubscribe.js?id=MXw3Mw=="></script>

<!-- Initializing the AkUnsub class -->
<script>
   const akUnsub = new AkUnsub();
</script>
```

INFO

If necessary, you can redefine the resource token and pixel domain in the script:

```
new AkUnsub({
    rtoken: 'example_token',
    pxl_domain: 'your_pxl_domain',
 })
```

The AkUnsub class has 2 public methods:

- `confirmUnsub` — method for confirming unsubscription
- `cancelUnsub` — method for canceling unsubscription. Used only **for delayed unsubscribe**, more on this in the next paragraph of the article.

Each of the methods accepts the following **arguments:**

- **SMID** is the identifier of the sent message (required parameter).
- **An array of categories** of the resource from which unsubscribing occurs (optional parameter).

Example:

```
AkUnsub.confirmUnsub('SMID', ['cat_to_unsub'])
AkUnsub.cancelUnsub ('SMID', ['cat_to_unsub'])
```

To call methods, you can add buttons to the unsubscribe page. Example:

```
<!-- Unsubscribe confirmation button -->
<button onclick="akUnsub.confirmUnsub('SMID',['category_1', 'category_2'])">Confirm unsubscribe</button>

<!-- Unsubscribe button -->
<button onclick="akUnsub.cancelUnsub('SMID',['category_1', 'category_2'])">Cancel unsubscribe</button>
```

Replace `SMID` and `['category1', 'category2']` with the corresponding values.

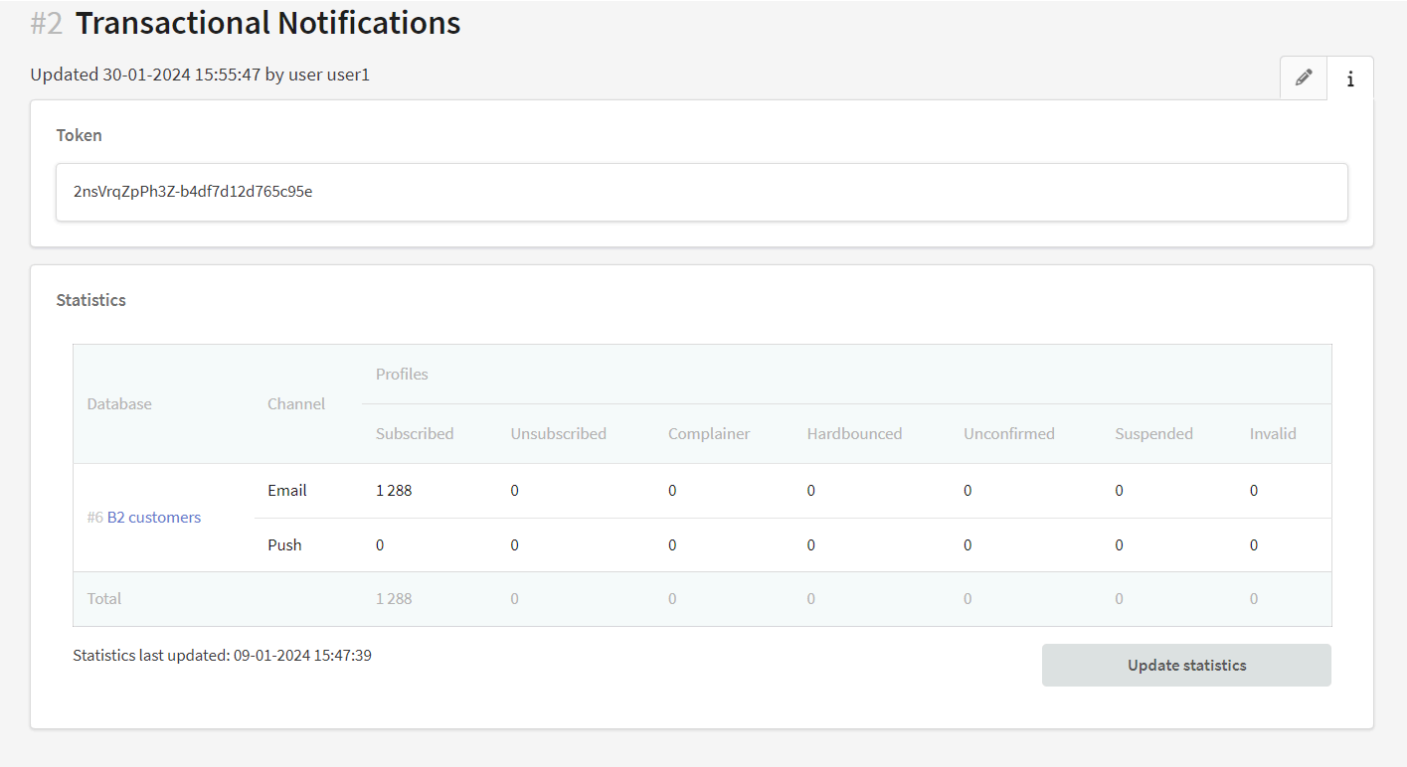
Unsubscribe mechanism without using the Altcraft SDK

CAUTION

This mechanism is implemented **on the server side**. It uses an API token and a resource token, which are confidential. They must not be included in page code or sent directly from the browser.

You can implement the unsubscribe mechanism on your side. To do this, add 2 parameters to the **Unsubscribe page URL**:

- **Send message ID** `{send_message_id}`. This is a mandatory parameter that allows you to identify the resource from which you want to unsubscribe.
- **Resource token** `{resource_token}`. This parameter is required for additional protection and to implement your own unsubscribe mechanism without using the Altcraft SDK. The resource token can be passed not in the URL of the unsubscribe page, but in the body of the API request. The token can be obtained in the resource editing window on the “i” tab — details:



If the profile has confirmed the unsubscription on your page by its actions, then it is necessary to transfer this data to the platform using the API request `/api/v1.1/profiles/message_unsub`. The status of the profile's subscription to the resource will change to "Unsubscribed". If you pass the `categories` parameter, the profile will be unsubscribed only from the specified resource categories.

Request Example:

```
{
"token": "abcdefghijklmnqrstuvwxyz", // API token
  "smid": "fGmXiWU9Y4NSIpCwl4QG6QXAAAQE", // ID of the sent message
  "rtoken":"2fHLS9YR9daS-a90b371eecc2952f", // Resource token
  "categories": ["cat_to_unsub"] // Resource categories to unsubscribe from
}
```

TIP

API request **/api/v1.1/profiles/message_unsub** is described in **documentation**.

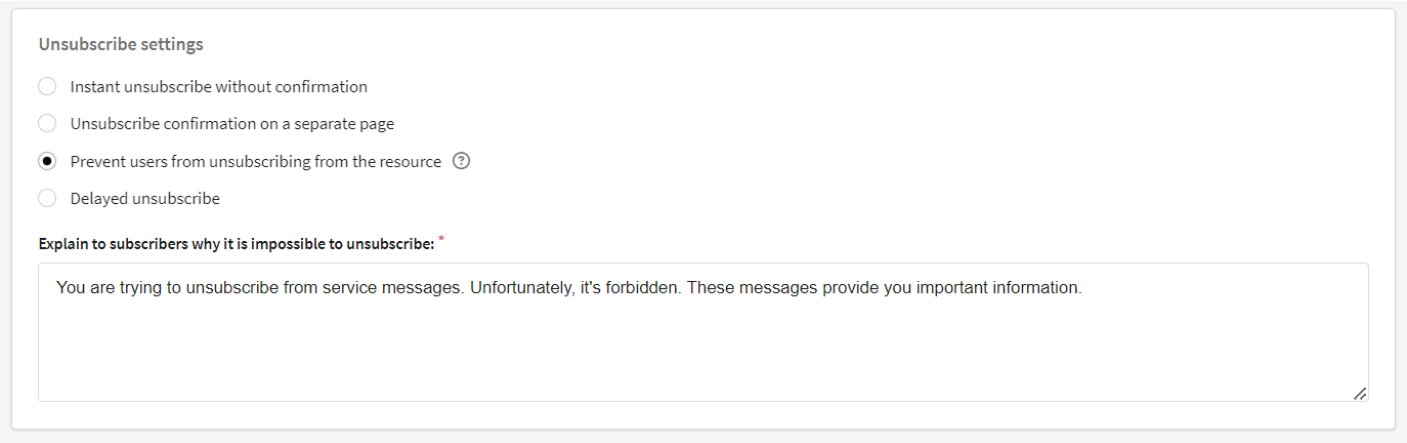
</ details>

Prevent users from unsubscribing from the resource

CAUTION

This unsubscribe setting is only available for users of the On-Premise solution. To activate these settings, add the `unconditional_subscriber` parameter to the `EXTRAS` property in the 'main' configuration file.json`.

For certain types of mailings, you can set a limit on unsubscribing. When this option is selected, all unsubscribe links specified in the template will lead to a page where the reason for not unsubscribing is displayed.



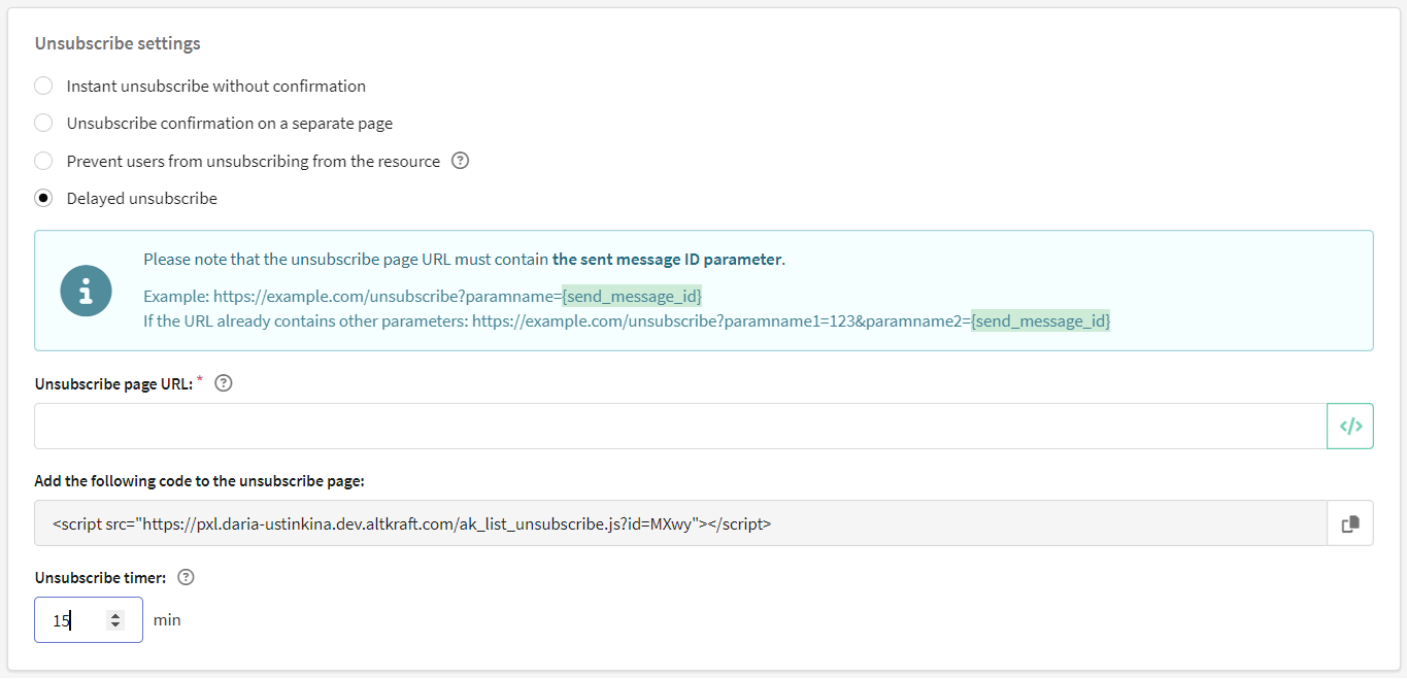
INFO

Unsubscribing via the list-unsubscribe link in an Email will also be prohibited.

Delayed unsubscribe

When you click on the unsubscribe link in the message, the profile will go to the user's unsubscribe page. Its status will change to "Unsubscribed" if:

- the profile will confirm the unsubscription by its actions;
- The profile will not cancel the unsubscribe during the time specified in the Timer field (the maximum timer value is 30 minutes).



Setting up fields:

Specify **Unsubscribe page URL** — this URL will override the URL of the system unsubscribe page.

In the URL of the unsubscribe page, add the parameter **ID of the sent message** `{send_message_id}`. This is a required parameter that will allow you to identify the resource and profile for which you want to unsubscribe.

Select the unsubscribe mechanism:

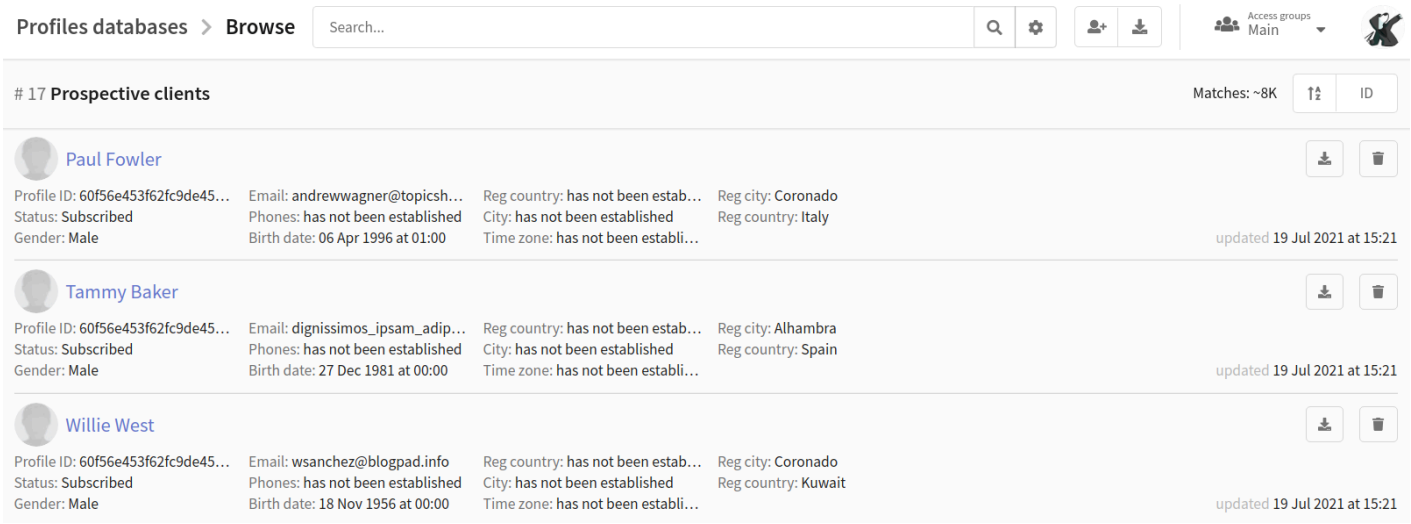
▶ Unsubscribe mechanism using the Altcraft SDK

Managing databases

Description

Modern people-based marketing techniques use real time behavior tracking and customers personal information.

Altcraft database management provides you with powerful and flexible tools to collect and make use of customer profiles data.



Flexibility and usefulness of database depends on how it is created, filled, and maintained. If data is correctly collected and organized, then it becomes possible to segment subscribers depending on certain attribute to separate groups and send targeted content to each group.

This process is called Segmentation, and for this we have certain element - segment, which has many intuitive settings.

Think about your marketing for a minute. What goals do you have, who is your audience and what information is interesting for them.
For example:

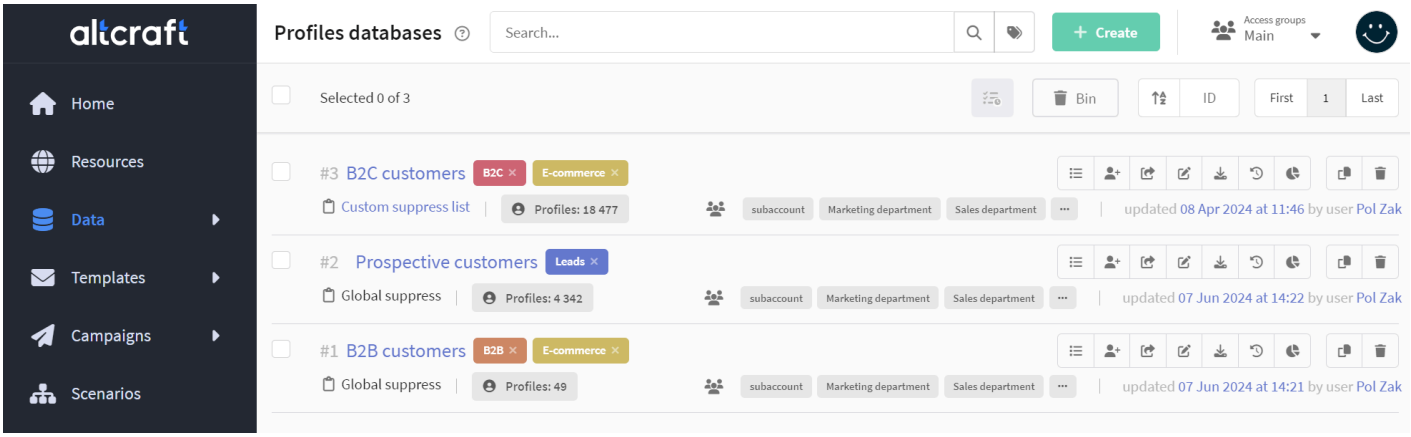
- Do you have your seasonal customers? Do they prefer certain product types?
- Is it necessary to send different content templates depending on customers occupation or hobbies;
- What other parameters affect your customers behaviour?

It can be convenient to figure out what customer information is necessary for you, and write it. You can use it to create custom fields for storing information about your customers in the database.

Thoughtful database setup will bring skyrocketing impact, especially in the long run.

Managing a database

To manage databases, select **Data —> Profiles databases** in the main menu. To create a new database, click **+ Create** in the top panel. To edit an existing database, click on its name.



The following functions are available in the context menu:

— view profiles

— import customer profiles

— copy profilesto another database

— update multiple profiles in the database. Using this instrument, you canupdate profiles, subscribe them to resources, attach or detach promo codes.

— exportcustomer profiles into a file. Beforeexport, you can specify data you need to download.

— export profile history. Download data on profile activity.

— create a new empty database with the same structure. Custom fields will be copied into a new database.

— delete database altogether with profiles.

TIP

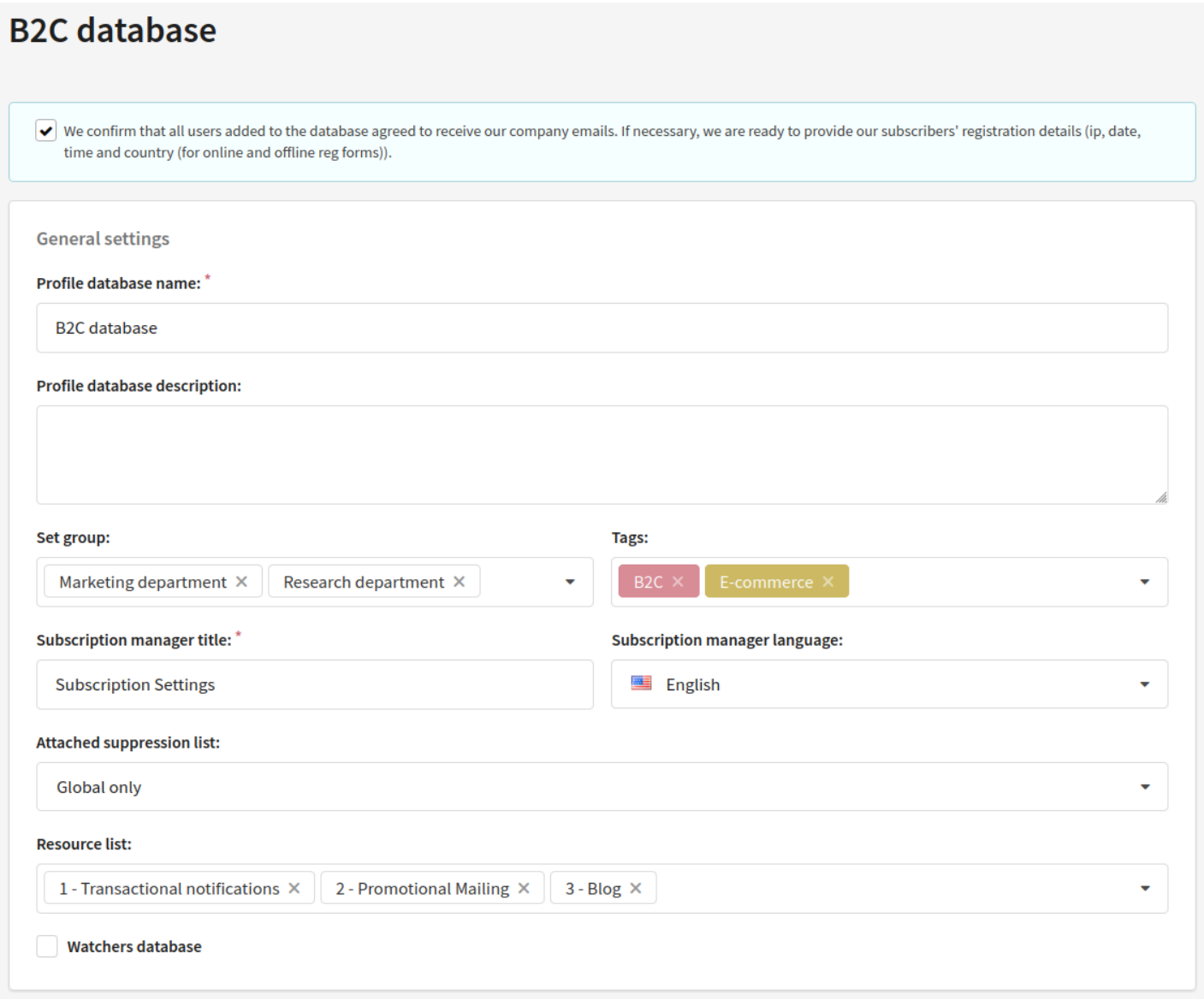
You can delete a database only if it does not contain any profiles that have had interactions within the account. If there have been interactions, such as sending mailings to these profiles, you must first delete those profiles from the database.

Creating a profiles database

General settings

The following settings can be configured for a database:

- **Database name:** name displayed in platform database lists.
- **Description:** database purpose description.
- **Set group:** database access group.
- **Tags:** custom tags that can be used for quick search and analytics.
- **Subscription manager title:** title displayed only in subscription manager.
- **Subscription manager language:** subscription manager page language for database profiles.
- **Attached suppression list:** a custom suppression list for this database.
- **Resources:** enable communication with database profiles through various channels within a specific topic. Instructions on how to configure the resource are provided in this article.
- **Watchers database:** select this option for a database of observers. They will receive messages when a mailing starts together with your customers. Databases with this option can be selected in Smart mailing settings.



Default fields

Standard profile fields are designed to keep the most frequently used customer information.

Default — this value is used when field value is unknown after profile import.

Make required — this field will be required during import. If you enter a Default value, profile import without this field value will be possible.

Subscription manager — this field value will be shown in subscription manager page.

Default fields

Field	Type	Default	Make required	Subscription manager
Email	Line		☑	☑
Phones	Phones list		■	☑
First name	Line		☐	☐
Last name	Line		☐	☐
Birth date	Date	%Y-%m-%d	☐	☐
Gender	Enum	Not set	☐	☐
Reg date	Date	%Y-%m-%d	☐	☐
Reg IP	IP	127.0.0.1	☐	☐
Reg city	Line		☐	☐
Reg country	Line	Not set	☐	☐
Reg URL	Line		☐	☐
IP	IP	127.0.0.1	☐	☐
City	Line		☐	☐
Country	Line	Not set	☐	☐
Region	Line		☐	☐
ZIP code	Line		☐	☐

INFO

Fields with customer geodata are divided into two types:

- The first (“reg.” In the field name) is the information received during registration: based on the IP address or static import data.
- The second is the latest data received from the client’s actions history: also based on IP address or transmitted data.

Custom fields

Custom fields are created by users of the Altcraft system to store specific for business customer information. You can create unlimited custom fields. They can contain 10 different data types:

- **Number** — positive and negative integer numbers.
- **Float** — floating-point number: it has a decimal place.
- **Line** — text in any language, including numbers, special characters, spaces. Unicode standard is used.
- **Date** — date in RFC 3339 format, like `2019-08-22T00:00:00.000Z`.
- **Boolean** — this field provides a choice of two options: `True` or `False` (Yes or No). It is used to define if customer has certain attribute, like VIP status.
- **Enum** — this field is similar to Boolean, but allows you to choose one option from unlimited number of options created by user. For example, customer tariff plan. Options are written in a comma separated line.
- **IP** — IP in IPv4 format. For example, `127.0.0.1`.

- **Tags** — in this field, several values can be written simultaneously. They may be products or services purchased by customer, or other values, composition of which may vary. Tags are written in line, separated by `"," "|" or ";"`. Therefore, these characters can not be used in tag itself. For example: `Product-1`, `Product 2`, `Product_3`.
- **Object** is a field for storing undefined data: a set of flags or complex data structures.
- **Array of objects** is a field for storing a subtable, which, for instance, may contain a list of managers, contracts, etc.

INFO: Validation in objects and arrays

1. **Naming rules.** In object field names, only Latin letters (in any case), digits from 0 to 9, the underscore `_`, and the hyphen `-` are allowed.

2. **Data type.** You can store data of `int`, `float`, `string`, `boolean` types in the “Object array” or “Object” fields.

Example

```
[
    {
        "key1": 12.3,
        "key2": 12,
        "key3": true,
        "key4": "text"
    }
]
```

3. **Object structure.** If you use such types as “Array of objects” or “Object”, you can only store peer objects or peer objects that are nested within other objects. If you use objects with a more complex structure, connect the SQL database and create queries to select profiles.

Details

Valid values:

```
[
        {"x":1},
        {"y":2}
]

//OR

{
  "key_1":
    {
      "sub_key_1":
        {"x":1,"y":2}
    },
  "key_2":
    {
      "sub_key_1":
        {"x":5,"y":3}
    }
}

//OR

[
    {"key": "value_1", "new_key": "new_value1"},
    {"key": "value_2", "new_key": "new_value2"}
]
```

Invalid values:

```
[
        ["x","y"],
        ["z","w"]
]

//OR

{
  "key":
    [
      ["x","y"],
      ["z","w"]
    ]
}

//OR

[
    {"key1": "value_1", "new_key10": "new_value1"},
    {"key7": "value_2", "new_key300": "new_value2"}
]
```

Custom fields

Field	Type	Default value	Make required	Subscription manager	
Custom_Integer	Integer	123	☑	☐	✖
Custom_Float	Float	123.45	☐	☑	✖
Custom_String	Line	test string	☐	☐	✖
Custom_ISO_date	Date	2022-01-10	☐	☐	✖
Custom_Boolean	Boolean	True	☐	☐	✖
Custom_List	Enum	Option 1, Option 2, Option 3	☐	☐	✖
Custom_IP	IP	127.0.0.1	☐	☐	✖
Custom_Tags	Tags	Tag 1, Tag 2, Tag 3	☐	☐	✖
Custom_Object	Object	{"X":1,"Y":2,"Z":3}	☐	☐	✖
Custom_Array	Objects array	[{"key":"value_1"},{"key":"value_2"}]	☐	☐	✖

+ Add custom field

TIP

You can find detailed information about the types of fields in the database and the allowed values in **this article**.

Email settings

These fields are required for creating a new database:

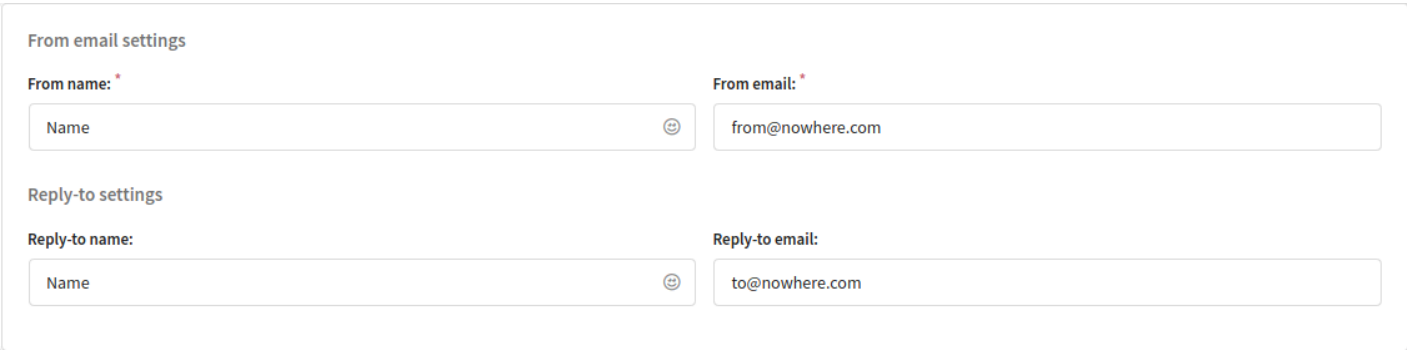
- From name

```
  <label>From email</label>
</strong>
```

And these you will need to gather replies:

```
  <label>Reply-to name</label>
</strong>
```

```
  <label>Reply-to email</label>
</strong>
```

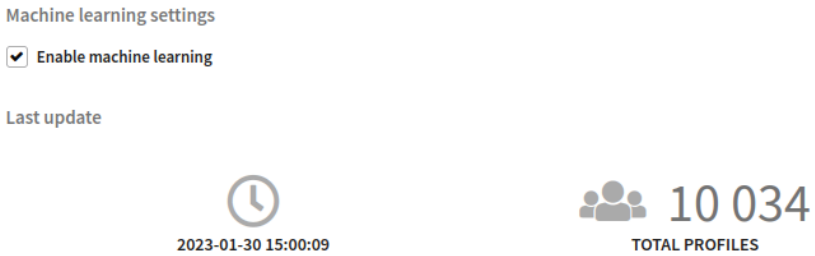


Machine learning settings

Enabling machine learning in the database allows you to determine the best time to send a message to a profile. This data can then be used to segment profiles and set up scenarios.

To get started with this module, you need to add fields to the platform configuration file.

After saving the settings, you will see a window with information about when the recalculation of profile data in the machine learning database is scheduled. The first best time data will be displayed after the first calculation.



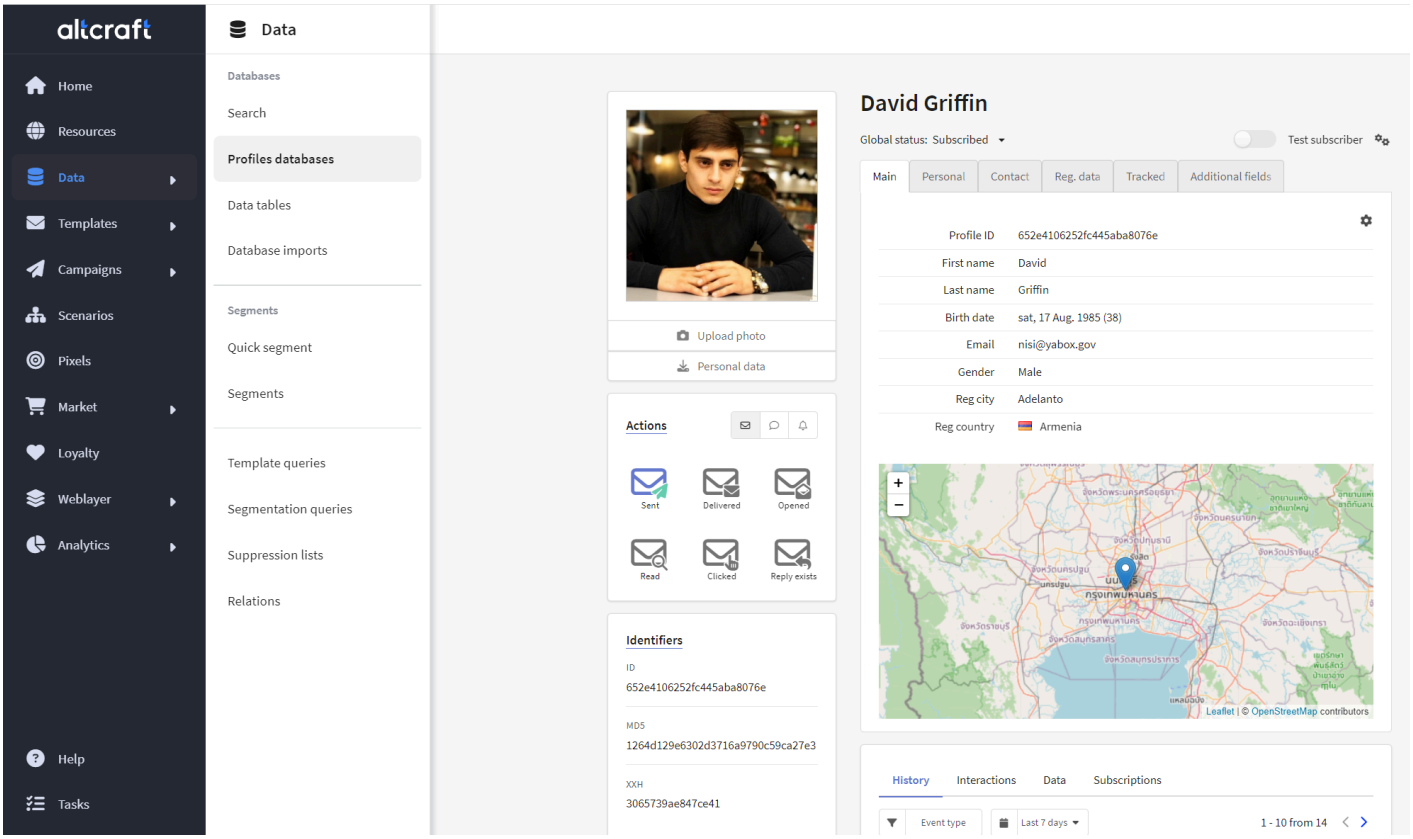
INFO

The schedule for updating machine learning data is set in the configuration file in the `ML_REFRESH_TIME_HOURS` (UTC +0) field. The default is at 00:00.

Subscriber profile

Description

Information about mailing recipients — customers, partners, mailing watchers — is stored in profiles organized in databases. Profile database determines the structure of general and custom data fields of each profile. Resources make communication possible.

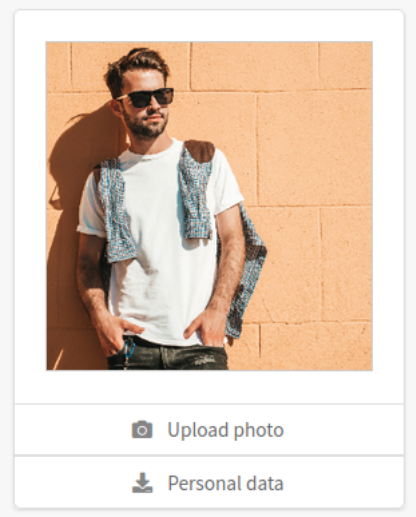


Subscriber profile combines several data types. Static data is displayed on the main page and is organized into several tabs. Other information is displayed below and contains:

- **History — events, connected to this customer: mailing and browsing action history.**
- **Interactions — profile participation in static segments and automation scenarios; you can manually send a profile to a scenario, add to a segment or run a trigger mailing.**
- **Data — information about promo codes, orders and relations of the profile, as well as a list of forms he or she filled out.**
- **Subscriptions — profile's subscriptions to various Resources.**

Avatar and data export

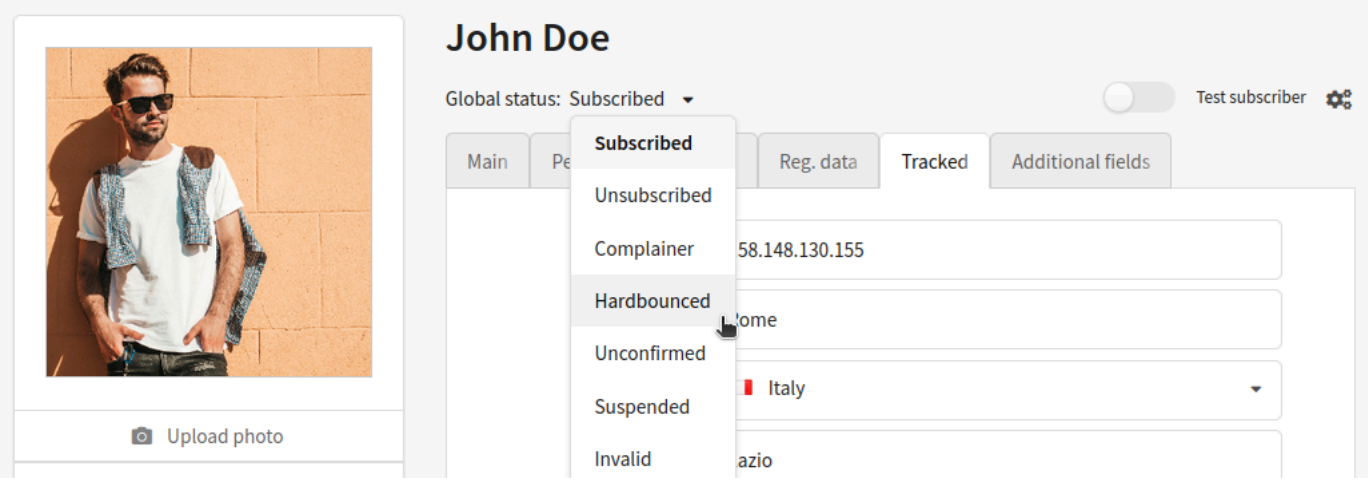
The top left side of the profile page contents your client's avatar and buttons that help you change it manually or export profile's personal data.



Global profile status

Global profile status is defined by the ability to communicate with customers through any communication channel. Individual subscription statuses depend on global status. If the profile has several active subscriptions, but its global status does not allow communication, a message will not be sent. 7 subscription statuses are available:

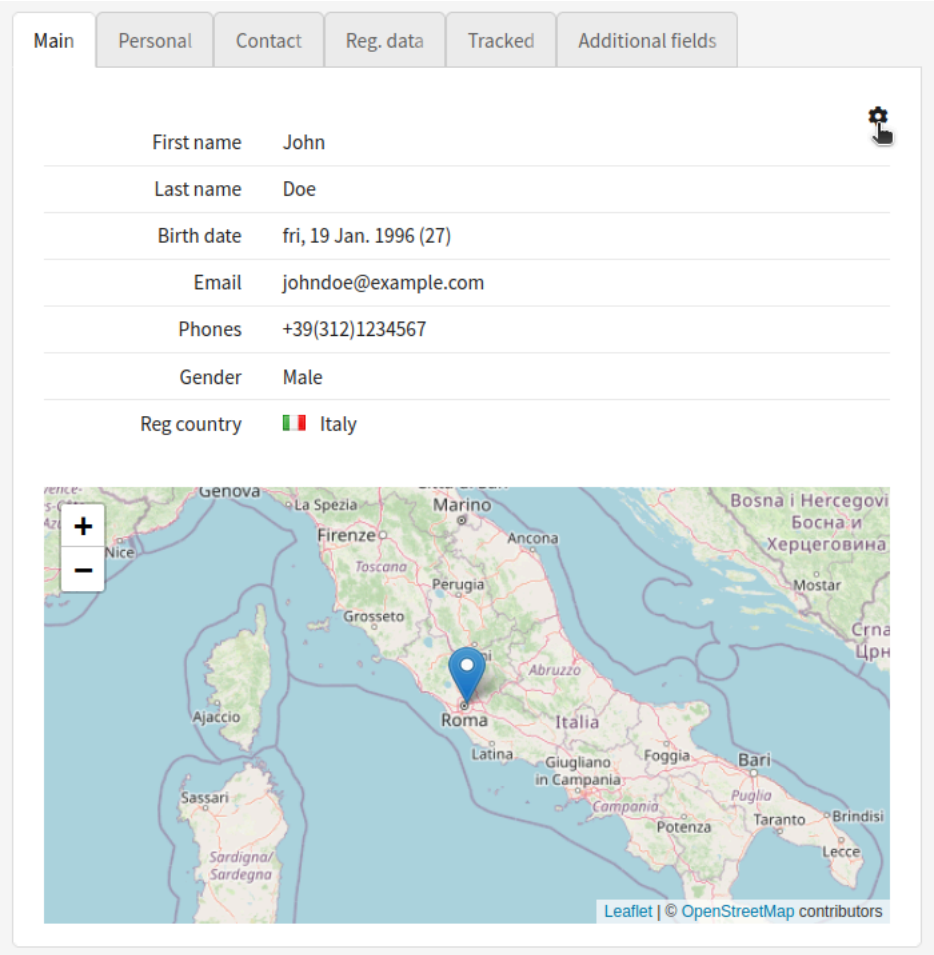
- **Subsribed — available for communication by all channels**
- **Unsubscribed — unsubscribed from receiving messages by all channels**
- **Complainer — reported to Spam. Communication is stopped.**
- **Hard bounced — the provider completely refused to accept messages to this address. Most likely mailbox does not exist and attempts to send something will decrease your reputation. Invalid addresses are recorded in the global suppression list.**
- **Unconfirmed — subscriber registered but did not confirm his consent to receive messages. It is used with Double opt-in subscription. After confirmation, the status is changed to Subscribed.**
- **Suspended — subscription is suspended for some time.**
- **Invalid — excluded for marketing mailings due to other reasons.**



Profile data

- **Main — general information and location pin. Pressing a cog button will allow you to add or remove fields to customize tab appearance.**
- **Personal — customer's demographic data.**
- **Contact — customer's basic contacts.**
- **Reg. data — opt-in information.**
- **Tracked — IP and location of the latest transaction.**
- **Additional fields — custom database fields containing business information.**

Email and Phones fields in the Contact tab are used to identify profiles in the database. Marketing mailings use resources.



TIP

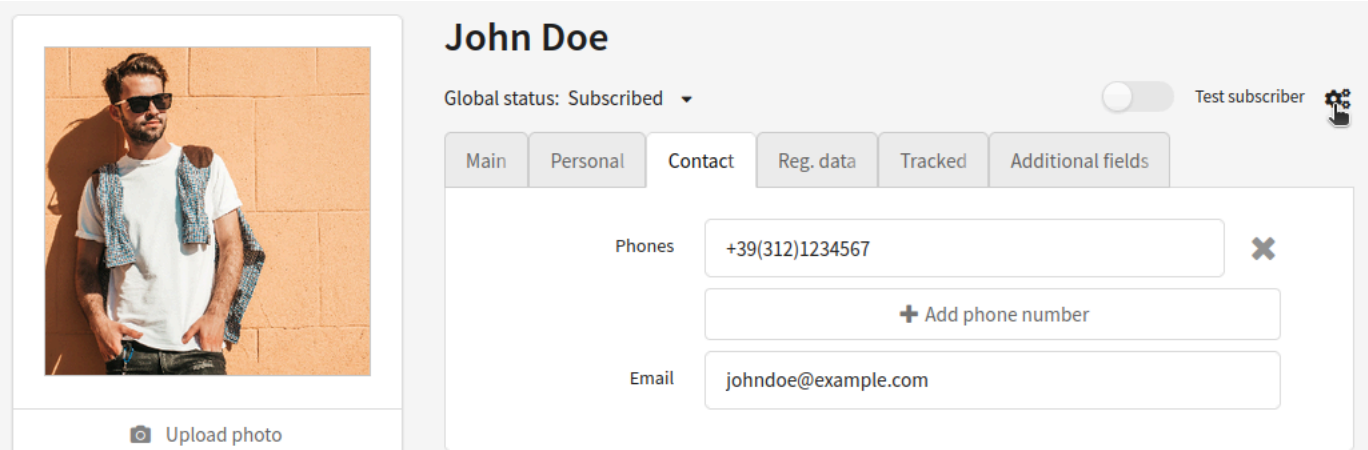
Subscriber location data can be set manually. Two types of default fields are used for this:

- **Latest action data — Country, Region, ZIP code, Timezone**
- **Registration data — Reg Country, Reg City**

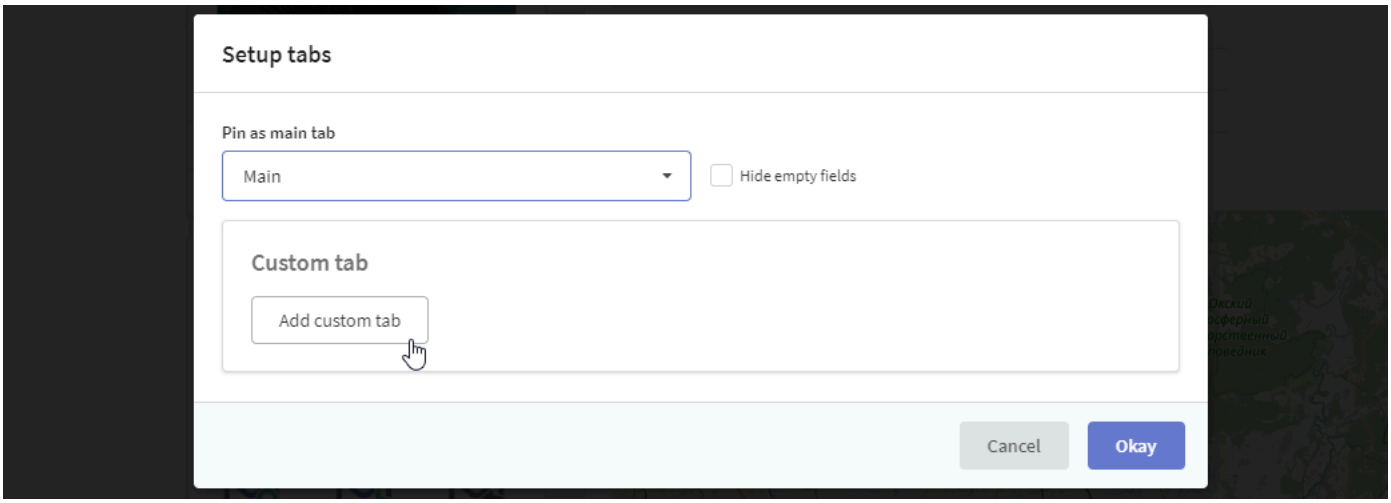
Location in Altcraft Marketing is identified automatically using the subscriber's IP address. This happens when customers open letters, click links or perform actions on a website. Geographical coordinates are displayed on the built-in map widget.

Automatic location is also available during profile import. For registration data, Reg IP field is used. Auto-detection option must be activated beforeimport from file. For API import, this option is enabled by default.

Whenever you open a customer profile - the main tab will be displayed by default. To customize it press the cogs button on the right:



Select a tab to be displayed by default - or create your custom tab and customize data to be shown:



INFO

Custom data fields are created to store business-specific customer information. Thesefields may contain 8 types of data:

- **Number — positive and negative integer numbers.**
- **Float — floating-point number: it has a decimal place.**
- **Line — text in any language, including numbers, special characters, spaces.**
- **Date — date in RFC 3339 format, like `2019-08-22T00: 00: 00.000Z`.**
- **Boolean — thisfield provides a choice of two options: True or False (Yes or No). It is used to define if the customer has a certain attribute, like VIP status.**
- **Enum — this field is similar to Boolean, but allows you to choose one option from the unlimited number of options created by the user. For example, customer tariff plan. Options are written in a comma-separated line.**
- **IP — IP in IPv4 format. For example, `127.0.0.1`.**
- **Tags — in thisfield, several values can be written simultaneously. They may be products or services purchased by a customer, or other values, composition of which may vary. Tags are written inline, separated by `","` `"|"` or `";"`. Therefore, these characters can not be used in the tag itself. For example `Product-1`, `Product 2`, `Product_3`.**

Custom fields are configured in the profiles database structure.

You can find detailed information about the types of fields in the database and the allowed values in this article.

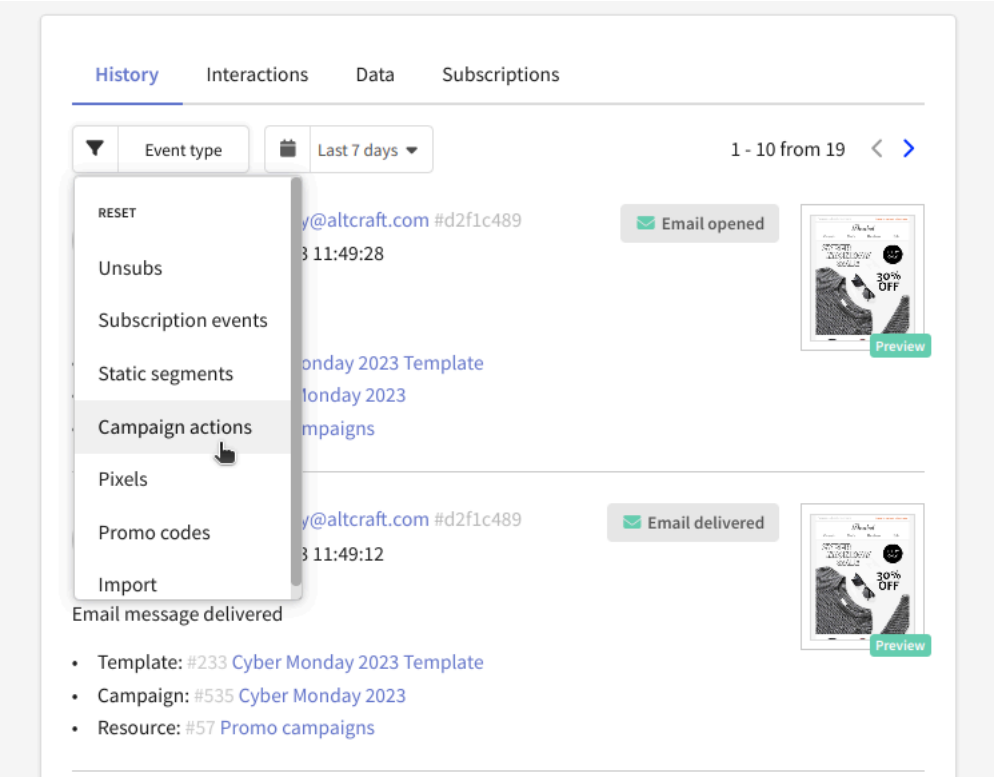
History

Profile's action history collects all the events connected with the profile. It can be used in behavioral segmentation:

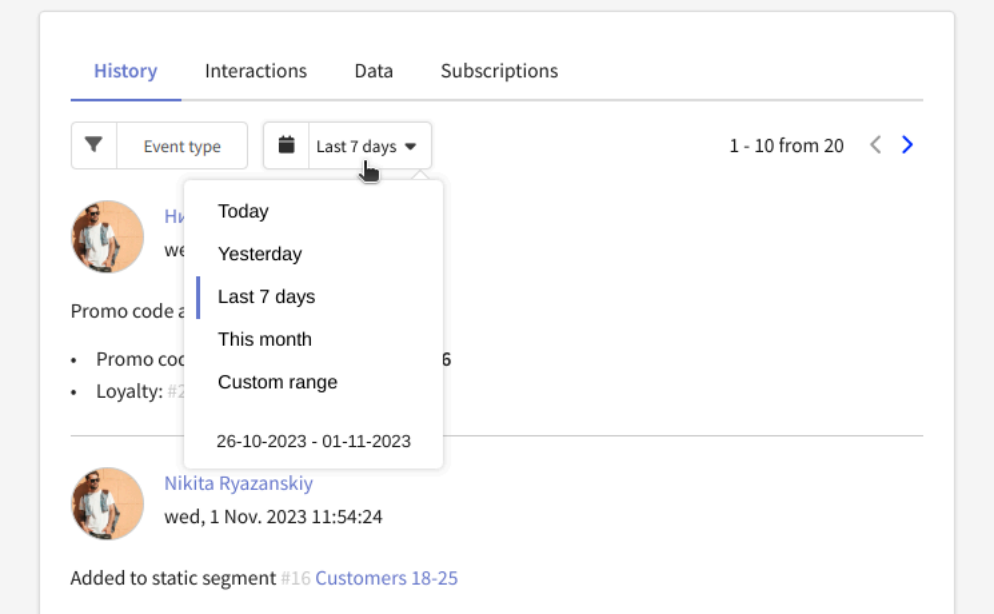
- **Import. How the profile was added to the database (manually, via API import,**
- **Subscription events. The profile was subscribed to a specific channel of the resource (Email / SMS / Push / Telegram / WhatsApp* / Notify / MAX / custom channel), or the profile was unsubscribed from the channel in the resource by changing the subscription status to "Unsubscribed".**
- **Mailing actions. Includes key events tracked in mailings (Sends, Deliveries, Opens, Reads, Clicks, Confirmation Link Clicks), as well as change of subscription status to "Hardbounce" or "Complainer".**
- **Unsubs. Unsubscribe events that are fixed when a profile clicks on an unsubscribe link in a message.**
- **Pixels. The profile performed an action on the site or in the mobile app that triggered the tracking pixel. The achievement of the goal and its cost are fixed.**
- **Promo codes. Includes events such as promo code attached, detached or activated.**
- **Static segments. The profile entered or exited a static segment.**

**To check a specific event type, use a filter. When you click on **

the conditions in the filter will be reset.



By default, profile activity history is displayed for the last 7 days, but you can select the period you need:



TIP

When the subscription status changes (in the interface, via API, or during import), events are registered only for the following statuses:

- **Subscribed**
- **Unsubscribed**
- **Complainer**
- **Hardbounced**

When the status changes to Subscribed, the event is registered only if the previous subscription status was Unsubscribed, Complainer, or Hardbounced.

For example, if the subscription status is set to Suspended and then changed to Subscribed, the event will not be registered.

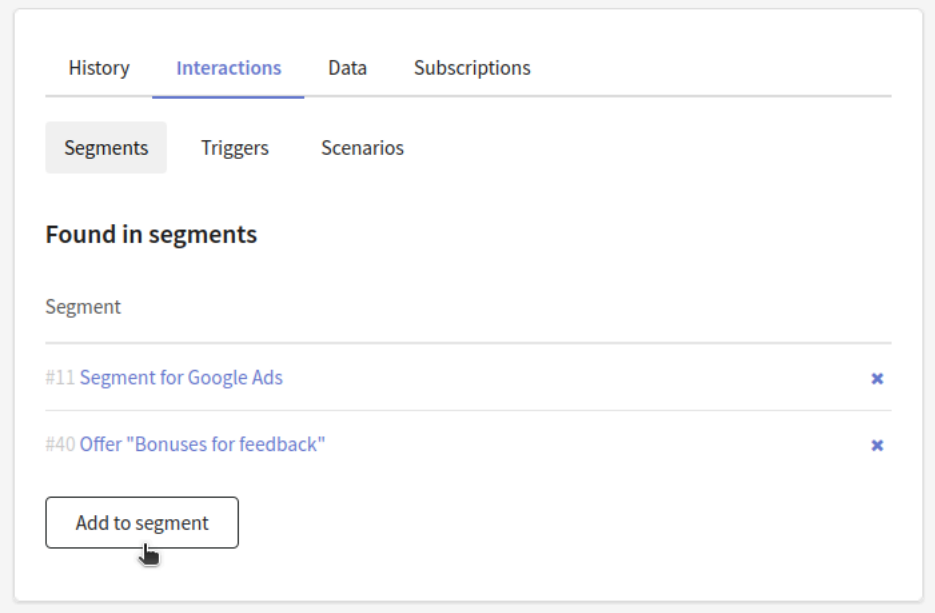
Events are not registered for the following statuses:

- **Unconfirmed**
- **Suspended**
- **Invalid**

Interactions

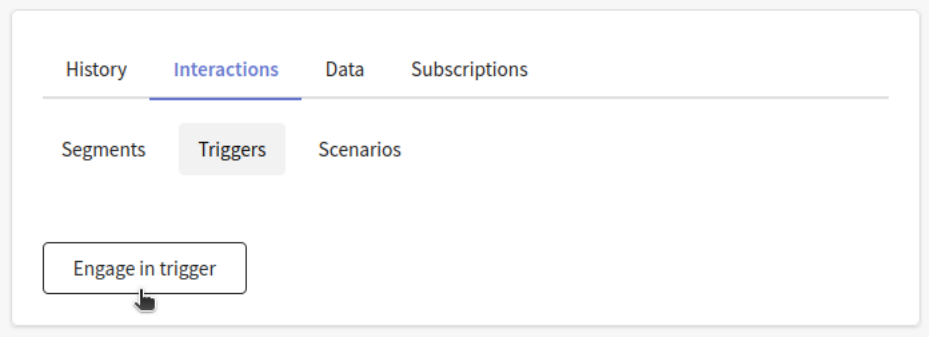
Segments

Static segments are used for advanced analytics of customer fixed groups. There is a ready-made integration to export static segments data via SQL. Here you can add the profile to a segment or remove.



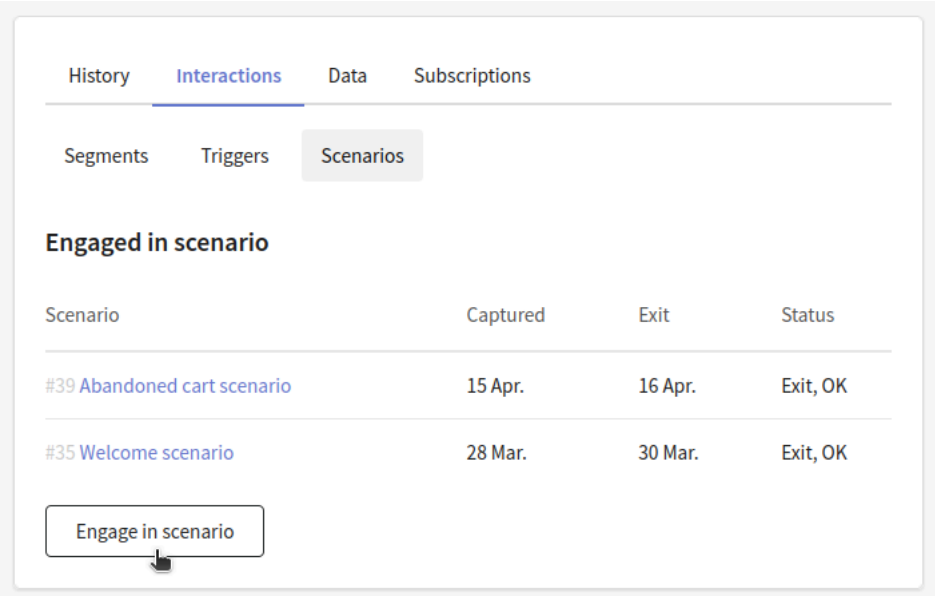
Triggers

Triggers are mailings that are launched when a certain event occurs (the profile clicks on a link, fills out a form, etc.). On this tab, you can manually send a trigger message to a profile.



Scenarios

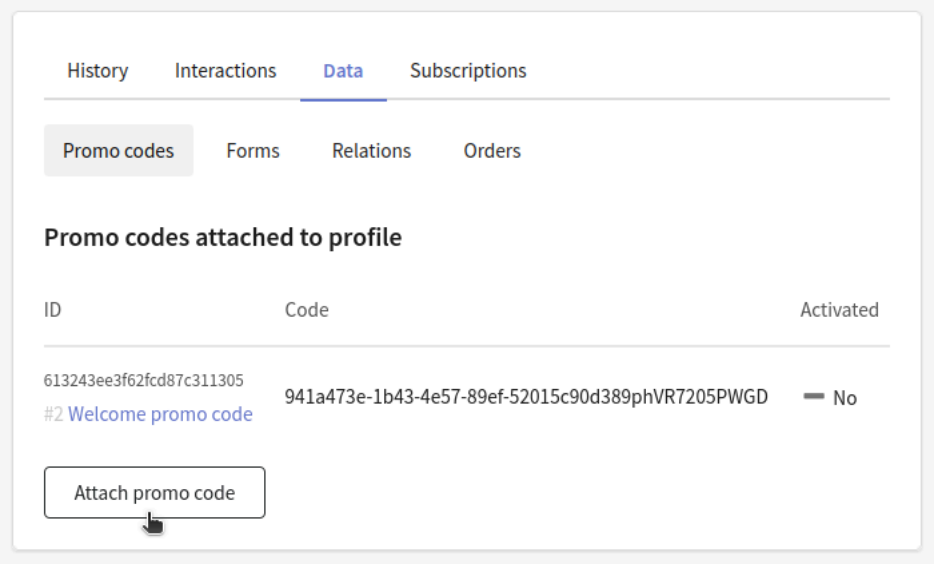
Scenarios help you automate your marketing workflows. This tab contains the history of a customer's scenario participation and results. And you can engage a customer in a scenario - manually.



Data

Promo codes

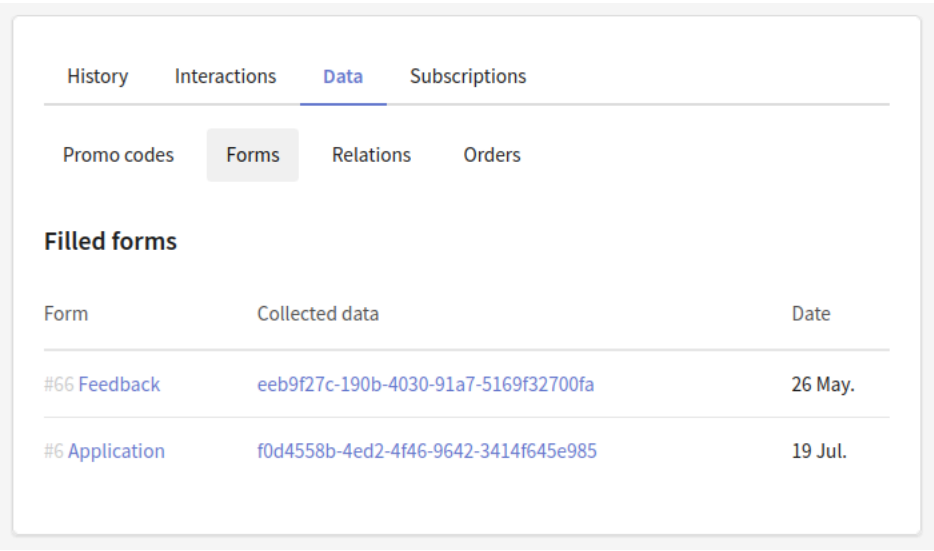
Promotion codes can encourage deals and they can be managed by Altcraft Marketing loyalty programs. On this tab all the relevant information is displayed. You can manually assign a promo code as well.



Forms

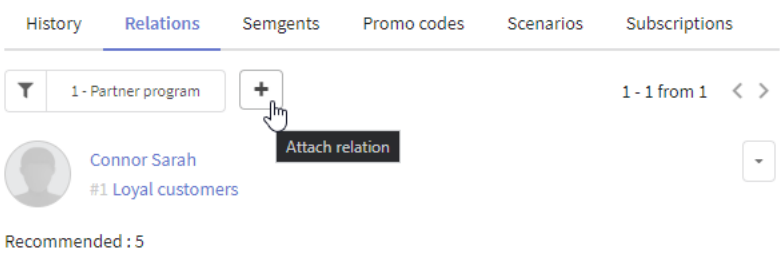
Forms are a tool for collecting data from your customers. With its help, you can implement various mechanisms using external data, for example, add new profiles to the database, manage subscriptions to various resources, and receive feedback from customers.

On this tab, you can see which forms the profile has already filled out. When you click on the collected data, you will be taken to a new window that displays all the data left by the profile in the form.



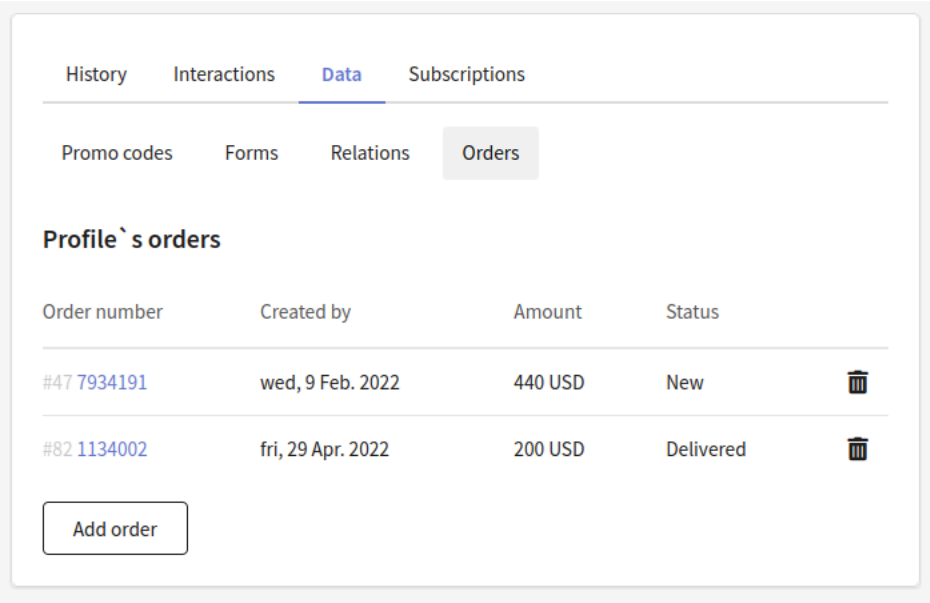
Relations

Relations between profiles can be used to register interactions between your clients — likes in a social network or service recommendations can serve as examples here. On this tab you can add a relation manually.



Orders

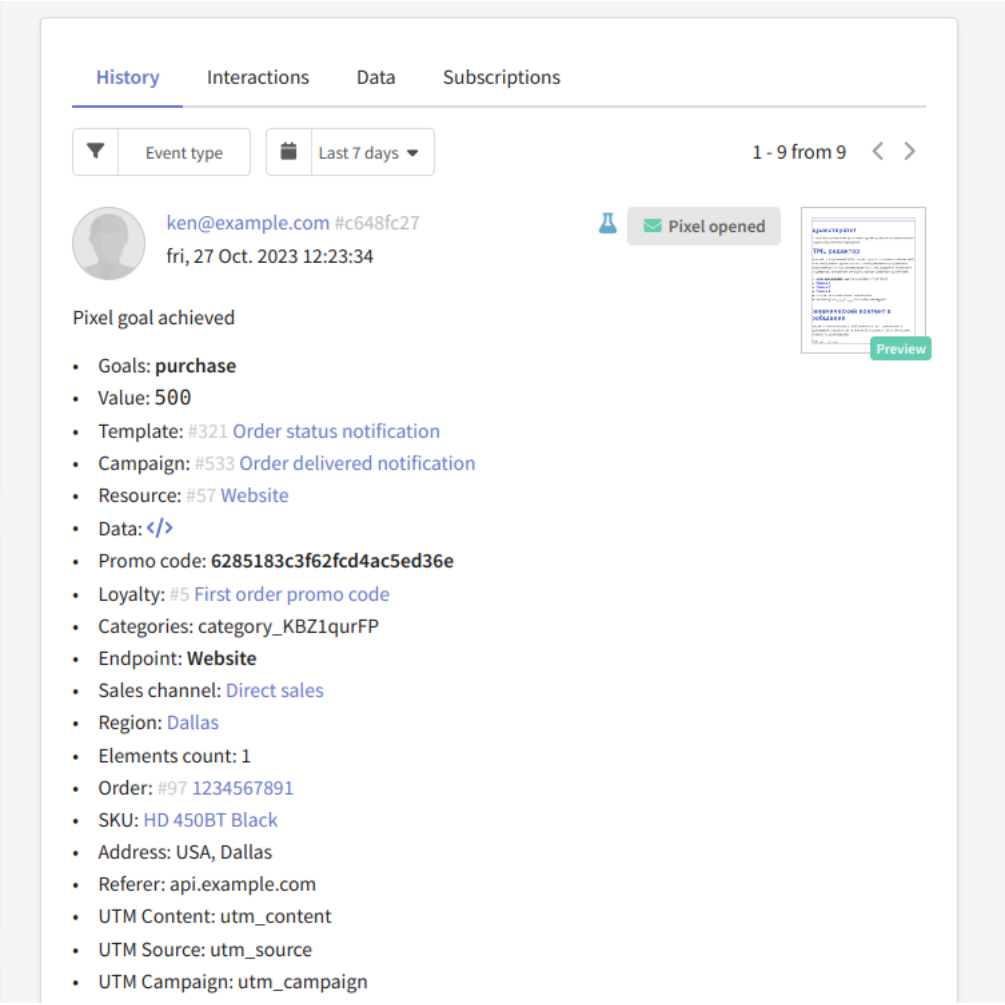
Orders are an element of the Market module. This module allows you to import product and order data to the platform and use this data to communicate with profiles. To activate this module, you must configure the appropriate access roles.



Pixels

A pixel is an HTML or JavaScript code placed on a website or in an application designed to track user actions. When a subscriber performs a specific action on the website or in the application, a request is sent to the server to record this action.

The pixel can also transmit information related to this action, such as the achieved goal, the value of the action, the URL of the website from which the subscriber came, profile geodata, information about sent mailings, and more. These data are recorded in the profile history and used for segmentation.



TIP

To view the data passed in the `pixel_data` object, press `</>` to the right of Data.

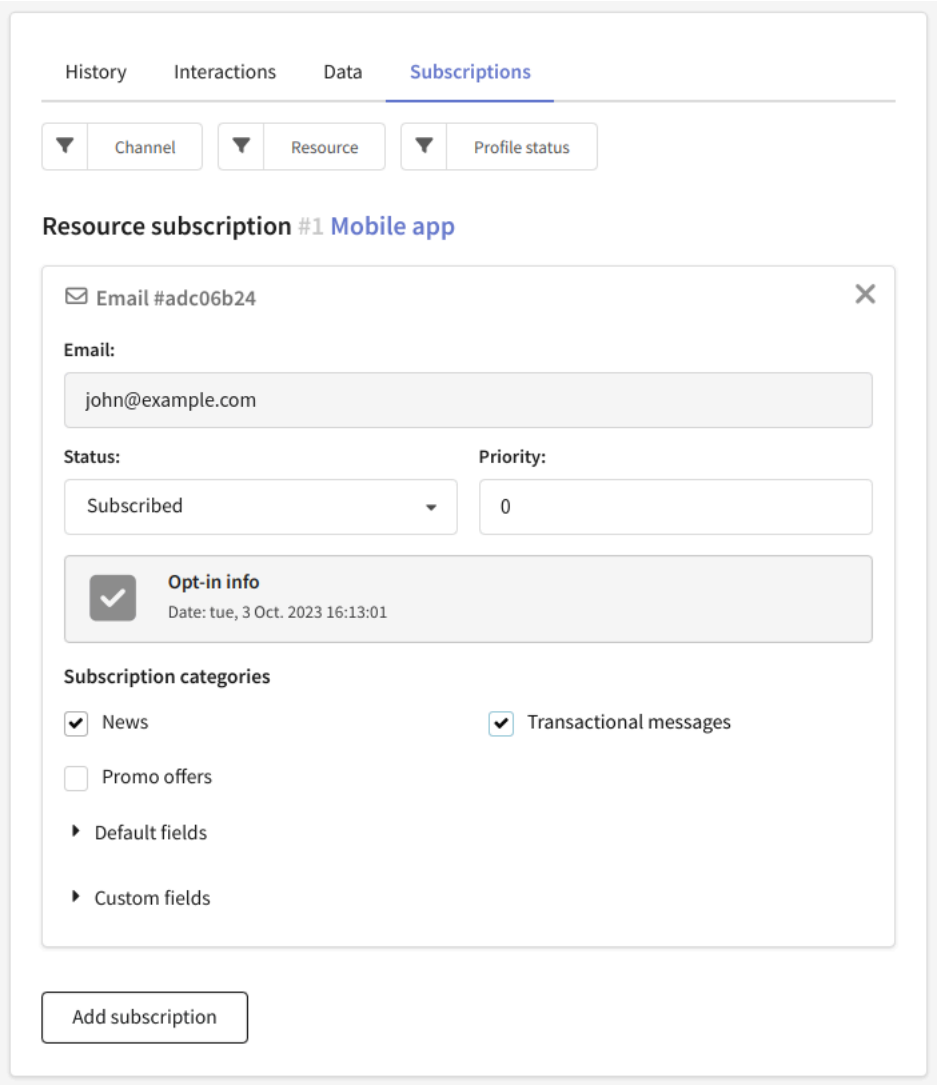
INFO

Goals can be registered for clients whose contact and personal information you do not yet have. Such events are called anonymous. Altcraft distinguishes them from each other based on connection data (cookies). If a client later registers using the same browser, their data is added to the already existing profile.

Profile subscriptions

Profile subscription is a record that combines contact data for a specific communication channel and information about registration. Every profile can have a number of subscriptions grouped by resource - the source of subscription. Every profile has a limit of 200 subscriptions or 20 subscriptions per resource in a channel. Also, different resources can use different contact information. For example, if a customer unsubscribes from one resource, they will continue to receive content from another resource.

Here you can manually change subscription status and priority:



Status

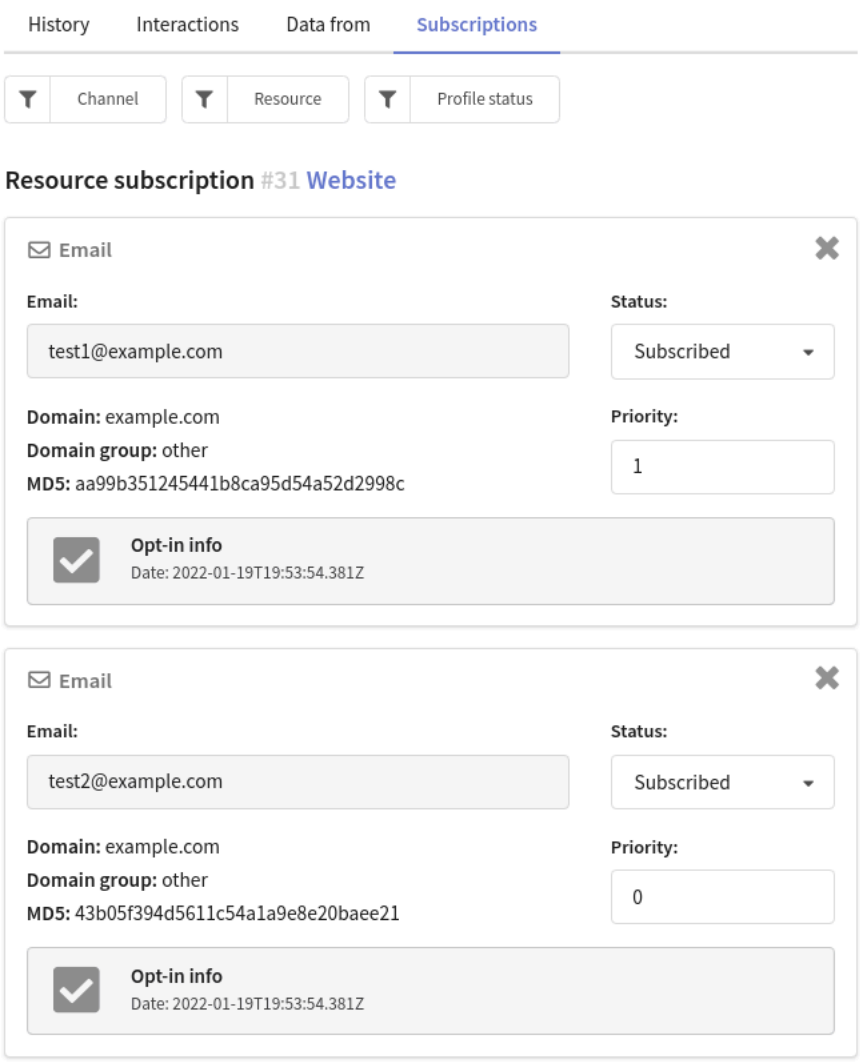
Subscription status defines communication possibility with a customer by specific contact data within a specific resource. Here, like for global profile status, there are 7 types:

- **Subscribed — available for communication by all channels**
- **Unsubscribed — unsubscribed from receiving messages by all channels**
- **Complainer — reported to Spam. Communication is stopped.**
- **Hard bounced — the provider completely refused to accept messages to this address, most likely the mailbox does not exist and attempts to send something will spoil your rating with the provider. Invalid addresses are recorded in the global stop list of the platform.**
- **Unconfirmed — the subscriber registered but did not confirm his consent to receive the newsletters. It is used with Double opt-in subscription. After confirmation, it changes to Subscribed.**
- **Suspended — subscription is suspended for some time.**
- **Invalid — excluded for marketing mailings due to any other reasons.**

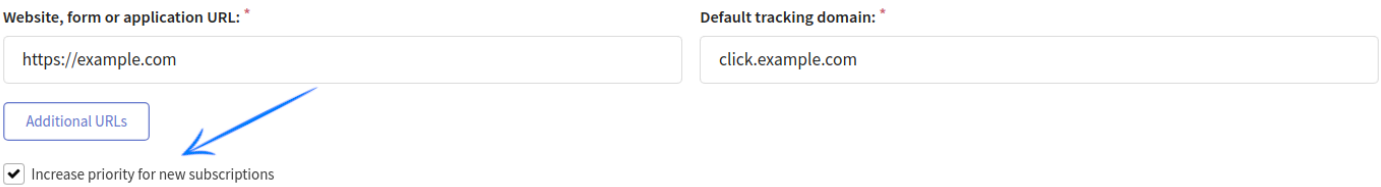
Priority

A user can subscribe to one resource several times by entering different contact details (email or phone number). The higher the value in the Priority field, the lower the priority of the subscription.

In the example below, mailings within the “Website” resource will be sent to `test2@example.com`, because a subscription with these contact details has the highest priority — 0.



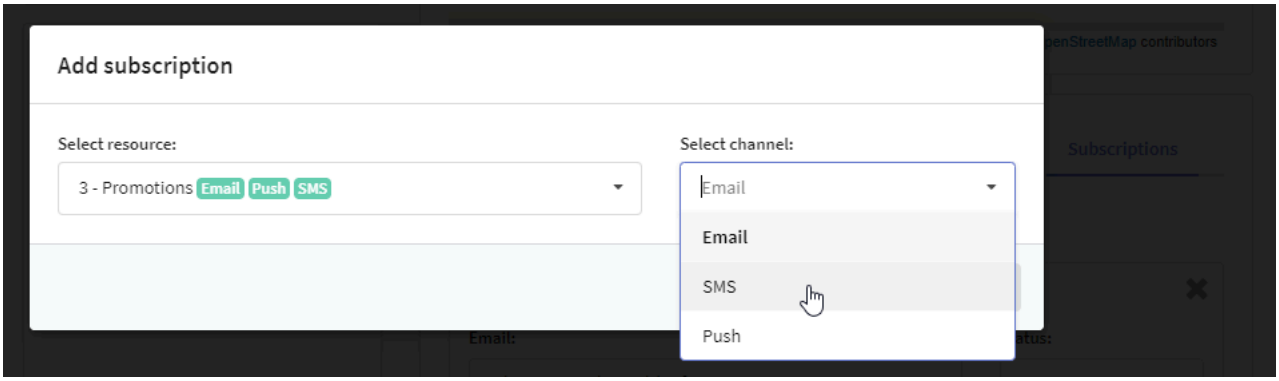
You can increase the priority of a subscription automatically. To do this, activate the option in the resource settings.



CAUTION

Please note that the "Increase priority for new subscriptions" option applies only if the priority is not explicitly specified when adding a new subscription. If the priority of the new subscription is explicitly specified (for example, in an API request), then the priorities of existing subscriptions remain unchanged.

If you want to subscribe a customer to a resourse manually, press the Add subscription button, specify a resource and a channel, enter contact data and save your changes:



TIP

The subscription process (opt-in) can be organized in two ways:

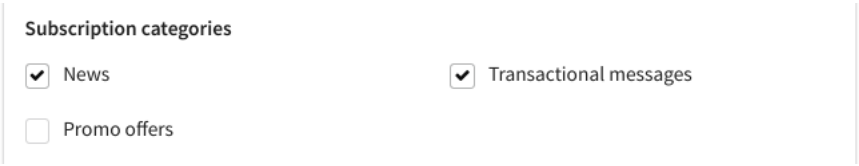
- **Single opt-in (SOI) - Without an agreement to receive content. It is enough to receive customer information (or create a subscription to a resource in Altcraft).**
- **Double opt-in (DOI) - Creating a subscription is not enough. Customers must send an email confirming their agreement to receive messages. This process allows you to exclude potential complaints from customers on marketing mailings, increase their effectiveness and improve sender reputation.**

Click here to find instructions on how to organize Double opt-in subscriptions in Altcraft.

Subscription categories

A resource subscription can include multiple categories. Categories enable personalization of offers. The client will receive messages only within the categories to which they are subscribed.

Unlike a subscription to a resource, a subscription to a category has only 2 statuses: “Subscribed” and “Unsubscribed”. To subscribe a profile to a category, check the appropriate box. You can also manage category subscription status via API requests.



Channel subscription fields

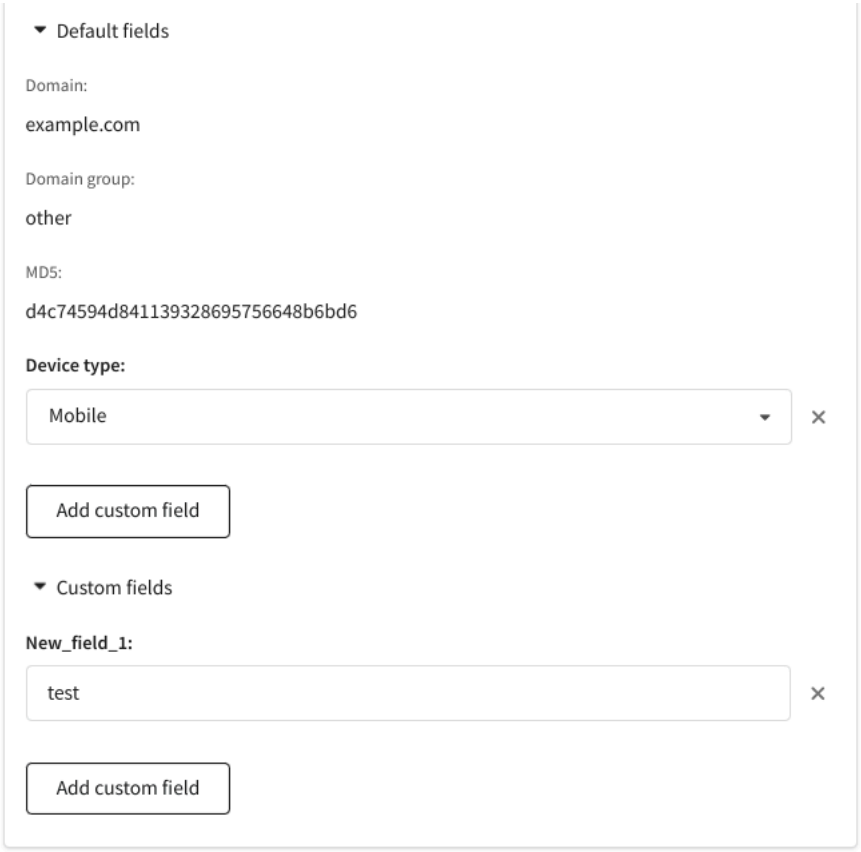
Data about a profile’s subscription to a specific channel is stored in standard and custom fields.

You can also include additional standard fields that store information about the device and profile application:

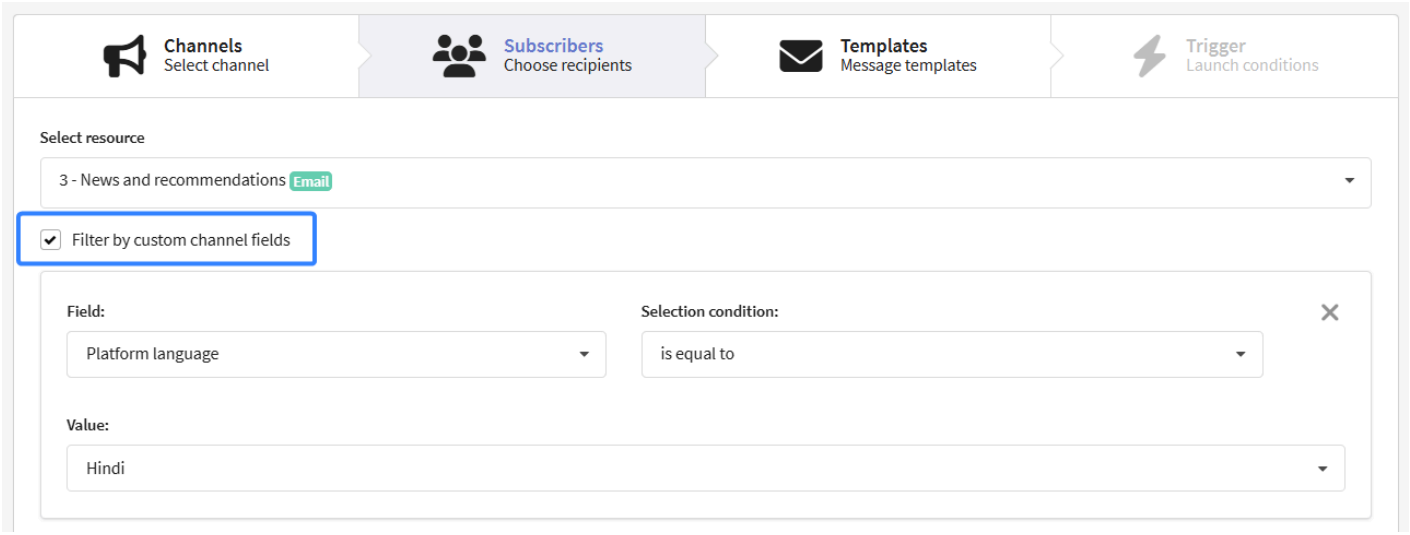
- **Device type (Desktop, Mobile, Other)**
- **Device model**
- **Device name**
- **Platform (Windows, Linux, Mac, Opera, iOS, BlackBerry, Android)**
- **Platform version**
- **Platform time zone**
- **Platform language**
- **Ad tracking allowed (True or False)**
- **Platform language**
- **Ad identifier**
- **Application ID**
- **Application installation ID**
- **Browser: Bots, Internet Explorer, Firefox, Safari, Facebook*, Chrome, Opera, Outlook, Thunderbird**
- **Application version**
- **Browser version**

To add a new field, click the Add custom field button and select a field from the list, then enter its value.

To store additional information about your channel subscription, use custom fields. They are created in resource settings.



You can use them to filter the recipients of a trigger mailing. This will make the mailing more personalized.



Subscriptions Limit

There is a subscriptions limit for a profile. A profile can have no more than 200 subscriptions, or no more than 20 subscriptions per resource in a channel. If the maximum number has been reached, a new subscription will replace one of the old ones.

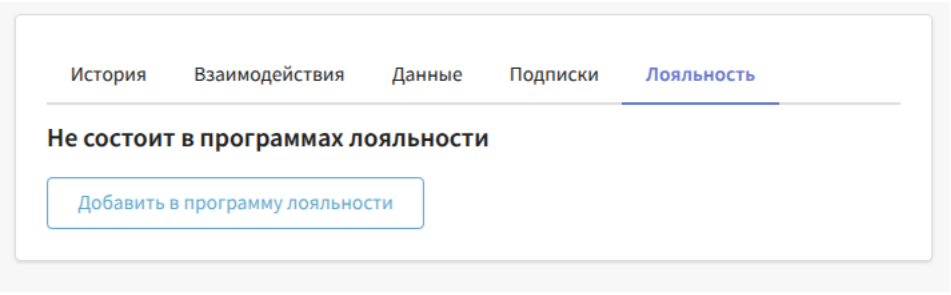
The following categories of subscriptions are being removed at first place:

- **All deleted subscriptions**
- **All subscriptions with statuses:**
  - **Invalid**
  - **Hard bounced**
  - **Complainer**
  - **Suspended**
  - **Unconfirmed**
- **Subscriptions with the lowest priority**
- **Oldest subscriptions by creation date**

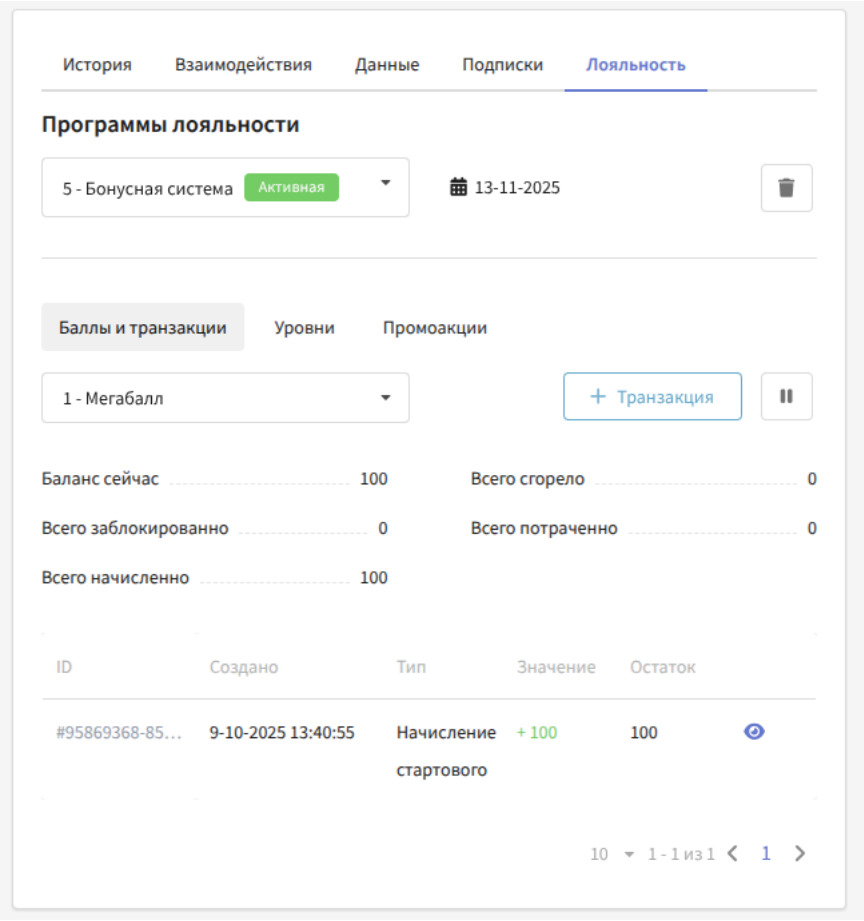
If you try to add replaced subscription, the platform will consider it a new one and re-add it to the list of subscriptions.

Loyalty programs

This section displays information about a profile's participation in loyalty programs. If the profile is not enrolled in a program, you can register it:



If the profile is already registered in one of the loyalty programs, this tab shows the profile's points and transactions, its tier in the loyalty program, and participation in the program's promotions. From here, you can also perform points transactions for the profile:



Identifiers

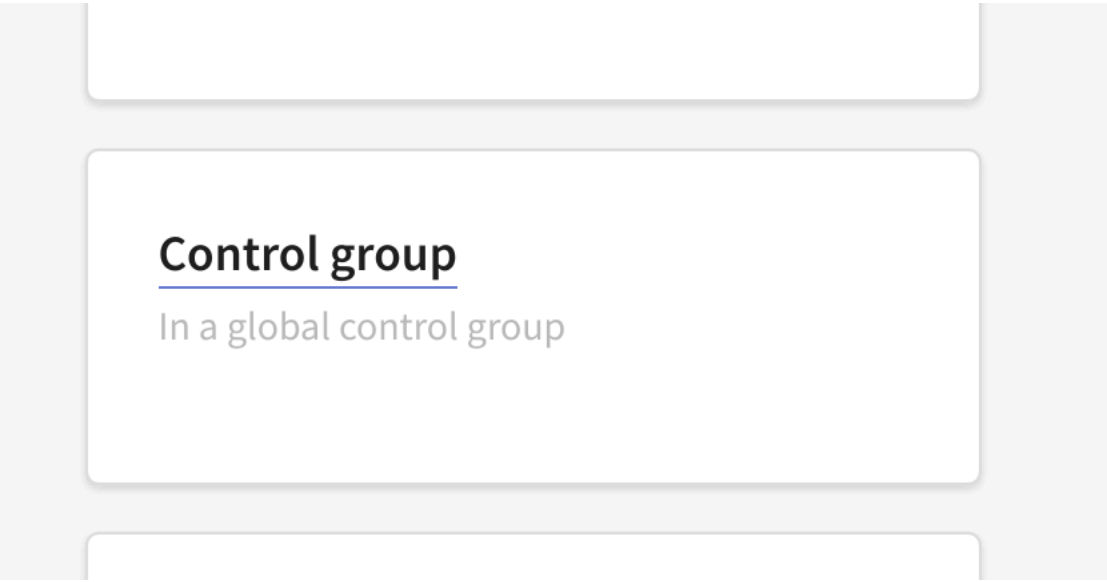
	Every subscriber profile in Altcraft Platform has a unique profile identifier profile_id — for all databases. It also gets unique MD5 and XXH identifiers. Within a database, the standard Email field is unique for profiles. However it is not required:if email address is missing, a new profile will get a **profile_id@localhost record.**

UTM

	UTM tags are updated in the subscriber's profile if they clicked on a link with UTM tags in the message template or submitted a [form](/user-guide/weblayers/forms/form-creating/37848225) with configured UTM tags (the "Update UTM data to profile" option is enabled in the form).

Participation in the control group

If the client profile is currently in the global control group, the corresponding mark will appear in their profile:

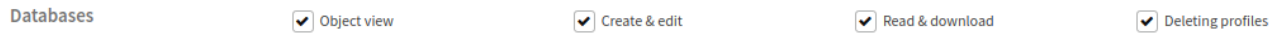


Also, the action history will show every time it gets in the GCG:

Deleting profile

	If you need to delete a profile from the database, you can do it using the "Delete profile" button. This action will also remove profile actions history and relations.

Special permission is necessary for deleting profiles. It can be configured in the user roles. You can read more about roles here.



You can also use an API request to delete a profile (see more).

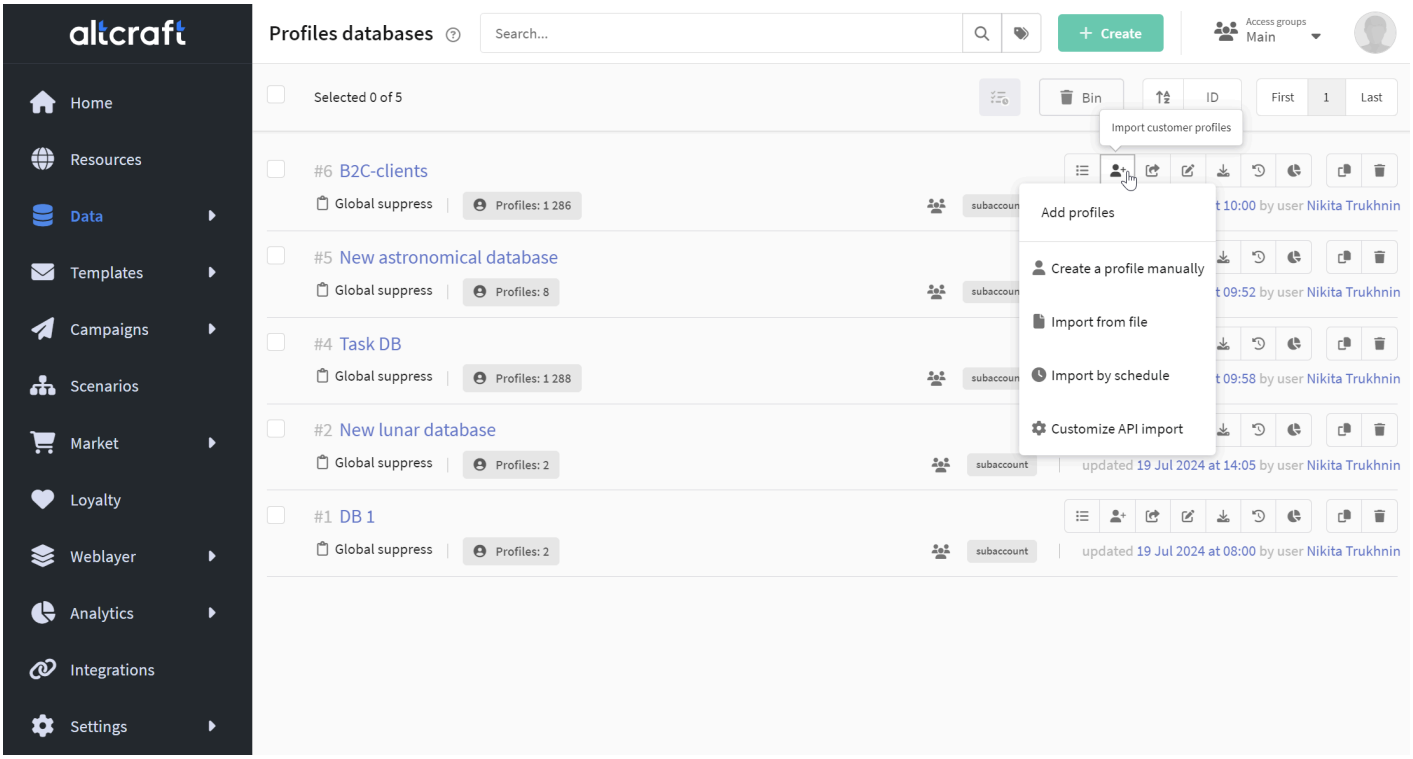
The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

Profiles import and data update

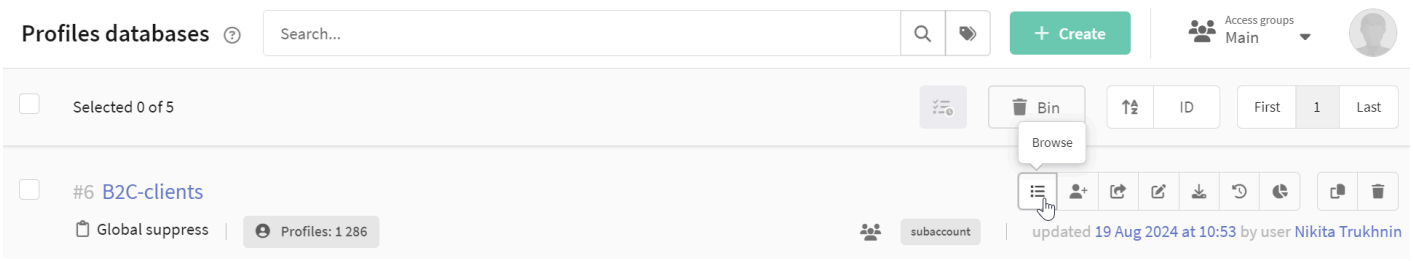
How to import data

There are three ways to proceed to import data:

1. **Under Data, select Profile Databases. Click on “Import Customer Profiles” next to the profile database you want and select the appropriate way to load the data:**

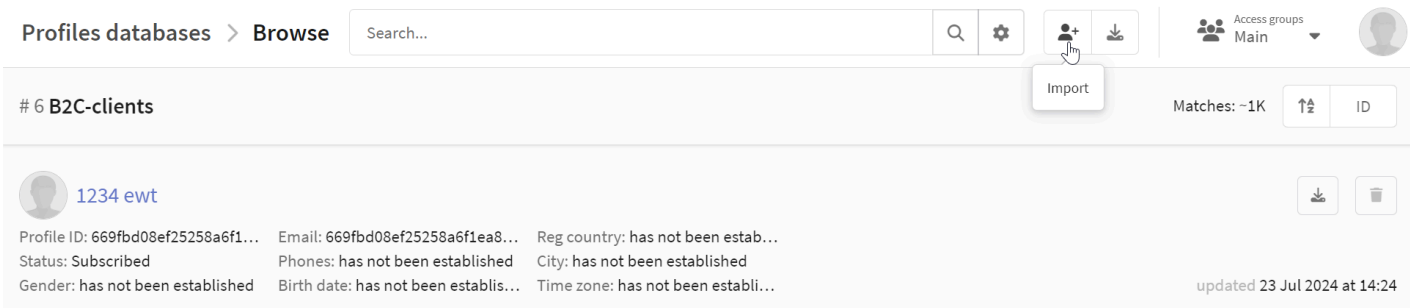


2. **Select your desired Profile Database and click on the Browse tab**



Click on the icon

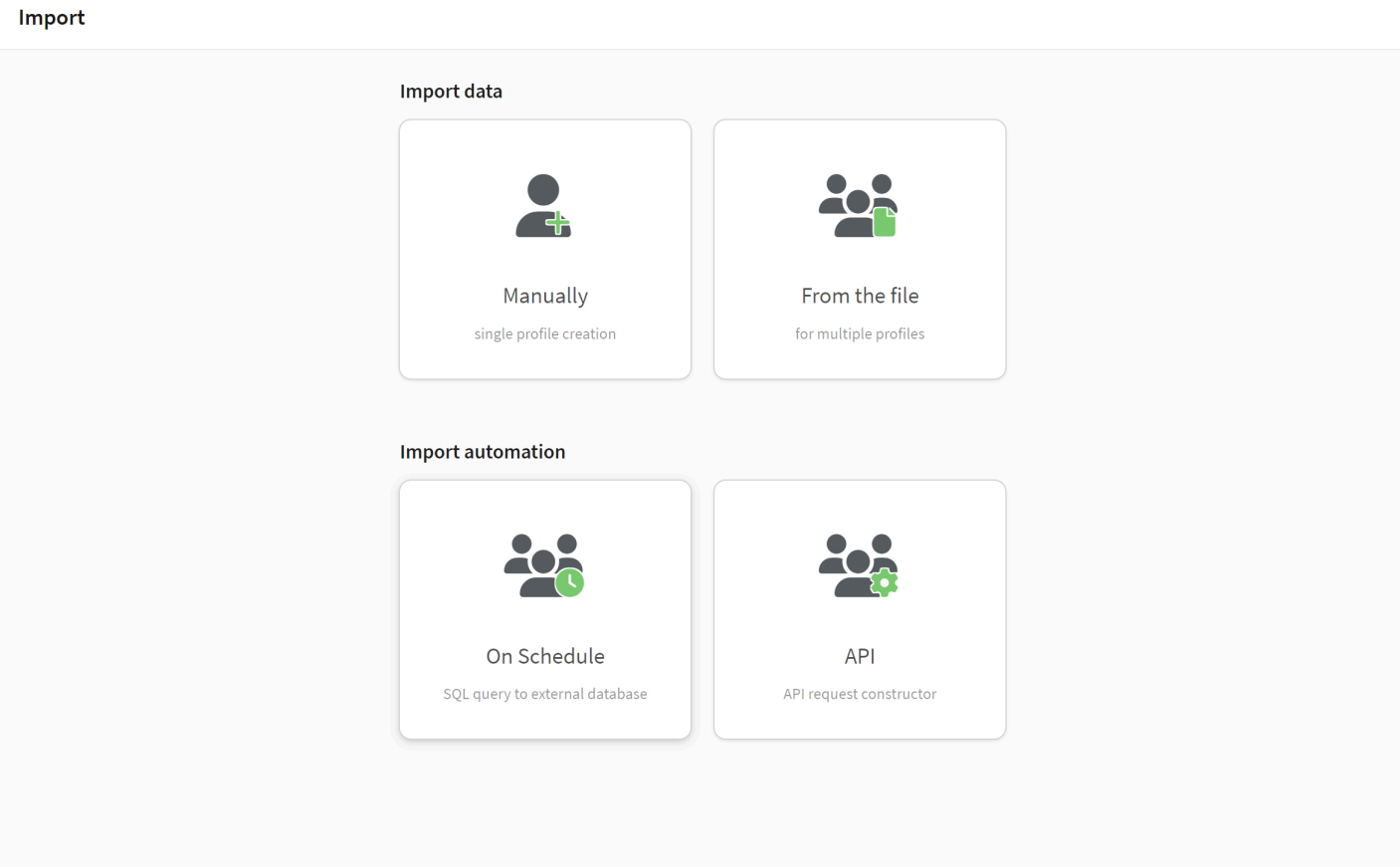
at the top of the screen and select the import method that suits you:



3. **In the Data section select Import.**

Customer profiles import and update can be organized depending on database size, new subscribers processing scenario (directly or from CRM), and company technical capabilities. There are 4 ways to import profiles into a database:

- **Manual import — add one profile at a time using the edit form**
- **From a file — import profiles from a file with customer data**
- **Via API — import profiles using API integration**
- **Schedule — import profiles from the external SQL database on schedule**



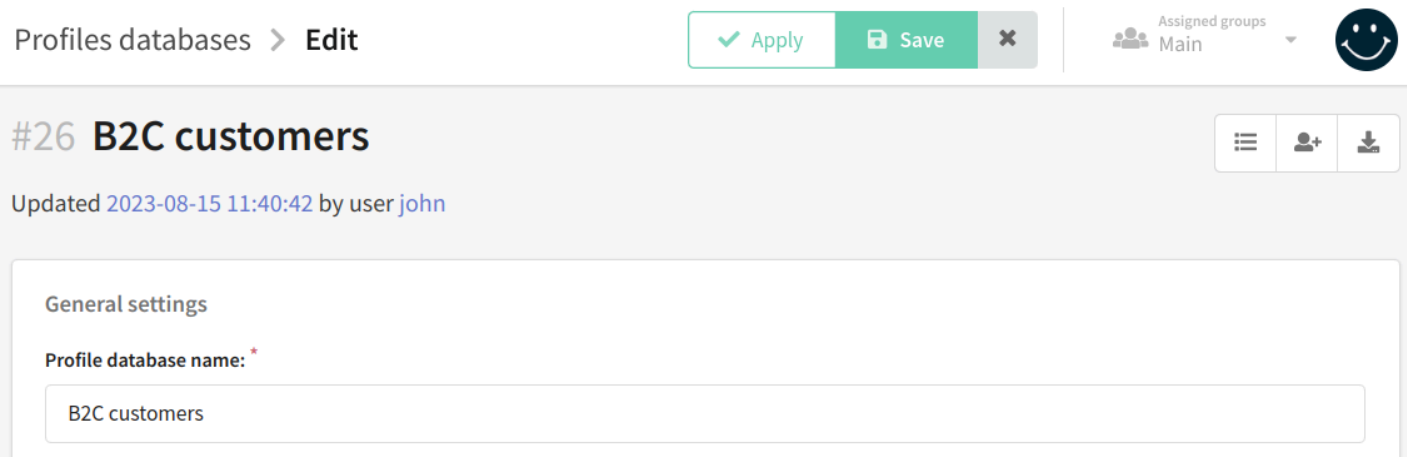
TIP

Altcraft Marketing also has integrations with external services:

- **Tilda — integration with Tilda data collection forms.**
- **LPGenerator — integration with LPGenerator data collection forms.**
- **Yandex AppMetrica — integration with Yandex application metrics service.**

Manual profile creation

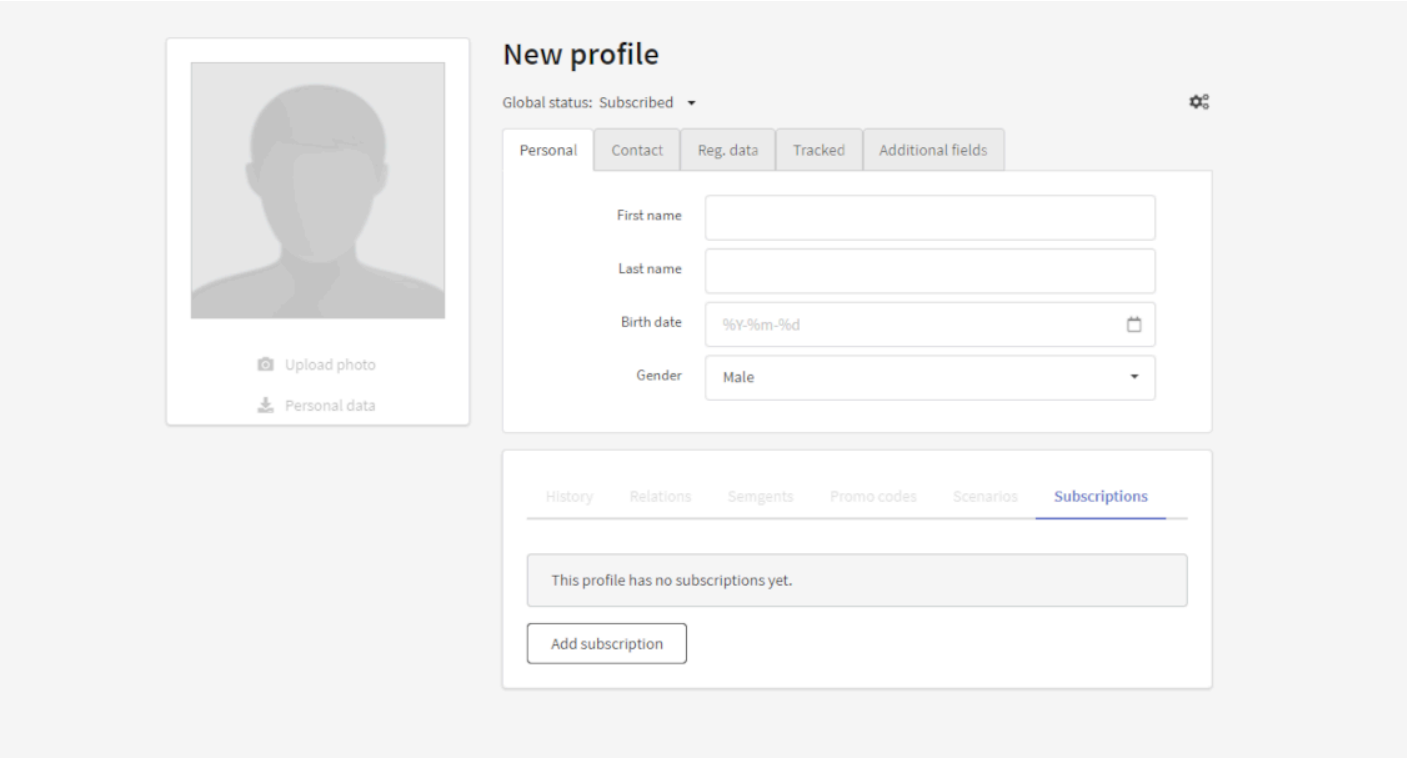
Select the import method - Manually.



The profile creation/editing form will open. During profile creation, you can add personal, contact, registration, and tracked data. This information is recorded in standard database fields.

Values for additional (custom) fields are specified in the "Additional fields" tab. Mandatory fields are marked with a red asterisk. You can create additional fields and set them as mandatory on the database editing page.

Below, on the Subscriptions tab, you can immediately assign a client to a specific resource when creating the profile.

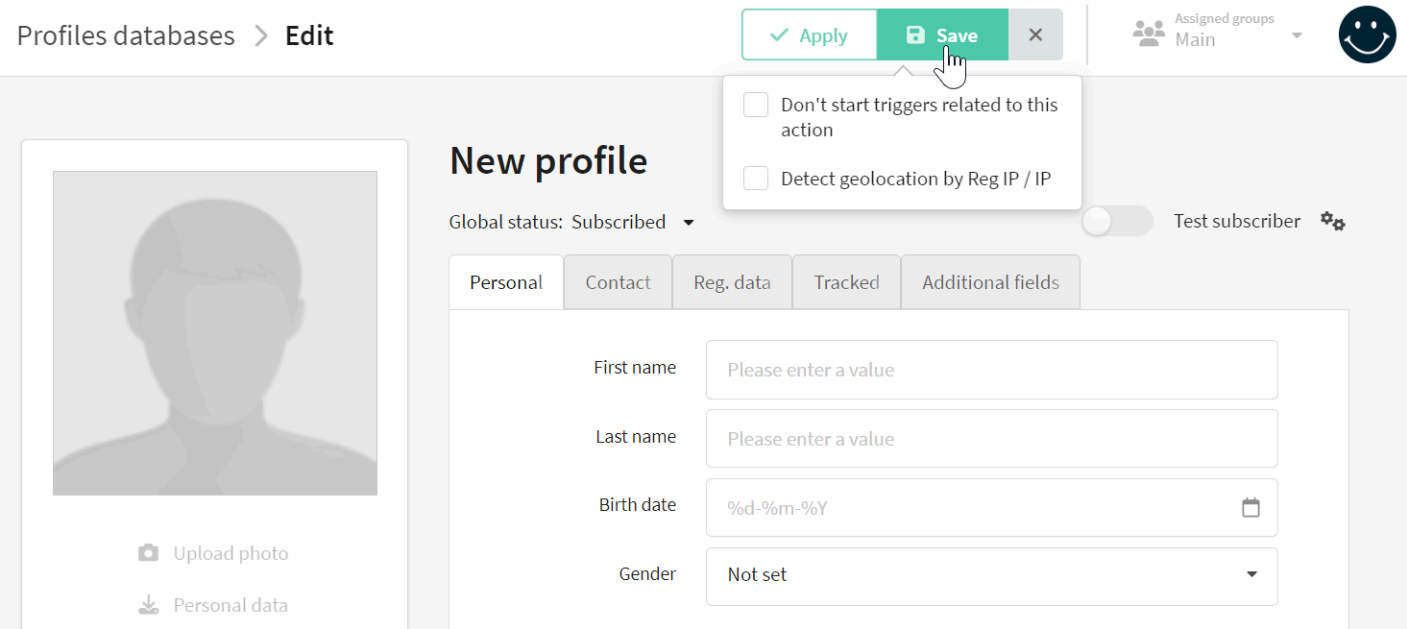


CAUTION

Email and Phones fields on the Profile tab are not used for communications. You can store contacts in these fields and use them in subscriptions. Youcan also leave these fields empty and store customer information in subscriptions.

By default, the Email field is used for profile identification along with profile_id. You cannot create a profile manually without this field.

When saving a new profile, you can disable triggers and enable location by Reg IP (registration IP address)/IP (last known IP address):

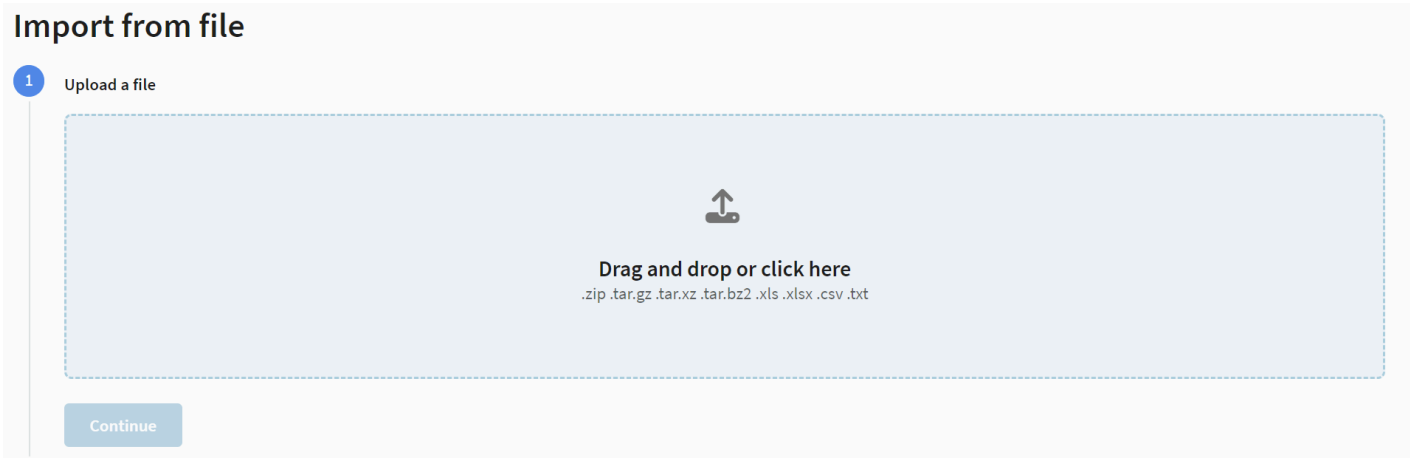


Import from file

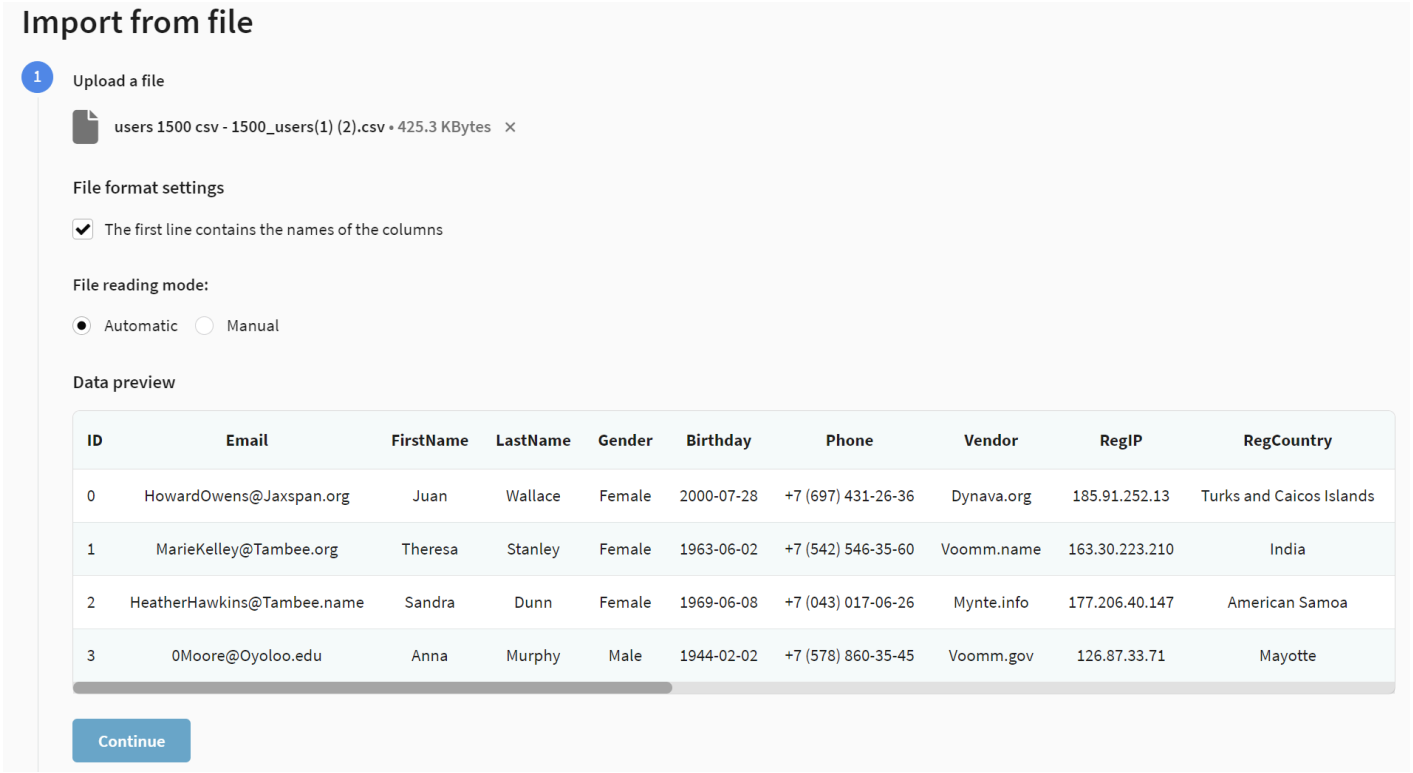
Uploading a file

You can use import from files to upload many profiles or update existing data. csv, txt, xls, xlsx, zip, tar.gz formats can be used.

Multiple table files can be uploaded into import from file form simultaneously. Press X next to file name to remove it. To add a new file, drag it to upload area or click on it and select file:



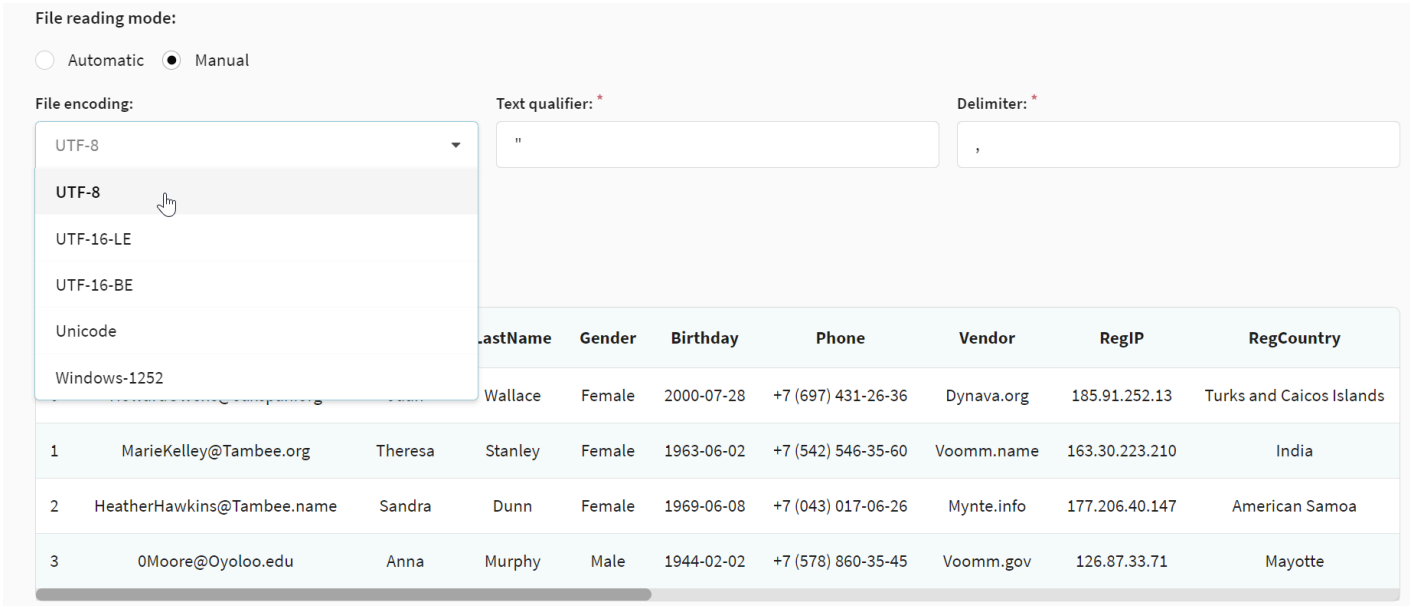
Check the “The first line contains the names of the column” box if the first line in your profile file is used to designate columns.



ID	Email	FirstName	LastName	Gender	Birthday	Phone	Vendor	RegIP	RegCountry
0	HowardOwens@Jaxspan.org	Juan	Wallace	Female	2000-07-28	+7 (697) 431-26-36	Dynava.org	185.91.252.13	Turks and Caicos Islands
1	MarieKelley@Tambee.org	Theresa	Stanley	Female	1963-06-02	+7 (542) 546-35-60	Voomm.name	163.30.223.210	India
2	HeatherHawkins@Tambee.name	Sandra	Dunn	Female	1969-06-08	+7 (043) 017-06-26	Mynte.info	177.206.40.147	American Samoa
3	0Moore@Oyoloo.edu	Anna	Murphy	Male	1944-02-02	+7 (578) 860-35-45	Voomm.gov	126.87.33.71	Mayotte

Select File Reading Mode. Two modes are available: Automatic and Manual.

The automatic mode uses UTF-8 encoding, if your file has a different encoding, you can go to manual reading mode and choose the one that suits you:



			LastName	Gender	Birthday	Phone	Vendor	RegIP	RegCountry
			Wallace	Female	2000-07-28	+7 (697) 431-26-36	Dynava.org	185.91.252.13	Turks and Caicos Islands
1	MarieKelley@Tambee.org	Theresa	Stanley	Female	1963-06-02	+7 (542) 546-35-60	Voomm.name	163.30.223.210	India
2	HeatherHawkins@Tambee.name	Sandra	Dunn	Female	1969-06-08	+7 (043) 017-06-26	Mynte.info	177.206.40.147	American Samoa
3	0Moore@Oyoloo.edu	Anna	Murphy	Male	1944-02-02	+7 (578) 860-35-45	Voomm.gov	126.87.33.71	Mayotte

- **Text qualifier — a character used to indicate the beginning and end of text data within a table cell. Qualifiers are required when the data within a cell contains characters that can be seen as delimiters or other special characters.**
- **Delimiter — a character that is used to separate data into separate fields (cells) in a table row. Delimiters are used to indicate the boundaries between columns in a table.**

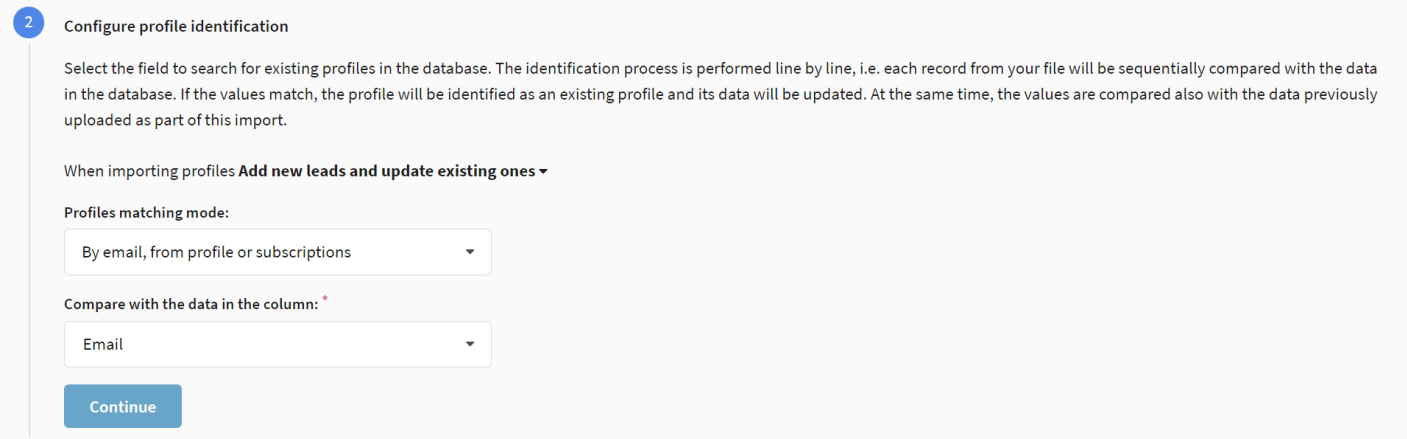
Profile identification

Select Import mode — it will depend on whether you add new records or update existing in the Altcraft profiles database:

- **Add new leads and update existing ones**
- **Add new leads, do not update existing ones**
- **Update only**

Profile matching mode defines what information from your file will be used in Altcraft for profiles search. This is necessary to update data without duplicating records. The following modes are available:

- **By email, from profile or subscriptions — search by email from database or subscriptions.**
- **By email from profile — search by database profile email.**
- **By phone, from profile or subscriptions — search by phone in database field or SMS subscriptions.**
- **By custom field — search by custom database field, like customer ID in company CRM.**
- **By subscription — serach subscription data.**
- **By profile ID - by unique profile ID (only when updating data).**



TIP

It is necessary to take into account the following peculiarities of matching:

- **The value of the search field should be unique for each profile in the database, otherwise the platform will not be able to unambiguously identify the profile being searched for**
- **Only one search mode should be used for one profile database. This is to avoid creating duplicate profiles in the database.**
- **If you are going to use custom field matching, you should contact the platform administrator to create an index for this field. Thanks to the index, searching for a profile in the database will take much less time.**

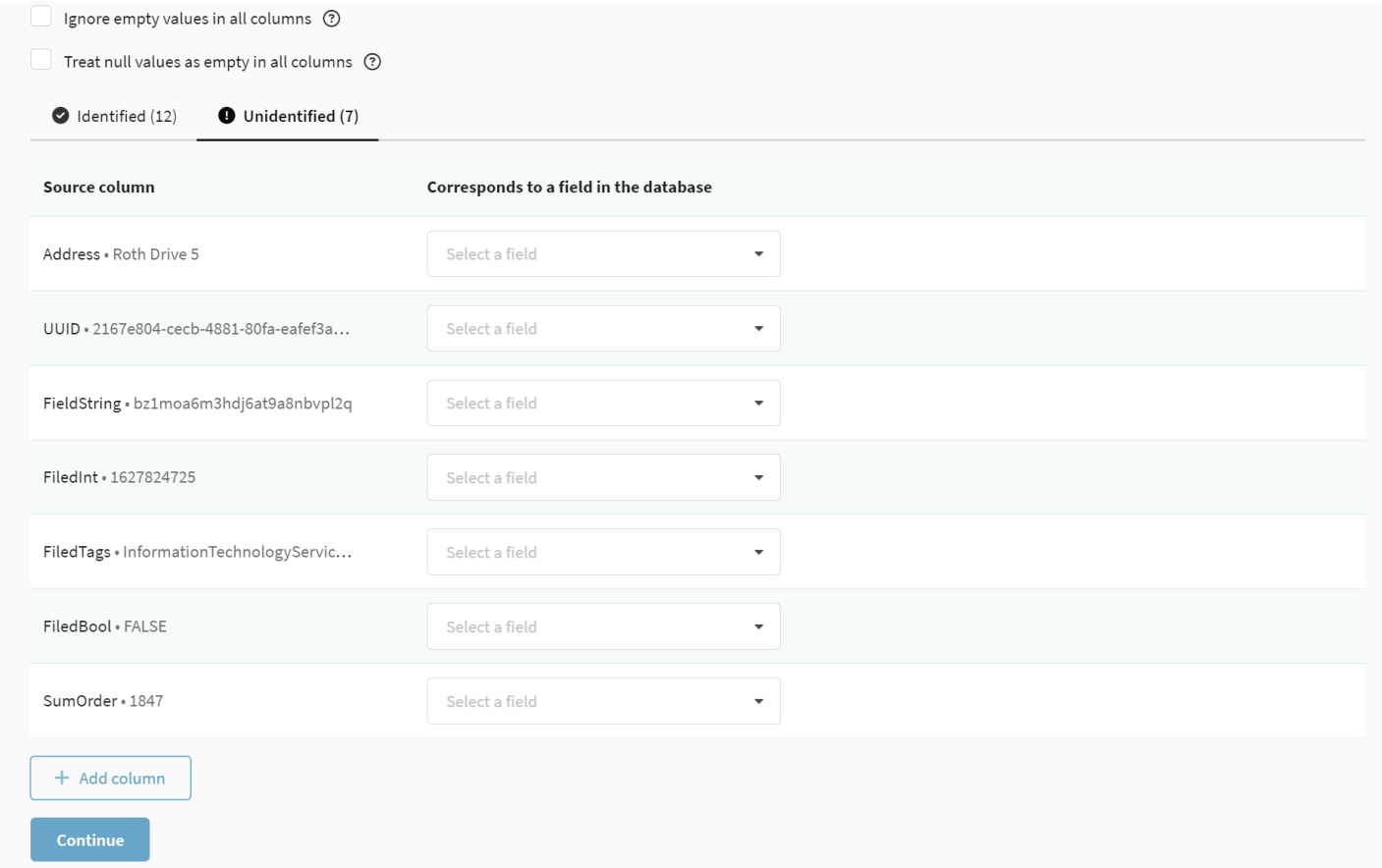
Fields customization

TIP

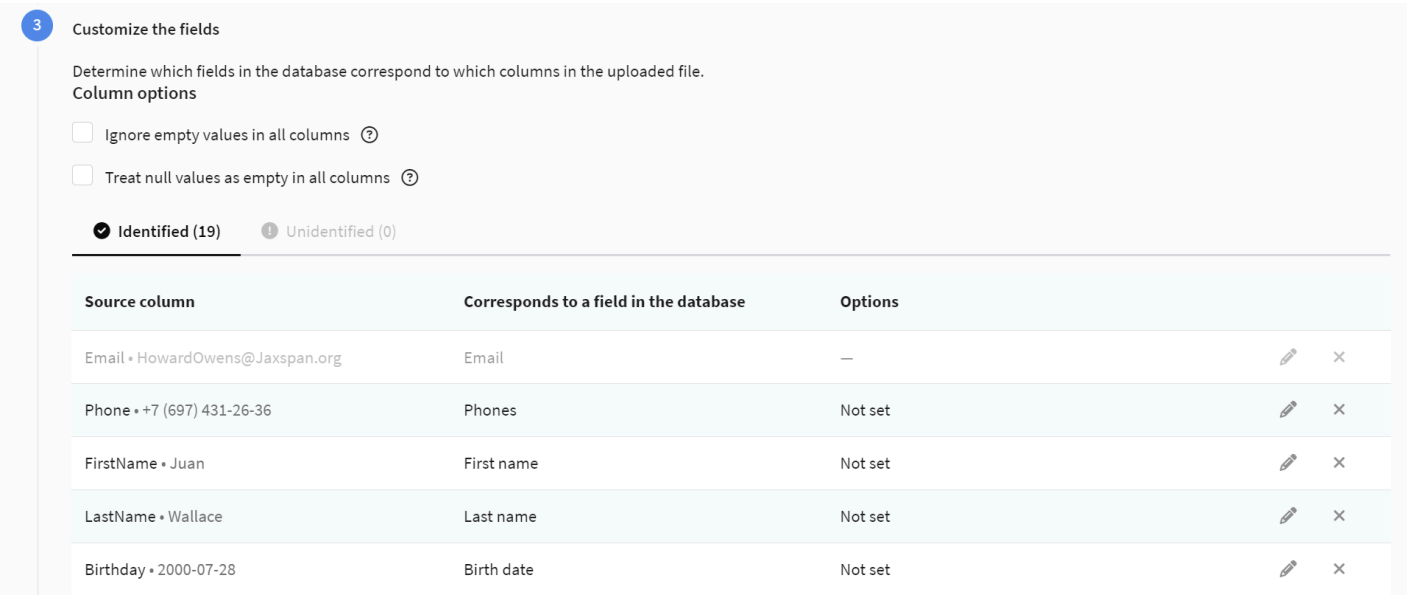
Pass profile contact information to Altcraft resources subscriptions (see the previous paragraph). You can store this data in database fields in order to search profiles. If customer identification is configured using CRM or other identifiers, email and phones may not be stored in appropriate fields.

You need to match columns in Altcraft database and columns in your file. Altcraft will detect the columns automatically if their names match the system ones, however, if your columns do not match the system ones, you can customize them manually by going to the "Unindentified" column:

- **Source column — select data source from your file. The firstrecord will be displayed for each column for data type proper selection.**
- **Corresponds to a field in the database — select customer profile field to store data. You can select standard and custom fields.**



After you select matches for the undefined columns, they will automatically be moved to the Identified column. If you have set an incorrect match, click on the cross to the right of the field you want, then it will be moved back to the Unindentified column.



The “Ignore empty in all columns” option is used when updating profile data that is already in the database. For example, the profile field “vendor” already contains the value “test.com”, but now you are importing a file where the “vendor” field for this client is empty. The platform will overwrite the field with an empty value. To avoid this, enable the “Ignore empty in all columns” option for the corresponding column. In this case, the value in the “vendor” field will remain the same.

If your file has unknown client data written with values such as `null`, `nil`, `none` or `undefined`, the platform can also proccess these values as empty. To do this, you need to enable both the "Treat null value as empty and Ignore empty values fields. In this case, the platform will not update the field.

Options

When configuring the import of data from a file, certain options are available for each field type that determine how the data will be processed and saved in the system:

Option	Description	Type of field
Ignore empty values	**Prevents overwriting profile data with empty values during import, preserving existing information if the corresponding fields in the imported file are empty.**	**All**

Option	Description	Type of field
Treat null values as empty	**The platform will treat values of type null, nil, none or undefined as empty if this option is enabled.**	**All**
Automatic register correction	**Converts letters to the selected register. For example: All letters are lowercase, Each word starts with an uppercase letter, or All letters are uppercase.**	**First name, Last name, Subscriptions(type of field - Line)**
Select the date format from the file	**Allows you to select the date format that is represented in the imported file.**	**Date**
Default value	**Preset value that is automatically filled in the field if there is no data for this field in the uploaded file.**	**Date**
Match the values of columns and fields	**If the values in your column do not correspond to the system values of the platform, you can specify their correspondence.**	**Gender(type of field - List)**
Increase the field value by the value from the column	**Adds a value from the imported file to an existing value in the specified database field, instead of replacing it.**	**Integer, Float**
Decrease the field value by the value from the column	**Subtracts the value from the imported file from the existing value in the specified database field, instead of replacing it.**	**Integer, Float**
Round values to a specified number of decimal places	**Automatically reduces a numeric value to a specified number of decimal places during data loading to ensure accuracy or format compliance. Rounding is applied to the data from the file, not to the total value that will be written to the profile field.**	**Float**
Tag delimiter	**A character or set of characters used to separate multiple tags (labels) in a single file field so that the platform can properly recognize and import them as separate values.**	**Tags**
Add or update data by object field	**An option that either creates a new value for the specified object field if it does not exist, or updates an existing value in the database based on data from the imported file. Please note that this update will work correctly only if the objects in the array have the same structure ([{key":“value_1”},{“key”:“value_2”}]).**	
Tags, Array of objects		
Delete data by object field	**An option that removes objects with the same value in the specified field.**	**Array of objects**
Import to property	**Writes a value from a file column to the specified property of the object.**	**Object**

Field import specifics

When importing data into the system, the platform expects specific value formats for each field type. All valid formats are described in this article. Make sure the data in your source matches these requirements, otherwise the platform will not be able to process it correctly.

For Object and Array of Objects fields, the platform expects the value to be a string containing valid JSON. To set this value, you need to first convert the JSON object into a string and then escape it according to the CSV rules described in RFC 4180 (double quotes " are replaced with “”).

Example: `"[{""field1"": 123, ""field2"": {""subfield"": false}, ""field3"": ""abc""}]"`

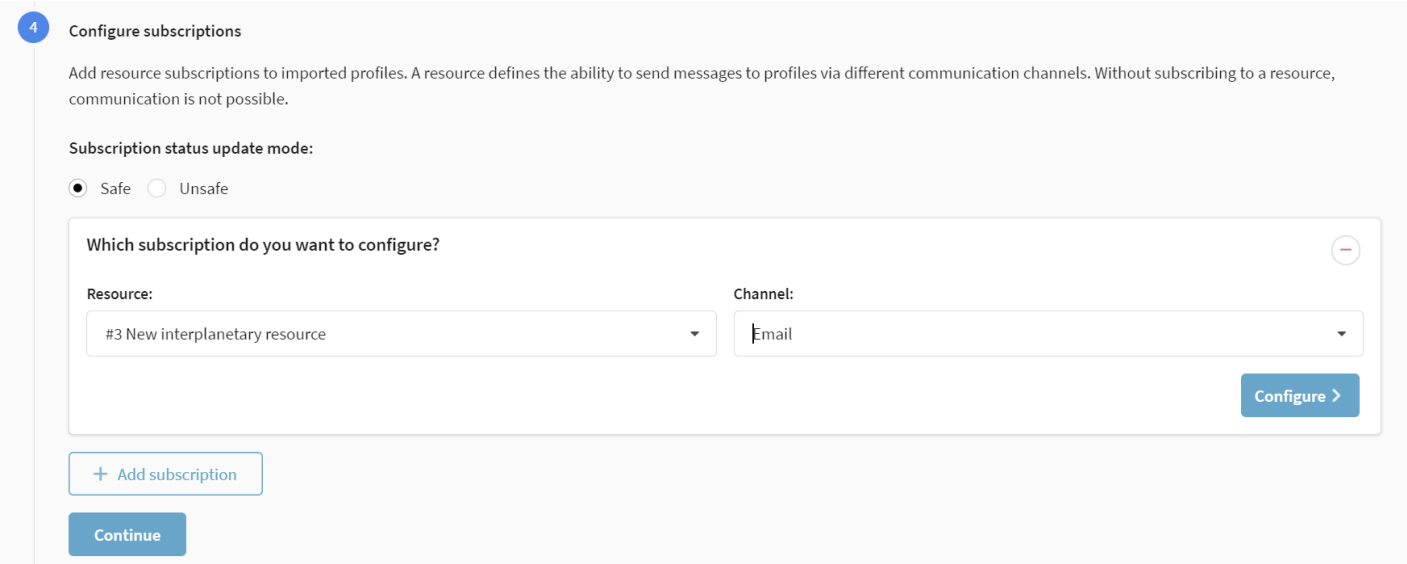
For Enum fields, you can import either the values themselves or their indexes. For example, if you have created a enum list in the database with options “tariff_1”, “tariff_2”, “tariff_3”, you can specify the same values in the source column or use the indexes: *1 — “tariff_1”*, *2 — “tariff_2”*, *3 — “tariff_3”*, *0 — “Not selected”* (default value).

Please note that values consisting only of digits cannot be imported directly — the platform will treat them as indexes. For example, if the list contains the values 100 and 200, you need to import the corresponding indexes: 1 — 100, 2 — 200.

Subscriptions configuration

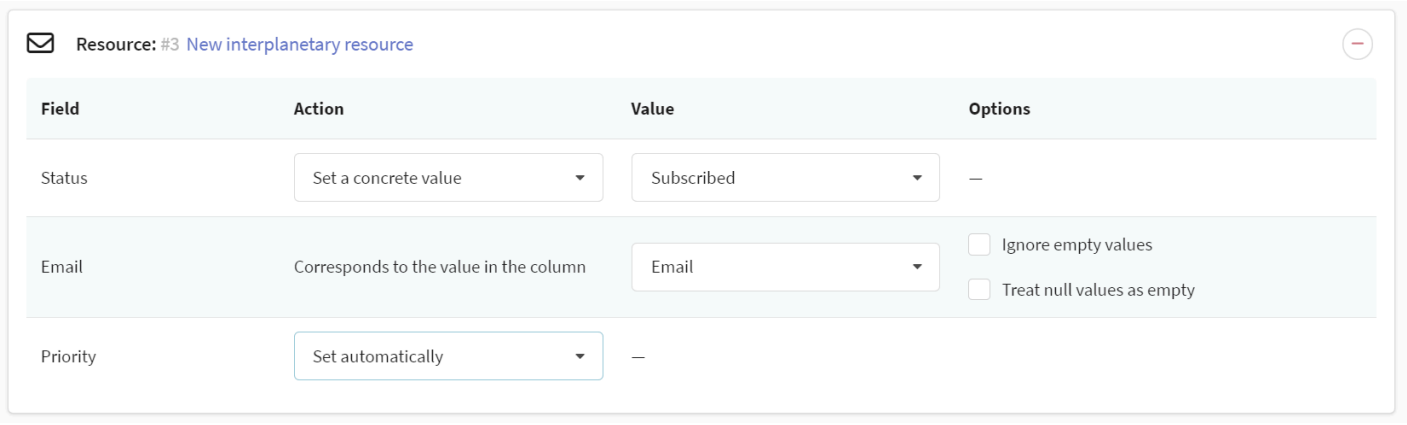
When importing from a file, subscribe your clients to the necessary resources. A resource defines the possibility to send messages to profiles via different communication channels. Without subscribing to the resource, communication is impossible. In the Customize subscriptions block, click the + Add subscription button.

You have two modes of updating the status of your subscriptions: safe and unsafe. If you choose unsafe mode, subscriptions of profiles with the following statuses will be updated: Unsubscribed, Complainer, Hardbounced, Not valid. If a profile already has a subscription to the selected channel in the resource with one of these statuses, after import its status will be changed according to the specified settings.

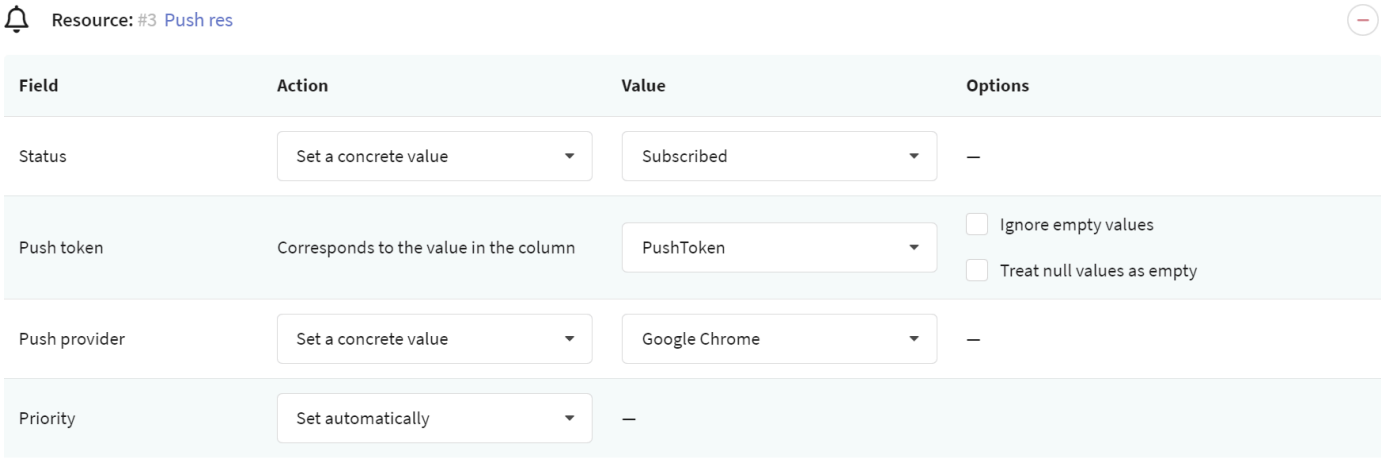


For each subscription, select Resource, Communication channel, Subscription status and set the field matches.

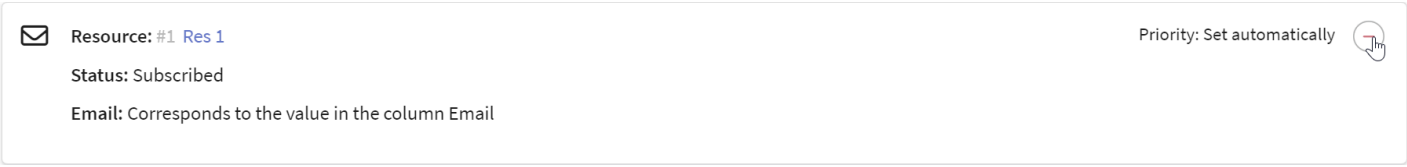
Each profile is limited to no more than 20 subscriptions per resource.



For subscriptions to push notifications, additionally specify push service provider and push token:

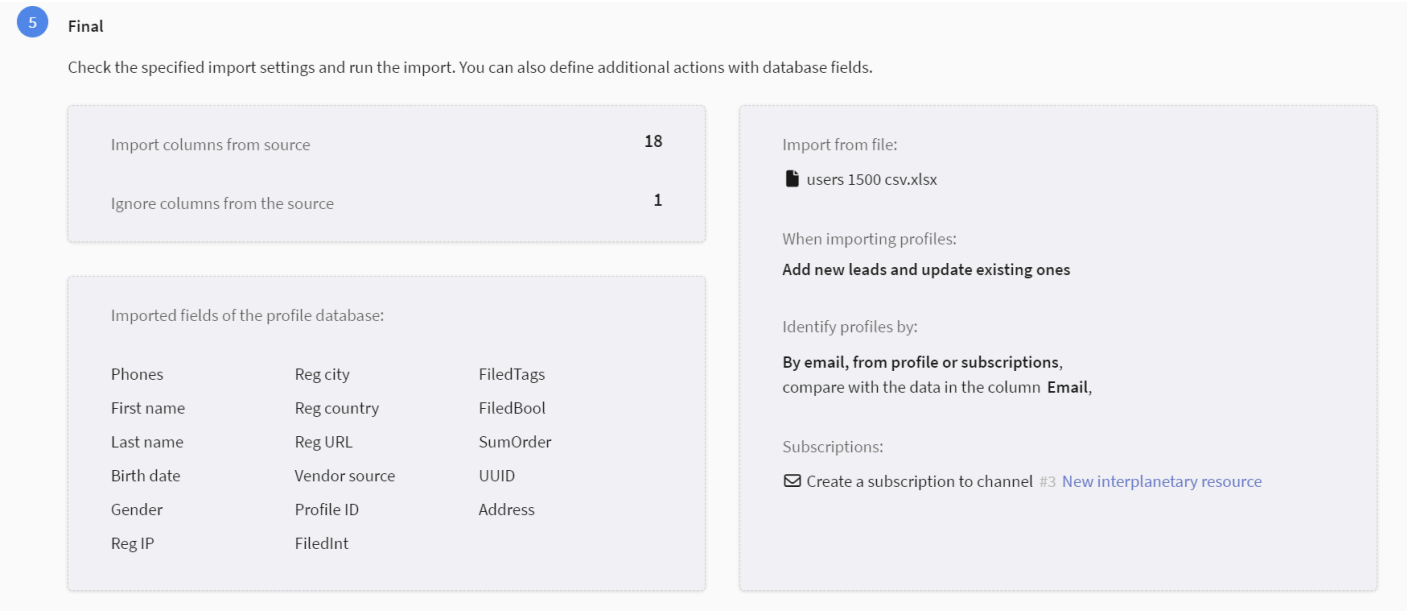


Once a subscription is set up, you can edit it by clicking on it, or delete it by clicking on the red minus icon.



Final

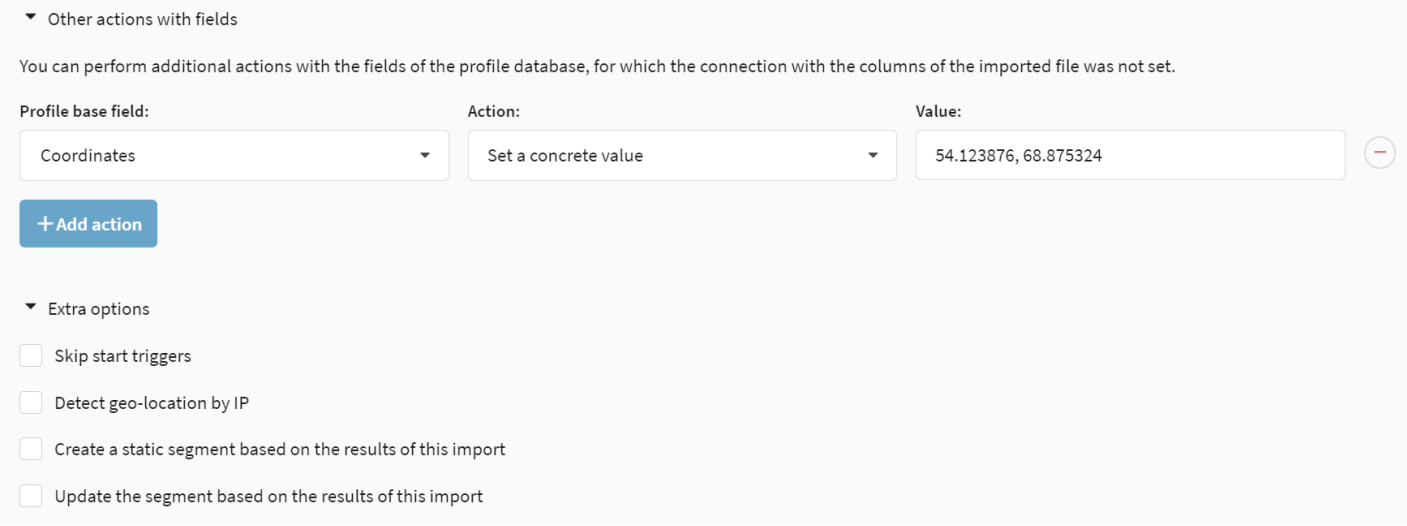
In the final step of the import you can check your settings and run the import, as well as define additional actions with database fields.



In the Other actions with fields block, you can do additional actions on the profile database fields that have not been matched.

Furthermore, additional functions are included in the Extra options block:

- **Skip start triggers — disable on-import triggers and scenarios for this operation.**
- **Detect geo-location by IP — When you submit data to the Reg IP or IP fields in IPv4 format, Altcraft will automatically detect geolocation and you will be able to select profile fields such as Coordinates, Country, City - or Country Reg. and City Reg. to fill in, and profile location on map widget will also be noted.**
- **Create a static segment from the results of this import — when importing profiles static segment is automatically created.**
- **Update a segment from the results of this import — when importing profiles, the segment is automatically updated.**



Transferring status on import

When customizing fields and subscriptions, you can pass the profile global status and subscription status, respectively. The status in the file column can be specified in either text or numeric format.

Possible field values:

- **0, “subscribed” - available for communication via all communication channels**
- **1, “unsubscribed” - unsubscribed from receiving messages on all channels**
- **2, “complained” - complained about spam, communication stopped**
- **3, “hardbounced” - ISP has completely refused to accept messages to this address**
- **4, “unconfirmed” - subscriber registered but did not confirm consent to receive mailings**
- **5, “suspended” - subscription is suspended for some time**
- **6, “invalid” - excluded from marketing mailings for any other reasons**

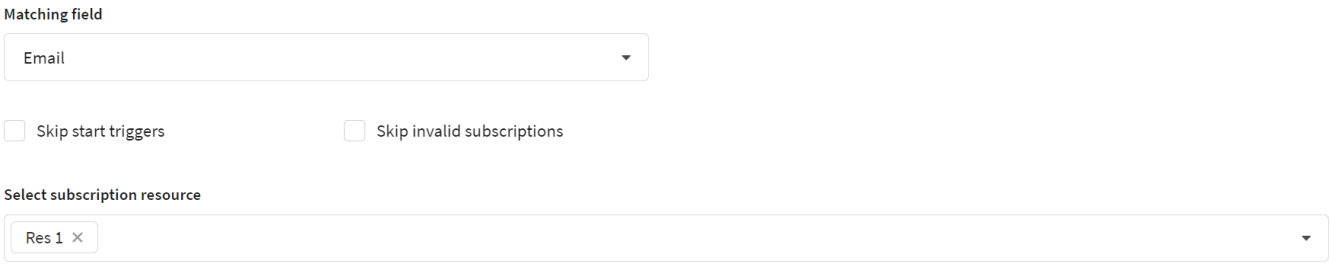
Import via API

API is used for integration with external systems — sites, applications, customer data management (CRM) services, data collection forms. In the API request constructor you can set certain parameters for importing data and pass them to your developers.

In the “Matching field”, select the data that will be used to search for existing profiles:

- **Email - by profile field Email.**
- **Phone - by profile field Telephones.**
- **Profile ID - by unique profile ID (only when updating data).**
- **Custom field - by custom database field. For example, customer ID in a CRM company.**

Also select resource to subscribe.



Check the Skip start triggers box if you do not want to run scenarios and trigger mailings configured to import into the selected database.

Check the Skip invalid subscriptions box if you do not want to add a profile subscription if invalid data is present in the query.

Below you will see a template of the API request:

Template v1.1 | cURL v1.1

POST URL

https://nikita-trukhnin.dev.altkraft.com/api/v1.1/profiles/import

HTTP Headers

Token name	Value
Content-Type	application/json

Query body

```
{
    "token": "<Your previously configured token>",
    "data": {
        "email": String ,
        "phones": phones ,
        "_fname": String ,
        "_lname": String ,
```

TIP

- **Use API query to import or update a single profile.**
- **Other import API queries are described in the Profiles section.**
- **Managing profile subscriptions is described in the Subscriptions section.**

Scheduled Import from External SQL Databases

TIP

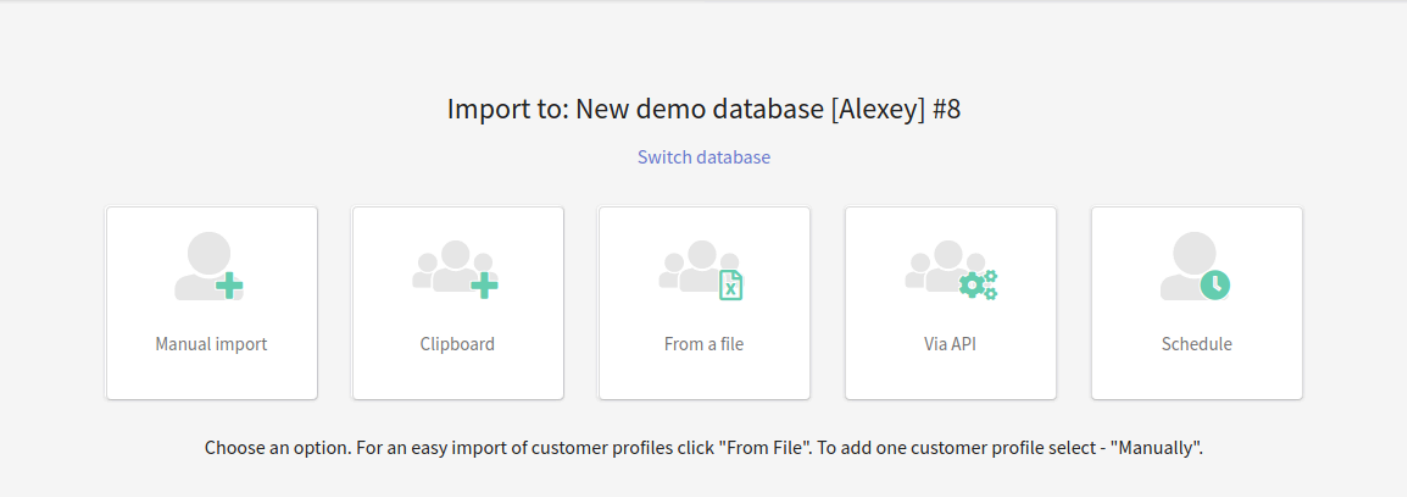
Altcraft can automatically collect data to import new customers profiles and update existing profiles from ClickHouse, MySQL, MS SQL, PostgreSQL and Oracle RDBMS.

Instructions for configuring scheduled import are in a separate article.

Scheduled customer data import

Customer data import can be automated using scheduled Database import. The import module takes data from external data tables. To configure automatic data collection, you need to configure SQL connector in admin panel and bind it to your account. In your account, to proceed to import creation, go to Data —> Database Imports.

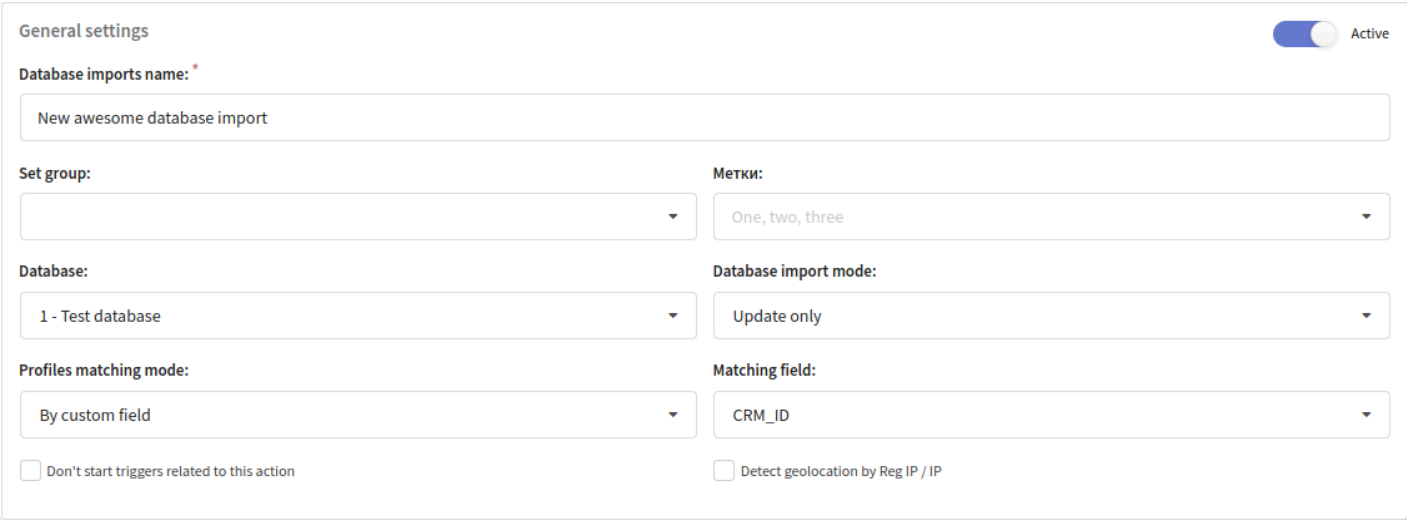
You can also go to this section from the list of databases by choosing “Import customer profiles” in the context menu on the right:



General settings

In general settings of import, you can set import name, assign access groups and labels and set how the import will process data in the platform account:

1. **Database — import will add or update fields in the selected database of customer profiles. The set of available fields for storing information depends on the database.**
2. **Database import mode — import can work in three modes:**
   - **Add new leads and update existing ones**
   - **Add new leads, do not update existing ones**
   - **Update only**
3. **Profiles matching mode — You can search by contact data, subscription data (including Push), by profile ID to update existing profiles — and by custom database field.**
4. **Matching field — if you search for profiles by custom field of the database, you need to select it in this menu.**



For scheduled import IP location is also available. If you have configured trigger mailings to import or update profiles, you can restrict their triggering for scheduled import.

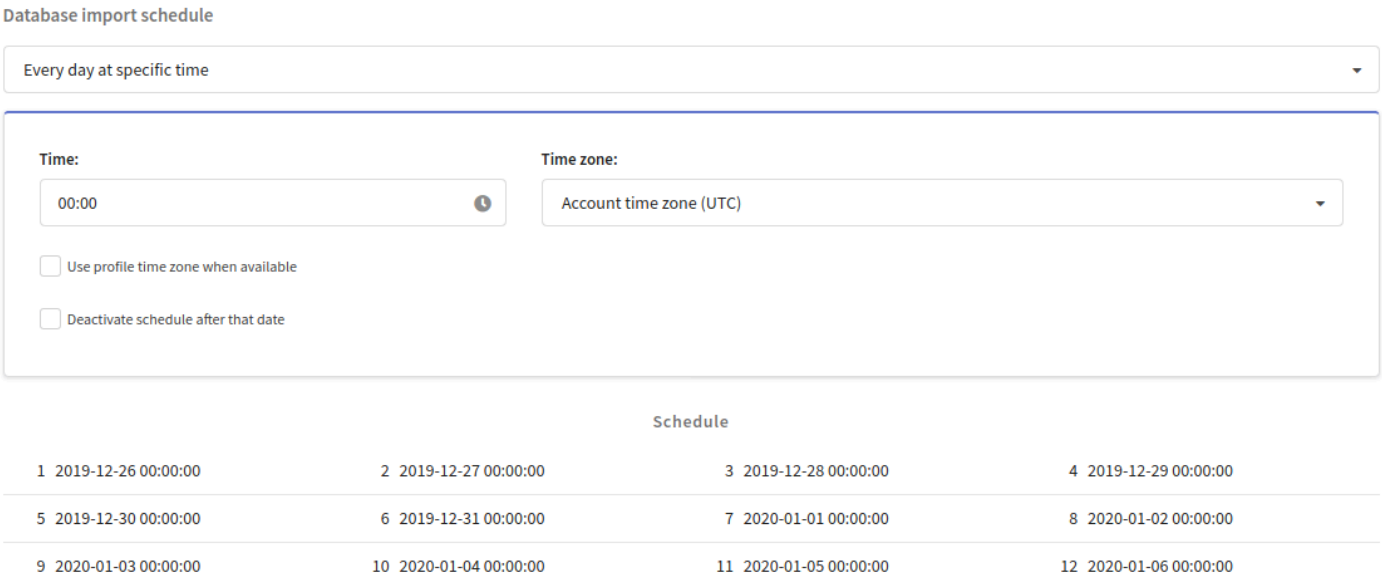
In the main settings there is an option to instantly start import in addition to the schedule. If this item is checked, after clicking the Apply or Save button, the import will be launched.

Database import schedule

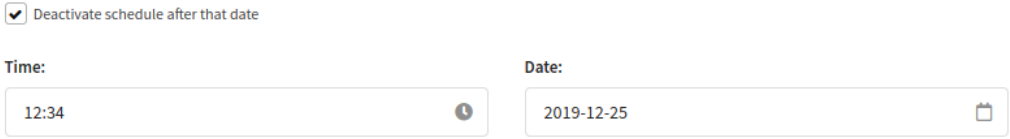
You can flexibly configure schedule for regular import. Choose schedule plan from the menu and configure it depending on your import purpose.

TIP

Mailing schedule is described in detail on [this page](#).

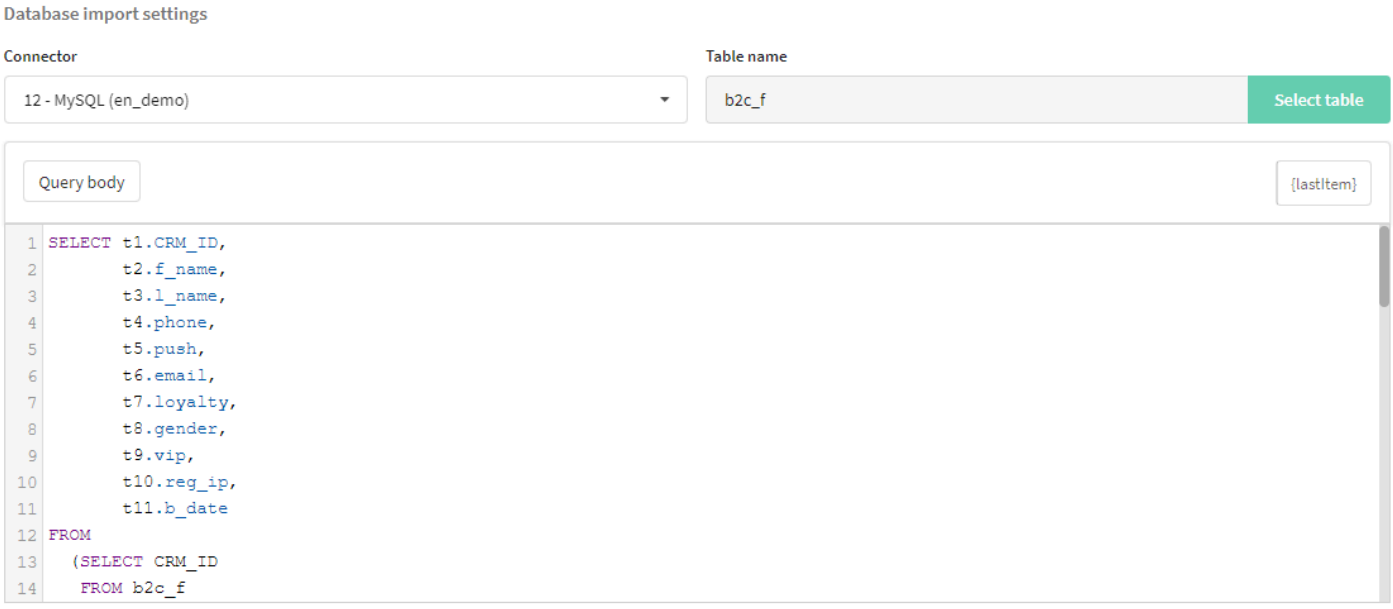


For each schedule plan, you can set the time zone for the calculation. You can also set import deactivation date:



Database import settings

Choose the necessary database connector (configured in administrative panel) in database import settings and the data table to take data from:



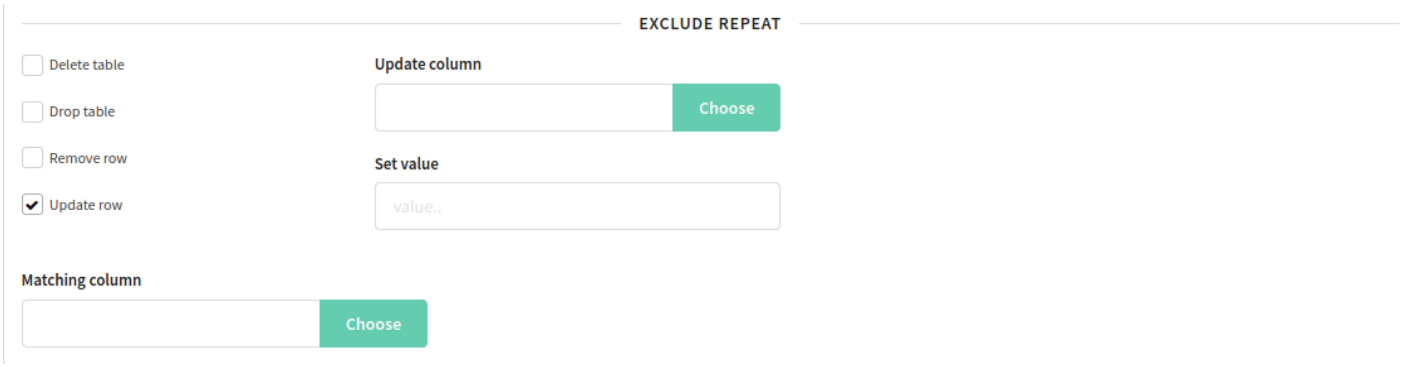
Create Query body depending on DB management system you use: MySQL, PostgreSQL, MSSQL, Oracle, Hive or ClickHouse. SELECT operator is necessary on your queries. There are no any other restrictions.

Prevent duplication

These functions are available to prevent repetative data update:

- **Delete table — deletes all records in data table**
- **Drop table — deletes table from database**
- **Remove row — if the value of the specified column in the record from the table and in the profile in the database coincides, such record will be deleted.**

- **Update row — updates values in selected column for imported records. It identifies records by Matching column (unique for every customer profile and differs from profile search column) and updates data in Update column.**



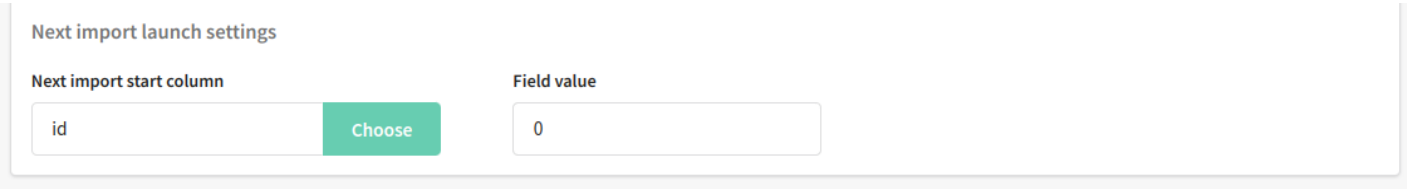
Splitting import into parts

When you import big amount of records, you can divide this process, to distribute loading. To do this, you need to add `{lastItem}` variable into Query body for the column with unique ID. It is recommended to sort IDs when making a query.

```
Query example: SELECT * from TestTable where id>{lastItem} LIMIT 10
```

Next, you need to specify from which row in the table you want to start the first import. To do this, select the column with the ID in Next import start column and specify Field value.

An example of settings for starting import from the first row (if the data in the `id` column is sorted in ascending order):

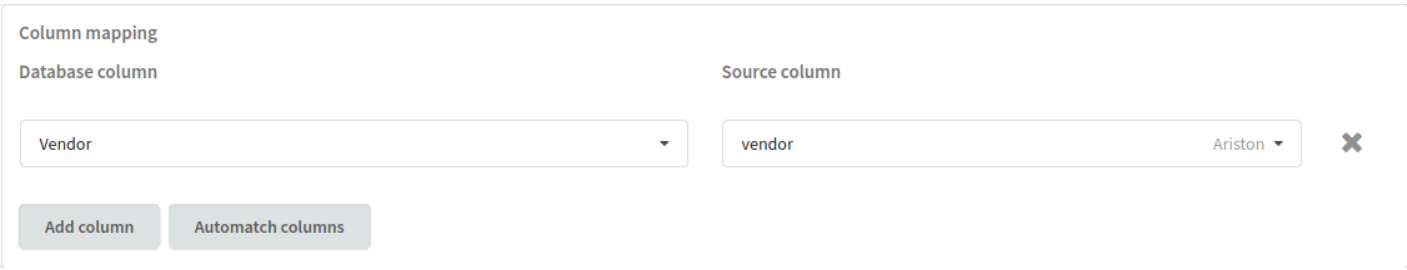


After starting a regular import, Field value will be automatically updated — it will be filled with the last ID after the previous run.

Column mapping

In this block, you need to determine the correspondence of profile fields in the Altcraft database and columns of the external SQL database. After clicking on the Add column button on the left of the drop-down menu, select Altcraft profile field for updating or recording data.Dur

When you choose column of external SQL database, its first value will be shown to the right as gray text, like in example below:



Field import specifics

When importing data into the system, the platform expects specific value formats for each field type. All valid formats are described in [this article](). Make sure the data in your source matches these requirements, otherwise the platform will not be able to process it correctly.

For Object and Array of Objects fields, the platform expects the value to be a string containing valid JSON. To set this value, you need to first convert the JSON object into a string and then escape it according to the CSV rules described in [RFC 4180]() (double quotes " are replaced with “”).

Array of objects example:

```
[
    {
        "field1": 123,
        "field2": {"subfield": false},
        "field3": "abc"
    }
]
```

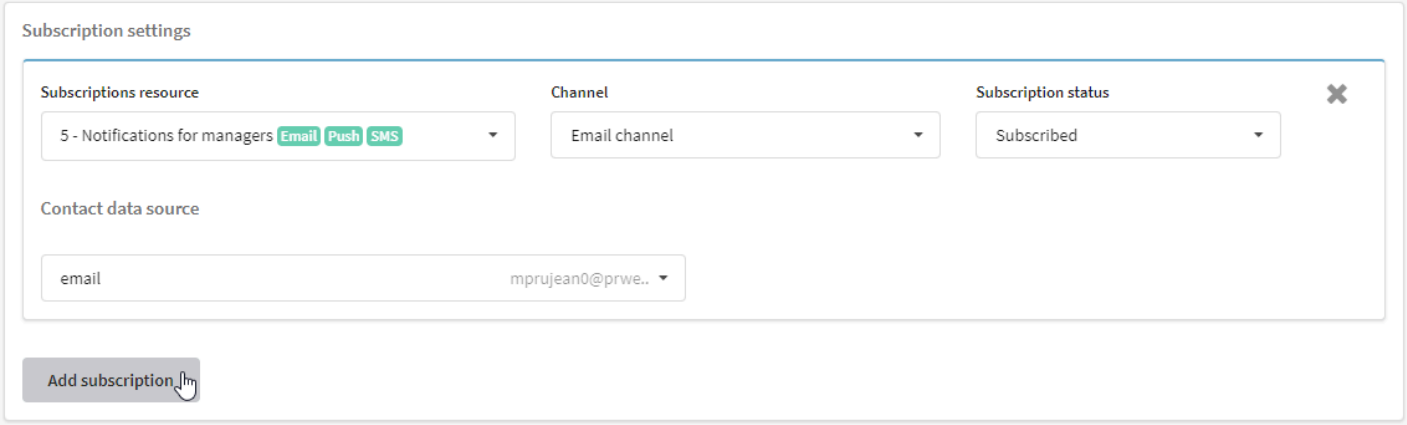
Expected value: `"[{""field1"": 123, ""field2"": {""subfield"": false}, ""field3"": ""abc""}]"`

For Enum fields, you can import either the values themselves or their indexes. For example, if you have created a enum list in the database with options “tariff_1”, “tariff_2”, “tariff_3”, you can specify the same values in the source column or use the indexes: *1 — “tariff_1”, 2 — “tariff_2”, 3 — “tariff_3”, 0 — “Not selected”* (default value).

Please note that values consisting only of digits cannot be imported directly — the platform will treat them as indexes. For example, if the list contains the values 100 and 200, you need to import the corresponding indexes: 1 — 100, 2 — 200.

Subscription settings

During import, you can automatically subscribe customers to certain resources - through one or more channels. For every subscription you need to select a resource, communication channel and select column in the data table from which contact data for subscriptions will be imported.

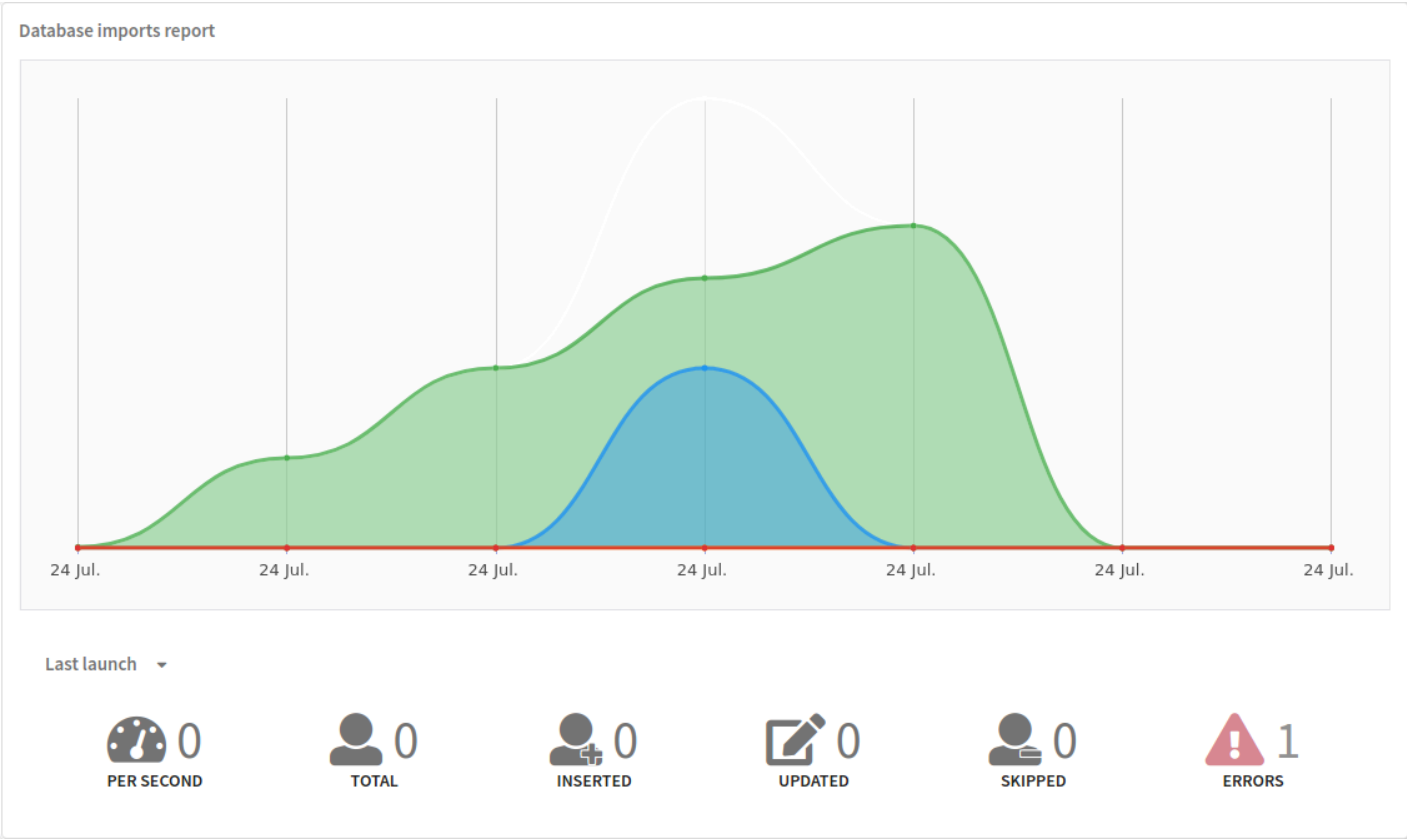


You can set status for every subscription:

- **During import, you can give "Subscribed" status for Single opt-in, or "Unconfirmed" status for Double opt-in logics.**
- **During data update, you can change subscription status depending on user actions, like suspending or unsubscribing**

Database imports report

In this block, you can see information about last 10 imports. If any errors occurred during last import, you can load report about them.



Automatic data collection

Altcraft Marketing platform provides a set of tools that keep profile information up to date and enrich it with user behavior and activity details.

This data can be further used for precise targeting segmentation or analytic reports.

Message tracking

The basic type of data collection automation is tracking message action history. It works by default for all available communication channels.

Message tracking records basic actions for messages like sent, delivered, opened or clicked.

Email tracking

Email action history tracking is processed by preset pixels — special tracking elements — that can be transparent images, code fragments or parts of URL links.

They can capture the following events:

- **message opened — a subscriber has opened the message**
- **message read — a subscriber has opened the message and kept it open for at least 10 seconds**
- **message clicked — a subscriber has clicked any link within the message**
- **link clicked — a subscriber has clicked a specific link within a message**
- **confirmed — a subscriber has clicked the subscription confirmation link**
- **unsubscribed - the unsubscribe link has been clicked by a subscriber**

Email sender provides you with email carrier information for profiles:

- **Delivered — the message has been delivered.**
- **Undelivered — the message hasn't been deliverd.**
- **Softbounced — sending client errors. The platform changes the profile status to "Hard bounced".**
- **Hardbounced — the provider completely refused to accept messages to this address.**
- **Complained — a subscriber reported to Spam. Communication is stopped.**

A single message link click history can be analyzed using Altcraft clicks map with content change history.

Email tracking data can also be used for cohort and lifetime value analytics.

SMS tracking

SMS links are also tracked by Altcraft platform. They are automatically shortened for SMS message versions:

`example.com/s/GrlX73`

SMS sender also tracks events like Delivered or Undelivered provided by SMS carrier.

Web Push tracking

Web push actions are processed by push providers: Mozilla, Google Firebase or Apple Safari.

As a result you receive events like "Push delivered", "Push opened"and "Push link clicked".

App Push tracking

App push actions are tracked by the push providing application.

For events like "Push delivered" and "Push opened" there are two links in each message `"ack"` and `"open"`, activated by the application GET request.

Optionally you can use event "Push link clicked" with `"click_action"` parameter.

AK Pixel

Pixel tracking scripts are implemented into your web site and track site visitors' custom actions. For more information see the specified documentation page.

API integrations

Using the API, you can create and edit profiles.

- **Profiles**
- **Profile subscriptions**

Scenarios

A scenario is a customizable logical process of automatic data processing.

The Refresh Data node directly modifies the data in the selected profile fields.

The Feedback node sends a message to the specified server, which can use the Altcraft API or work directly with the database.

Action hooks

Action hooks are used for sending events information to custom interfaces.

It can process more events than triggers — and they can both make http requests and send messages directly to RabbitMQ.

Profile update link attributes

You can use your messages links to update specific profile fields. The attribute notation looks like:

`update-field_name="value"`

It changes values in existing profile fields.

```
<a href="http://www.example.org" name="link" update-clicked_link="true">Update clicked_link!</a>
```

To subscribe your customer to a different resource on click — use the following link attribute:

```
<a href="http://www.example.org" subscribe-to="1">Subscribe me to resource #1</a>
```

Altcraft Marketing platform provides a set of tools that keep profile information up to date and enrich it with user behavior and activity details.

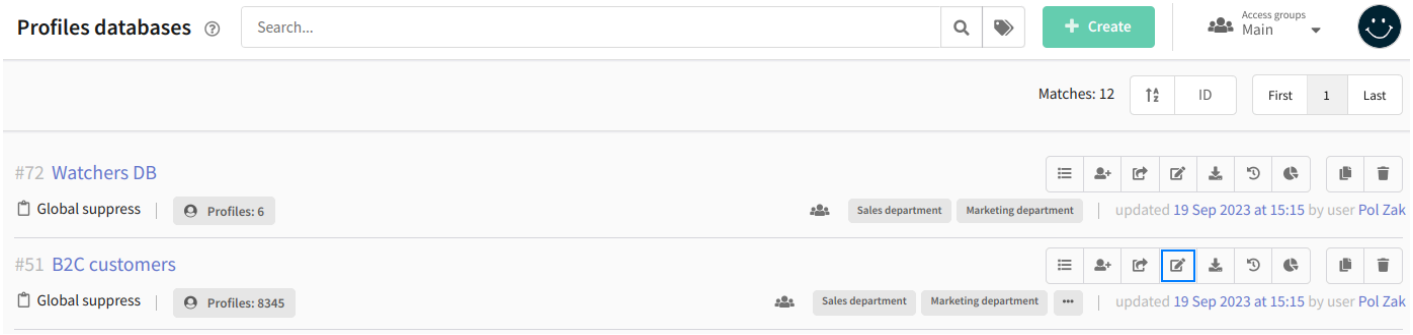
This data can be further used for precise targeting segmentation or analytic reports.

Bulk customers profiles update

Bulk profiles update allows you to update manually Database or the Segment of customers. In this case, conditions can be configured, to make update query more accurate. The following functions are available:

- **Adding and changing customer profile fields value**
- **Adding and updating [profiles subscriptions](/user-guide/profiles-and-databases/9405665)**
- **Managing [promo codes](/user-guide/loyalty-programs/)**
- **Deleting profiles from a database**
- **[Global profile unsubscribe](/user-guide/profiles-and-databases/9405665#Subscriberprofile-Globalprofilestatus)**

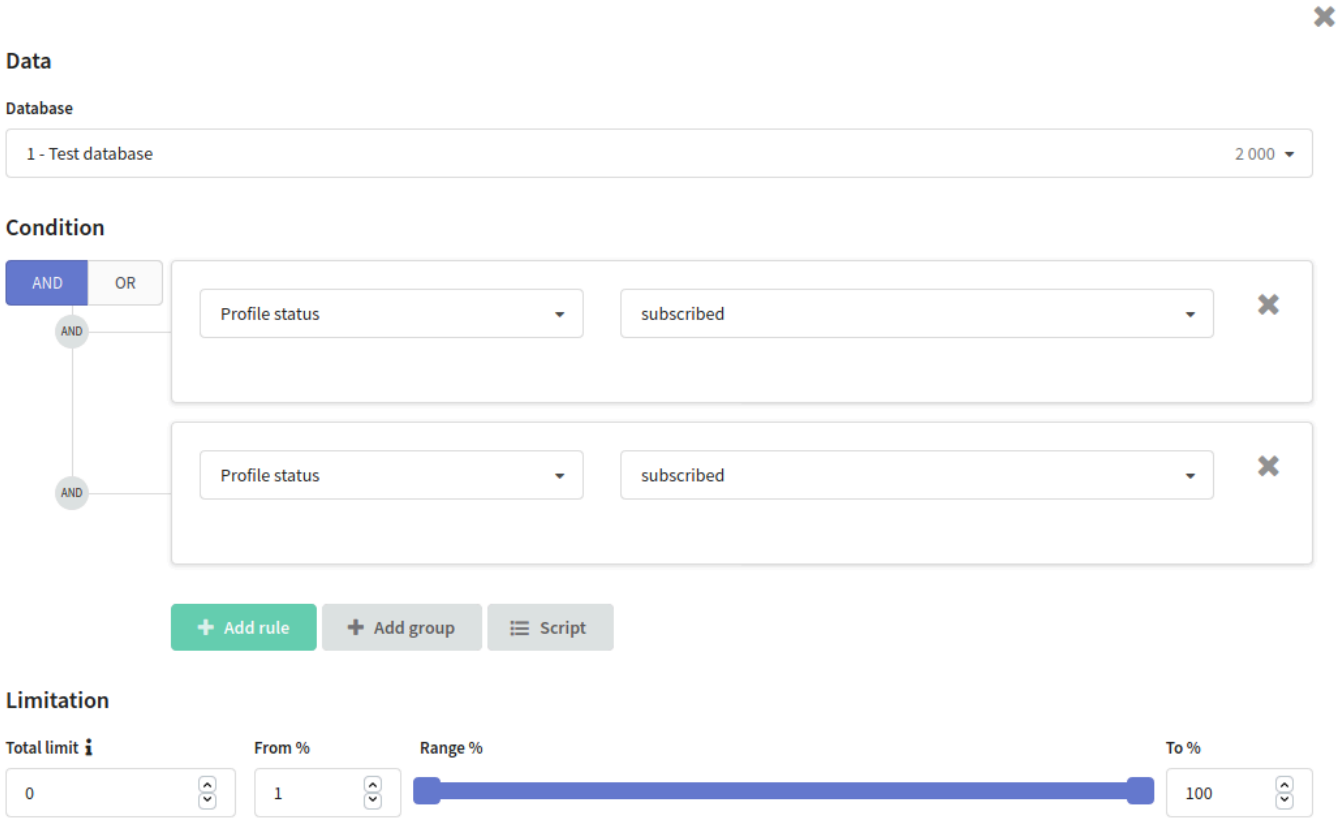
To open the bulk update tool, select Data —> Profile databases or Data —> Segments in the main menu. In the context menu of the profile group, select Update profiles:



Selecting customer profiles

Select the database where profiles that need to be updated are stored. If you want to update only a part of the profiles in the database, add specifying segmentation rules.

You can also add multiple rules and combine them using logical AND (all conditions must be met) and OR (at least one condition must be met).



TIP

In this section of the documentation, you can find information on segmenting customer profiles based on personal data, subscription data, and action history.

You can also set a limit on the resulting sample using two options that can be applied simultaneously — the maximum profile count limit and the range.

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can be included. If more people match the segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 profiles will be included in the segment. This restriction applies to the sample result of the segment. If a range is also used, the range restriction is applied first, and then the profile count limit is applied to the selected portion of the database.

The range determines which portion of the entire database will participate in segmentation. By default, it is set to 100%, meaning the conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in the database is assigned a numeric value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls within the selected range. This allows you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% and a range from 80 to 100% will never intersect.

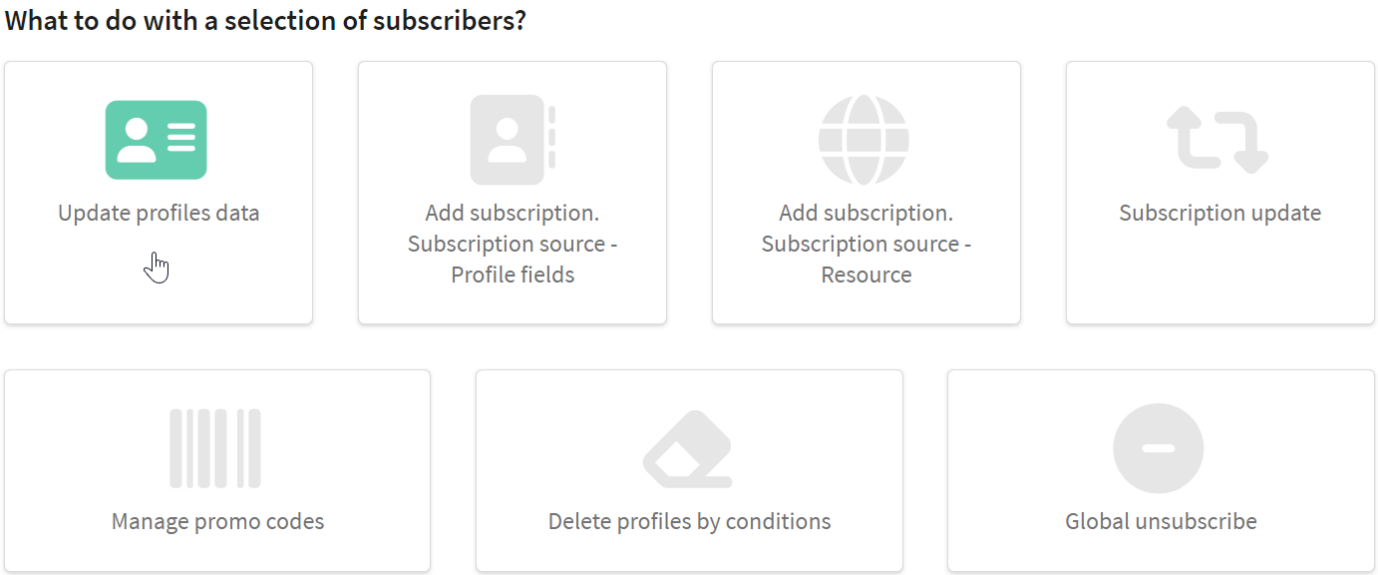
When new users are added to the database, they are distributed according to the same rules: if a 20% range is selected, a new profile has the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the database, and then the limit takes effect within that portion.

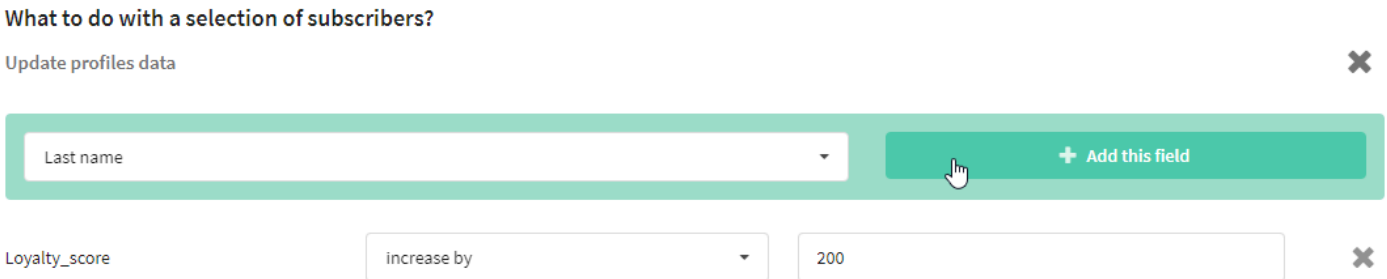
Customer profiles update

Updating profiles data

Select the option Update profiles data:



All database fields are available in the dropdown list. Choose the field you want to update and click + Add this field. Then select the update option: change or delete the value. If you want to update, choose a new value depending on the field type:



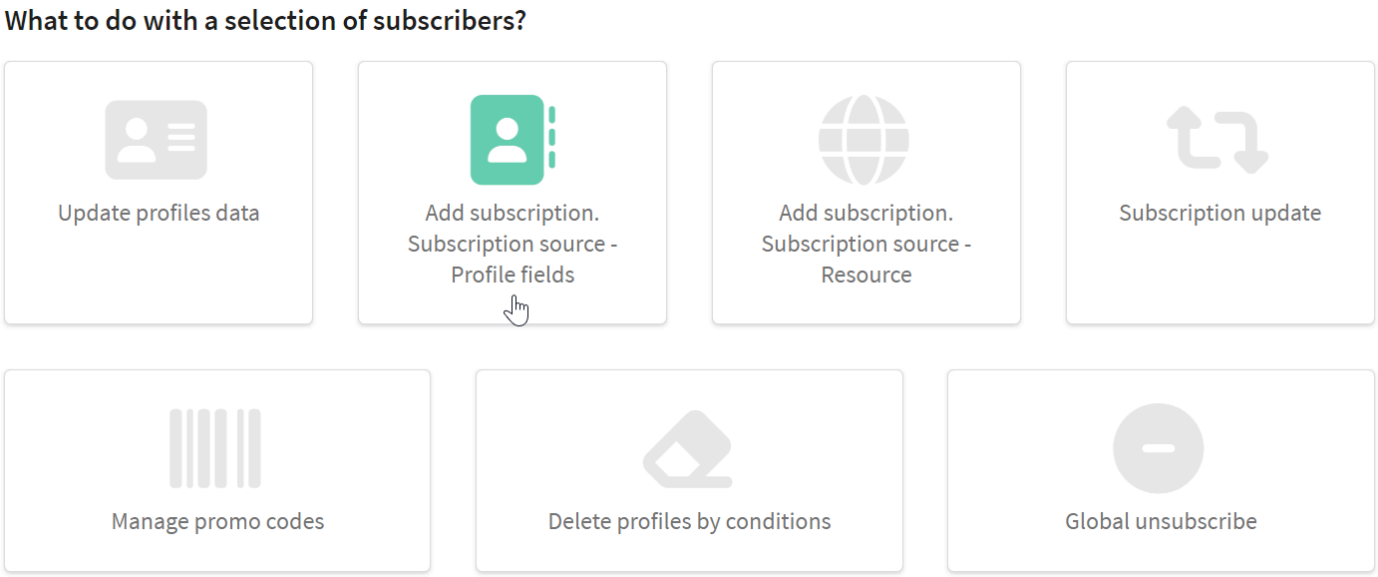
Adding a subscription to a new resource

Depending on the source of your contact information, there are two ways to add a subscription:

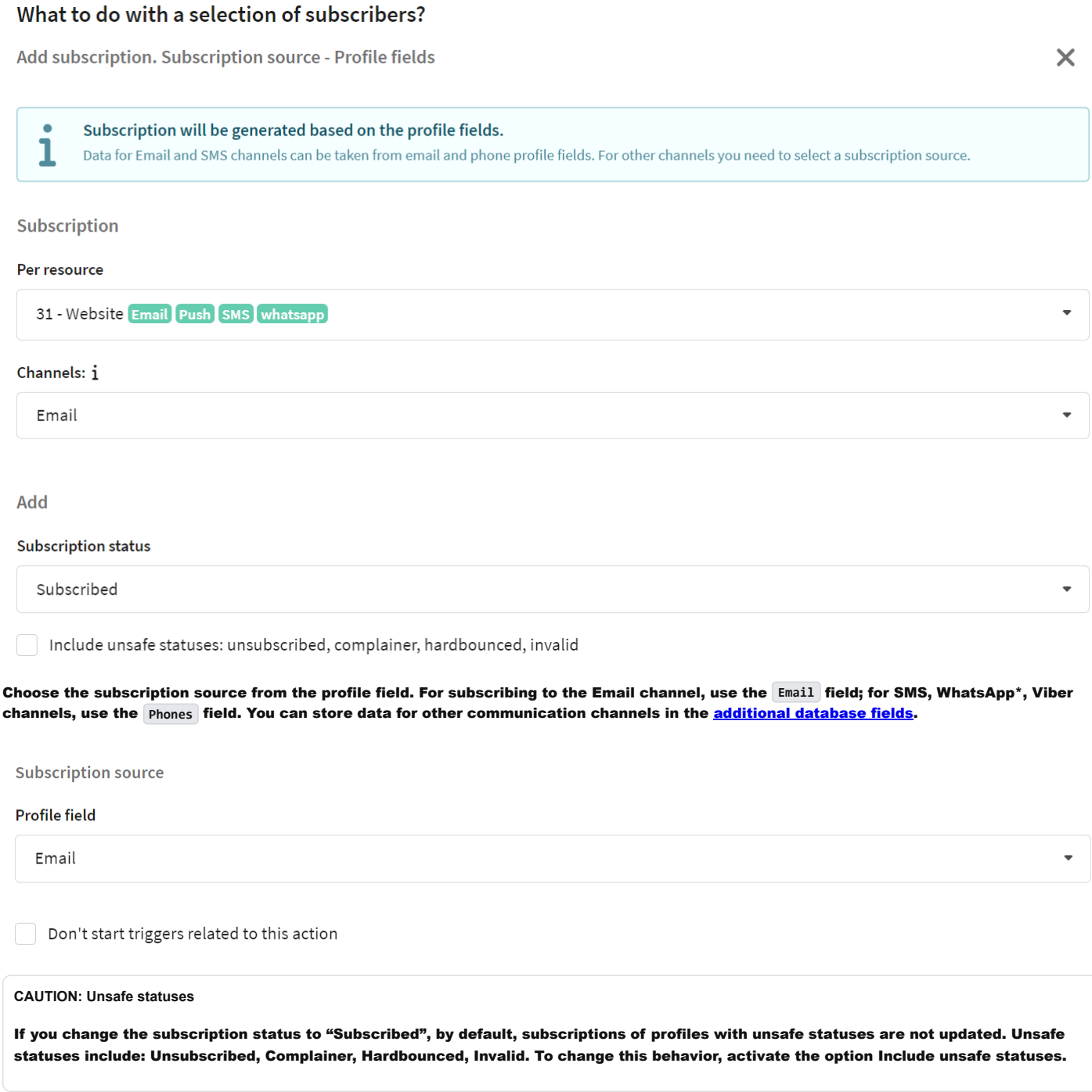
- **subscription source — profile fields**
- **subscription source — resource**

Subscription source — profile fields

Subscription is created based on contact details (phone number, email address, etc.) provided in the profile fields.



Select the resource and channel to which you want to add the subscription. The resource must be attached to the database. Then specify the subscription status.

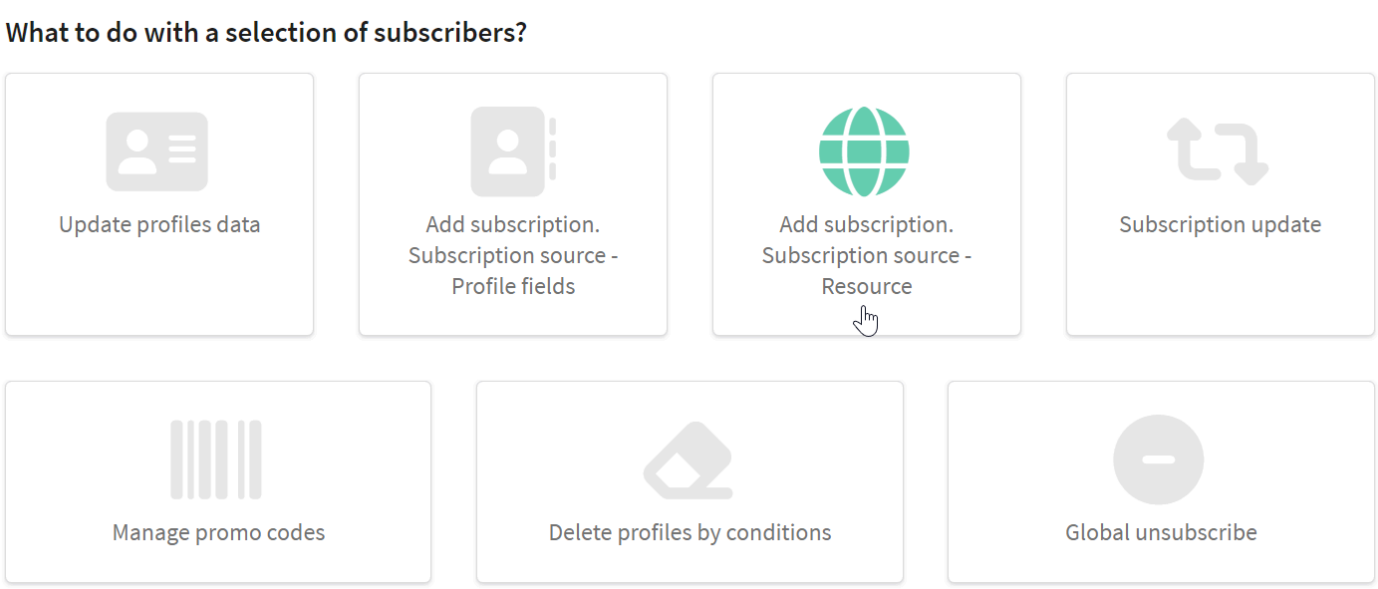


CAUTION: Unsafe statuses

If you change the subscription status to “Subscribed”, by default, subscriptions of profiles with unsafe statuses are not updated. Unsafe statuses include: Unsubscribed, Complainer, Hardbounced, Invalid. To change this behavior, activate the option Include unsafe statuses.

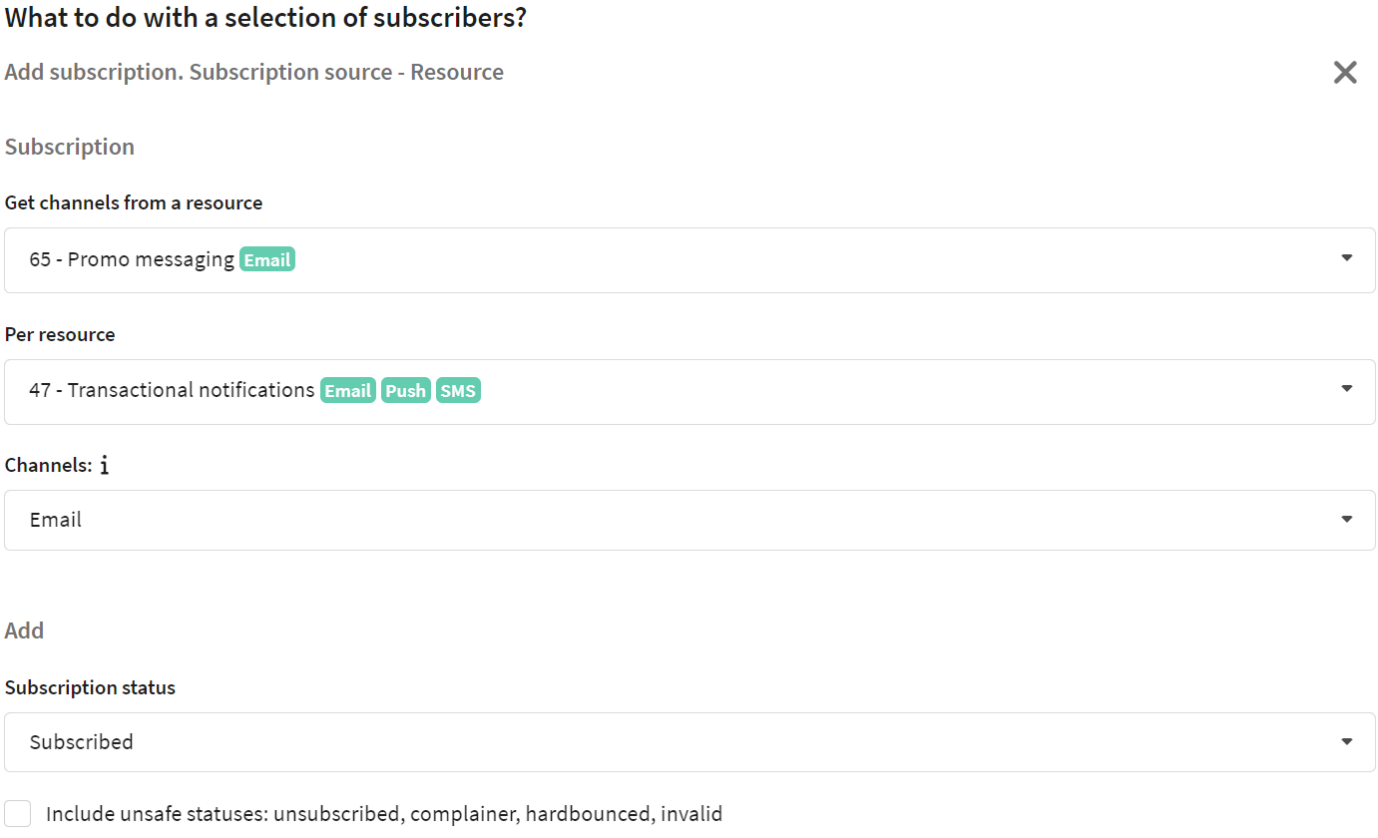
Subscription source — resource

Subscription is created based on the contact data used in the subscription to another resource.



You need to onfigure the following fields:

- **Get channels from a resource. The platform updates contact data using the subscription from the selected resource. If a profile lacks a subscription to this resource, the platform won't add a new subscription.**
- **Per resource. The platform adds subscriptions to the selected resource for profiles.**
- **Channels. The platform adds subscriptions to the selected channel in the resource. Note that this channel must be configured in both resources.**
- **Subscription status. The platform adds a subscription with the selected status.**



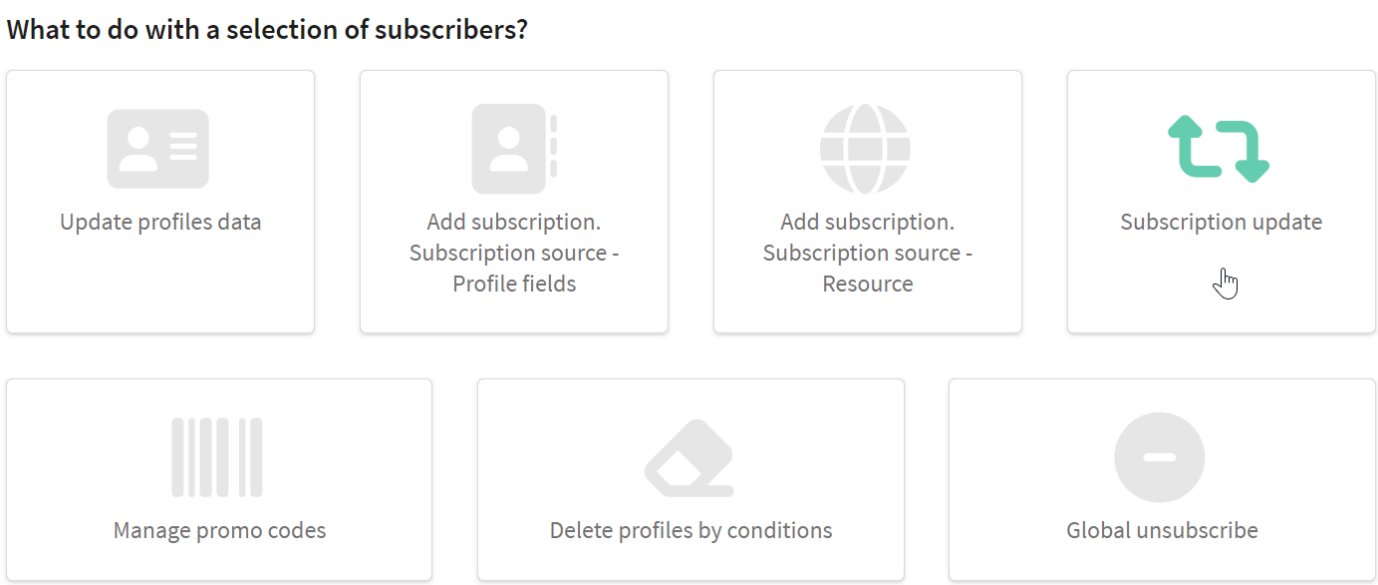
CAUTION: Unsafe statuses

Changing the subscription status to "Subscribed" won't update subscriptions of profiles with unsafe statuses by default. Unsafe statuses include: Unsubscribed, Complainer, Hardbounced, Invalid.

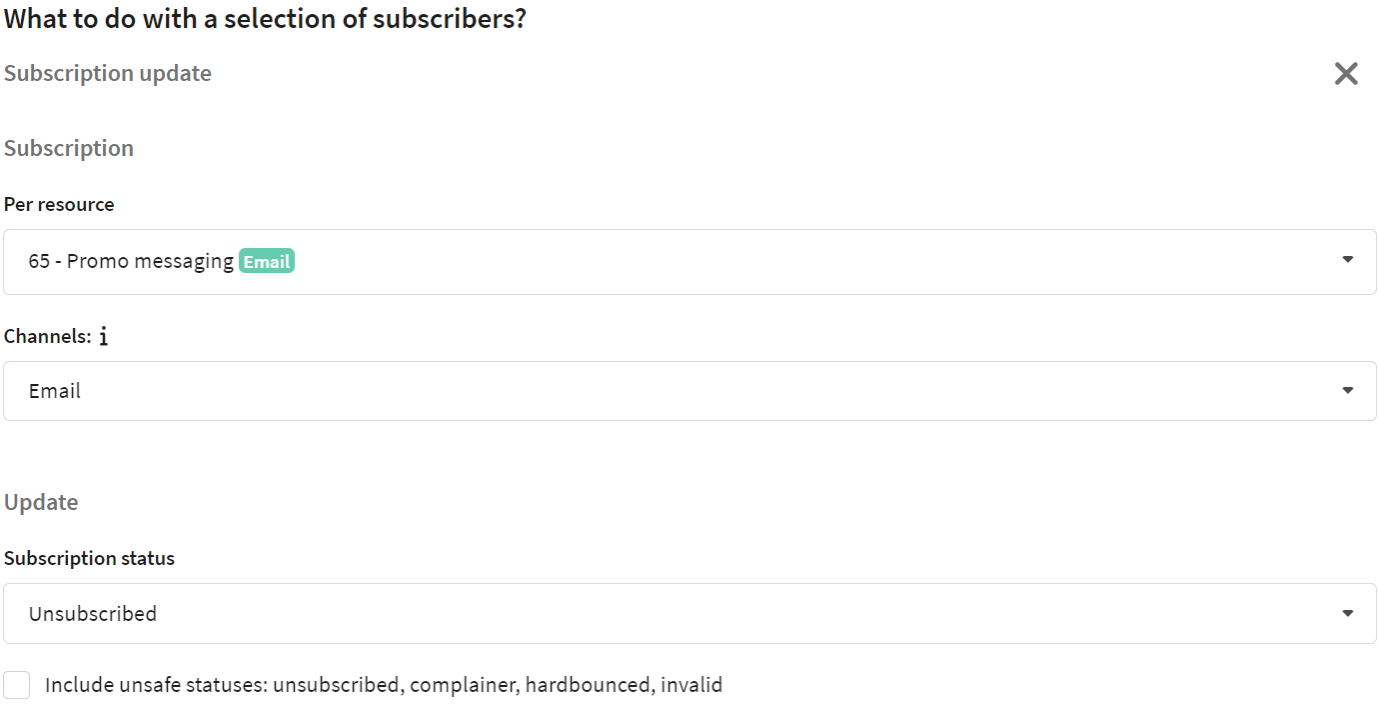
Additionally, the subscription will not be updated if you're using data from a subscription with an unsafe status. To change this behavior, activate the option Include unsafe statuses.

Updating subscription

Updates the status of existing subscriptions to the resource. This feature is handy if you want to unsubscribe all inactive clients at once.



Choose the resource and channel, then specify the status:

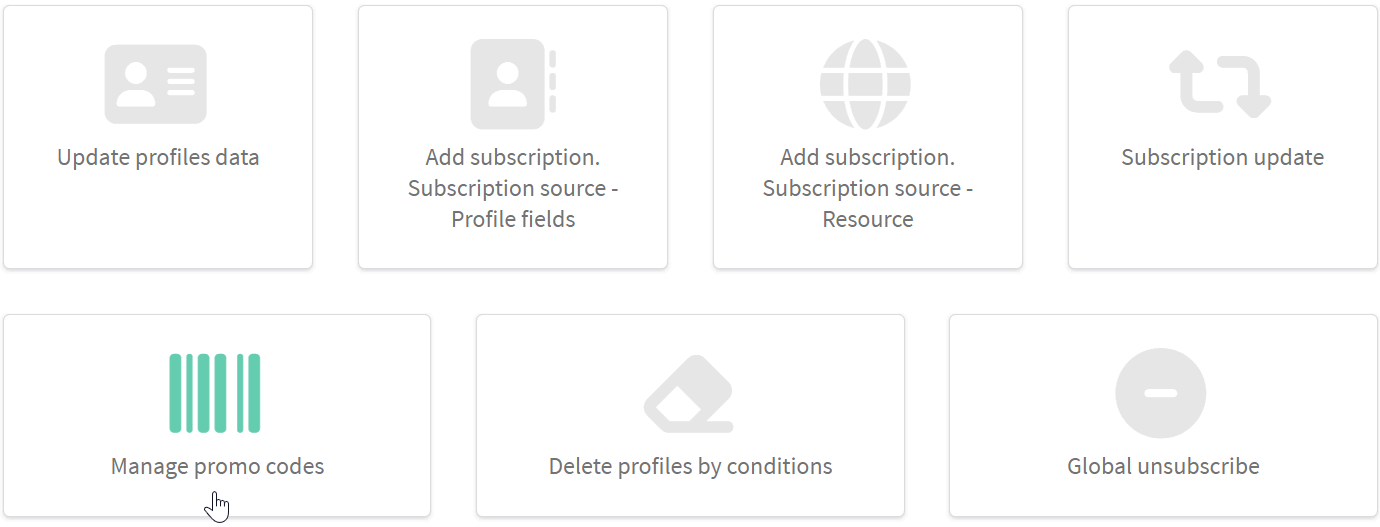


CAUTION: Unsafe statuses

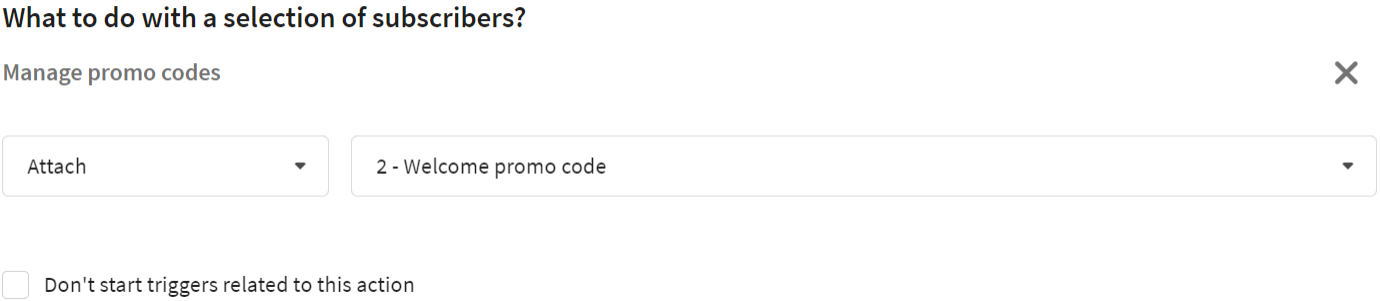
If you change the subscription status to "Subscribed", by default, subscriptions of profiles with unsafe statuses are not updated. Unsafe statuses include: Unsubscribed, Complainer, Hardbounced, Invalid. To change this behavior, activate the option Include unsafe statuses.

Managing promotion codes

This option allows you to attach or detach specific loyalty program promo codes to/from profiles.

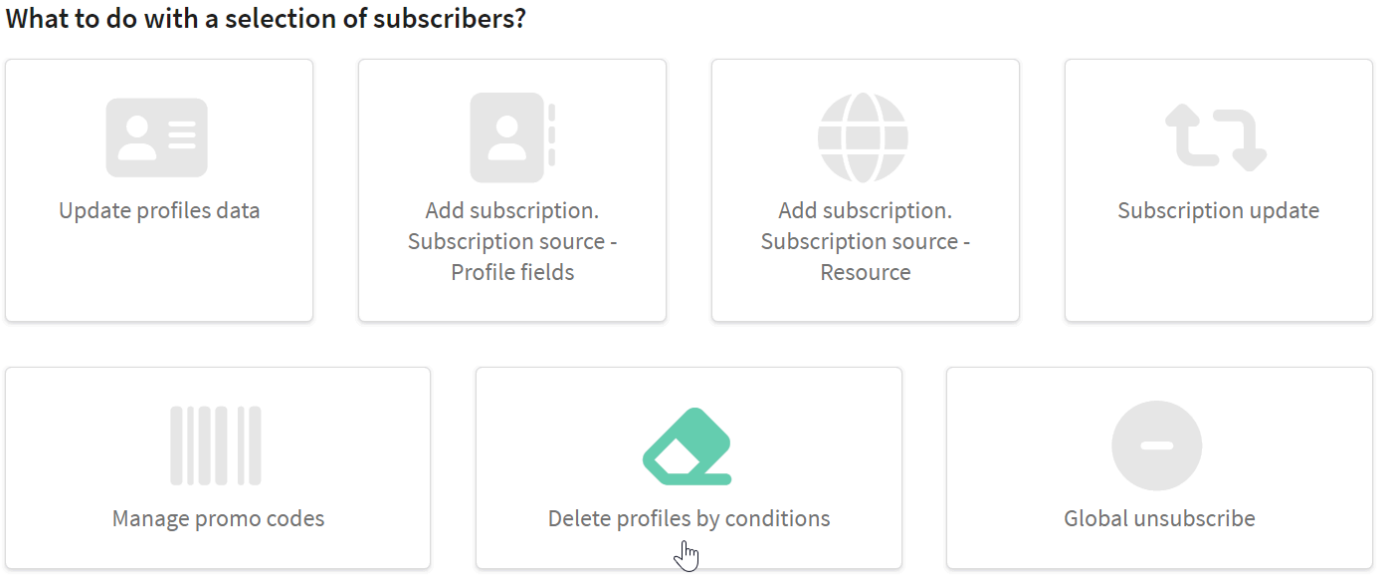


Select the action and the loyalty program:

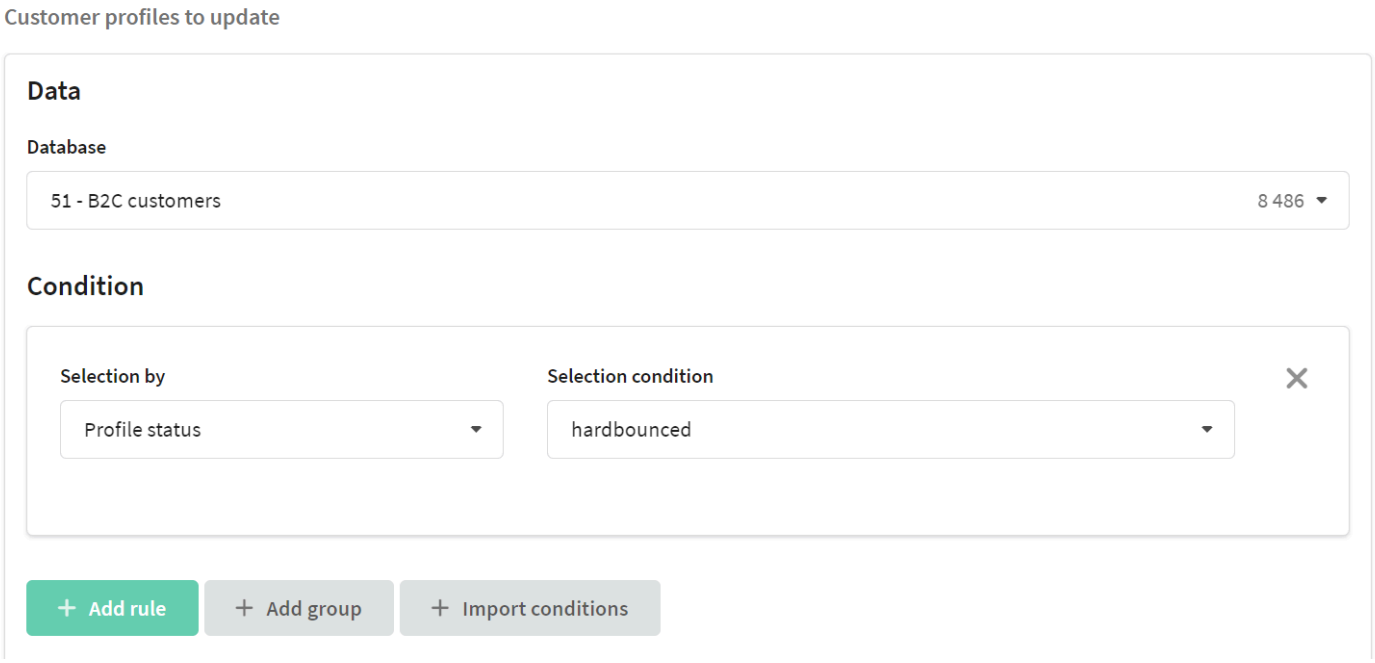


Deleting profiles by conditions

This option allows you to remove profiles from the database that match the specified conditions. The statistics of actions of deleted profiles is saved in reports.



Using segmentation rules, select the profiles you want to delete, then click the “Update” button in the top right corner:



The platform does not delete all selected profiles, but only those that had no interactions during a specific period.

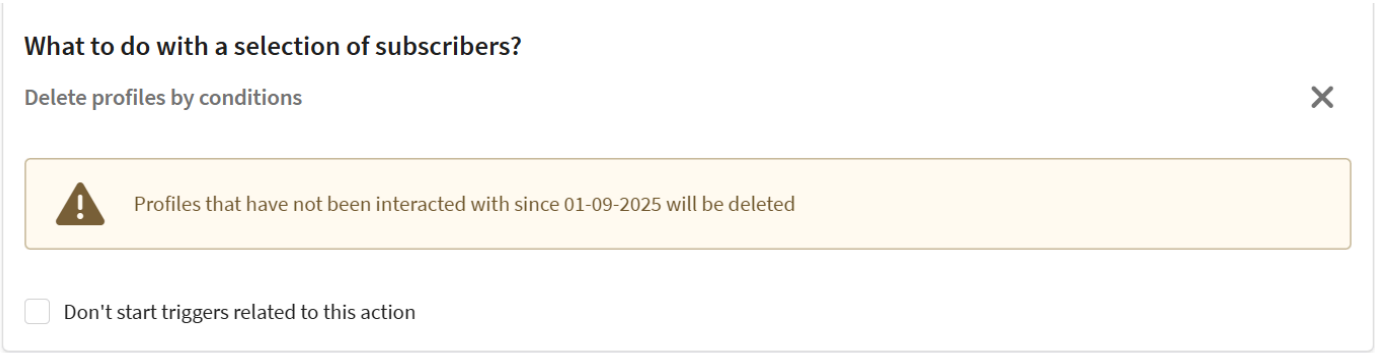
In the Cloud version, this period is fixed and equals one month before the start of the current month. For example:

- **If today is October 1, 2025, the system will delete profiles that had no interactions since September 1, 2025.**
- **If today is October 31, 2025, the same profiles will be deleted — the period does not change within the month.**

TIP

In the On-premises version, this period is configured in the system settings file using the parameter `PROFILES_REMOVAL_MIN_MONTHS`. Learn more.

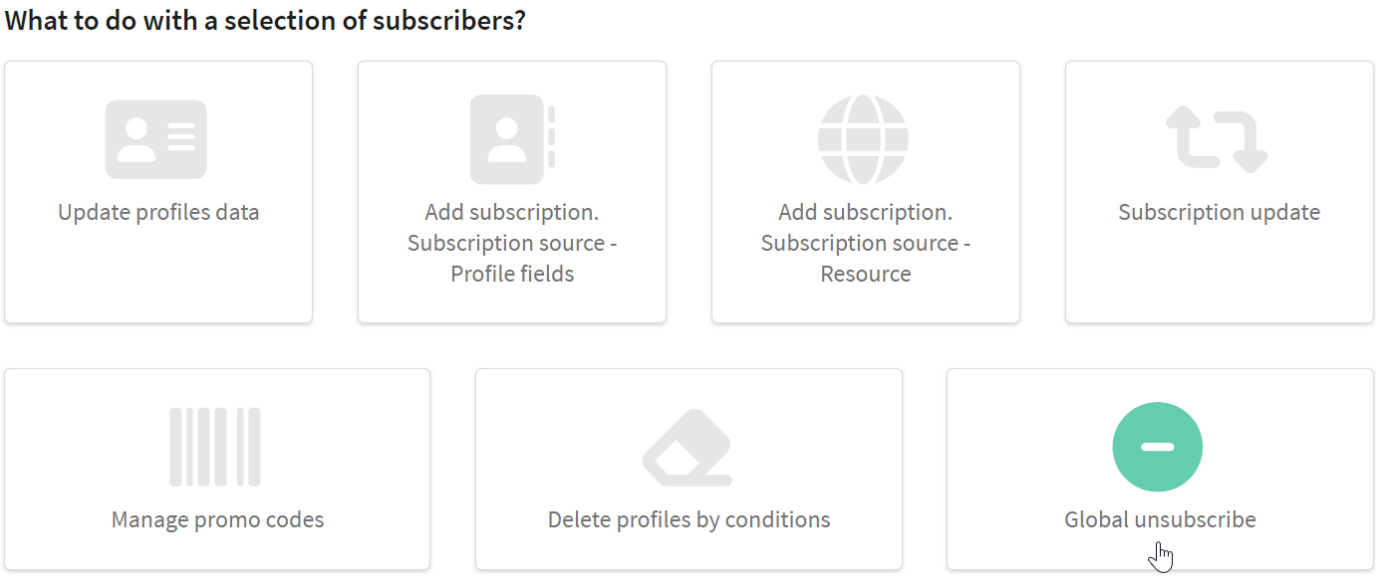
An interaction means any action recorded in the profile's activity history. Information about the current period is displayed in the interface:



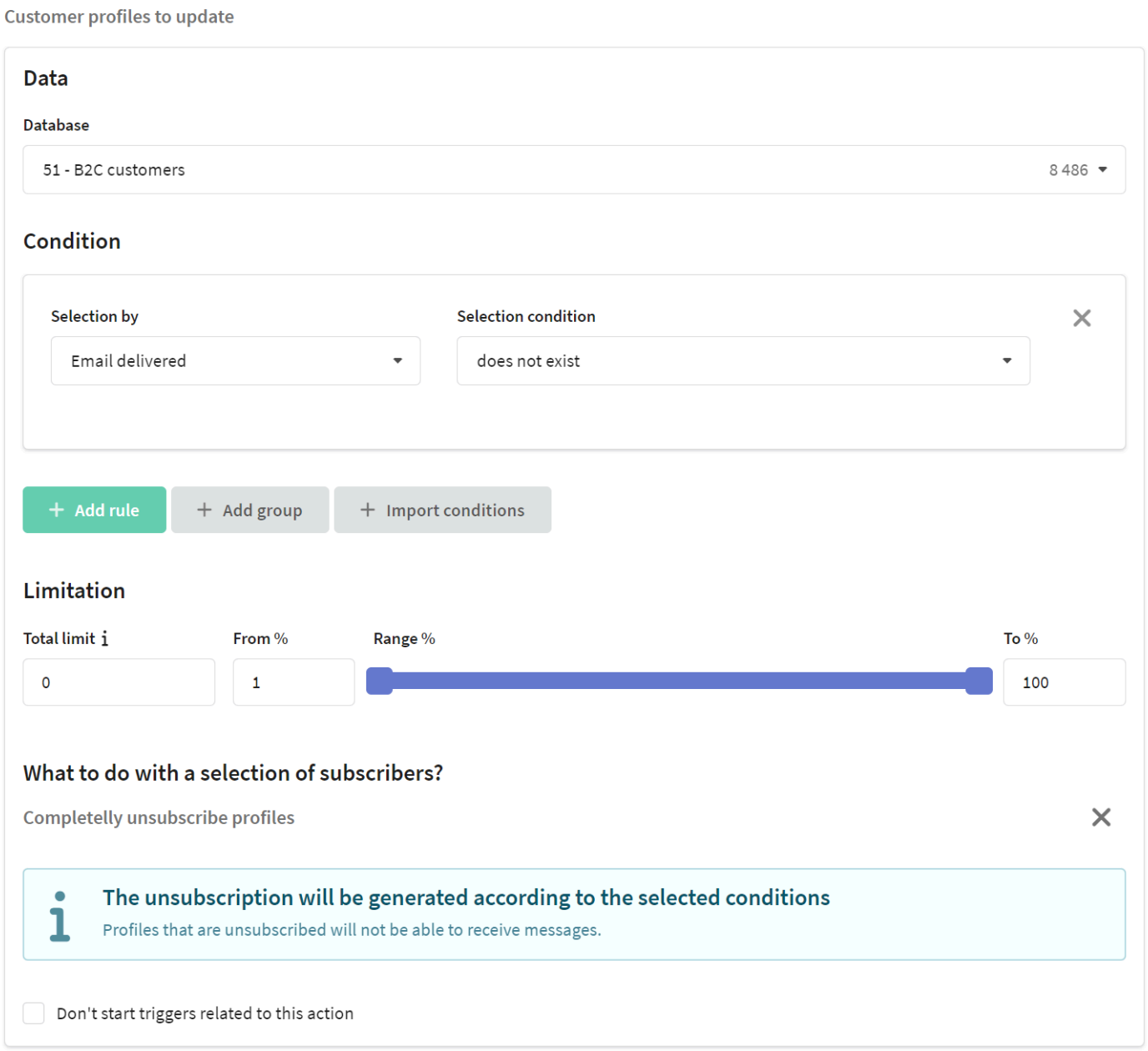
After launching the update, you can check the Tasks section to see how many profiles have been deleted. If a profile matches the selection conditions but was not deleted, it means that some interactions with it were recorded during the specified period.

Global unsubscribe

This option sets the "Unsubscribed" global status for all profiles, completely stoping communication via any channel. Even if a profile has active subscriptions to resources, mailings won't send messages to them.



Using segmentation rules, select the profiles you want to unsubscribe, then click the “Update” button in the top right corner:



In the platform, trigger mailings or scenarios can be configured to activate upon subscription addition or update, profile field update, or promo code attachment. If you don’t want these triggers to activate after the current update, enable the option Don’t start triggers related to this

action.

****The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.***

Double opt-in subscription

Overview

Double opt-in (DOI) is a form of mass communication subscription, organized in two steps:

1. **Subscribers provide their contact data and an agreement to receive content.**
2. **Subscribers confirm their agreement, clicking a special message hyperlink**

Double opt-in is applied over Single opt-in (SOI) when several legal or loyalty factors matter:

- **ISP's sender reputation**
- **Compliance with messaging legal regulations**
- **Higher marketing mailing ROI**
- **Clients' loyalty and lifetime value**

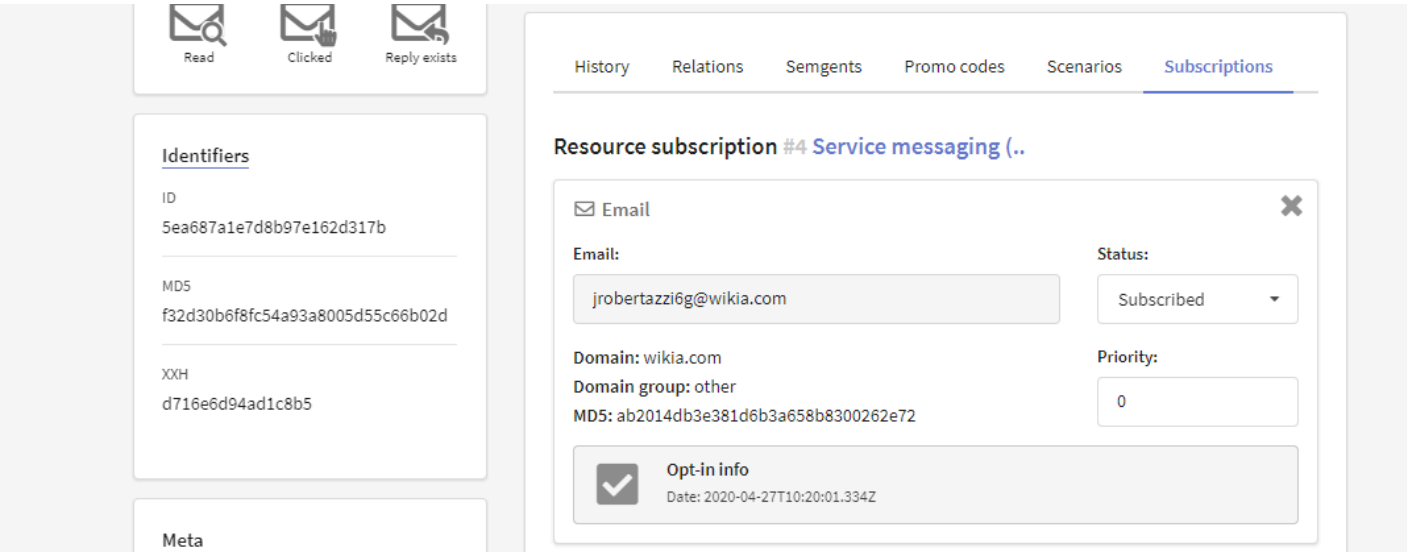
> CAUTION
>
> **Double opt-in is available only for subscriptions to resources.**

Step 1: Subscription

When a client gets subscribed to a Resource, the Subscription status should be set to Unconfirmed.

Adding an Unconfirmed subscription to a single profile

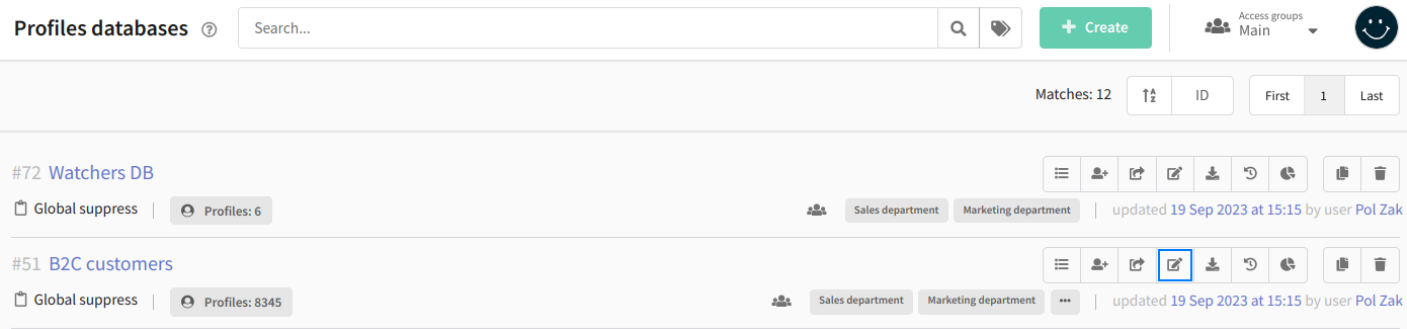
Open a client's profile in a Database and go to Subscriptions tab. Select a Resource and its Email channel. Enter a subscription email address or copy it from profile default fields. Select "Unconfirmed" in Status menu and save your changes.



After you save the subscription it obtains an md5 hash identifier and opt-in information block. It may be necessary for subscribers who doubt their subscription legal status. The opt-in info contains date in UTC and the client's IP address.

Adding Unconfirmed subscriptions to multiple profiles

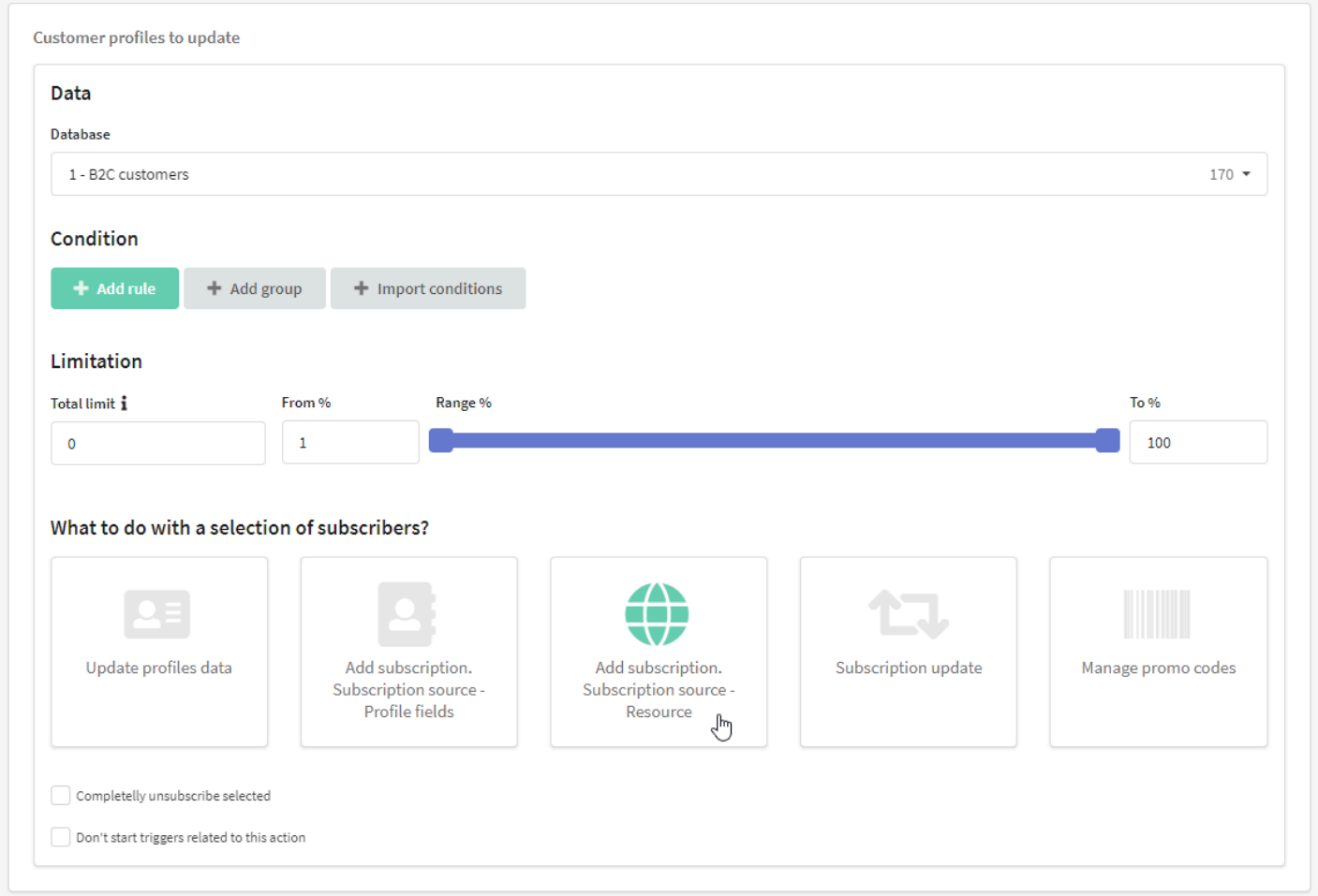
To organize double opt-in process manually for a group of clients, open a list of Databases or Segments and select Update customer profiles from the context dropdown menu on the right side of Database or Segment tile.



On the following screen you can correct a part of audience you want to subscribe. The update screen uses Altcraft MP segmentation rules.

> CAUTION

Profiles bulk update is configured for each Database of a Segment, provided the Segments contains more than one.



In Manage subscription section select Add subscription, specify a resource with Email channel and set Status to “Unconfirmed”. You can choose between profile data and other subscriptions data as contacts source:

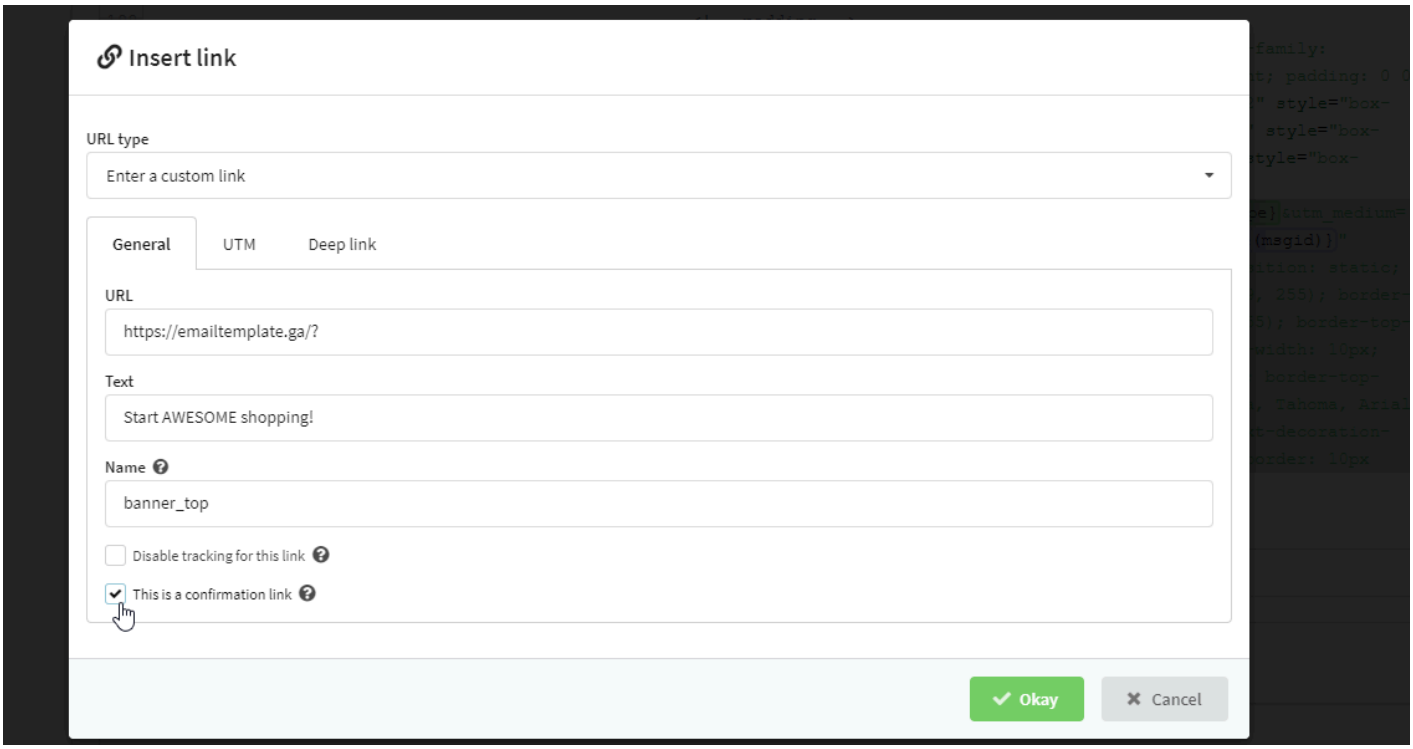
After all necessary settings are configured, click Update button. A Task will be created where you can watch update process.

NOTE

For API profile import and subscription requests the following property should be added to subscription objects `"status": "unconfirmed"`

Step 2: Confirmation

Profiles with unconfirmed subscriptions can receive only messages containing a confirmation link. This can be any hyperlink: in the message template add confirm-link attribute to a hyperlink or select Insert link tool and check “This is a confirmation link” option:



After the template is ready, add it to a mailing, using the same Resource you have subscribed your DOI audience to. Message recipients by clicking the confirmation link will verify their intention to receive messages and thus their subscriptions' status will be changed from Unconfirmed to Subscribed.

NOTE

A click on the confirmation link can be a capturing event for trigger mailings and automation workflows. In trigger or workflow settings set Type: Message action —> Confirmed.

Suppression lists

	Suppression list is a list of email addresses, email domains and phone numbers, which you do not want to send messages to. Suppression list filters an email address or a phone number at moment of mailing launch. Suppression lists are formed to follow CAN-SPAM law rules.

Three types of black lists are used in the platform:

- **Global Platform suppression list — affects mailings in all platform Accounts. If hard bounce is returned for the email address, it goes to global list. Valid addresses, that must be restricted from mailing, are stored separately.**
- **Global Account suppression list — it is created automatically in every account of the platform. Added contacts will be blocked from all Account mailings.**
- **User suppression lists — they are created by users in platform accounts. They can be attached to Profile databases or to Messages Templates.**

Several suppression lists with certain rules are considered, because segment (which can contain several databases with suppresion lists) and several messages can be used in the mailing.

Global suppression list of the platform

Global suppression list of the platform affects all platform Accounts. In consists of two parts:

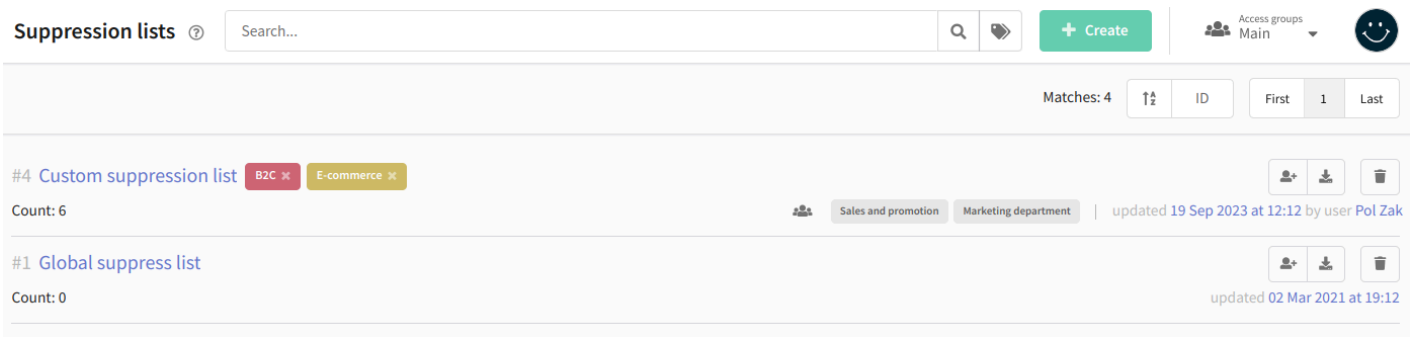
- **Invalid emails list — all emails, for which hard bounce (address does not exist) was received from email provider, go to this list.**

> CAUTION
>
> **Take your notice, that when such answer has been received, Subscription status of the customers changes to hard bounce. These subscriptions will not be used in the mailings, because sending to non-existing addresses makes sender reputation with mail service worse.**
>
> **If you change status of these subscriptions to Subscribed — they will stay in global lost and will not be processed. To resubscribe these customers, you need to remove them from global suppression list of the platform.**

- **Global suppression list — any addresses or any domains for all Accounts of the platform. Platform Administrator can add data in this list, using Administrative panel.**

Account suppression lists

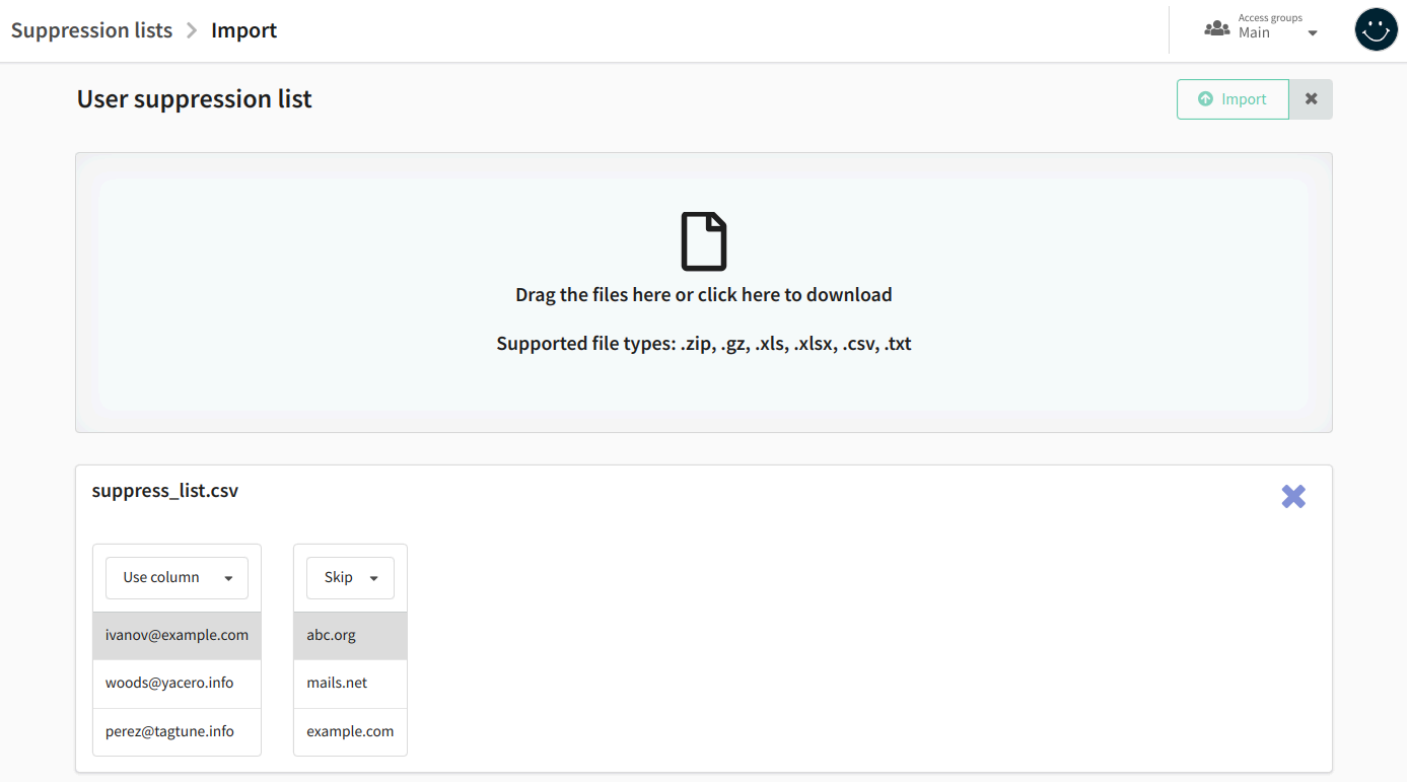
To access Account suppression lists management, choose Data —> Suppression lists in the main menu. This will open a menu displaying the existing suppression lists for your account. From there, you can use the quick actions menu to begin working with the desired suppression list immediately:



File import

Choose appropriate point

in quick actions menu to add contacts from file to suppression list. Drag your file with contacts list in CSV or XLS format into the form opened. You can upload several files, and there can be several columns in these files. Next, you can choose, which column you are going to use for the import to suppression list. After that, click Import:



Manual import

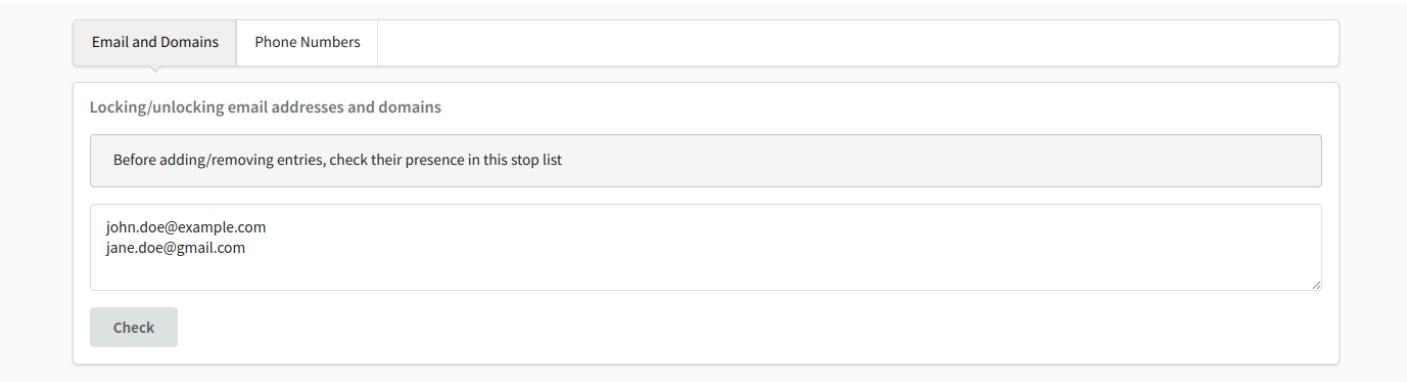
If you want to add customer contacts manually, open the desired suppression list. You can add customer email addresses, email domains, and phone numbers to the blocklist. Depending on the contact type, open one of the tabs and enter the contact details you want to add to the list in the Lock/Unlock section.

TIP

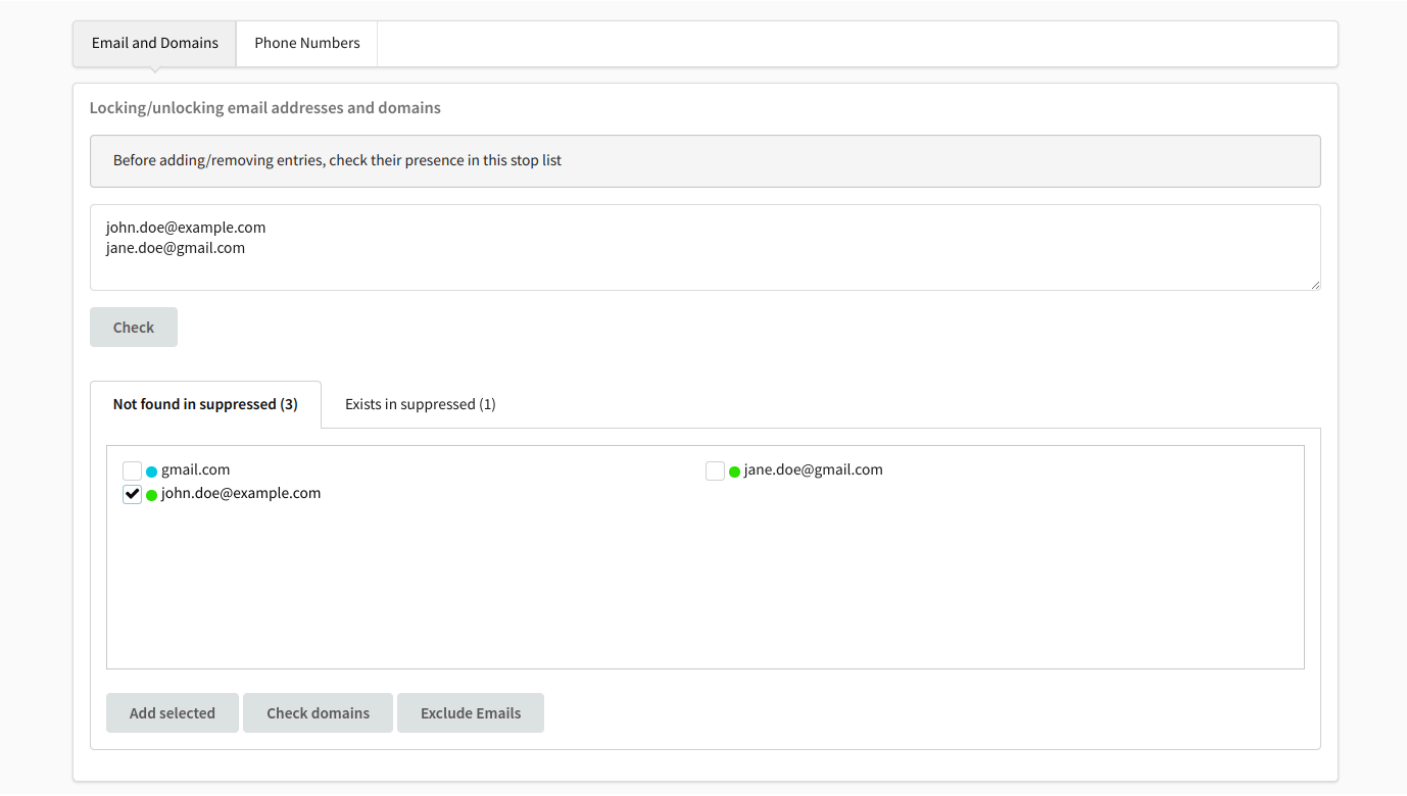
Multiple entries should be separated by commas, spaces, or placed on new lines. You can specify up to 500 email addresses, domains, or phone numbers.

Phone numbers should be in international format (e.g., +79123456789).

Then click Check:



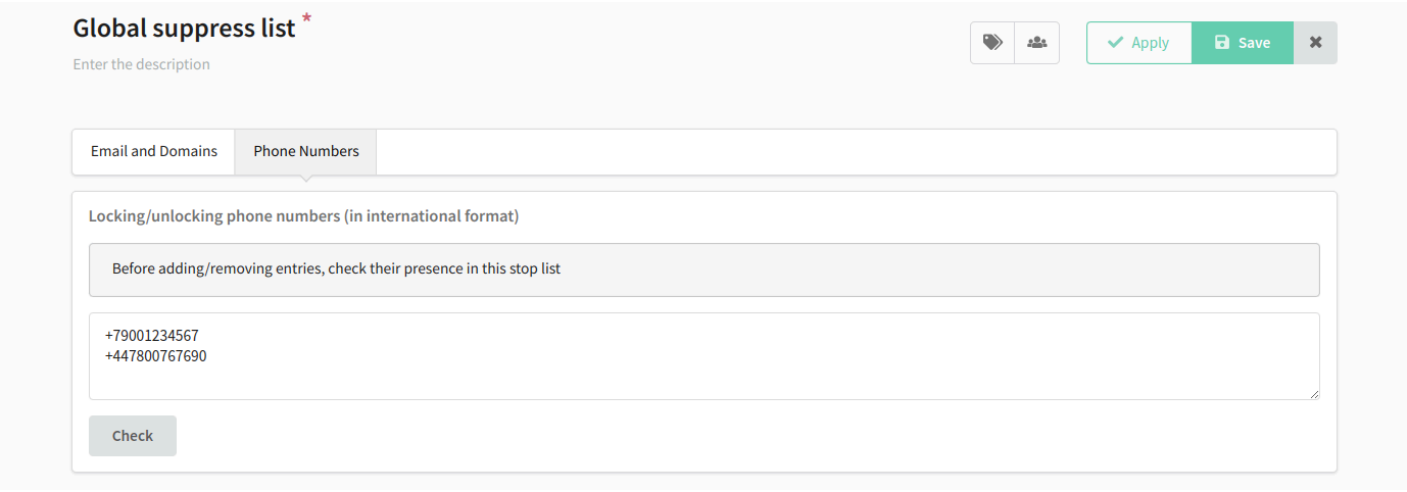
Two tabs will appear: Not found in suppressed and Exists in suppressed. If a domain, email address, or phone number is not in the suppression list (it will appear in Not found in suppressed), you can select it and add it (Add selected). If it is in the suppression list, you can remove it (Exclude). Additionally, you have the option to select all email addresses, all domains, or all phone numbers in tabs (Check):



Global suppression list

In every Altcraft account, one global suppression list is created automatically, which works always - for all mailings within the Account. You do not need to attach it manually to Profiles databases and Messages templates.

You can add subscriptions manually or import from the file to the global suppress list:



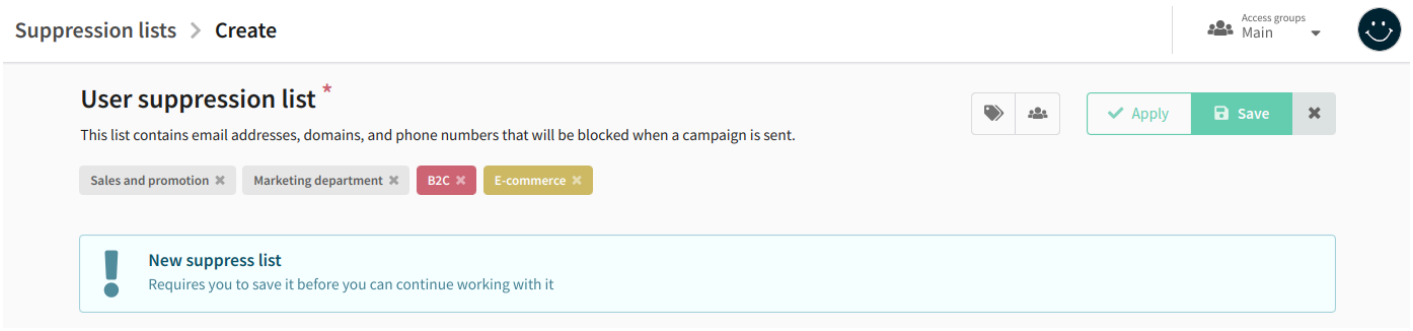
If you want subscriptions to be automatically added into the global suppress list, you need to use a link in the message - URL of global unsubscribe variable `{globalunsubscribe}`. Address of subscribers, who follow this link, will immediately go to global suppress list and will be excluded from all future mailings in the account.

TIP

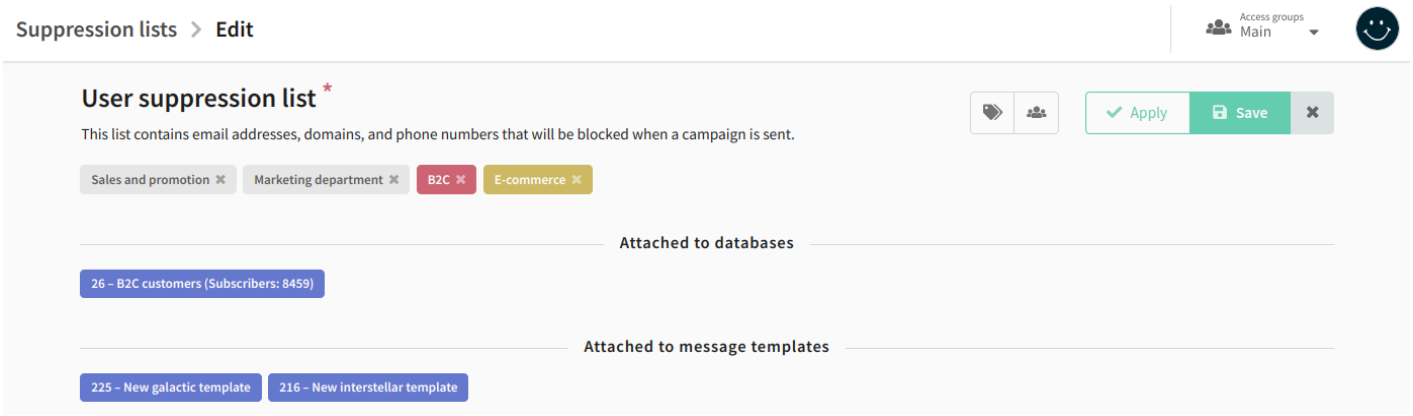
Note that communications will stop only for the email address to which the message with the unsubscribe link was sent. If the profile has a subscription with a different email address, messages will continue to be delivered to it as usual.

User suppression lists

Press + Create in lists menu to add user suppression list into the account. Select Suppression list name, enter Description, set access group and tags for quick search, if it is necessary:

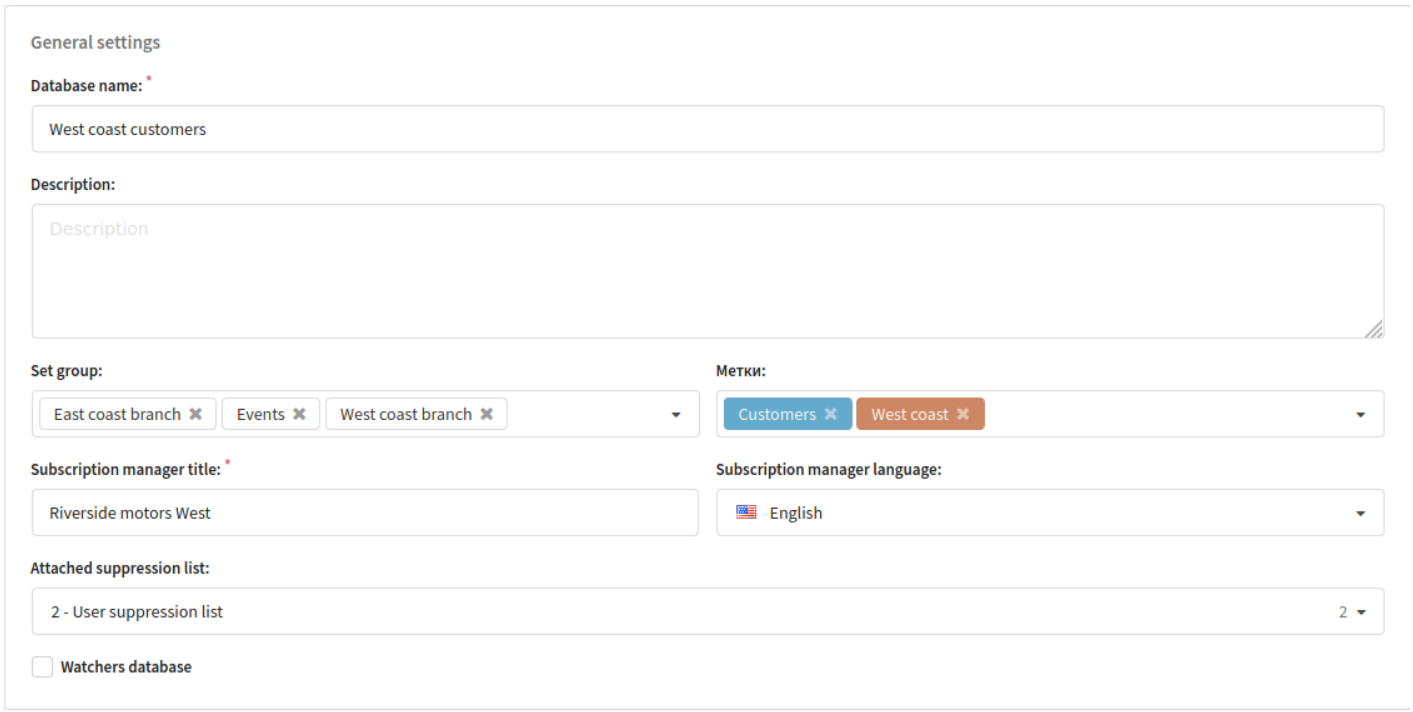


After saving it, you can manually add customer contacts to the created suppression list or import them from a file. Additionally, you can attach the list to profile databases and templates. You also have the option to check at any time which platform objects the list is attached to.



Suppression list for a Profiles database

User suppression lists can be attached to a certain Profiles database of the Account. This list will be active only during orchestrating Mailings for an audience, which includes this profiles database. For this, choose Attached suppression list in General settings of Database editing form:



Suppression list for a Message template

The second option of using custom suppression lists is attachment to certain Messages templates. This will allow you to control, which audience should not receive certain content: all unsubcriptions by template will go this list and therefore, communications will be blocked.

Open the necessary Template, and go to Options tab, and choose necessary suppression list in Extra settings block:

Rules of suppression lists interaction

You can use several suppression lists during mailing. Subscribers are excluded from the mailing in the following way:

1. Global suppression lists are checked, and all subscribers there are excluded from the mailing.

2. Next, suppression lists, attached to Profiles databases, excluded subscribers of their databases from mailing.

INFO

If there are two same subscriptions in two databases, and suppression list, which contains them, is attached only to the first database, then messages from mailing, which contains the segment with this databases, will be completely sent. This happens, because suppression list will act only on the database, that it is attached to, which is the first one.

3. Next, suppression list, which is attached to the Message template, will exclude only subscribes, who would receive this message.

Profile relations

TIP

To attach, detach, edit or modify relations automatically you would need API integration. Check this Section of API documentation.

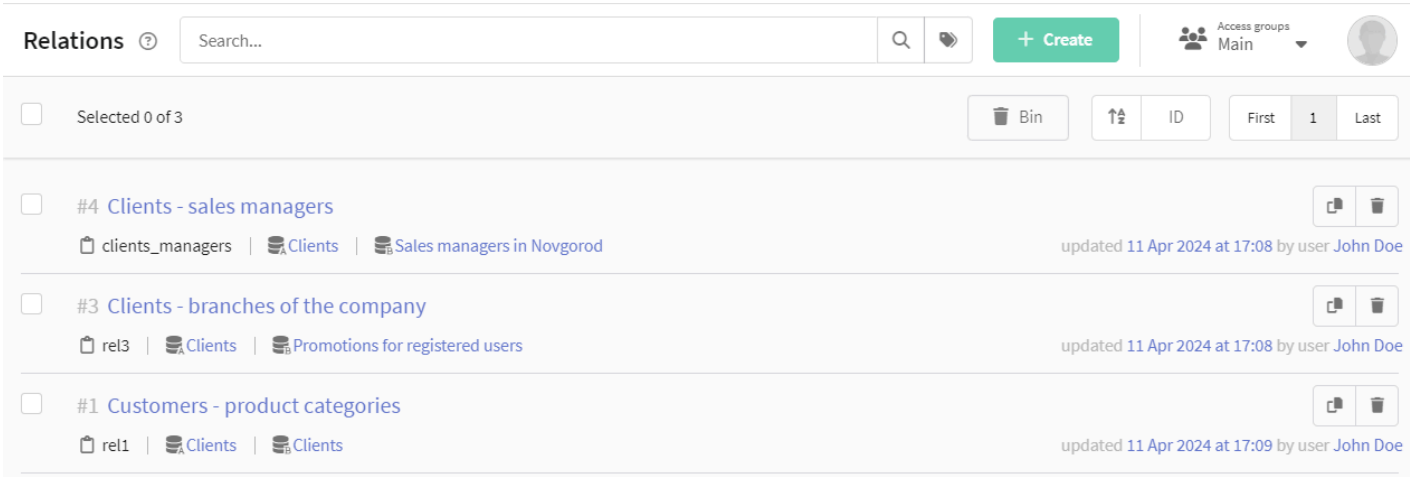
The relations between profiles allow you to implement customer interaction strategies on the platform. You can relation profiles within the same database or relation profiles from different databases. There are two types of communication:

Direct — outgoing. Shows the relationship of the profile to other profiles. For example, the relationship of a referral client with his clients. Feedback is incoming. Shows how other profiles relate to the selected profile. For example, clients of one agent or manager of a company evaluate the quality of his work. Each relationship has properties - relationships between clients. Properties can be in the form of numeric or Boolean (false - true) values.

Relation objects are created in the user interface. You can add, remove, strengthen, or change a relation either manually or using special API requests described in this section of the documentation.

Managing relations

In the main menu search for Relations in the Data section. Here all your existing relations are enlisted, accompanied by created-updated summary and databases involved. Using actions menu on the right side you can quickly clone or delete a relation. To edit an existing relation, click on its name. To add a new record, click + Create at the top of the page:



General settings

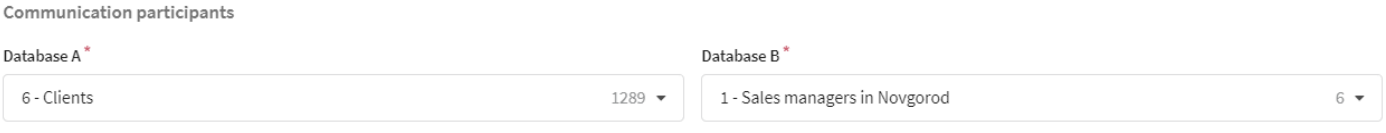
In General settings enter the new relation's name and short name as well. Short name will be used in message template editor and relations API requests.

Below select Databases containing profiles you are going to relate:

- **Database A — contains profiles that are collecting relations. These can be your products, managers, agents — or partners. We can call them objects.**
- **Database B — contains profiles that initiate relations — your or your partners' customers, reviewers or subjects we can call them.**

TIP

If you are using one database relations, select it in both drop-downs!



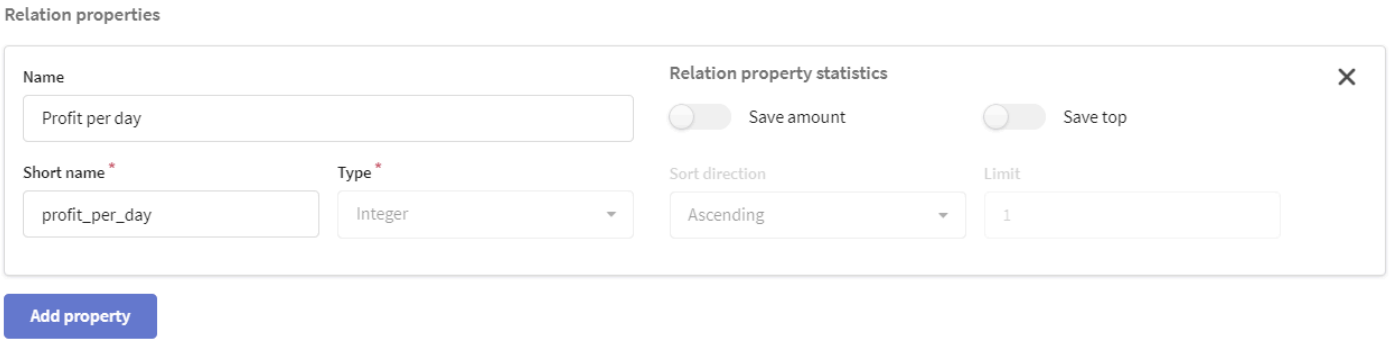
Relation properties

One relation can contain up to 11 properties, describing the value or weight of interaction between profiles. For each new property enter a name and a short name for templates and API.

Property type sets which data will be used to measure relation impact:

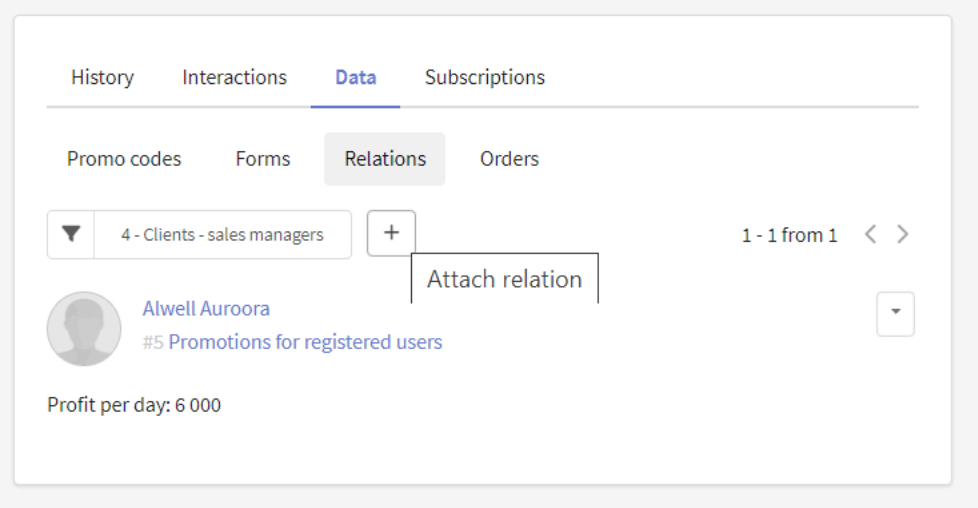
- **Integer — any integer number. Fits well for quality assessment, numbers of purchases, social network profile views and post likes.**
- **Float number — a floating-point number. Monetary values and other more complex numbers can be recorded in this property.**
- **Boolean — a boolean true-false value. Can be used for one-time or on-off actions: subscriptions, recommendations, or availibility.**

For numeric properties you can store statistics: save amount and save top. The sorting direction determines whether to store the highest or lowest values in the top, and the limit (from 1 to 20) sets the number of relations for statistics:



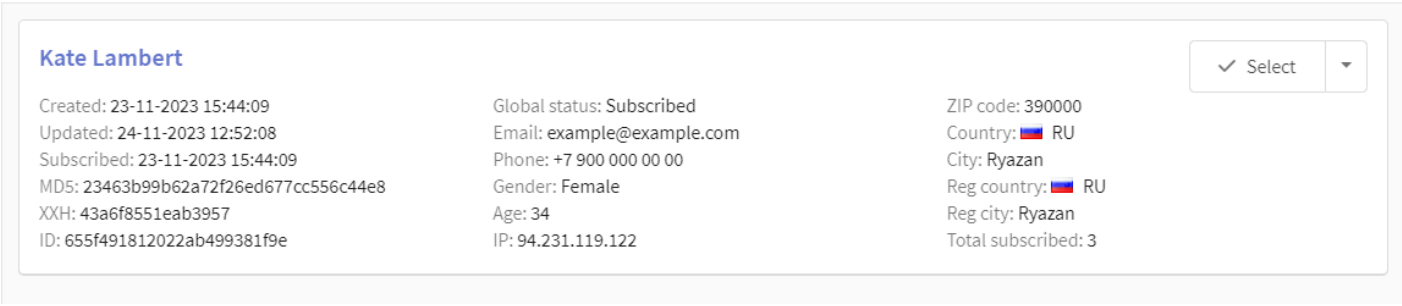
Relating profiles manually

Open the subscriber profile in Database A. On the Data tab, go to the Relation section. The available relations for the profile of the current database are displayed here.

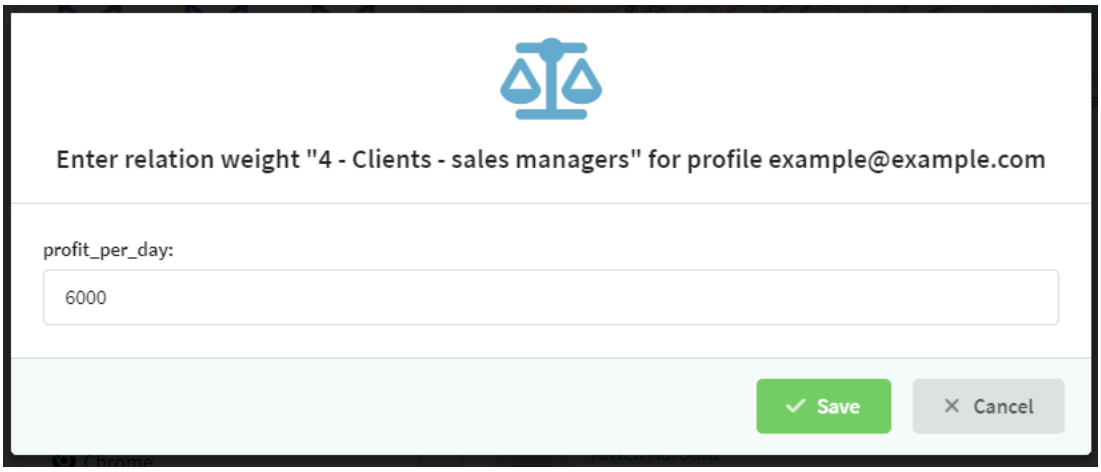


In the window that opens, specify the rule for selecting a profile from the Database B. Standard rules are used, as in the search or segmentation of profiles.

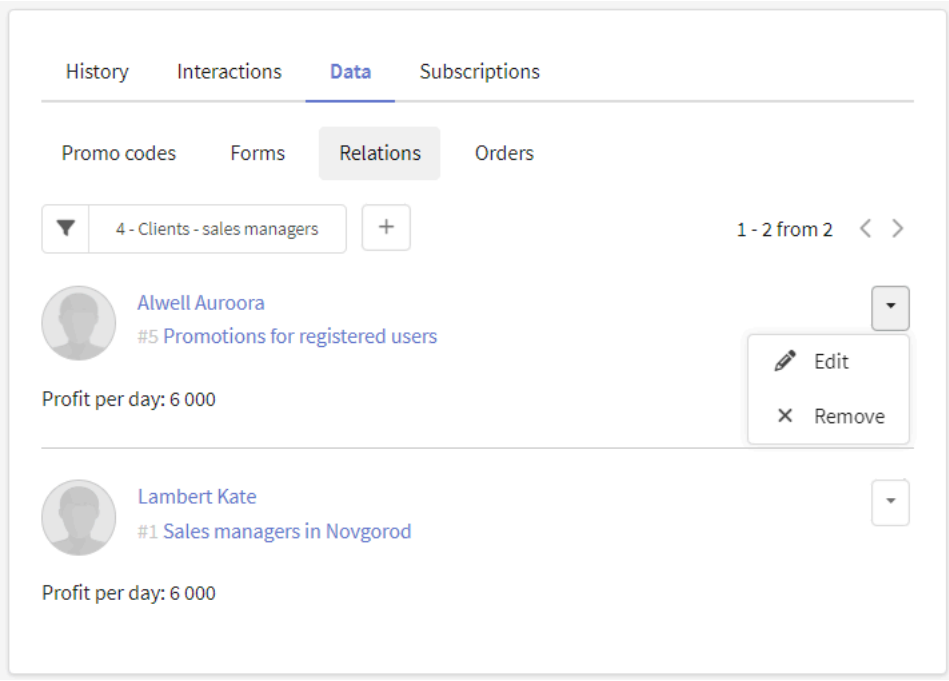
Click **Apply ** and Select:



Then enter relation's property weight:



After adding a relation, a brief summary and a top on the properties of the relation will be available in the profile, as well as a list of related profiles from the Database B. You can change the weight of the relation or delete it using the drop-down menu on the right side of the entry:



What are relations and relation values used on the platform for?

1. Segmentation

They can be used to create a static or dynamic segment. Due to this, the target audience is "separated" from all the clients of one database based on the attitude of its participants to each other.

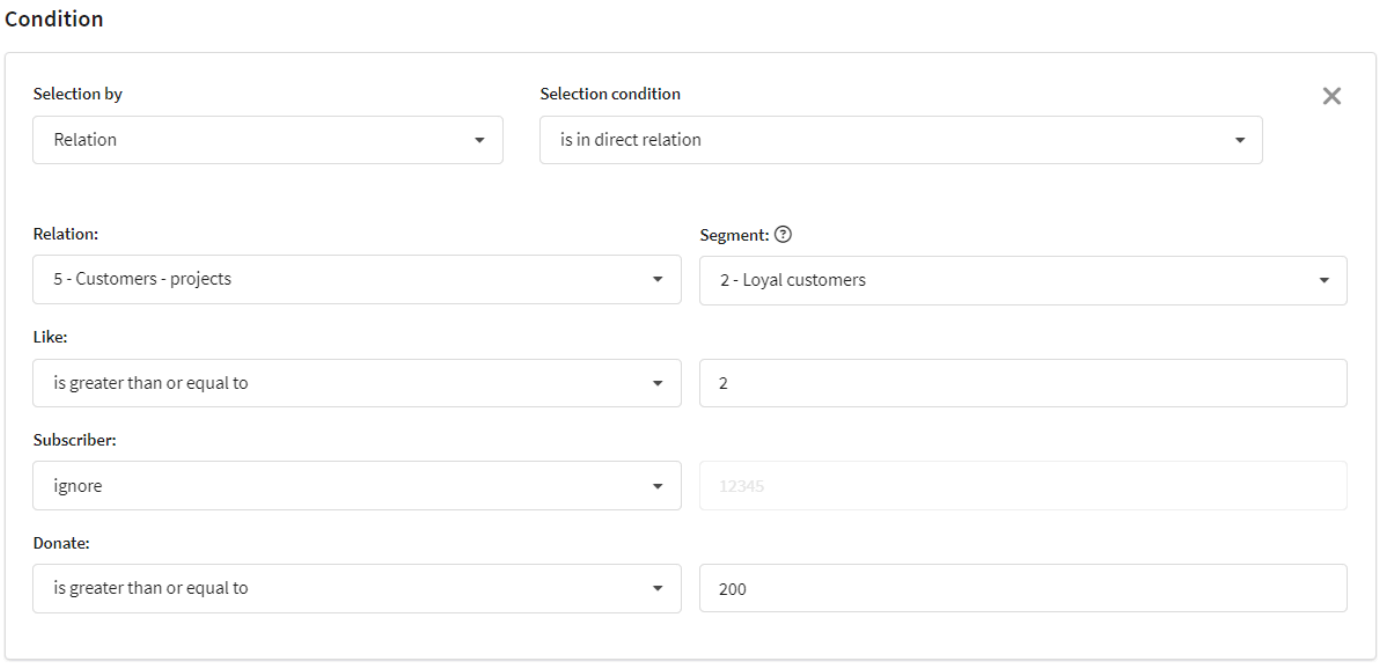
Additionally, you can add checks for certain communication properties:

- **false or true — for Boolean properties**
- **more/less, more/less or equal, equal/not equal — for numeric properties**

In the segmentation rule, select the presence or absence of direct or reverse, and add the necessary parameters.

TIP

For example, you can use the rule below to find customers who approve of your projects and even donate more than 200 currency pieces.



2. Dynamic content in templates

Relation values can be added to message template using special variables.

In the variables menu, select direct or feedback and the desired value for substitution.

> TIP
>
> **For example, for an online training center, you can notify methodologists about how many hours students spent studying with certain instructors, and for clients - provide an up—to-date schedule of the instructors they prefer.**

Relation variables can also be inserted manually using the following editor script:

```
{relation.[short_connection_name].[direct for direct or reverse for reverse].[property name].[count – quantity, total – sum, top – top relations]}
```

The number of direct relations to the profit property:

```
{relation.managers_loyal.direct.profit.count}
```

Top feedbacks with the `profit` property. A loop is used to output list of values.

> TIP
>
> **The loop substitutes the data of those profiles that are specified in the relation, and not from the database through which the mailing is sent. Thus, if you send an email to your agent, it may use personal customer data stored in a separate database.**

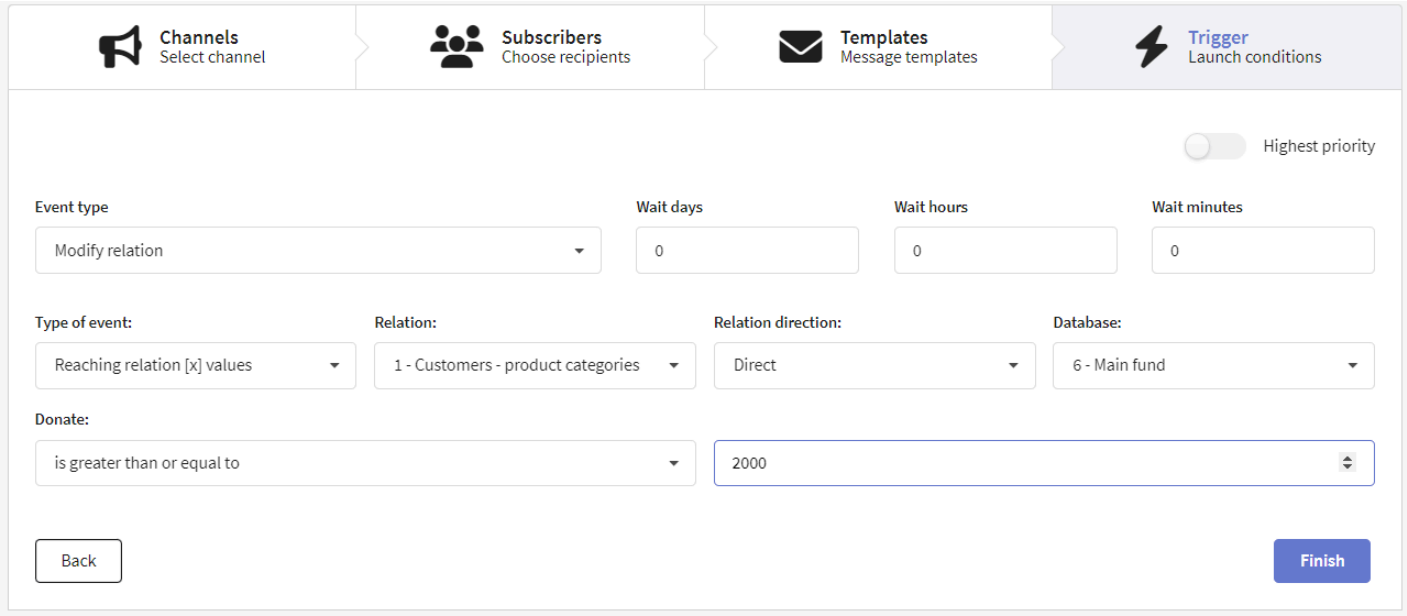
```
{for $index $item = relation.managers_loyal.reverse.profit.top}
{$item.lead._fname} {$item.lead._lname} ({$item.value} profit)
{else} There is no ‘profit’ for you :(
{end}
```

The sum of the value of the `profit` property for direct relation:

```
{relation.managers_loyal.direct.profit.total}
```

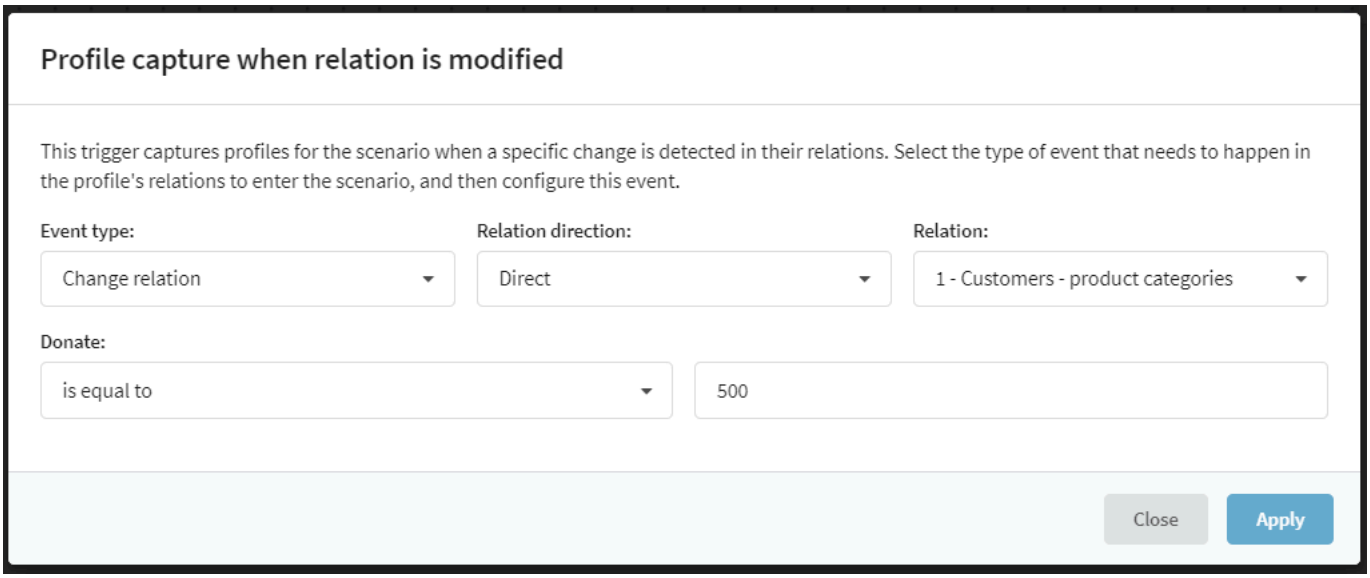
3. Trigger mailing or automation scenario

A trigger mailing can be configured to create or delete a relation:



The trigger can be triggered when both the direct and reverse relation between profiles changes. This may take into account the appearance, change or deletion of the relation, as well as the achievement of the relation property set in the trigger value.

Or connections can be implemented in automation scenario:



TIP

For example, if you have a referral program, set up 2 different automated workflows for both referral and for customers engaged by the referral:

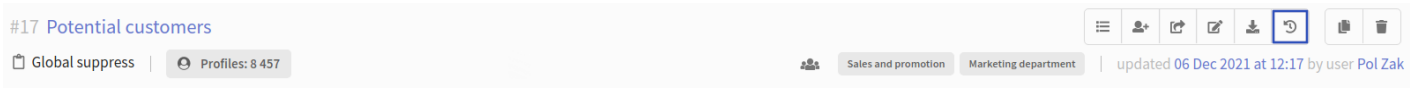
for the referral — a trigger "When a direct relation is attached". It will launch when a referral engages a new customer for you. Use this scenario or trigger to further motivate your referral. You can additionally specify values, like profit of the initial transaction.

for the engaged customers — a trigger for reverse relation. Use it to promote your products and motivate customers to become new referrals.

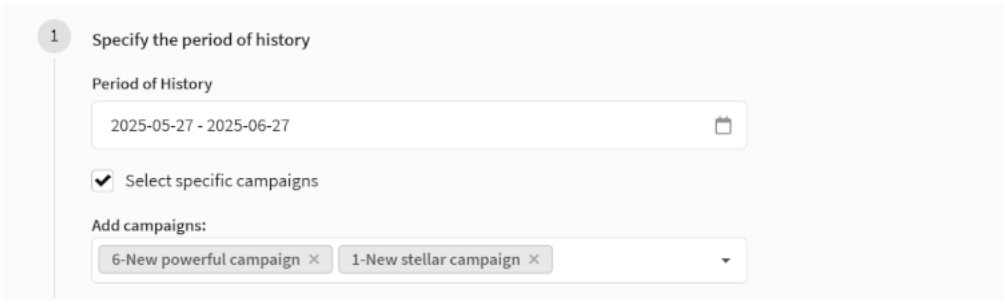
Profile history export

Profile history export allows you to download data on profile activity from the platform. The data is usually in a specific database or a segment. The data will be exported to a CSV file.

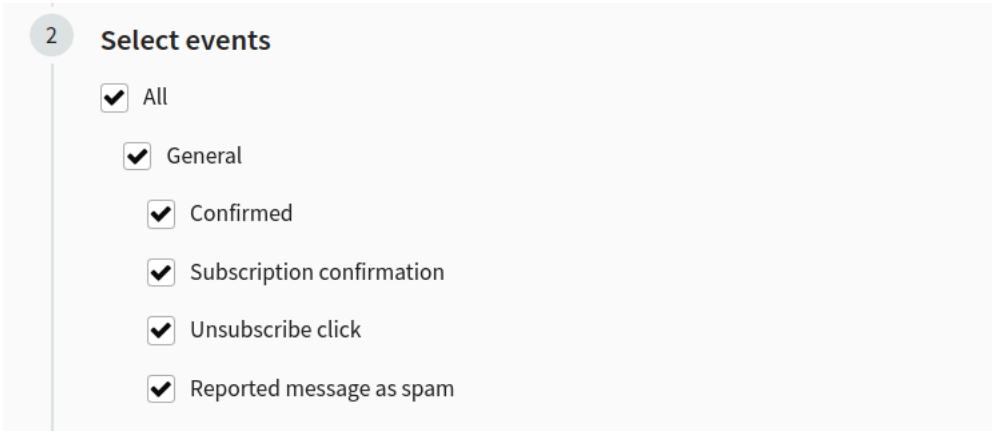
- **Open a list of databases or segments, find a necessary base and click the icon Export profile history.**



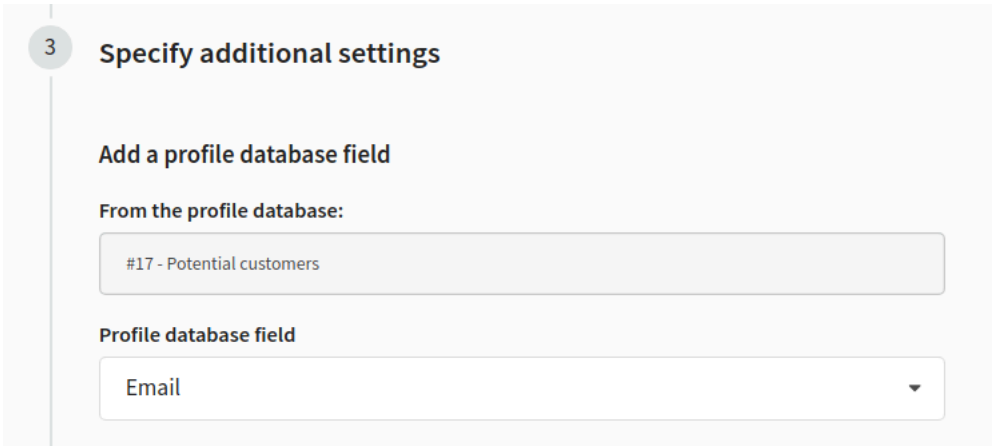
- **Specify the period for which the data needs to be unloaded. You can also select mailings if necessary:**



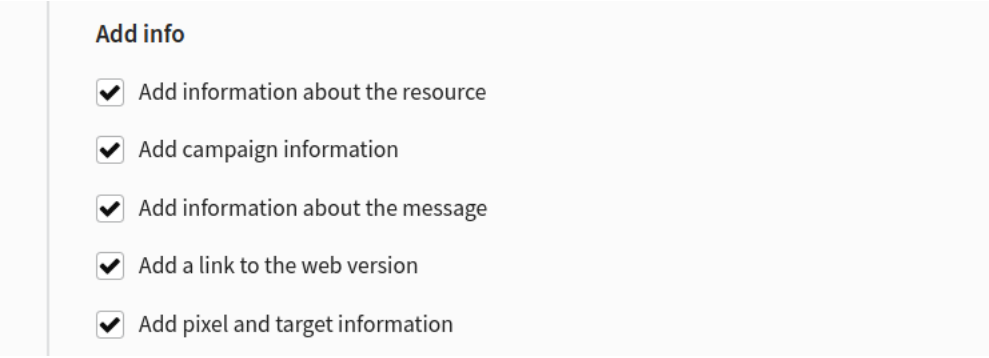
- **Select events to export:**
  - **Basic (subscription confirmation, complaints, unsubscription, etc.)**
  - **Activity in various communication channels (message delivery, openings, clicks, etc.)**
  - **Pixel and goal registration**
  - **Promo codes**
  - **Connections**



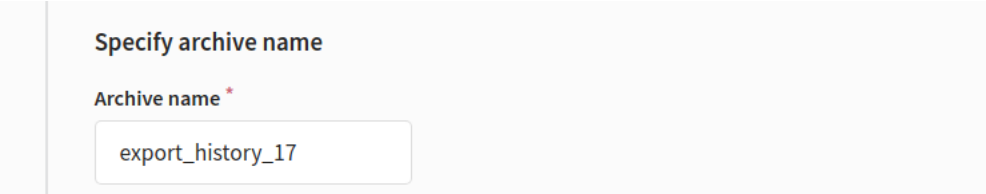
- **Specify the profile database field by which you want to group the unloaded data:**



- **Add additional information for export:**
  - **resource— add ID and the name of the resource to which the profile has subscribed to the target file**
  - **mailing — add ID and the name of the mailing in which the event occurred;**
  - **message— add ID and the name of the template used;**
  - **add a link to the web version of an email;**
  - **pixel and goal — add ID and the name of the pixel as well as the name and value of the goal registered by the pixel**

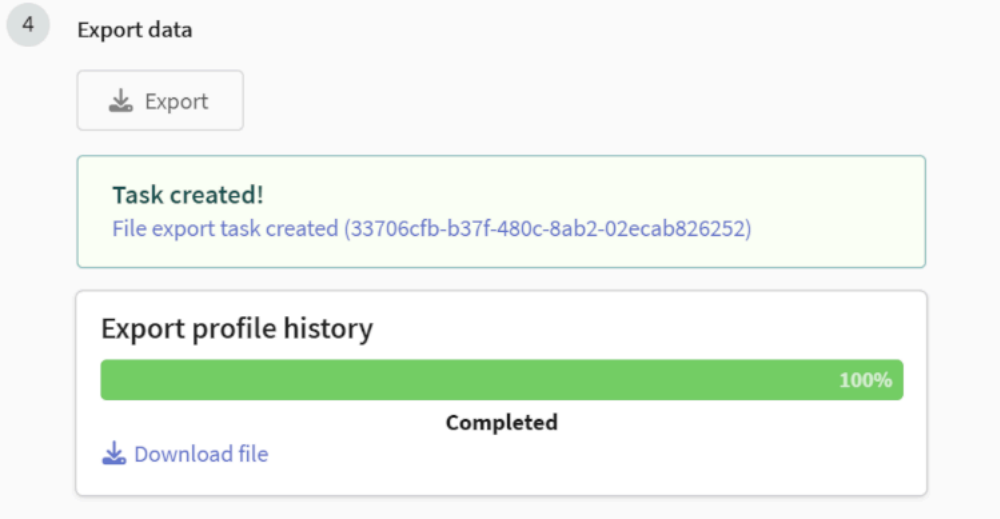


- **Enter an arbitrary name for the archive:**



- **Export the data by clicking Export. An export task will be generated automatically. Its progress will be displayed immediately:**

Once the task is complete, click Download file. Archive in `.zip` format will be downloaded to your device:

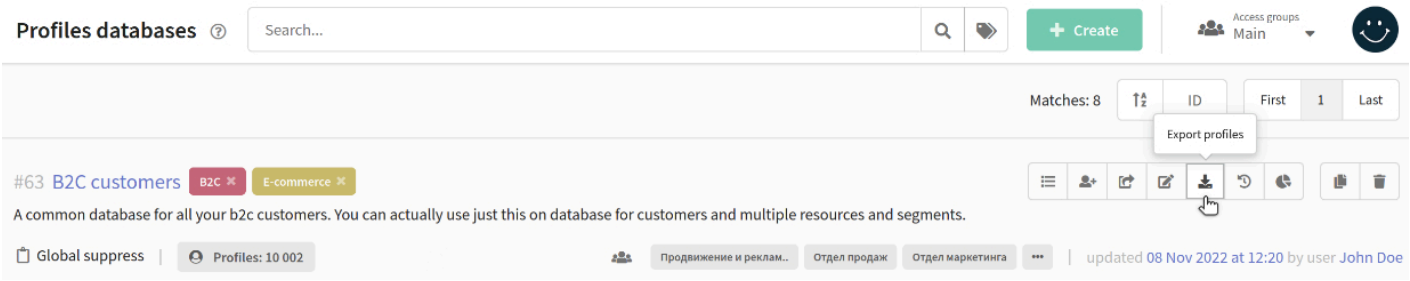


Profile export

Profile history export allows you to download information about all profiles in the database or segment. The data will be downloaded as a csv file. You can choose the data you want to receive.

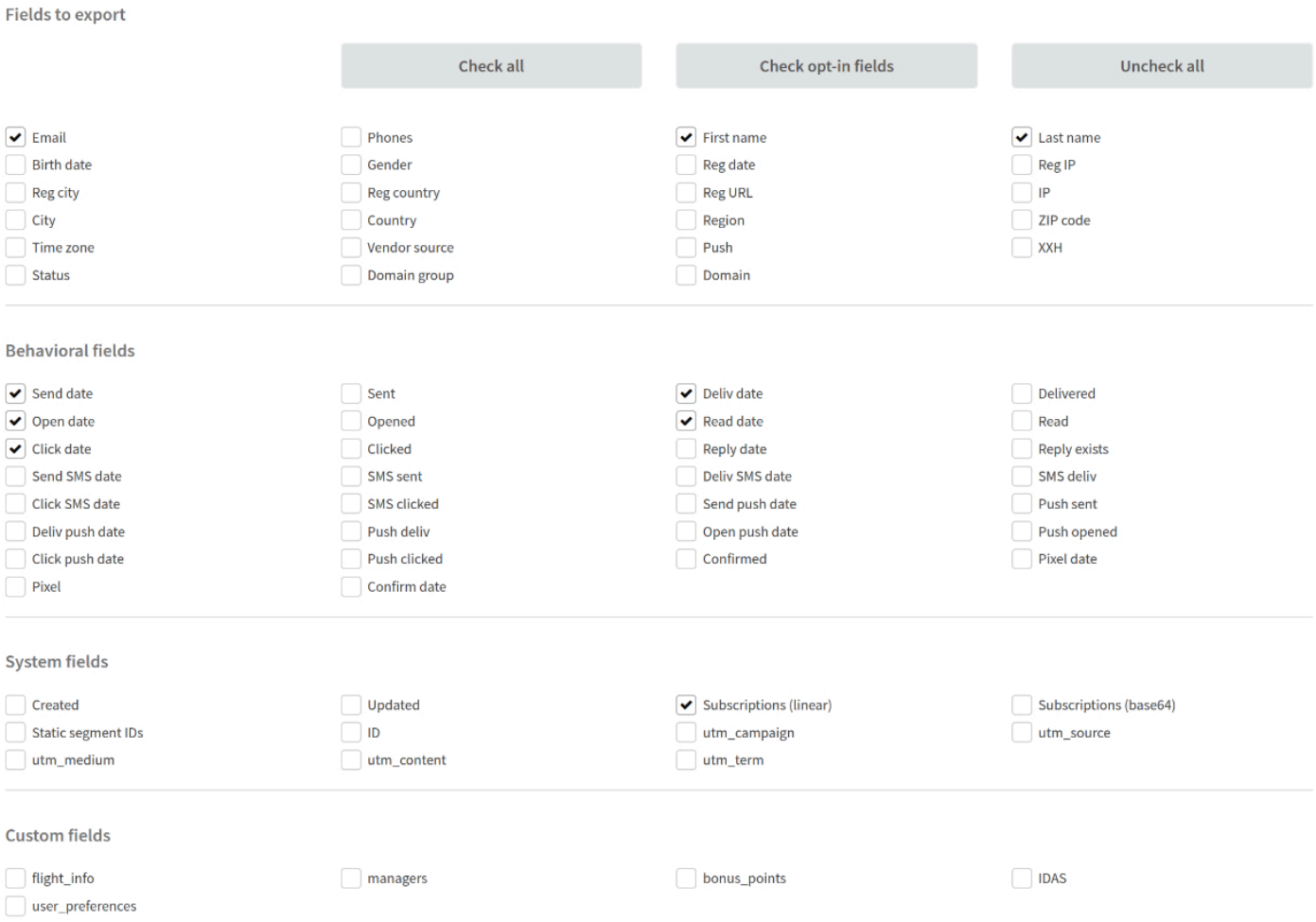
To start exporting, select a database or segment and click

in the toolbar:



Export fields

Check the profile fields to be downloaded. The buttons at the top of the page can help here — check all fields, uncheck all of them, or check only opt-in fields. Opt-in data includes registration information: email, date, links and IP address at the time of registration.



The following profile fields are available for export:

- **default fields;**
- **behavioral fields;**
- **system fields;**
- **custom fields.**

The table below describes behavioral and system fields:

Behavioral fields

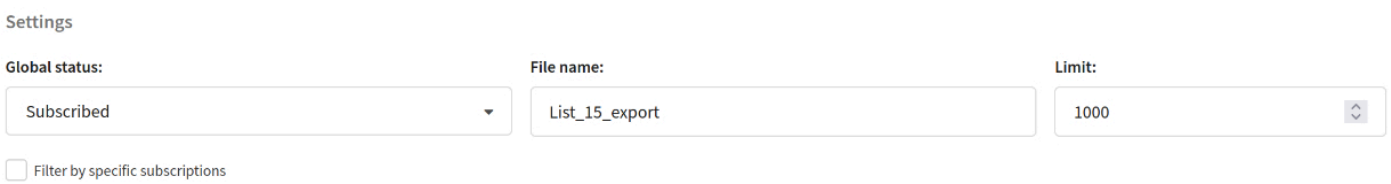
Export field	Column in file	Description
Send date **Send SMS date** **Send push date**	**SendDate** **SendSMSDate** **SendPushDate**	**Date and time of the first message sent (Email, SMS, Push)**
Open date **Open push date**	**OpenDate** **OpenPushDate**	**Date and time when the message was last opened (Email, Push)**
Click date **Click SMS date** **Click push date**	**ClickDate** **ClickSMSDate** **ClickPushDate**	**Date and time of the message was last clicked (Email, SMS, Push)**
Deliv date **Deliv SMS date** **Deliv push date**	**DelivDate** **DelivSMSDate** **DelivPushDate**	**Date and time when the message was last delivered (Email, SMS, Push)**
Read date	**ReadDate**	**Date and time when the message was last read (Email)**
Reply date	**ReplyDate**	**Date and time when the message was last read (Email)**
Confirm date	**ConfirmDate**	**Date and time when the confirmation link was clicked**
Pixel date	**PixelDate**	**Date and time when the pixel was last activated**
Sent **SMS sent** **Push sent**	**SendExists** **SendSMSExists** **SendPushExists**	**Whether a message was sent for the entire period of interaction with the profile or not (Email, SMS, Push)**
Opened **Push opened**	**OpenExists** **OpenPushExists**	**Whether a message was opened for the entire period of interaction with the profile or not (Email, Push)**
Clicked **SMS clicked** **Push clicked**	**ClickExists** **ClickSMSExists** **ClickPushExists**	**Whether a message was clicked for the entire period of interaction with the profile or not (Email, SMS, Push)**
Delivered **SMS delivered** **Push delivered**	**DelivExists** **DelivSMSExists** **DelivPushExists**	**Whether a message was delivered for the entire period of interaction with the profile or not (Email, SMS, Push)**
Read	**ReadExists**	**Whether a message was read for the entire period of interaction with the profile or not (Email)**
Reply exists	**ReplyExists**	**Whether a message was replied for the entire period of interaction with the profile or not (Email)**
Confirmed	**ConfirmExists**	**Whether there was a click on the confirmation link in the message for the entire period of interaction with the profile or not**
Pixel	**PixelOpenExists**	**Whether any pixel was activated for the entire period of interaction with the profile or not**

System fields

Export field	Column in file	Description
Created	**CreateTime**	**Date and time the profile was created**
Updated	**UpdateTime**	**Date and time the profile was updated**
Subscriptions (linear)		**The following subscription data is exported:** • **subscription_resource_id** • **subscription_channel** • **subscription_status** • **subscription_priority** • **subscription_hash_id** • **subscription_is_delete (whether the subscription to the resource was deleted or not)** • **subscription_email** • **subscription_phone** • **subscription_subscription_id (push subscription ID)** • **subscription_provider (push provide)** • **subscription_date** • **subscription_ip** • **subscription_url**
Subscriptions (base64)		**Subscription data is exported as a base64 encoded string**
utm_campaign **utm_source** **utm_medium** **utm_content** **utm_term**	**UtmMailing** **UtmSource** **UtmMedium** **UtmContent** **UtmTerm**	**UTM tags**
Static segment IDs	**StaticSegments**	**ID of the static segment that the profile belongs to**
ID	**Id**	**profile_id**

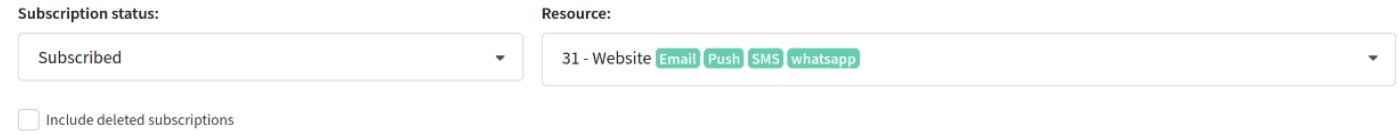
Export settings

In this block, you set additional export settings. You can export the data of profiles that have a certain global status, as well as limit the number of downloaded profiles:



To download information about profiles with a subscription to a specific resource, activate the “Filter by specific subscriptions” option. Select a resource and optionally a subscription status.

In some cases, a resource subscription is added to a profile, but deleted after some time. If you want the file to contain profile data with a deleted subscription to the selected resource, check the box “Include deleted subscriptions”.



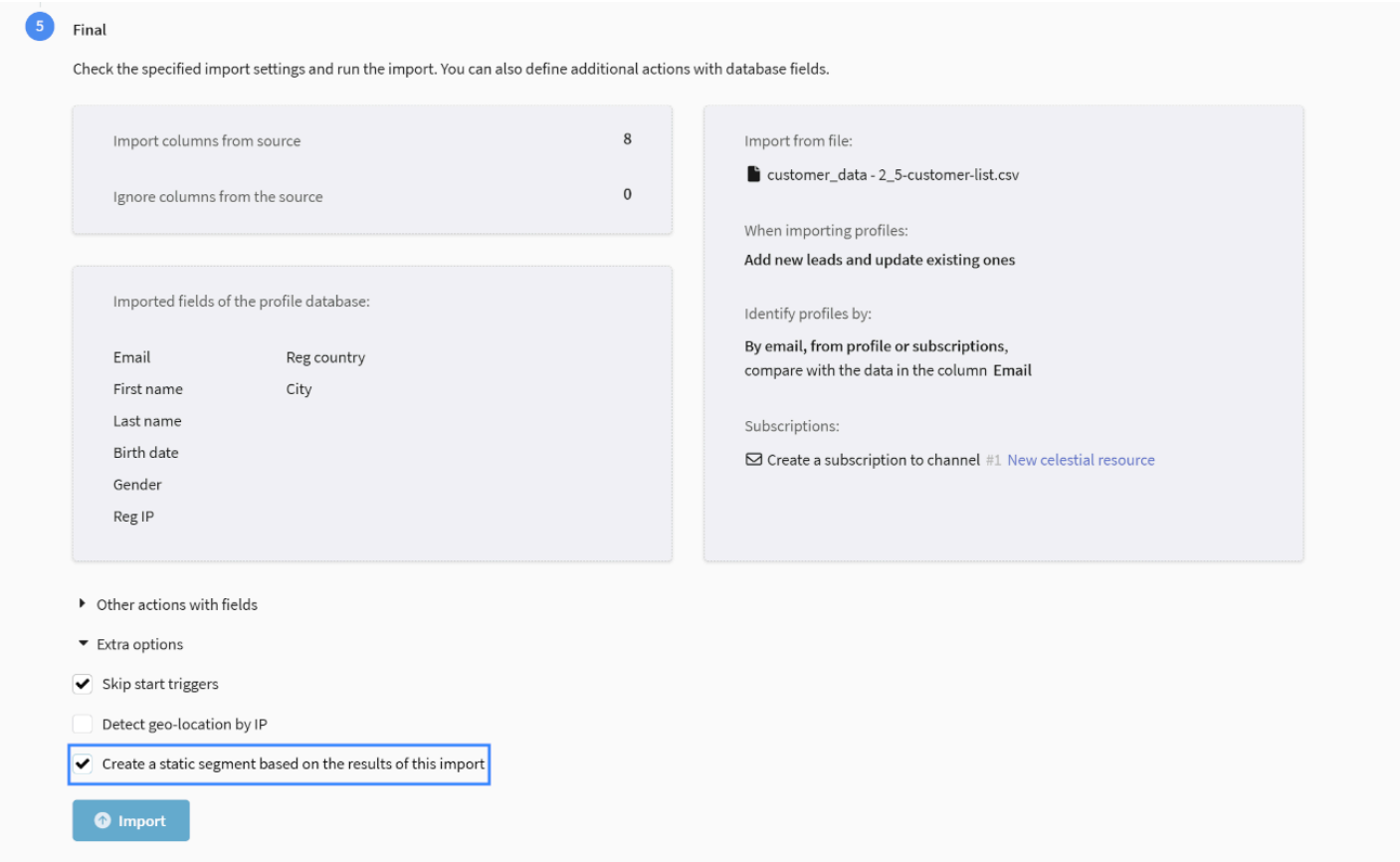
Once you have selected the fields to export and set up the export details, click on the Export button. The platform will start the process. Go to the “Tasks” section to download the generated file containing profile data.

Creating a static segment based on import results

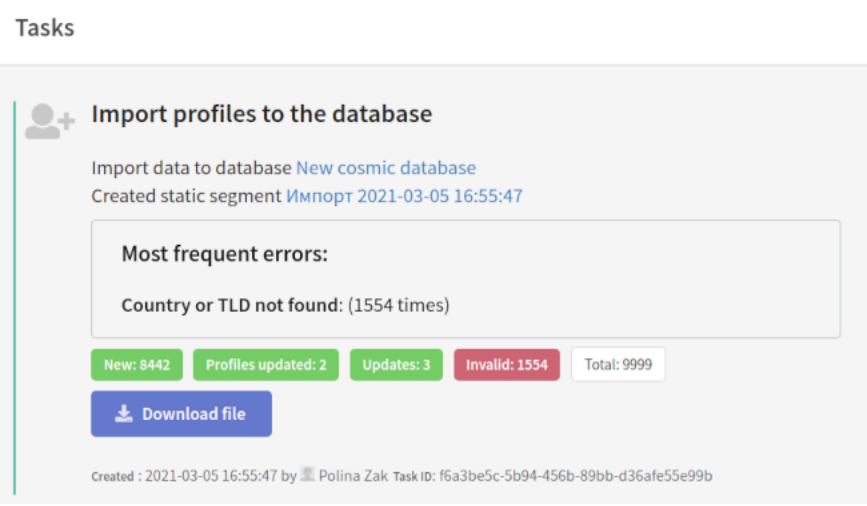
If we import profiles into the database using a file, in the import details, we can enable the option Create a static segment from the results of this import.

INFO

If new profiles are regularly added to the same database, a static segment is an easy way to group them for a one-time mailing



We can check the process of import and the process of creating a segment in Tasks:



This way, as soon as the import is completed, a new static segment will be created. It will include the newly imported profiles. The name of the static segment will be generated automatically (for example, `"Import 2021-03-51 16:55:47"`).

How to open a CSV file

What is CSV

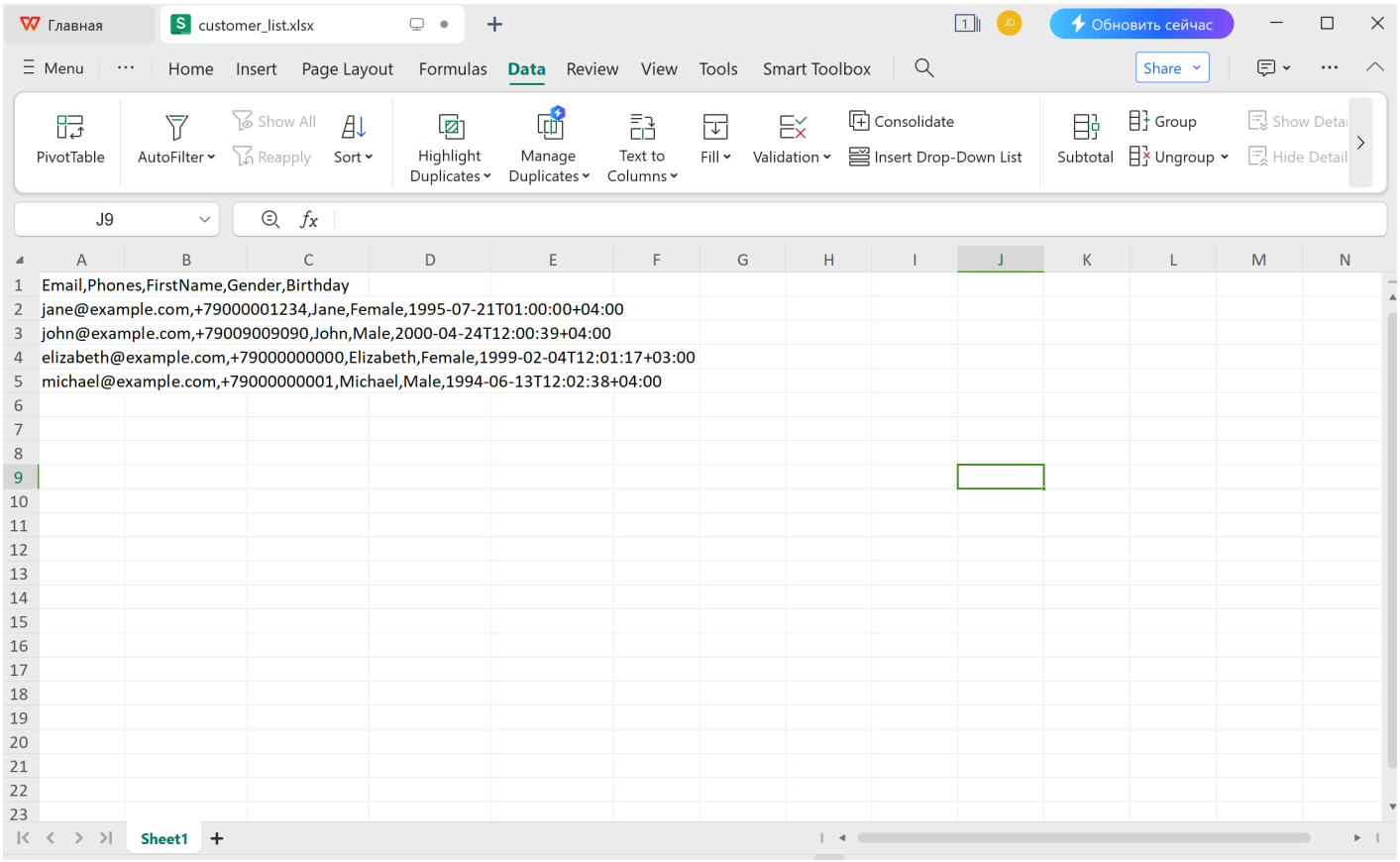
CSV (Comma-Separated Values) is a text format designed to represent tabular data. The data in such a file is separated by commas or other delimiters, such as semicolons or tabs.

```
Email,Phones,FirstName,Gender,Birthday
jane@example.com,+79000001234,Jane,Female,1995-07-21T01:00:00+04:00
john@example.com,+79009009090,John,Male,2000-04-24T12:00:39+04:00
elizabeth@example.com,+79000000000,Elizabeth,Female,1999-02-04T12:01:17+03:00
michael@example.com,+79000000001,Michael,Male,1994-06-13T12:02:38+04:00
```

Using CSV files in the platform

CSV files are used in the platform for the following purposes:

- **Importing profiles into the database from a file**
- **Exporting profiles and their history from the database**
- **Generating reports**
- **Exporting task results, etc.**



Spreadsheet editors like Excel or WPS office are often used for working with CSV files. However, when opening a file, the data may sometimes be displayed incorrectly. The main issue is that Excel incorrectly determines the delimiter used and places all the data in a single column:

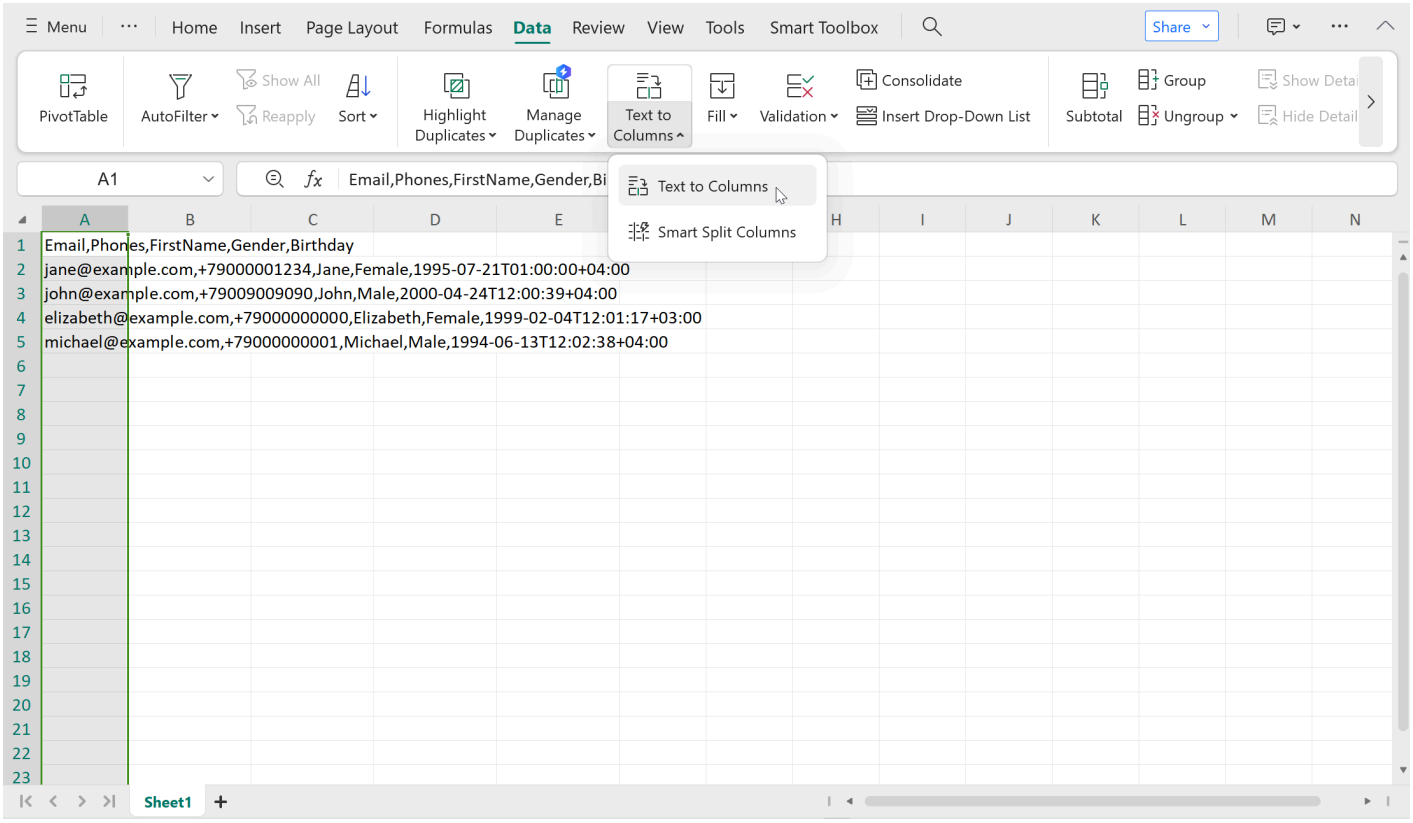
- **Excel incorrectly detects the delimiter used and places all data in a single column**
- **If the file is saved in UTF-8 encoding, Excel may fail to recognize Cyrillic characters and display them as unreadable symbols**

Below you will find instructions on how to properly open a CSV file. The interface may vary slightly depending on the program you use.

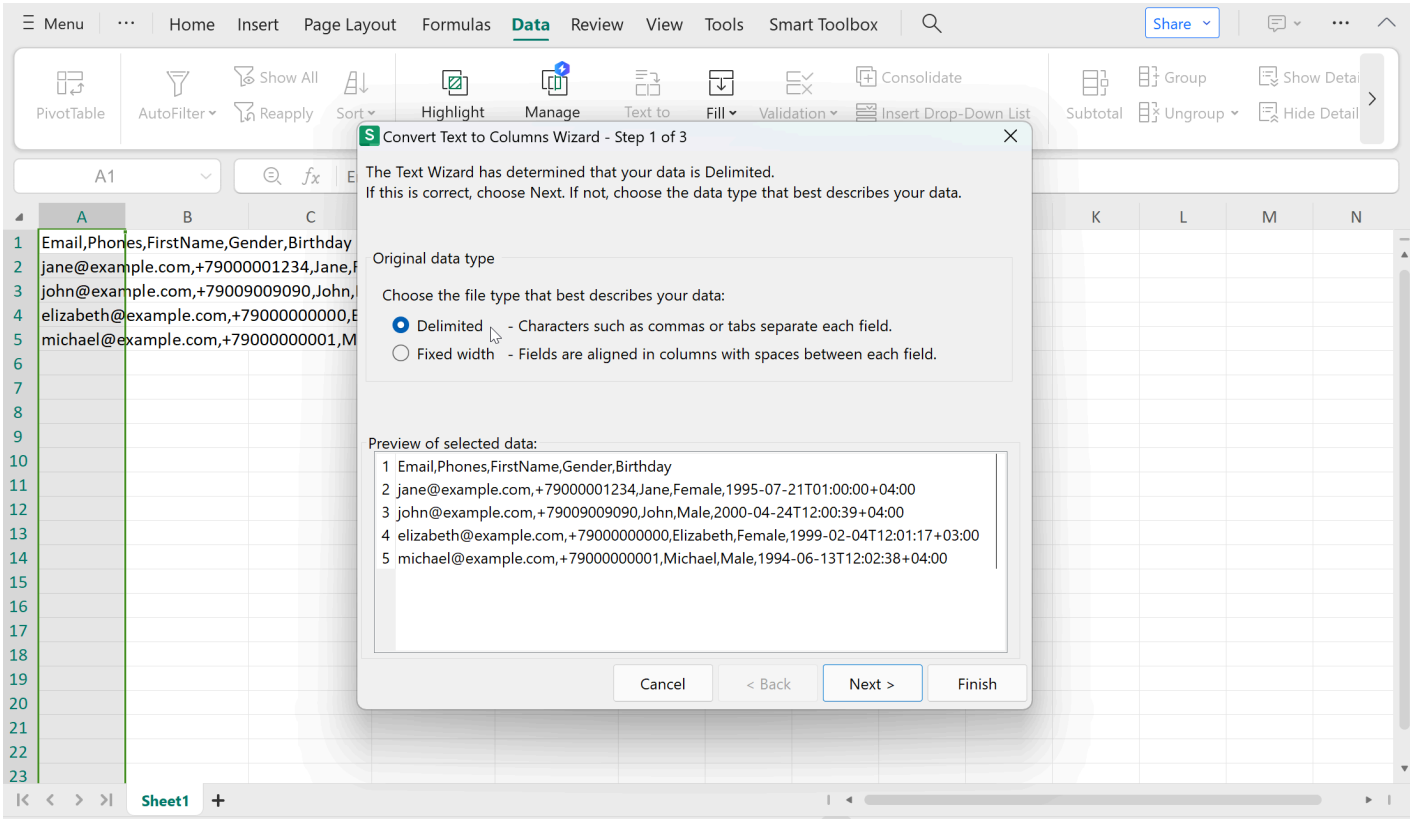
Guide to opening a CSV file

Manual CSV Data Parsing

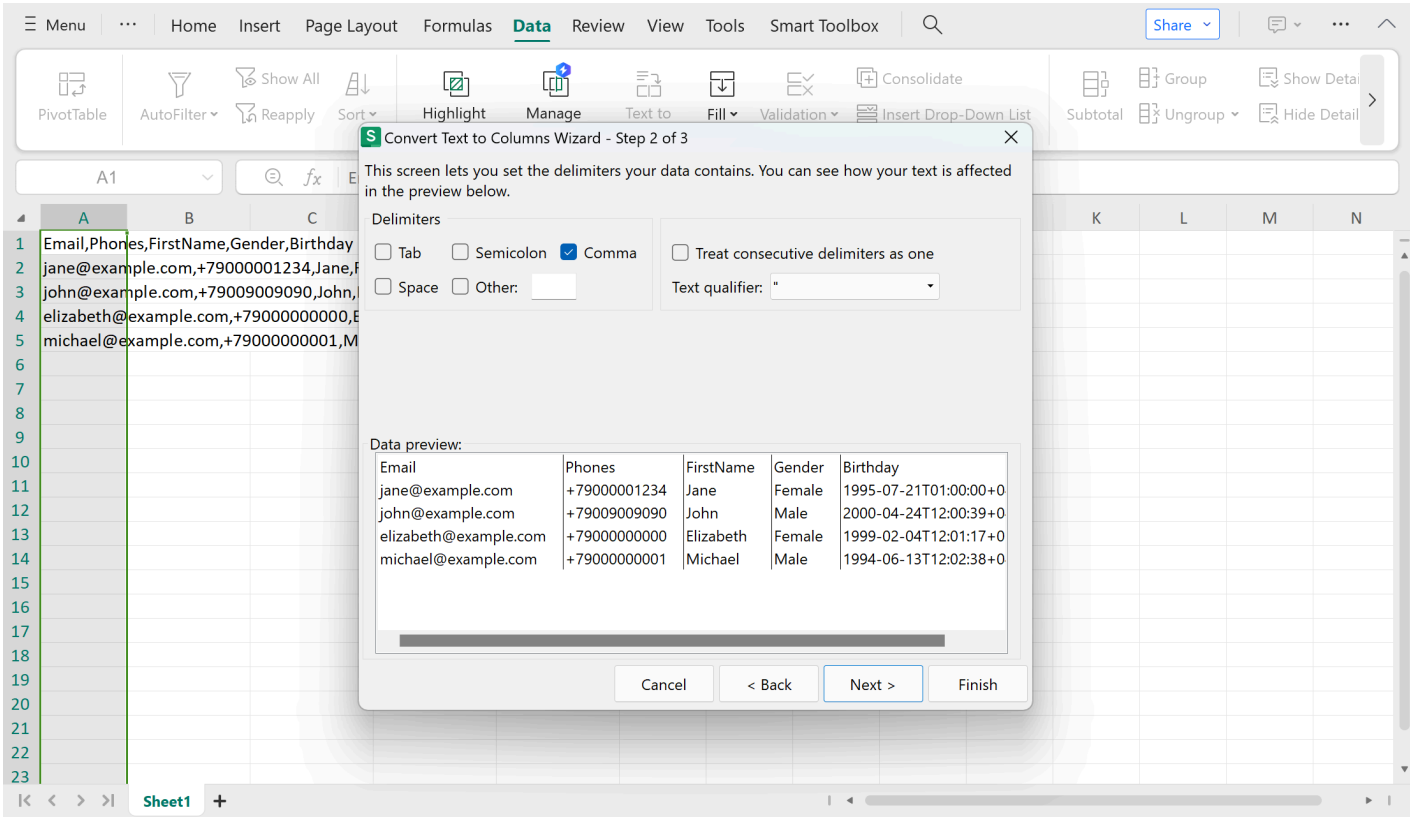
Open the file, select the entire first column, and go to the Data tab on the top panel. Click Text to Columns:



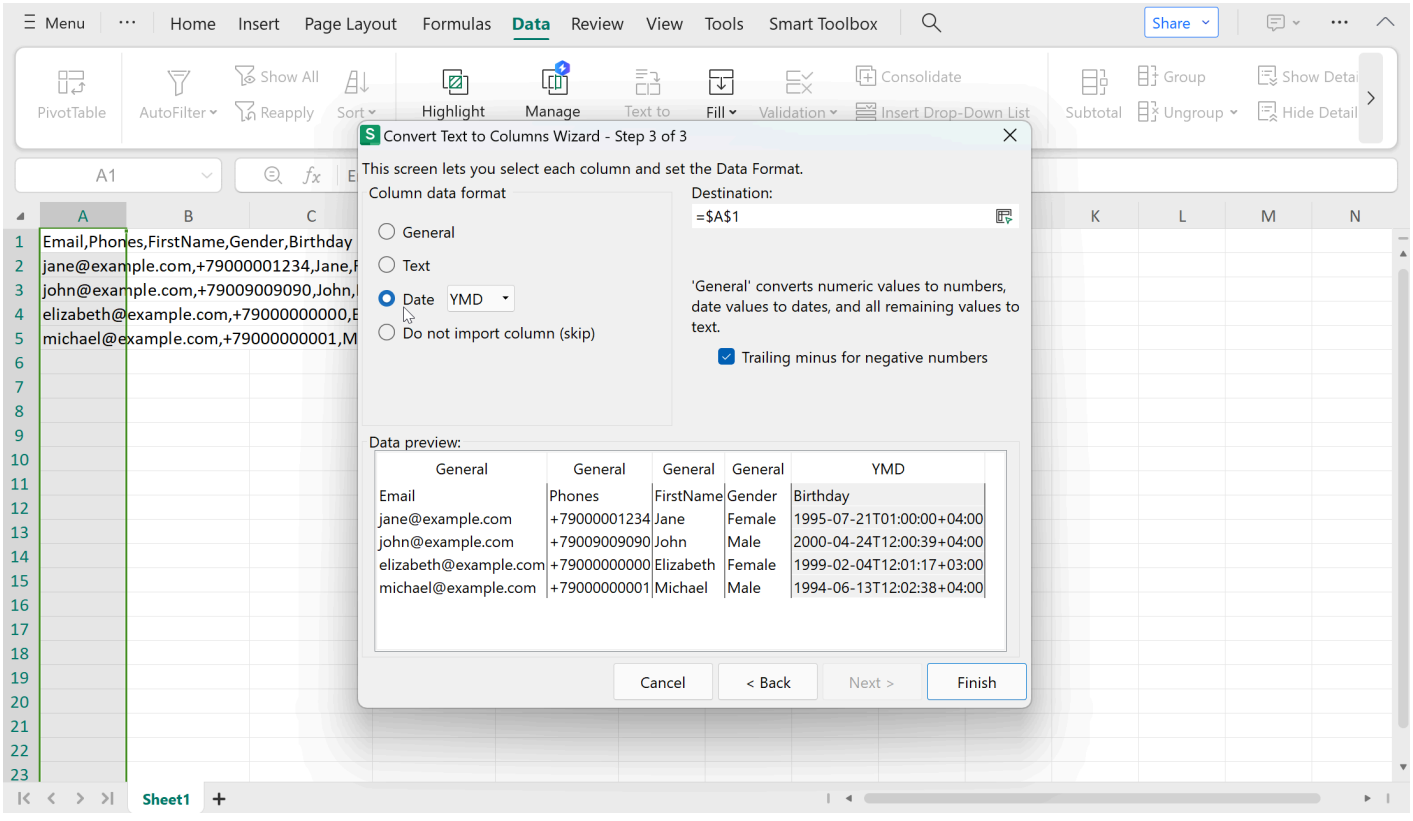
The program will open the text-to-columns wizard to help you correctly interpret the data. In the first step, specify the data format as Delimited. Click “Next”:



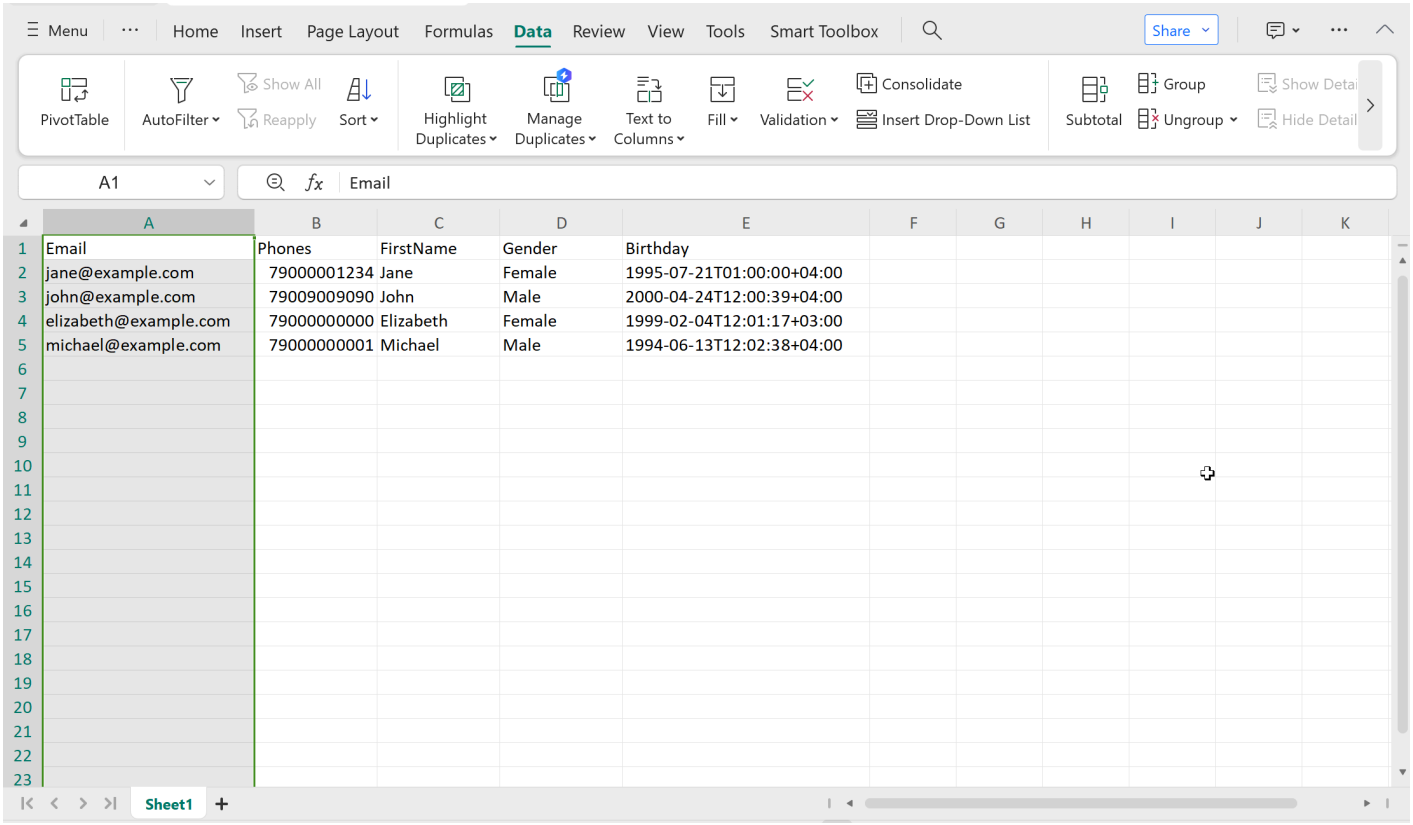
In the second step, choose Comma as the delimiter. Click “Next”:



In the third step, if necessary, you can specify the data format for the columns. In the example below, the Date format (year-month-day) is set for the Birthday column. Then click “Finish”:



If all steps are done correctly, the data in the file will be distributed across the appropriate columns:



	A	B	C	D	E
1	Email	Phones	FirstName	Gender	Birthday
2	jane@example.com	79000001234	Jane	Female	1995-07-21T01:00:00+04:00
3	john@example.com	79009009090	John	Male	2000-04-24T12:00:39+04:00
4	elizabeth@example.com	79000000000	Elizabeth	Female	1999-02-04T12:01:17+03:00
5	michael@example.com	79000000001	Michael	Male	1994-06-13T12:02:38+04:00

UTF-8 Encoding

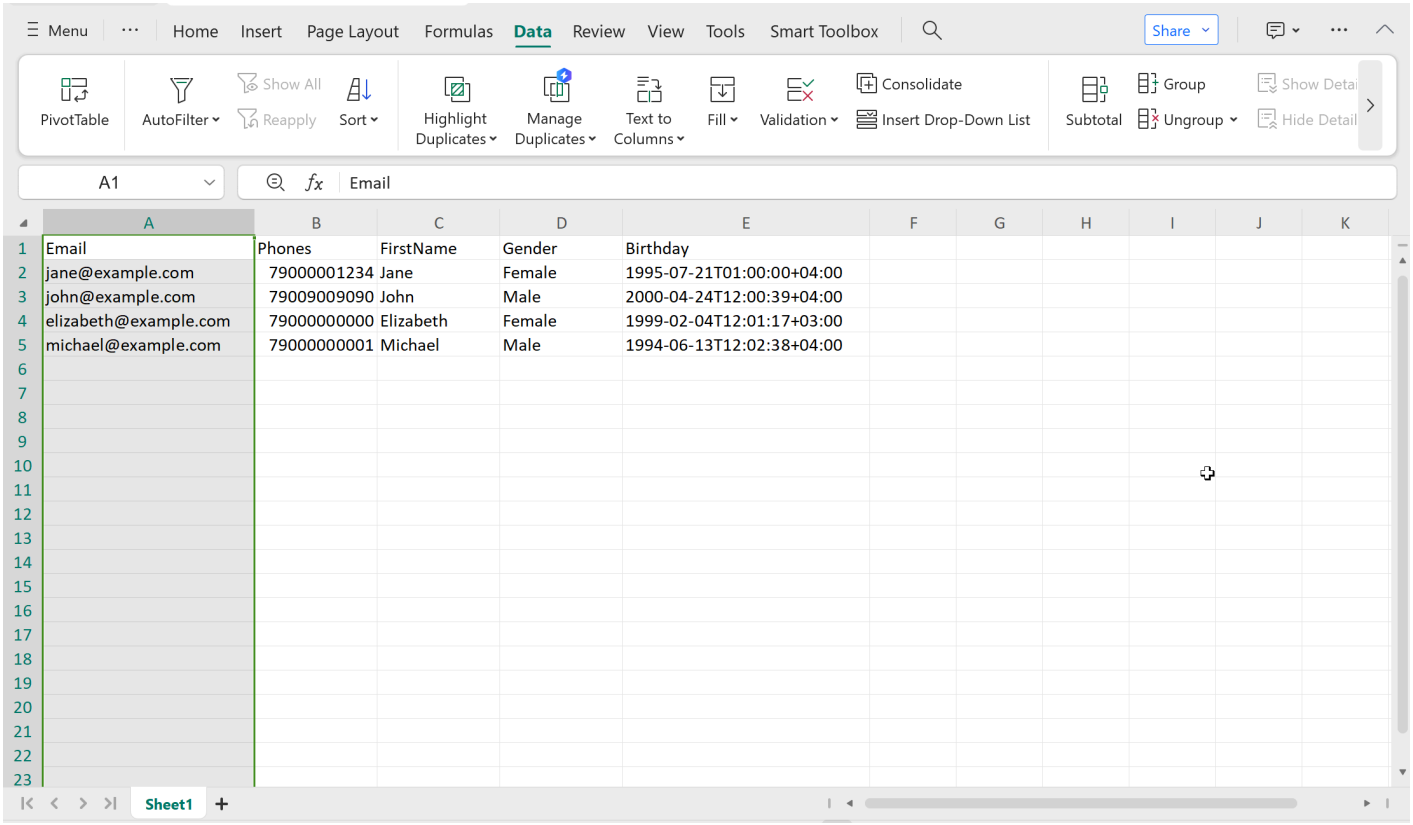
Opening files in UTF-8 (When the file contains Cyrillic characters that display incorrectly)

1. Go to the Data tab —> Import Data (or Edit Data Import if the file is already open):

2. Select the desired CSV file and specify UTF-8 encoding:

4. Select comma as the delimiter:

5. Click Finish, and the data will be properly distributed across columns:

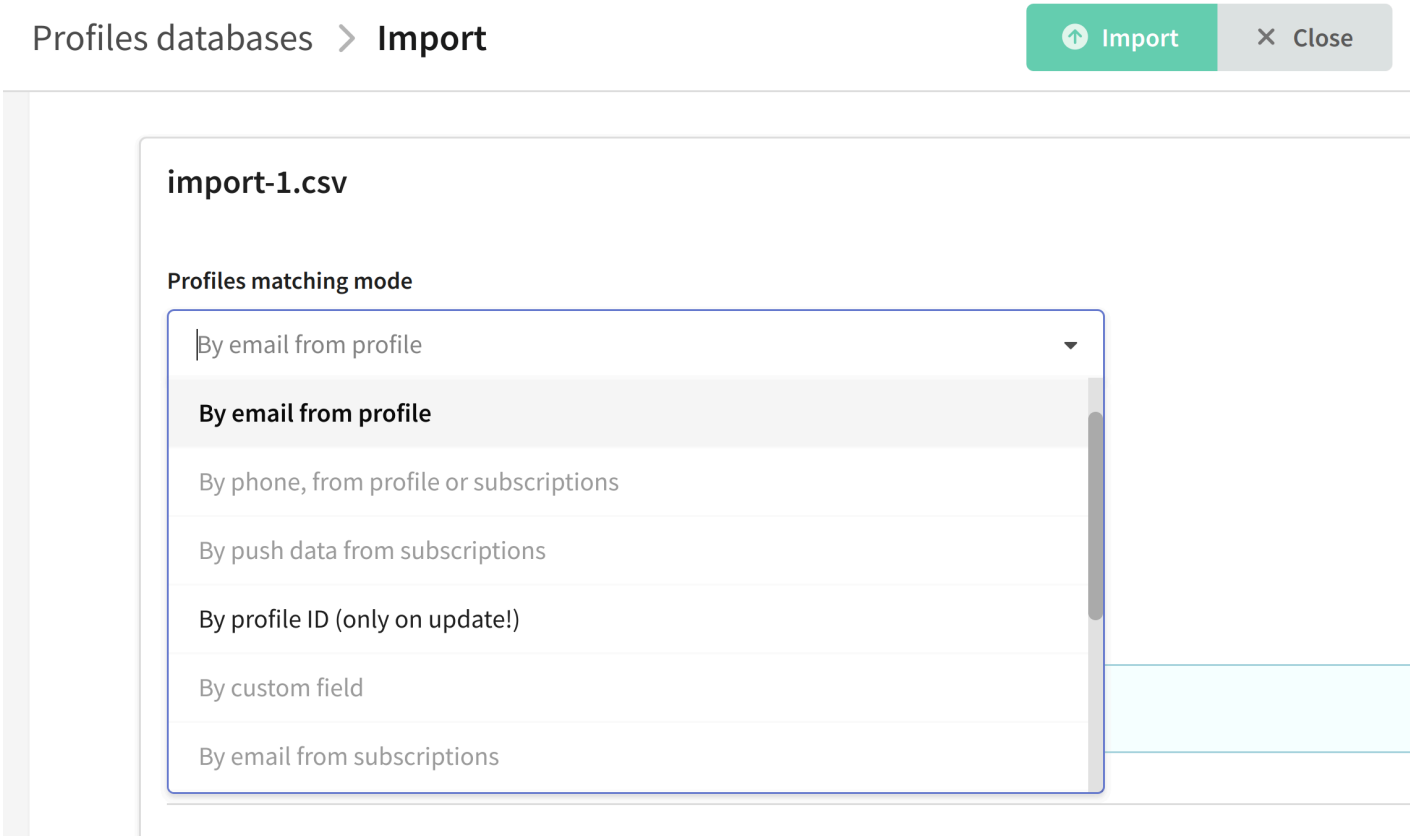


	A	B	C	D	E
1	Email	Phones	FirstName	Gender	Birthday
2	jane@example.com	79000001234	Jane	Female	1995-07-21T01:00:00+04:00
3	john@example.com	79009009090	John	Male	2000-04-24T12:00:39+04:00
4	elizabeth@example.com	79000000000	Elizabeth	Female	1999-02-04T12:01:17+03:00
5	michael@example.com	79000000001	Michael	Male	1994-06-13T12:02:38+04:00

Matching

Matching is the process of searching and identifying existing customer profiles in a database by specific data. It is used to update customer information and prevent the creation of duplicate records in the database.

When importing profiles into the database (from a file, via API query or from an external SQL table), you specify the profile search mode:



.

During the import process, the platform compares this data with what is already in the database. If it finds a profile with the same data, it updates it. If it doesn't find it, it creates a new profile.

Note, the data you are going to use for the search must be in the source. For example, if you are importing data from a file and select email search mode, make sure you have a column with email addresses in your file.

Matching modes

The following modes are available:

- **By email, from profile or subscriptions - search by email from database or subscriptions.**
- **By email from profile - search by database profile email.**
- **By email from subscriptions - search by email from subscriptions only.**
- **By phone, from profile or subscriptions - search by phone in database field or SMS subscriptions.**
- **By phone from SMS subscriptions - search by phone from SMS subscriptions only.**
- **By push data from subscriptions - search by the identifier of push subscription (device token).**
- **By profile identifier - search by system identifier profile_id.**
- **By custom field - search by custom database field, like customer ID in company CRM.**
- **By custom field from subscriptions- search by custom database field from subscriptions only.**
- **Multimatching - by Email or phone from profile and subscriptions (email_phone), or by email or phone from subscriptions (email_phone_sub).**

Matching features

There are several important nuances related to matchmaking:

- **The field by which the search is performed must be unique for each profile. The value of this field should not be the same for different profiles in the same database.**
- **It is not recommended to use different types of matching in the same database to avoid duplicate profiles.**
- **If you do not pass the search mode in the matching field when importing or updating profiles via API, the platform will search for the profile by email address.**

INFO

If you are going to use custom field matching, you should contact the platform administrator to create an index for this field. Thanks to the index, searching for a profile in the database will take much less time.

Types of fields in the database

Default database fields

This table provides a list of default database fields, their API matches, data types, and examples of values that can be used.

Field in the database	Field name in the API	Field type in the UI	The type of the field in the API	Example of the value
Email	**email**	**Line/Email**	**string**	**"example@example.com"**
Phones	**phones**	**Phone list**	**array**	**["+79001234567"]**
Name	**_fname**	**Line**	**string**	**"John"**
Last name	**_lname**	**Line**	**string**	**"Doe"**
Date of birth	**_bdate**	**Date**	**string**	**"2001-12-31" "2001-12-31T00:00+03:00" The time can only be transmitted via the API.**
Gender	**_sex**	**Enum**	**string**	**"male" "female"**
Date of reg.	**_regdate**	**Date**	**string**	**"2023-12-31" "2023-12-31T23:59:59+03:00"**
The IP address of the reg.	**_regip**	**IP**	**string**	**"127.0.0.1"**
City reg.	**_regcity**	**Line**	**string**	**"Moscow"**
Country reg.	**_regcountry**	**Line**	**string**	**"Russian Federation" "RU"**
Link reg.	**_regurl**	**Line**	**string**	**"https://example.com"**
IP address	**_ip**	**IP**	**string**	**"127.0.0.1"**
City	**_city**	**Line**	**string**	**"Moscow"**
Country	**_country**	**Line**	**string**	**"Russian Federation" "RU"**
Region	**_region**	**Line**	**string**	**"Europe/Moscow"**
Index	**_postal_code**	**Line**	**string**	**"190000"**
Time zone	**_tz**	**Line**	**string**	**"Europe/Moscow"**
Supplier	**_vendor**	**Line**	**string**	**"form #30"**

Custom database fields

The type of field determines which operators will be available in segments and in dynamic template variables, so we recommend carefully choosing the type of additional field in the database.

Field type in UI	Field type in API	Example of the value
Integer	**int**	**100**
Float	**float**	**100.5**
Line	**string**	**"string value"**
Date	**string**	**"2024-05-27T12:00:00Z"**
Boolean	**boolean**	**true false**
Enum	**string**	**"Option 1" You can also pass a value as a number, for example, 1. The number corresponds to the ordinal number of the item in the list.**
IP	**string**	**"127.0.0.1"**
Tags	**array**	**["Tag 1", "Tag 2", "Tag 3"]**
Object	**object**	**`{ "key_1": { "sub_key": "sub_value" } }`**
Array of objects	**array of objects**	**`[ { "key_1": "value_1" }, { "key_2": "value_2" } ]`**

INFO

When API importing or updating fields with the “Tags” type, their values will be overwritten, not added to the existing ones. To add new values without deleting old ones, use functional update.

Acceptable formats and restrictions

The platform expects certain value formats for each type of field. Make sure that the imported data meets the specified requirements, otherwise the system will not be able to process them correctly.

- **Email — the format must comply with RFC standards. A regular expression is used for validation:**

```
^[-a-z0-9!#$%&'*+/=?^_{|}~]+(\.[-a-z0-9!#$%&'*+/=?^_{|}~]+)*@((?:localhost|([a-z0-9]([-a-z0-9]{0,61}[a-z0-9])?\.)*([a-z]{1,24})))$
```

- **Phones — the number must correspond to the international format (without spaces, hyphens and other separators, except for the plus sign at the beginning). More information about the standard can be found in this article.**
- **Dates — are recorded in the format RFC 3339 (ISO 8601) indicating the time zone.**

- **Country — accepted as a two-letter code ISO 3166-1 alpha-2 or the full name (in any case).**
- **Gender — acceptable values are described in this article.**
- **Provider — the length of the string is no more than 26 characters.**
- **IP — IPv4 format.**
- **Line — it is not recommended to store more than 100 KB of text in one field.**
- **Objects and arrays of objects — you can write only peer-to-peer objects or peer objects nested in other objects. If you are using objects with a more complex structure, connect the SQL database and create queries to select profiles. In object field names, only Latin letters (in any case), digits from 0 to 9, the underscore `_`, and the hyphen `-` are allowed.Data type. You can store data of type `int`, `float`, `string`, `boolean` in the Object Array or Object fields.**
- **The limit on the amount of data of one profile in the database is 16 MB, including all metadata. However, it is not recommended to store a total of more than 2 MB in additional profile fields.**

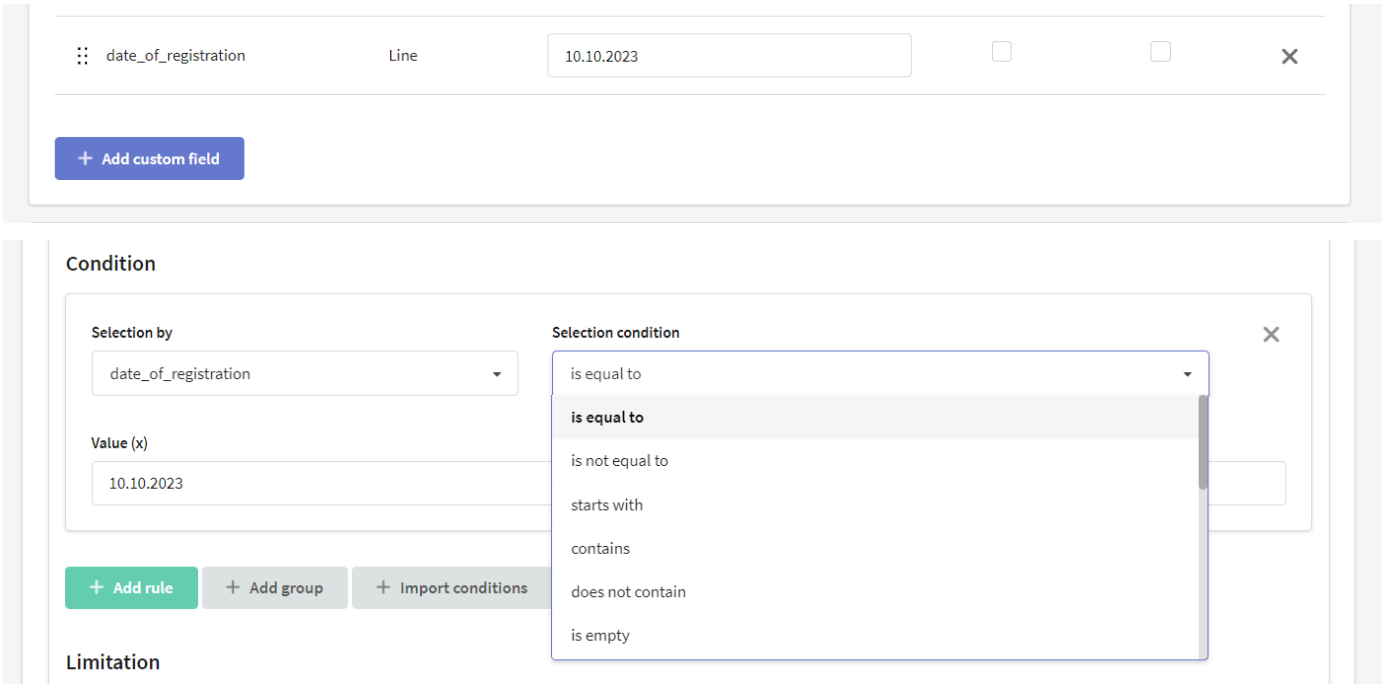
Matching of database fields and form elements

If you are configuring profile data import from form fields to database fields, make sure that you are using the correct field type for each form element.

Form element	Field in the database
Short text	**Custom field with the "String" type**
Email	**Default "Email" field**
Phone number	**Default field "Phones"**
Long text	**Custom field with the "String" type**
Number	**Custom field with the "Integer" type**
Yes/No	**Custom field with the "Boolean" type**
One choice	**Custom field with the "String" type**
Multiple choice	**Custom field with the "Tags" type**
Drop-down list	**Custom field with the "Tags" type**
Date	**Default or custom field with the "Date" type**
Range	**Custom field with the "Integer" type**

NOTE

Although you can write all user data in string fields, this will limit the possibilities for further segmentation. For example, if the date is written in a string field, you will not be able to use the comparison operators “date less” or “date more” in segments or templates.



Global control groups

What is a global control group

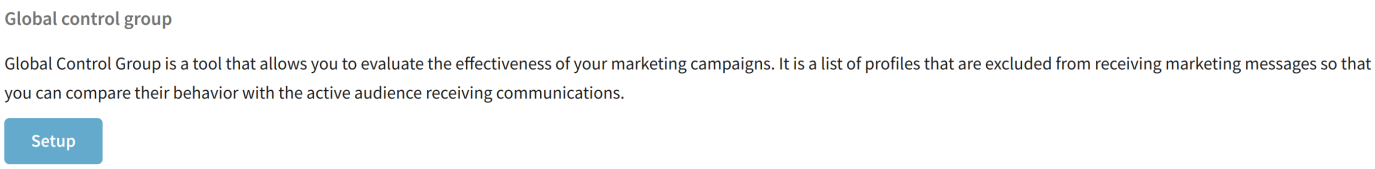
Global control groups are parts of the customer base that temporarily does not receive marketing communications in order to analyze the effectiveness of marketing strategies. After some time, the user can compare the key metrics of the main part of the customers and the control group. In case the metrics of the groups differ slightly, the marketer can conclude that his mailings are ineffective.

Having control groups allows to:

- **Evaluate the effectiveness of marketing strategies;**
- **Optimize marketing costs;**
- **Increase the conversion rate of marketing mailings.**

Configuring global control groups

You can customize the formation of a global control group for a profile database only after its creation. To do this, go to edit the created base and find the corresponding section at the bottom of the page:



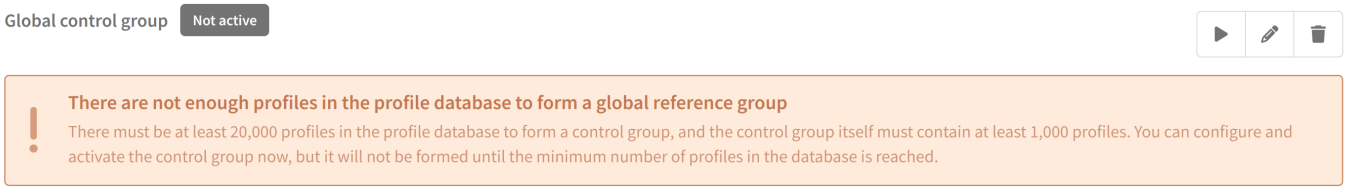
The first step is to select an source audience for the GCG. This source can be either a profile database or a segment. If you have selected a segment as the source for the GCG, you will need to set up a segmentation query. You can read more about segmentation and selection conditions in this section.

TIP

Regardless of the source of the audience, the following requirements exist for a global reference group:

- **The GCG must include between 1,000 and 10 million profiles;**
- **The GCG must be no less than 5% of profiles and no more than 25% of the entire profile base;**
- **There must be at least 20,000 profiles in the profile database to form a control group.**

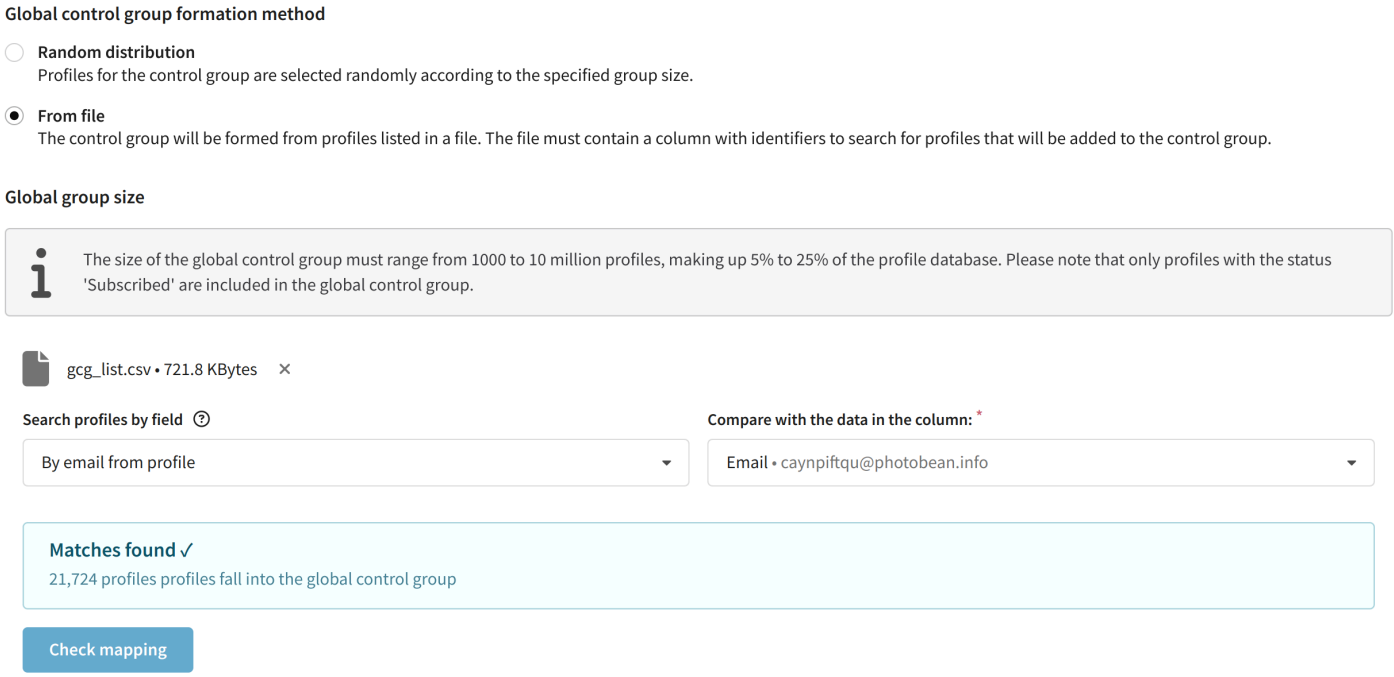
You can set up and activate a control group at any time, but it will only be formed after all of the above criteria have been met. In case of any problems related to the number of profiles in the GCG, the platform will notify you:



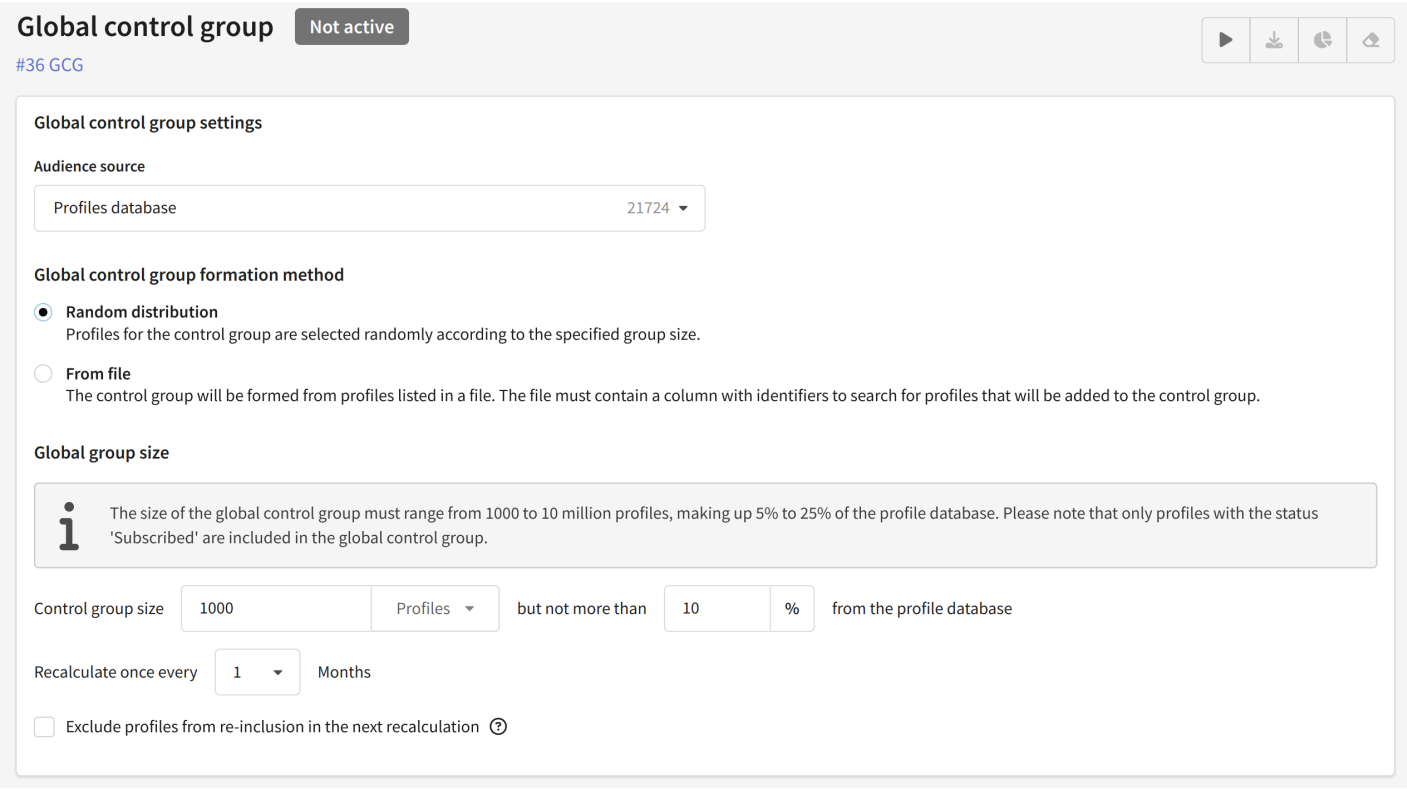
The global control group based on the profile database has two ways of formation:

- **Random distribution — profiles for the control group are selected randomly according to the specified group size.**
- **From file — the control group will be formed from profiles in a file. The file must contain a column with an identifier to search for profiles for the control group. Keep in mind that the GCG will be generated from the entire file.**

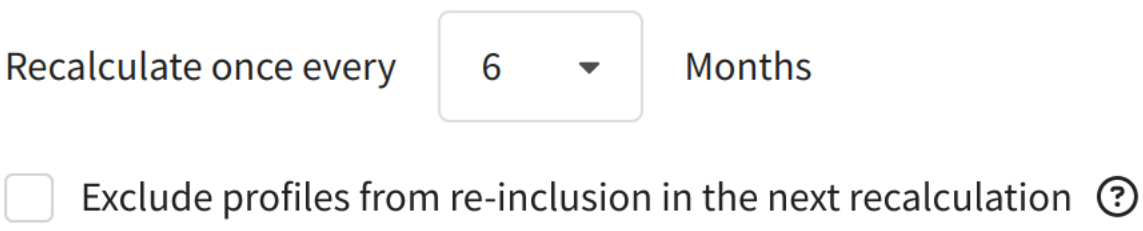
If you have chosen the option of forming from file, you will be offered to upload a file with the list of clients in .xls, .xlsx, .csv or .txt format, and then specify the field for searching profiles in the database and the column for comparison from the file. After clicking on the Verify compliance button, the platform will display a message indicating the number of profiles that will be included in the control group:



For random distribution, you can specify a specific number of profiles in the group, as well as the maximum percentage of GCGs to the entire database of profiles. The latter parameter is necessary to limit the size of the GCG in case the number of profiles in the database changes significantly:



Then you must specify the frequency of recalculation for the global control group, the minimum value is once every 12 months, the maximum value is once every 1 month. The recalculation takes place according to the specified schedule at 00:00 in the account's time zone:

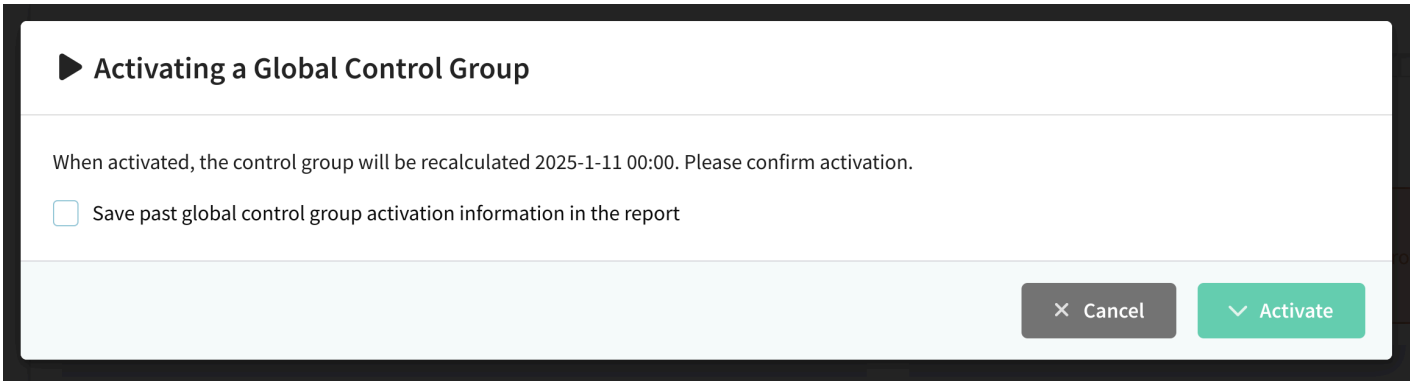


The parameter Exclude re-entry for a profile at the next recalculation allows to prohibit the profiles participating in the current control group to get there at the next recalculation. However, these profiles will be able to get to the GCG at the third and subsequent recalculations.

TIP

If the same profiles keep getting added to the control group, it can distort the results and lead to wrong conclusions. The “Exclude re-entry for a profile at the next recalculation” parameter helps change the control group, making the experiment’s results more accurate.

In order for the global control group to start working, you need to activate it and save it. Before activation, the platform will inform you of the next recalculation date of the control group and offer to save information about the past activation:

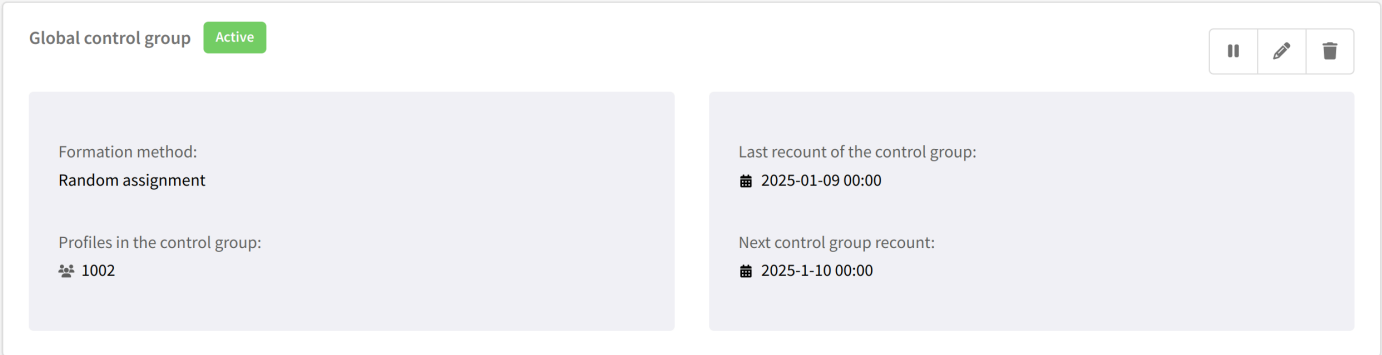


Once GCG is saved you will have access to quick-actions menu at the top of the screen:

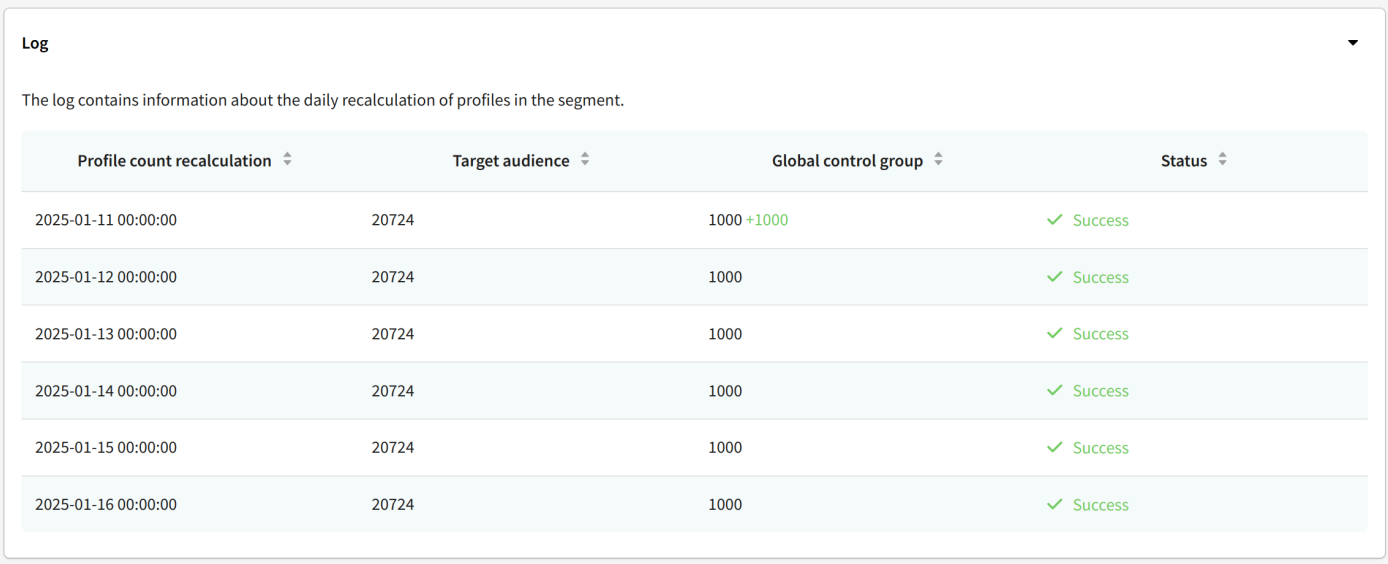
The following buttons are available to you in the menu:

- **Deactivate - stops the GCG from running**
- **Export Report - allows you to unload the list of profiles that are in the GCG**
- **Insights - opens [appropriate page] of analytics**
- **Reset Settings - resets GCG settings and clears all filled fields.**

Keep in mind that the first data and analytics on the participants of the GCG will be available after the first calculation. Once the CTG has been successfully set up and run, the Profile Database Edit screen will display information about the control group, the number of profiles in it, and the dates of the last and next recalculation:

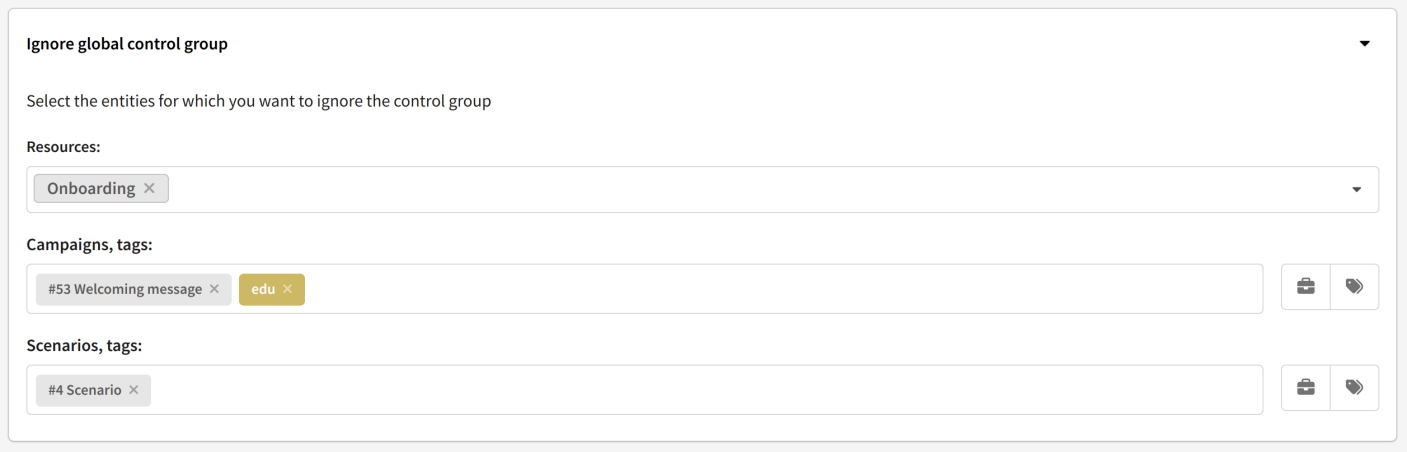


In order to view the log, which contains detailed information about GCG recalculations, you need to go to edit the control group and open the corresponding tab at the bottom of the screen:



Exclusions for the global control group

You can configure exclusions from the global control group in the Ignore global control group section. Within the selected resources, mailings and scenarios, marketing communications will be carried out even for those users who are in the global control group:



Here you can specify:

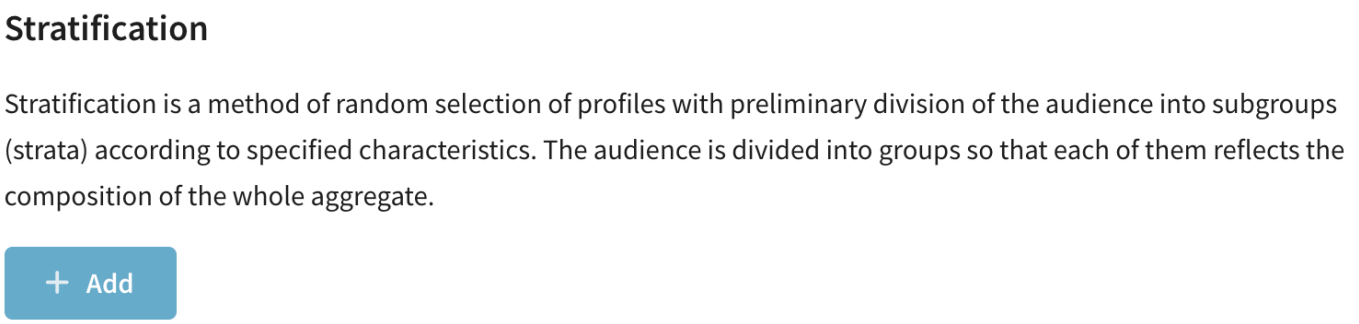
- **specific mailings and scenarios**
- **entities with certain tags**
- **entities with a specific attribute value.**

Stratification

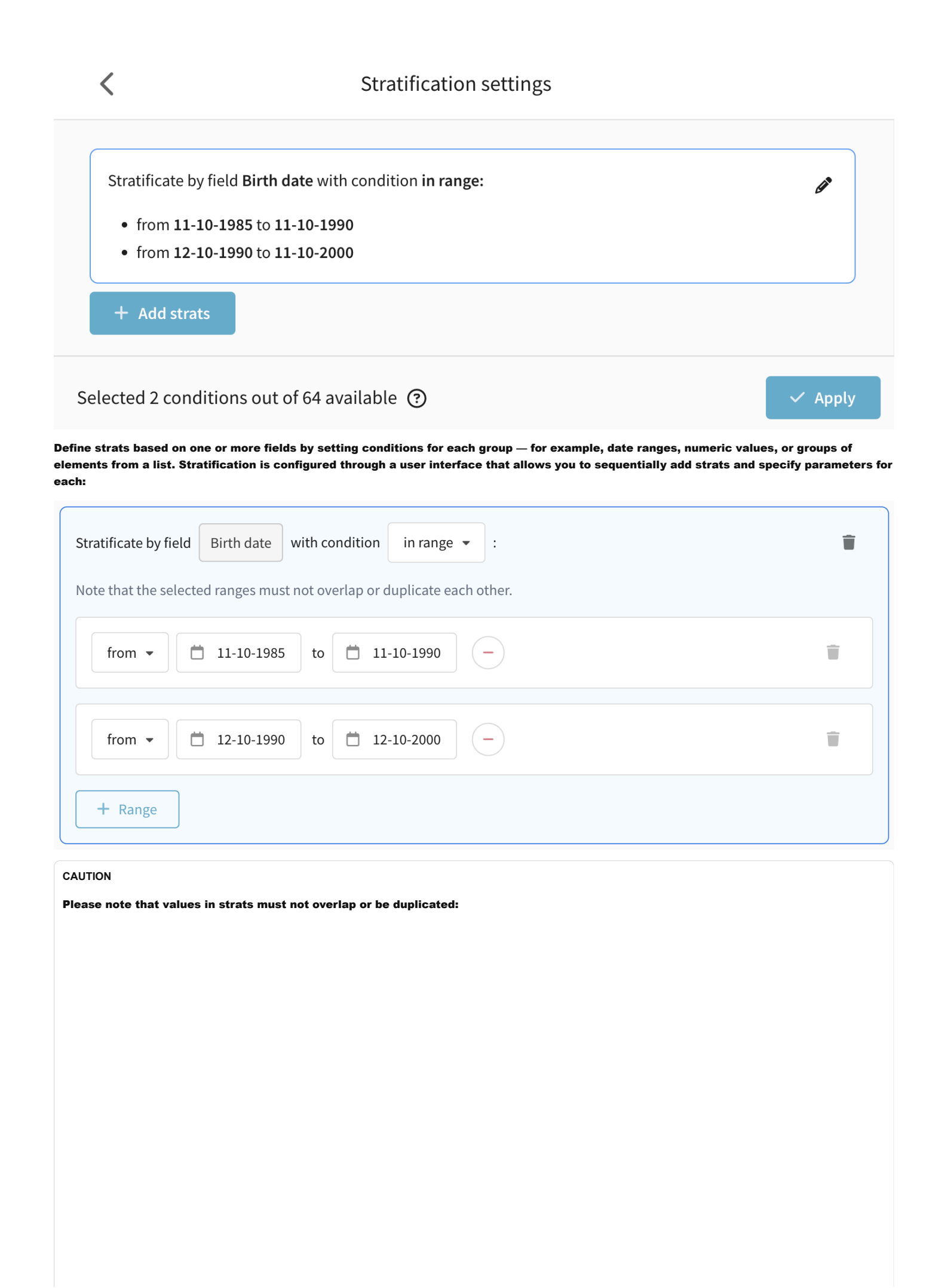
Stratification is a method of random profile selection that involves dividing the audience into subgroups (strats) based on specified characteristics. The audience is segmented in such a way that each group reflects the overall structure of the entire population.

A portion of profiles is then randomly selected from each group, proportionally to its size. This ensures that the control group has the same distribution of characteristics as the test group, making mailing results more accurate. The maximum number of strats depends on the number of selected characteristics and the number of groups defined within them. These limits can be configured in the system.

Add stratification by clicking the corresponding button:



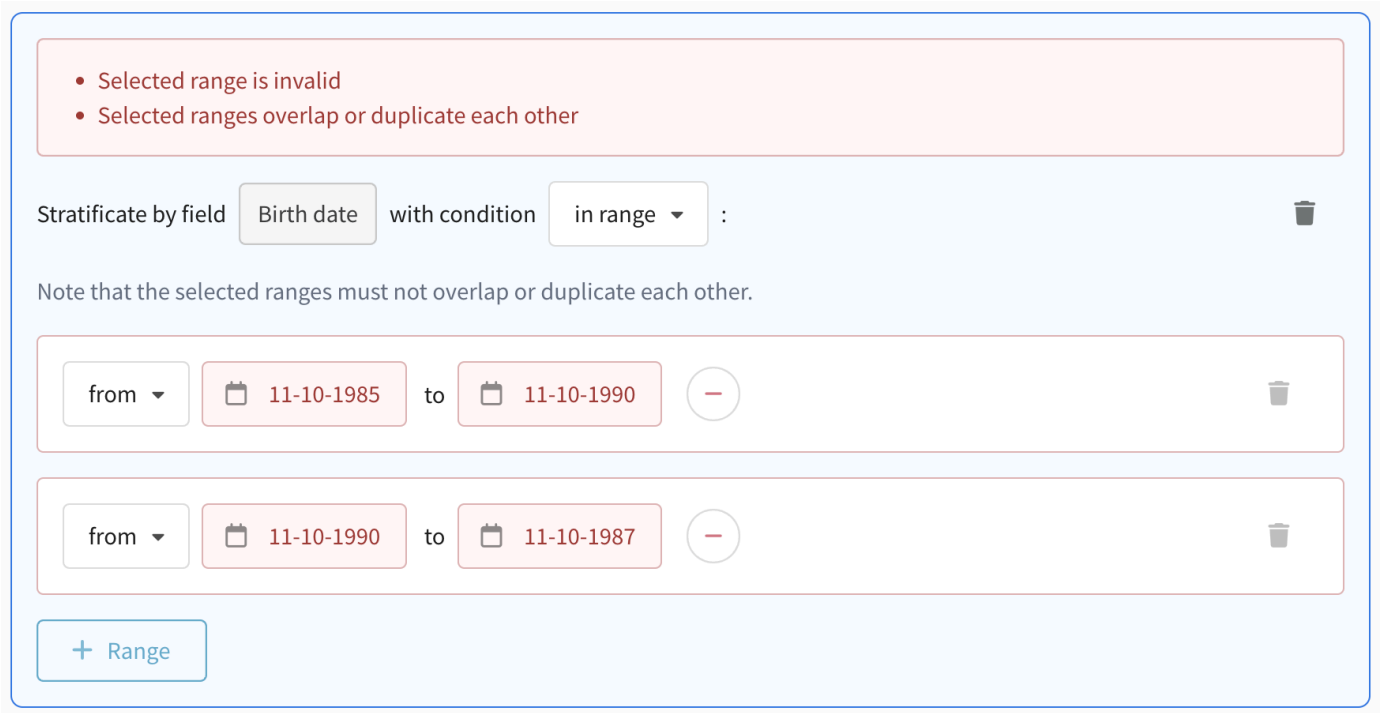
You will be taken to the stratification settings window, where you can create and edit strats:



Define strats based on one or more fields by setting conditions for each group — for example, date ranges, numeric values, or groups of elements from a list. Stratification is configured through a user interface that allows you to sequentially add strats and specify parameters for each:

CAUTION

Please note that values in strats must not overlap or be duplicated:



Supported field types for stratification:

- **boolean — logical values (e.g., yes/no);**
- **date — date values (e.g., birthdate, registration date);**
- **enum — predefined lists of values (e.g., region, category);**
- **float — decimal numbers;**
- **integer — whole numbers;**
- **unicode — string values (e.g., text labels or group names).**

Each of these field types supports appropriate condition formats: ranges, exact values, groups of values, etc.

TIP

The total number of condition combinations in stratification must not exceed 64. This value is calculated by multiplying the number of groups in each added stratum.

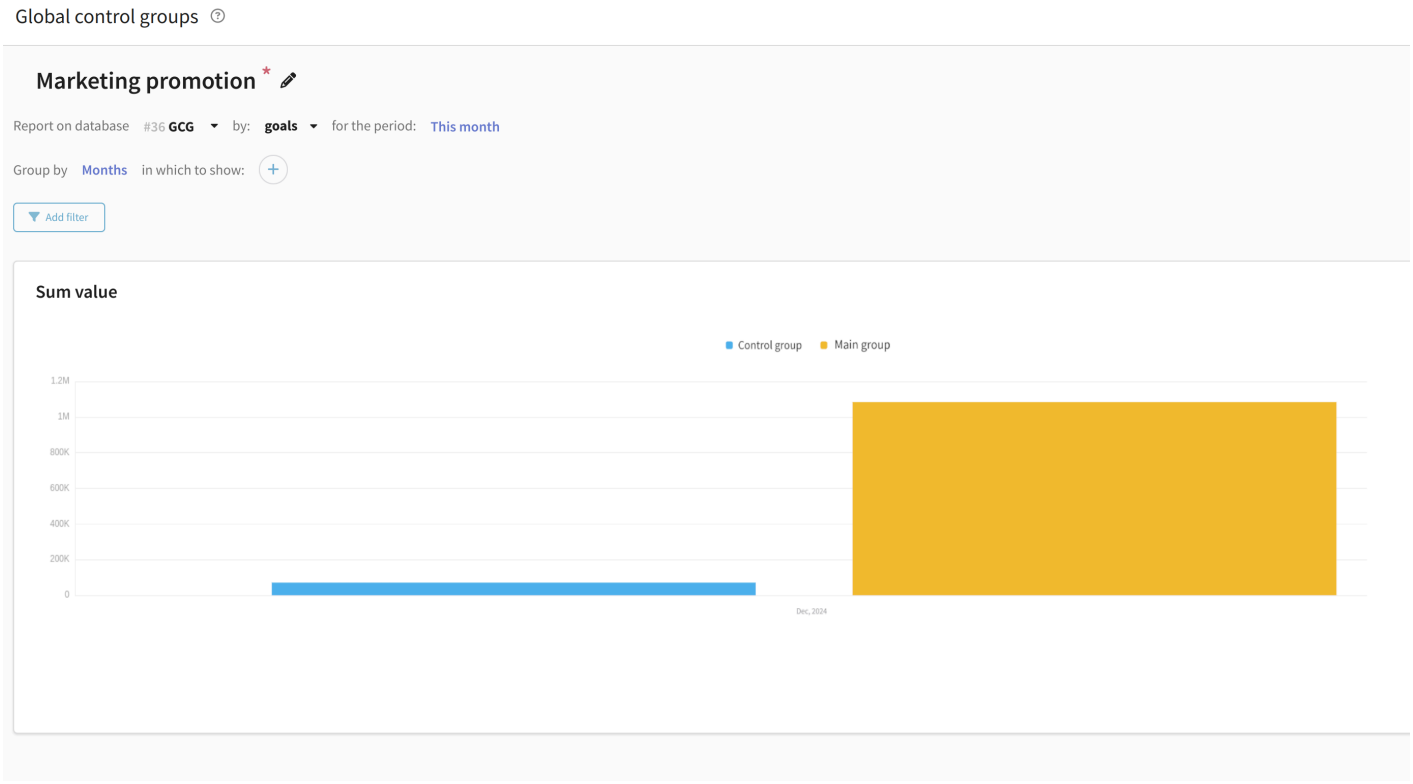
For example, if you select a text field and define 3 groups, then add a numeric field with 4 groups, the total number of combinations will be: 3 × 4 = 12 combinations.

If the number of combinations exceeds 64, the system will prevent you from saving the configuration — this limitation ensures stable performance and accurate sample generation.

GCG statistics

A comparison of metrics for the audience receiving communications and the control group audience can be viewed in Analytics - Global Control Group:

A graph comparing the main group to the control group will appear in front of you. You can select pixel and target to compare metrics derived from tracking pixels and sales channel for Market metrics:



Read more about analytics on global control groups here.

Subscription Manager

Description

Altcraft Subscription Manager is a page that allows subscribers to manage their profile data and subscriptions: unsubscribe, resume, or suspend them for a specific period (14, 30, 60, or 90 days). If your company already has a subscription manager in use, it can be integrated into the platform.

> INFO
>
> **The built-in subscription manager only works with the subscription to which the message with the manager link was sent.**

Subscription Manager Configuration

The subscription manager is automatically activated when a profile database is created. In the required field Subscription Manager Title, specify the name that users will see when entering the interface. In the adjacent field, select the interface language for the manager.

> INFO
>
> **The language of the manager is set in the profile database settings and does not depend on the user's region or browser language. If English is selected, the interface will always display in English.**

When users follow the link, they are directed to the built-in subscription manager page. By default, the manager interface displays the main recipient details: First Name, Last Name, Email, and Phone:

Configuration of Displayed Fields

In the subscription manager, you can configure the visibility of profile fields. In addition to the required email and phone number, you can enable the display of standard or custom fields by simply ticking the corresponding options in the Subscription Manager column.

> CAUTION
>
> **Any changes made by the user to the displayed fields in the subscription manager will automatically update the corresponding data in their profile. Make sure that modifying these fields does not cause any errors.**

After saving changes, the selected fields will appear in the subscription manager:

Categories and Additional Settings

Subscription Categories

Categories help detail the type of content to which the user is subscribed. If a user unsubscribes from one category, they will continue to receive content from others.

Unlike the subscription to the resource, categories have only two statuses (Subscribed/Unsubscribed). The status can be manually changed in the profile or via API requests. To allow the subscriber to manage a category on their own, add it to the subscription manager:

After saving changes, the selected categories will appear in the Categories section of the subscription manager:

Learn more about resource categories in the article.

Custom Fields

Each resource channel can include additional fields, similar to those used in the profile database. These fields help store extra information about the subscription.

Creating Custom Fields

- **Go to resource settings**
- **Click Add Field and select its type:**
  - **Integer — positive and negative whole numbers. For example, you can specify order limits, bonus points, etc.**
  - **Float — positive and negative floating-point numbers. Examples: "Average Check", "Coefficient", etc.**
  - **String — text in any language, including numbers, special characters, and spaces. Unicode is used. Examples: "Order Comments", "Interests", etc.**

    - **Boolean — a field that allows you to choose between two options: True or False (Yes or No). In the subscription manager, it appears as a slider. Examples: “Notify about sale start”, “Receive exclusive offers”, etc.**

- **Select the fields that will be displayed in the subscription manager.**

> CAUTION
>
> **Recipients can modify the values of custom fields added to the subscription manager. Ensure that changing these fields does not lead to errors.**

In the subscription manager, the selected fields will appear in the Details section:

Learn more about custom fields in the article.

Subscription Manager Integration

Adding a Link to the Message Template

To add a link to the subscription manager:

- **Open the template editor**
- **Select Insert Link**
- **Choose the URL type Subscription Manager (preferences) link**

Using an Alternative Subscription Manager

To connect a third-party subscription manager:

- **Create a custom unsubscribe link**
- **In the Link field, enter the URL of your subscription manager**

Now, when using the unsubscribe link in the template, the recipient will be redirected to the subscription manager page.

> CAUTION
>
> **In this case, clicking the subscription manager link will automatically unsubscribe the profile from the resource. To avoid this, the chosen manager needs to be configured. Please contact your platform administrator for assistance.**

Communication channels

Client messaging

Altcraft Platform offers you 6 communication channels out of the box: Email SMS, Push (both browser and mobile app), as well as Telegram, MAX, WhatsApp*, Viber*™ and Notify. In various configuration options you will also get access to custom communication channels, like banners or social networks. This article provides you with instructions on their settings and debugging.

You import your customers' digital profiles into a Profiles database. They get subscribed to your messaging Resources — via one or several communication channels and contact data entries.

In Altcraft Platform each Mailing can be sent over one specific channel. If you need to orchestrate omnichannel messaging, you can organize cascade sending with Scenarios.

Content for all the available communication channels is created in a single Message template. Thus different mailings can use only one template that provides you with extra analytics intersections.

Email channel

	Email allows you to send the largest amount of content, for a letter can contain different design elements and interactive sections, the ones to improve customer engagement and retention rates. To start messaging over SMTP Email, you will need a sender (From) domain to include in databases and resources settings.

INFO

This article will help you configure a sender domain for effective mailing.

INFO

And this one will provide recommendations on how to interact with email providers.

SMS channel

	Short Message Services are just too great for transactional messaging to ignore them. All you need is your customer's phone number — and you will reach him or her even if no 3G+ network is available. No subscription confirmation required, no mass messaging restrictions. Most of the SMS messages are opened. **To send SMS through Altcraft Platform, you need to integrate the platform with a telecom service. Such services often provide you with CALLER_ID — your "from name", which must be specified in the Resource when setting up an SMS channel.**

INFO

Set up your first SMS mailing with the help of our Step-by-Step Guide!

This article contains information about ready-made telecom gateway integrations.

INFO

You can as well require a custom integration. Contact support@altcraft.com for details.

Push channel

	Push notifications are one of the best ways to communicate with your subscribers as they are virtually free: no fee for sending a push compared to telecom operator fee with SMS.

INFO

For the push channel to work, first configure and activate it in Administrative panel.

INFO

Web browser push configuration is described in this article.

INFO

Mobile application push configuration is described in this article.

Telegram channel

	There are 2 channels available for communication with subscribers in Telegram: **Telegram Bot and **Telegram Group.** Telegram Bot will allow you to send direct messages to subscribers who have entered into a chat with the bot. With the help of Telegram Group, you will be able to publish posts in a Telegram group or channel through a bot.**

INFO

This guide will help you organize sending direct messages to a subscriber in Telegram. Please note that when using the Telegram Bot channel, the platform only sends messages. Subscriber data is collected and transmitted to the platform on the client side.

INFO

To communicate with subscribers in a Telegram group or channel, use this guide.

WhatsApp channel

	WhatsApp* is an effective communication channel that allows you to build a dialogue with the client in real time. Through this channel, you can inform your subscribers about upcoming events and changes in order status. Open rate in the messaging app reaches 90%.

INFO

This guide will help you start your first WhatsApp* mailing.

Viber*™ channel

	Viber*™ is a fast and convenient communication channel. You can send promotional mailings, transactional and trigger messages to your subscribers with Viber*™. The cost of such sendings is cheaper than SMS.

INFO

The integration can be implemented through the providers Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto. This guide will help you create your first Viber*™ Mailing.

Notify channel

	Notify — an effective channel for business communication with the client. Send transactional and service messages via Vkontakte and Odnoklassniki.

INFO

For how to create a connection and how to set up integration, see the corresponding documentation articles.

Set up your first Notify mailing with the help of our Step-by-Step Guide!

MAX

CAUTION

When setting up integration with this communication channel, consider the internal rules of the MAX messenger.

	Two channels are available in MAX for communication with subscribers: MAX Bot and MAX Group. MAX Bot is used to send personalized messages to subscribers who have started a chat with the bot. With MAX Group, you can publish posts in a MAX group or channel via a bot.

INFO

Step-by-step guide explains how to organize sending personalized messages to a subscriber in MAX.

Note that when using the MAX Bot channel, the platform only sends messages. Collecting and transferring data about users who started a chat with the MAX bot to the platform must be implemented on the client side.

INFO

To communicate with subscribers in a MAX group or channel, use this guide.

Communication channel limitations

Feature	Email	SMS	Push	WhatsApp™	Viber™	Notify™	Telegram	Max	Custom Channels
MVT testing	✅	✅	✅	❌	❌	❌	❌	❌	❌
Web version saving	✅	✅	✅	❌	❌	❌	❌	❌	❌

Feature	Email	SMS	Push	WhatsApp™	Viber™	Notify™	Telegram	Max	Custom Channels
Subscription from scenario	✅	✅	❌	❌	❌	❌	❌	❌	❌
Suppression lists	✅	✅	❌	❌	❌	❌	❌	❌	❌
Click map	✅	❌	❌	❌	❌	❌	❌	❌	❌

For more information on custom channel restrictions, check the administrator documentation.

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.***

***** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

First WhatsApp Mailing

	WhatsApp* is an effective communication channel that allows you to build a dialogue with the client in real time. Through this channel, you can inform your subscribers about upcoming events and changes in order status. Open rate in the messaging app reaches 90%. This guide will help you start your first WhatsApp* mailing. </td> </tr>

Step 1. Connect WhatsApp Business API through a provider

To send bulk messages in WhatsApp*, you need to connect the WhatsApp* Business API. In Altcraft Platform, such a connection is possible through such official providers as Edna or 360dialog.

Register an account with one of the providers.To do this, you need a phone number and a verified Facebook* Business Manager account.

TIP

If you want to register a WABA account through Edna, leave a request at this link.

360dialog collects applications through this form.

TIP

A phone number must be clean. This is a number that has not been registered for WhatsApp* Business API in the past, and is currently not registered on the WhatsApp* mobile app.

CAUTION

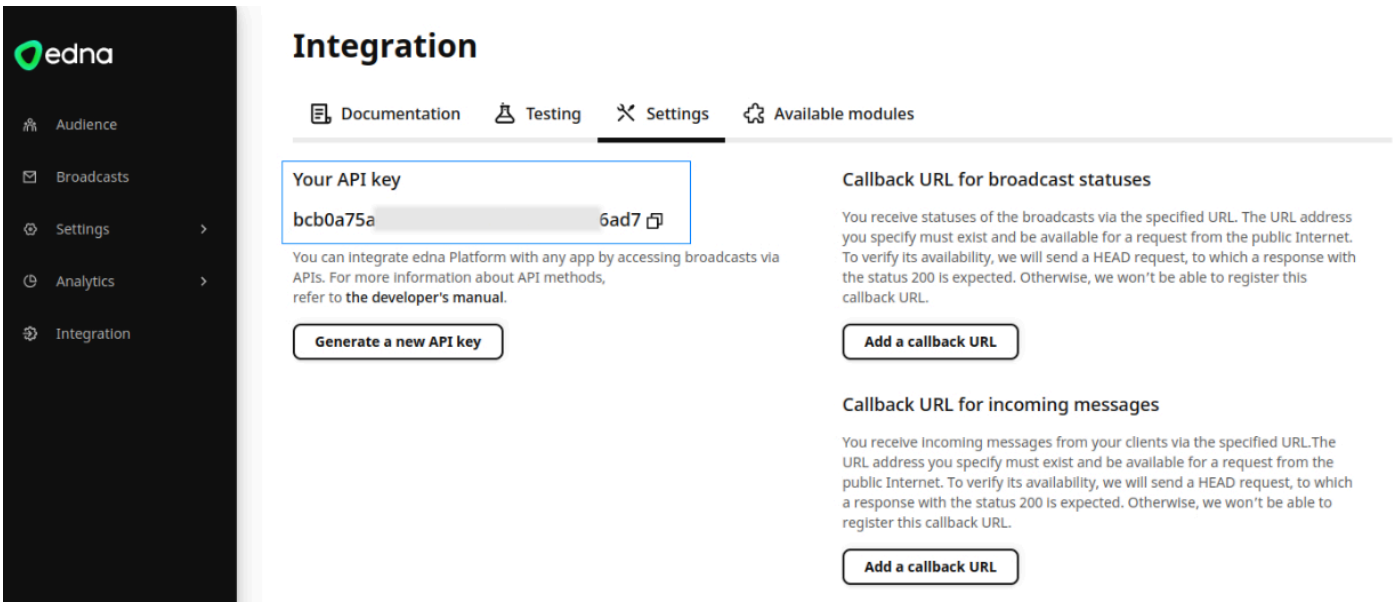
Integration with WhatsApp* Business API, as well as sending messages, are not charged additionally on the Altcraft side. However, you will need to pay for the services of your provider (360dialog or Edna). In addition, the cost of sending messages on WhatsApp* is also charged separately. You can find out more about the cost of WhatsApp* messages here.

Step 2. Obtain an API Key

The API key can be obtained in the provider's personal account:

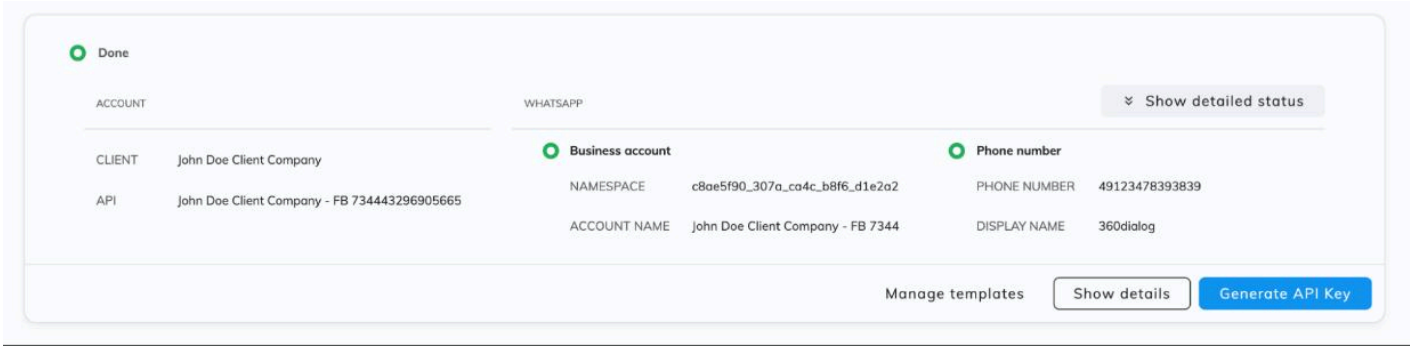
Edna

In your Edna account, go to Integration —> Settings:



360dialog

In 360 Client Hub go to *WhatsApp Accounts and click Generate API key:**

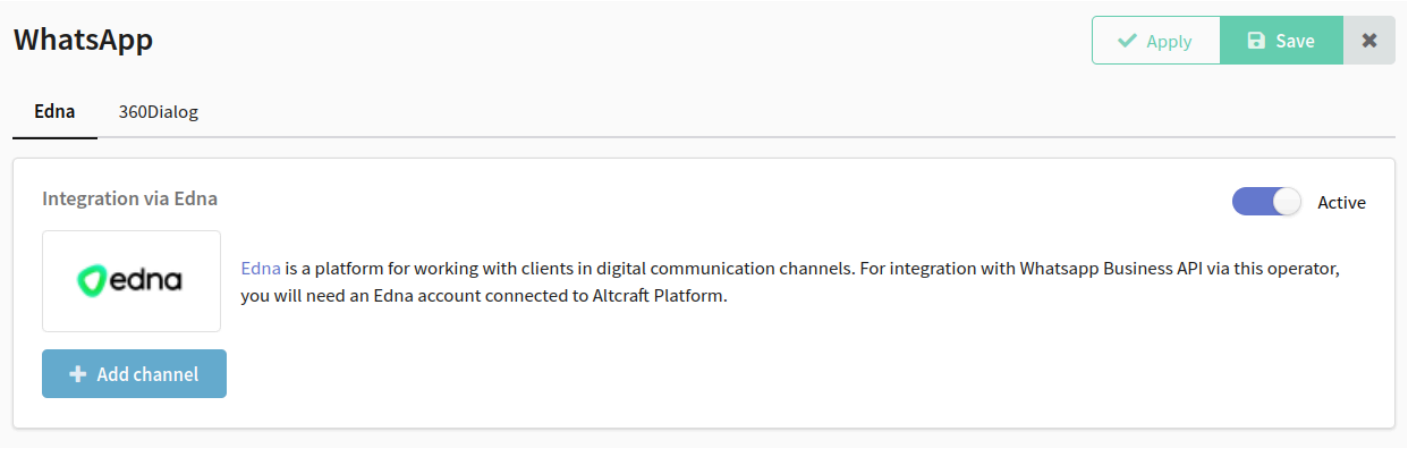


Step 3. Set up integration

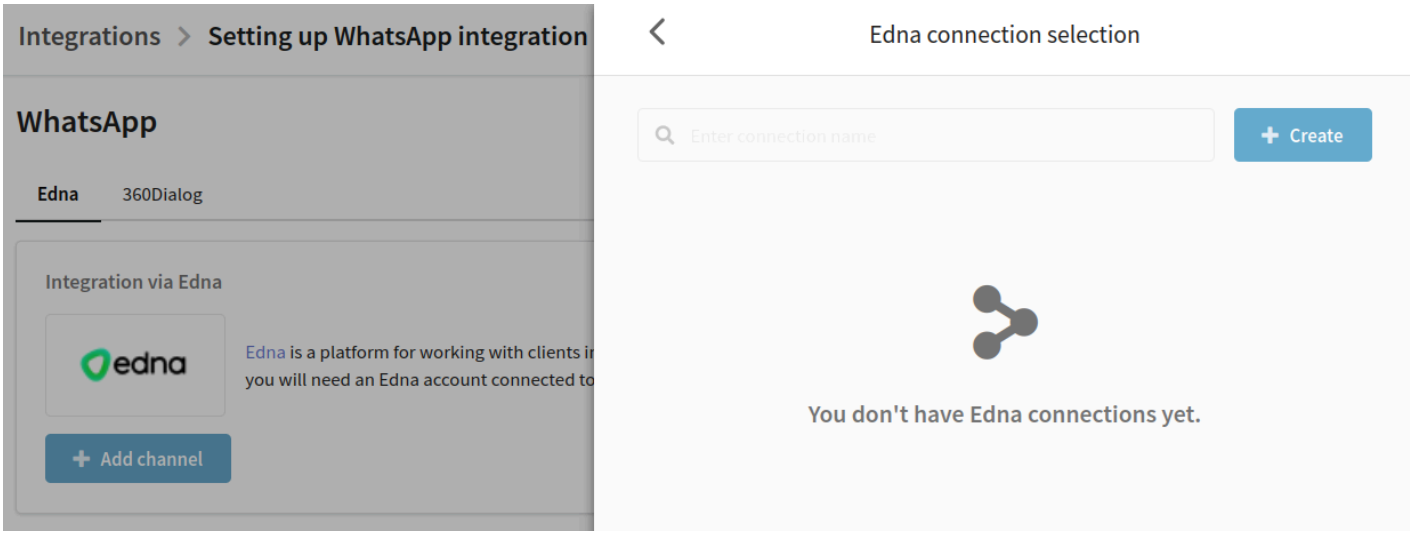
Go back to the platform and open Integrations. Select the type of integration — WhatsApp*. Switch to the tab of your provider.

Edna

Activate the integration using the toggle in the top right corner. Click Add channel:

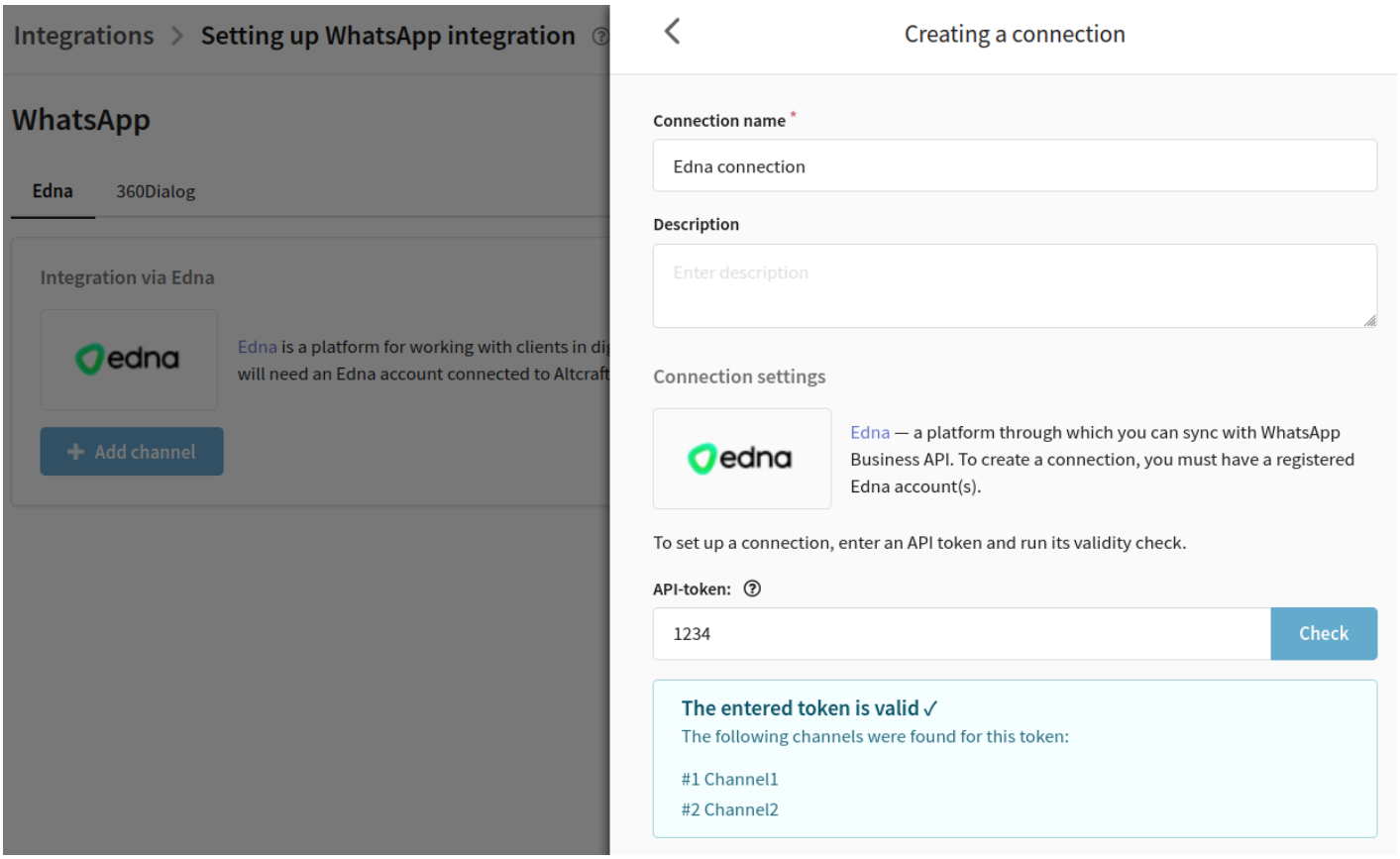


For integration, you need to link your WhatsApp* Business API account to Altcraft Platform. Linking accounts is done through connections. Click Create to set up a new connection:

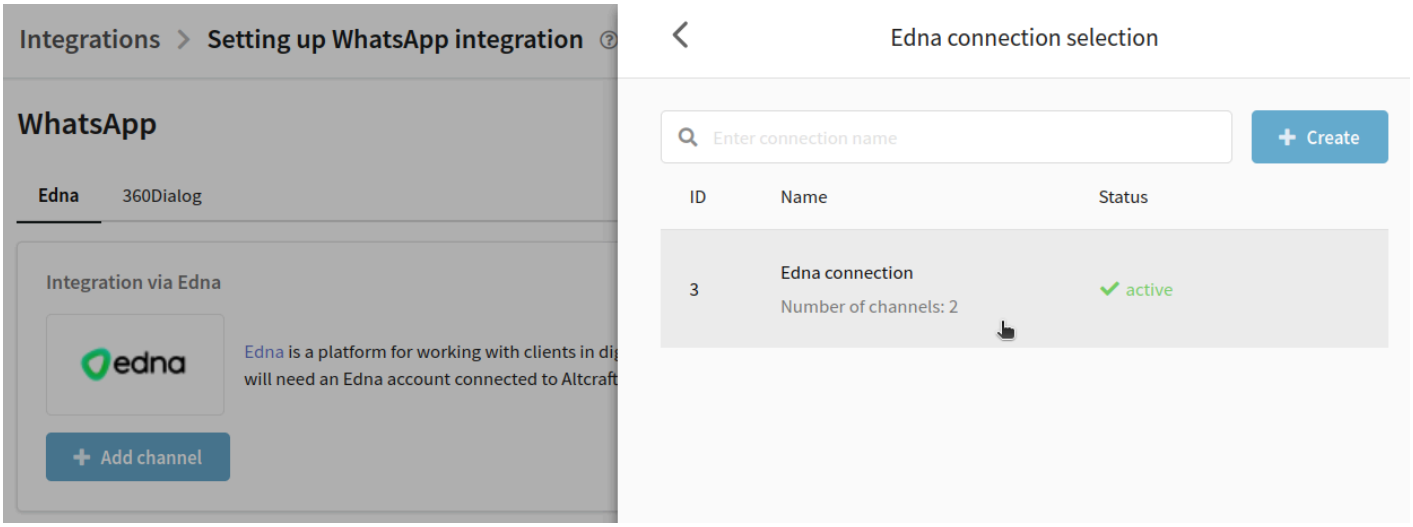


In the new window, enter a name and description for the connection.Enter the API token generated in the provider's personal account and run a check to confirm its validity.

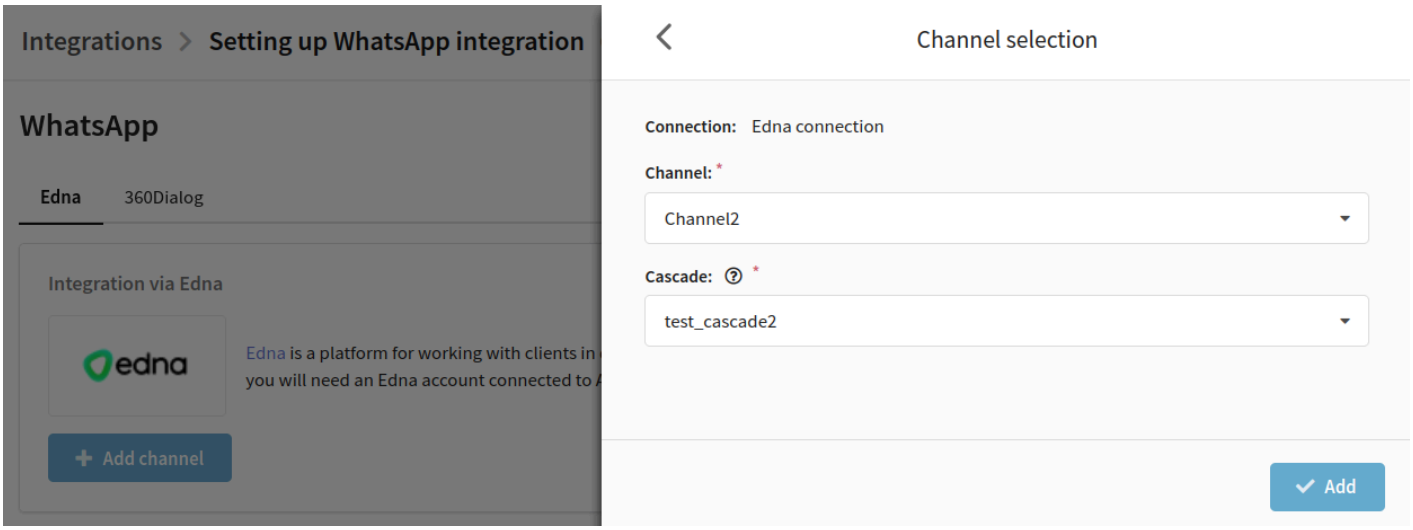
If the token is valid,the window will display data on the available channels in the account:



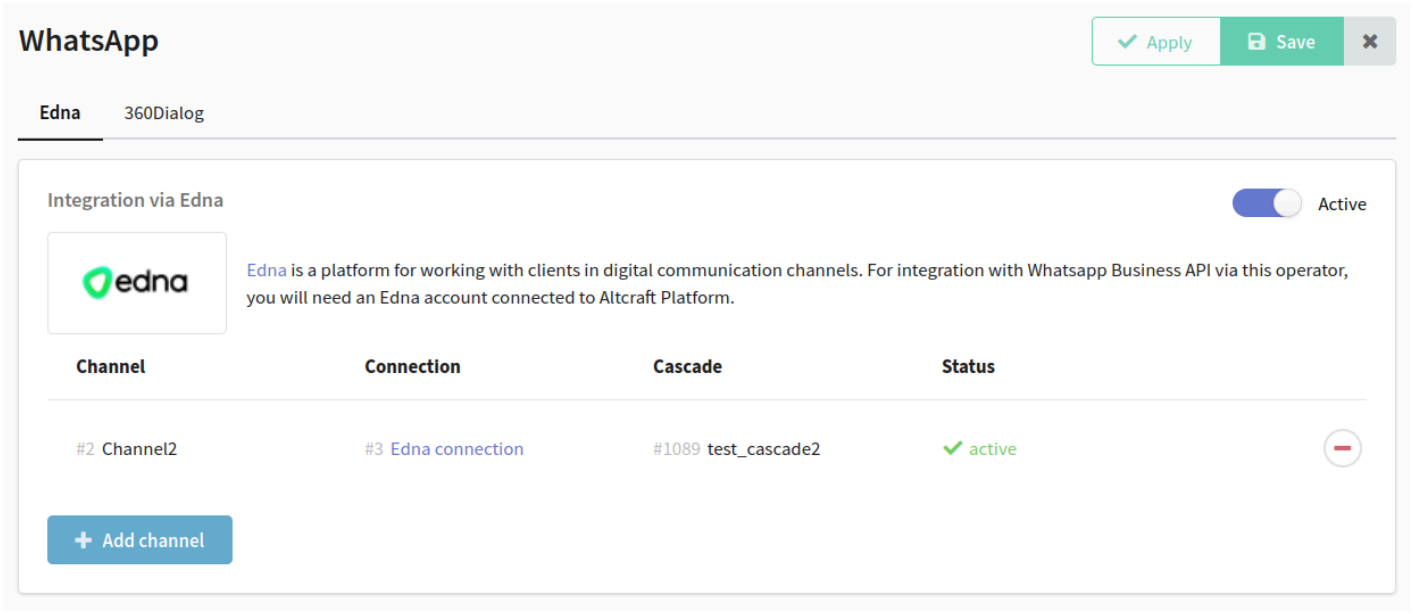
Next, click ✓Create in the lower right corner to save the connection. You will be taken back to the list of connections. Select the one you've just created:



Next step is to select a channel and cascade. The channel must be in the "Correct" status, and the cascade must use only one channel. Then click Add in the bottom right corner.



Data on the added Edna channel will be displayed in the integration:



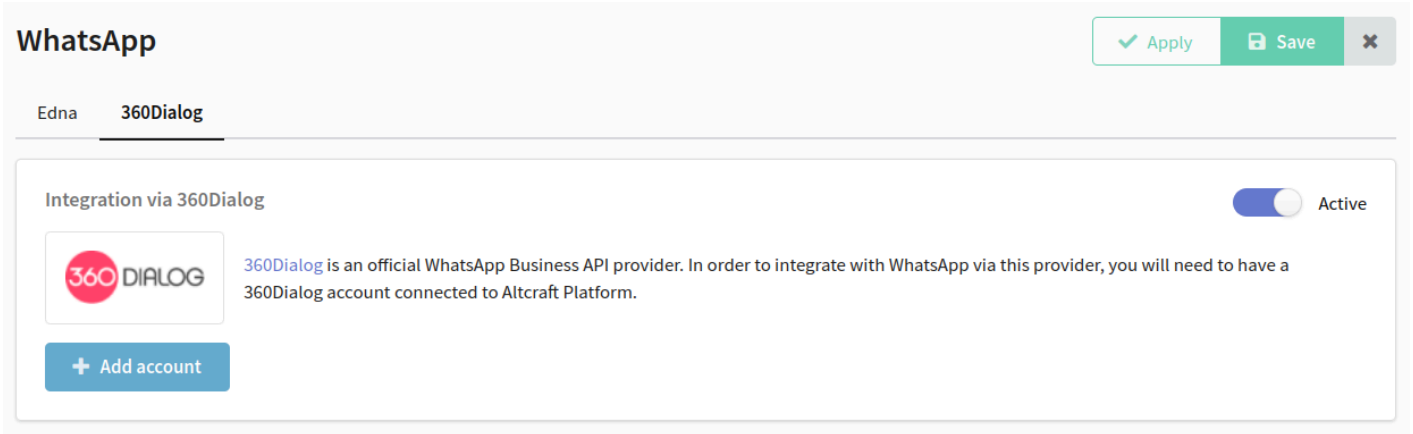
Save the integration using the buttons in the top right corner.

TIP

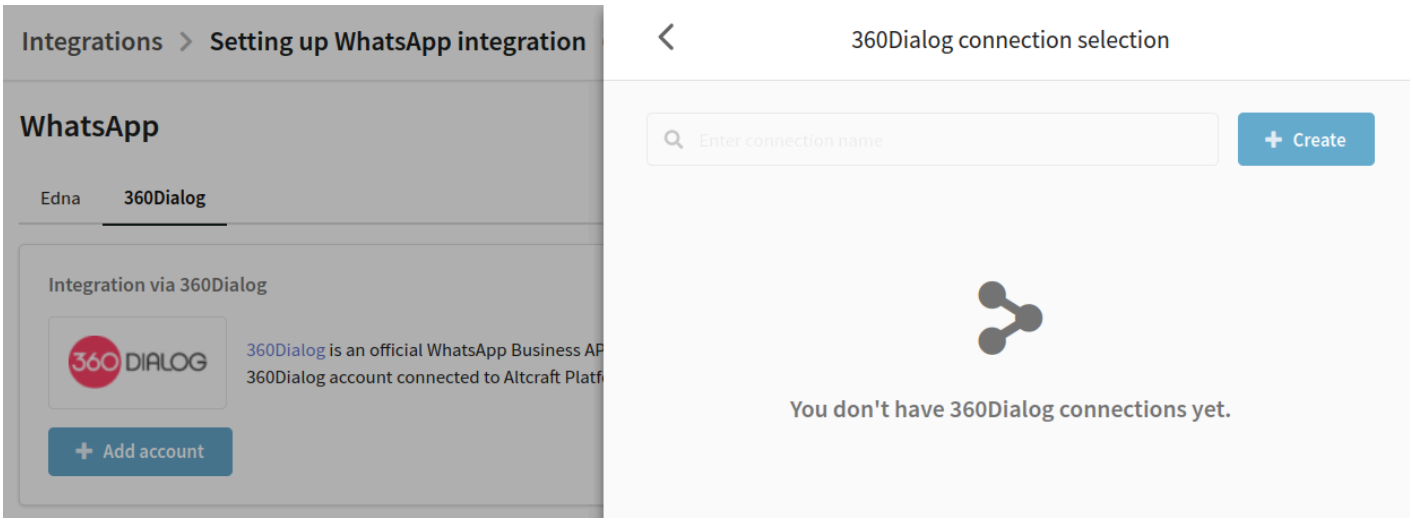
If necessary, you can add multiple channels for Edna integration.

360dialog

Activate the integration using the toggle in the top right corner. Click Add account:

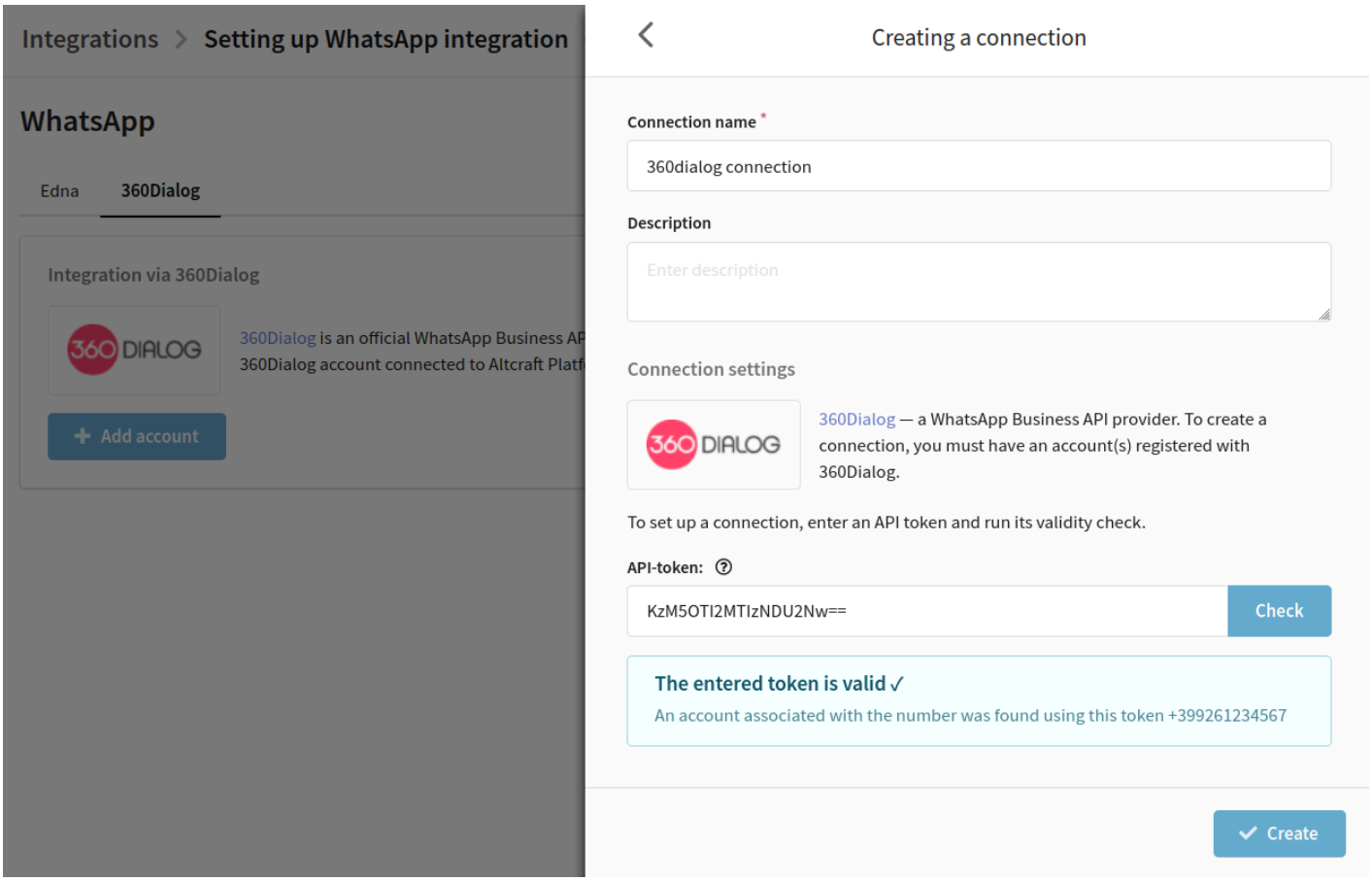


To set up the integration, you first need to link your WhatsApp* Business API account to Altcraft Platform. Linking accounts is done through connections. Click Create to set up a new connection:

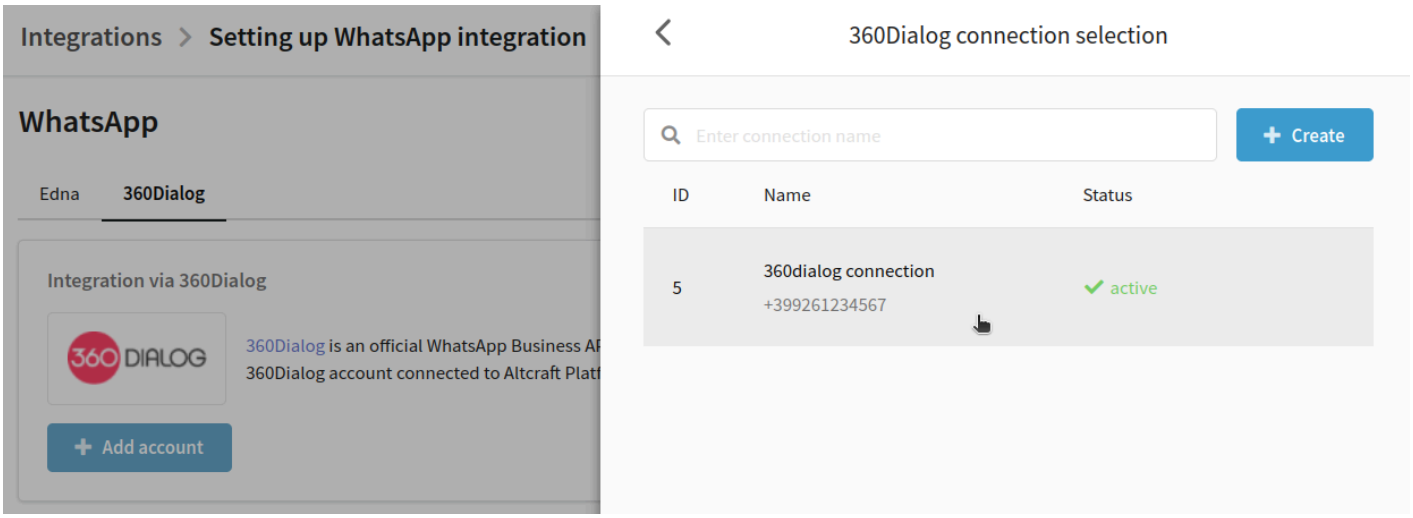


In the new window, enter a name and description for the connection.Enter the API token generated in the provider's personal account and run a check to confirm its validity.

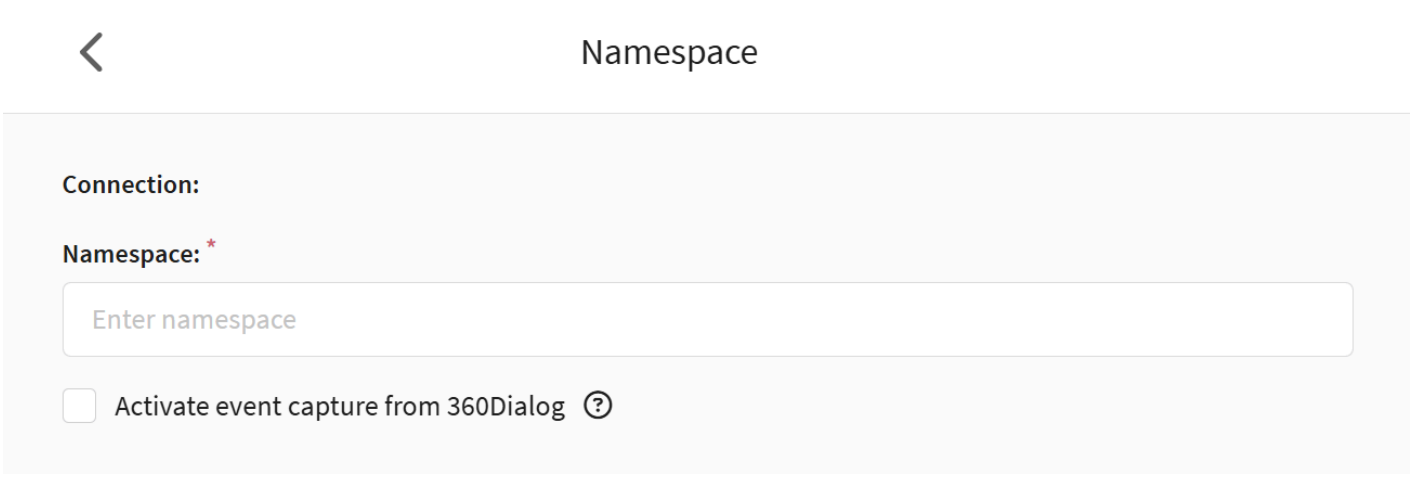
If the token is valid, you will see the phone number attached to the account you are connecting to:



Next, click ✓Create in the lower right corner to save the connection. You will be taken back to the list of connections. Select the one you've just created:

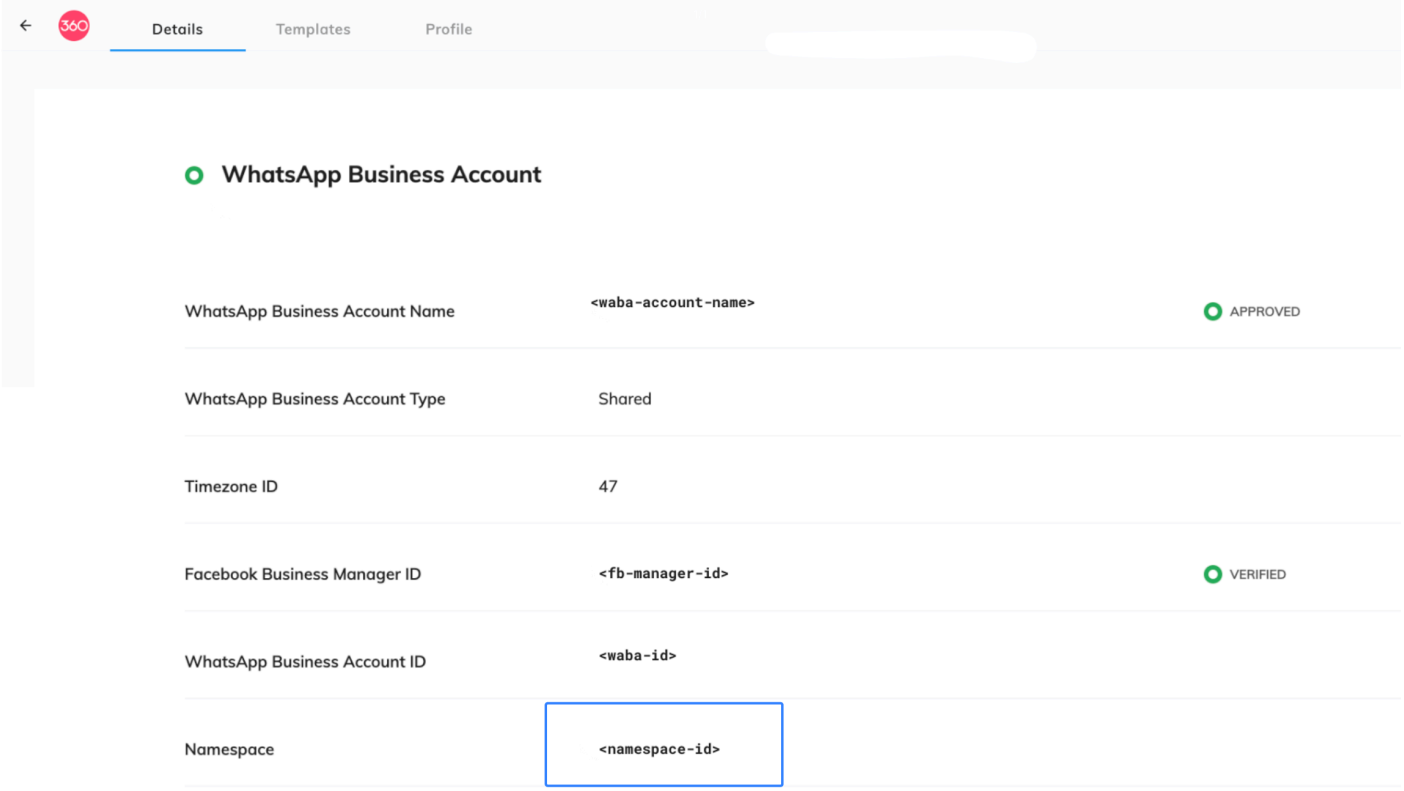


Fill in the Namespace field:

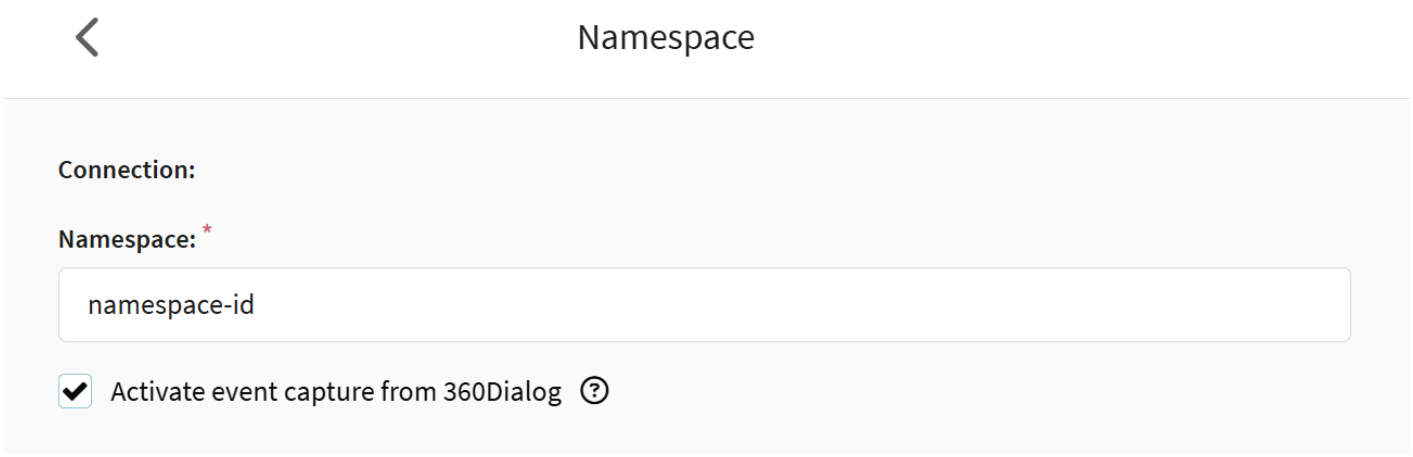


Namespace is a free string. Each 360dialog provider account stores the templates created for it and bound through this operator. Namespace can be viewed in the provider's personal cabinet in the account settings.

Go to the settings of your personal cabinet 360Dialog:



Copy the Namespace value from there, go back to the platform and paste it into the required field:



If you want, you can activate the Capture events from 360Dialog option.

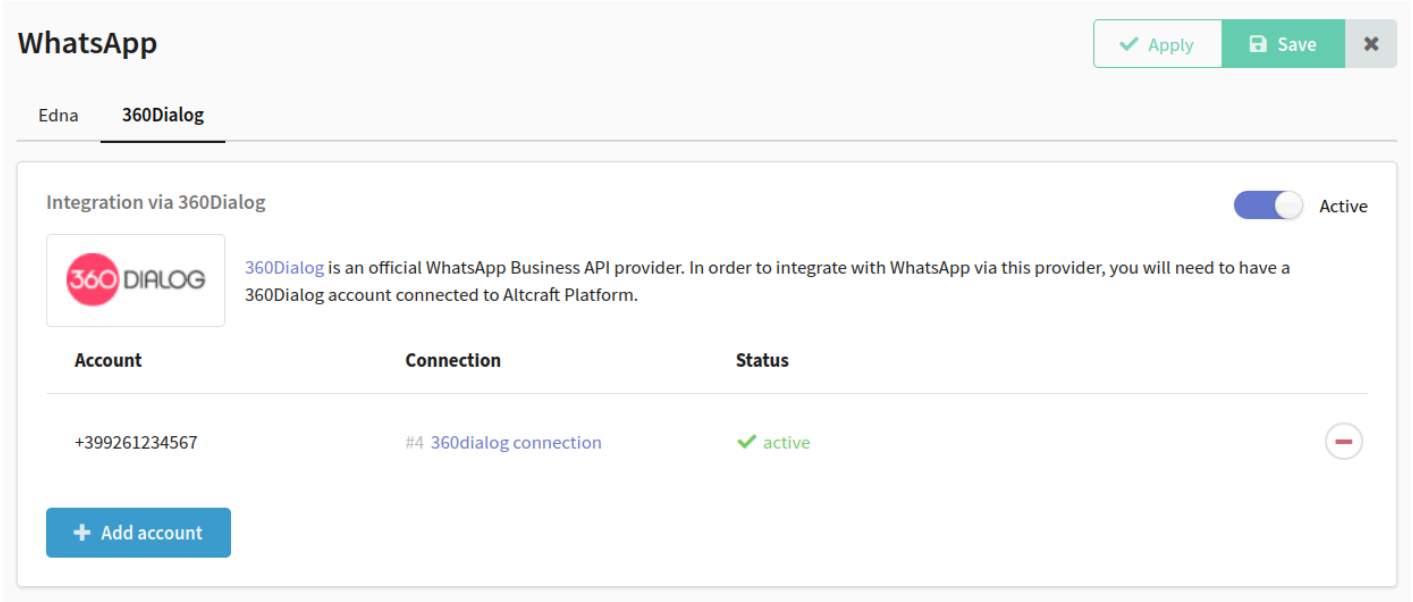
The Activate event capture from 360Dialog option enables sending WhatsApp event data to the Altcraft platform from 360Dialog.

If it is enabled, after saving the integration, a special link (Callback URL) will be automatically added to 360Dialog, through which Altcraft will receive information about message statuses, user actions and other events.

If the transmission of this data is not required, disable the setting.

Click Add in the lower right corner.

The integration is set up. You will see the data for the connected account:



Save the integration using the buttons in the top right corner.

TIP

If necessary, you can add multiple channels for Edna integration.

Step 4. Subscribe customers to WhatsApp channel

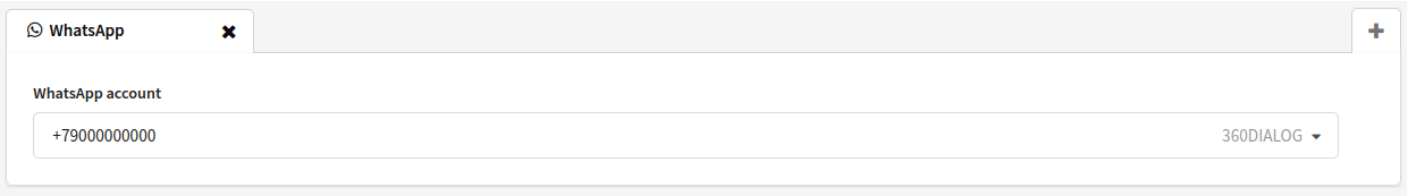
Sending a message to a user in WhatsApp* requires:

- **the user's phone number to be in the database;**
- **the user's profile to subscribe to the resource's WhatsApp* channel;**
- **the user's consent to receive messages from your company.**

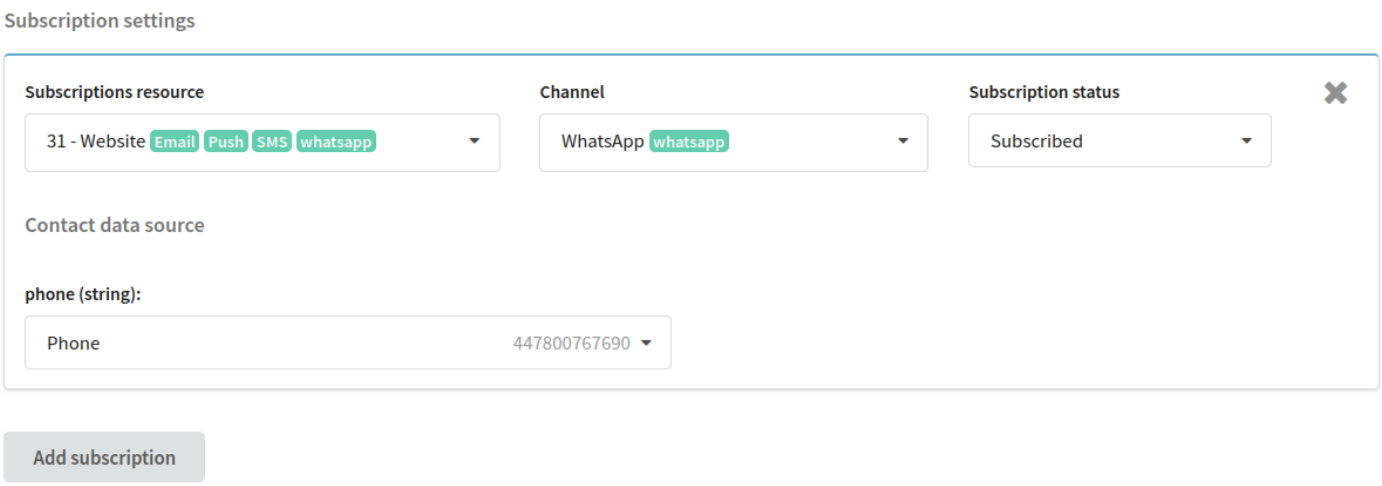
CAUTION

The profile phone number to which you send messages must be stored in international format. The following format must be used: +79001234567.

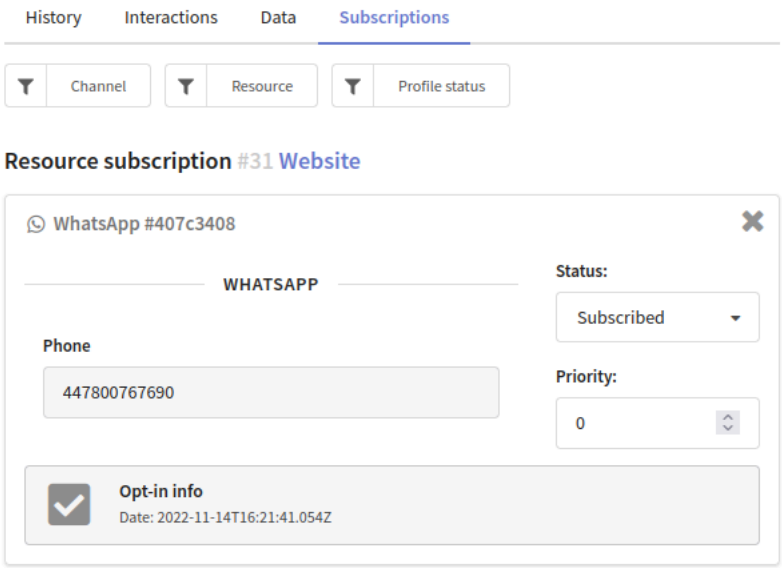
First, select or create a profile database that will store subscriber data, including phone numbers. Then create a new resource or edit an existing one by adding a WhatsApp* channel. To set up a channel, select the previously connected WhatsApp* account:



Next, [import subscriber data] into the platform, for example, via a file. When importing, add them a subscription to a resource with a WhatsApp* channel. Select the column in your file that contains phone numbers as the subscription source.



A [digital profile] will be created for each client. It combines personal data, activity history and subscriptions to resources. If a subscription to the WhatsApp* channel has been successfully added to the client during import, this data will be displayed in the client's history:



Step 5. Create a template in 360 Client Hub

There are two types of conversations in WhatsApp* Business:

- **User-initiated conversation. It is aconversation that starts in response to a user message. Whenever you reply to a user within the 24-hour service window, that message is associated with a user-initiated conversation. You can send free-form messages within this 24-hour customer service window.**
- **Business-initiated conversation. It is a conversation that starts from a business sending a user a message outside the 24-hour service window. Messages that initiate a business-initiated conversation require an [approved message template].**

Once the customer replies to the business's template message, the business can begin sending any type of message to the customer.

CAUTION

You can only use Altcraft Platform for business-initiated conversations. This means that you can only send approved templates messages (HSM).

A template message consists of fixed text and variables. Variables are text that can be changed before each mailing without the consent of Facebook*.

You need to create a template in your provider's account; in Altcraft Platform you can only add text into variables, set up buttons and add media files.

TIP

You can find more information about creating templates in your provider's documentation:

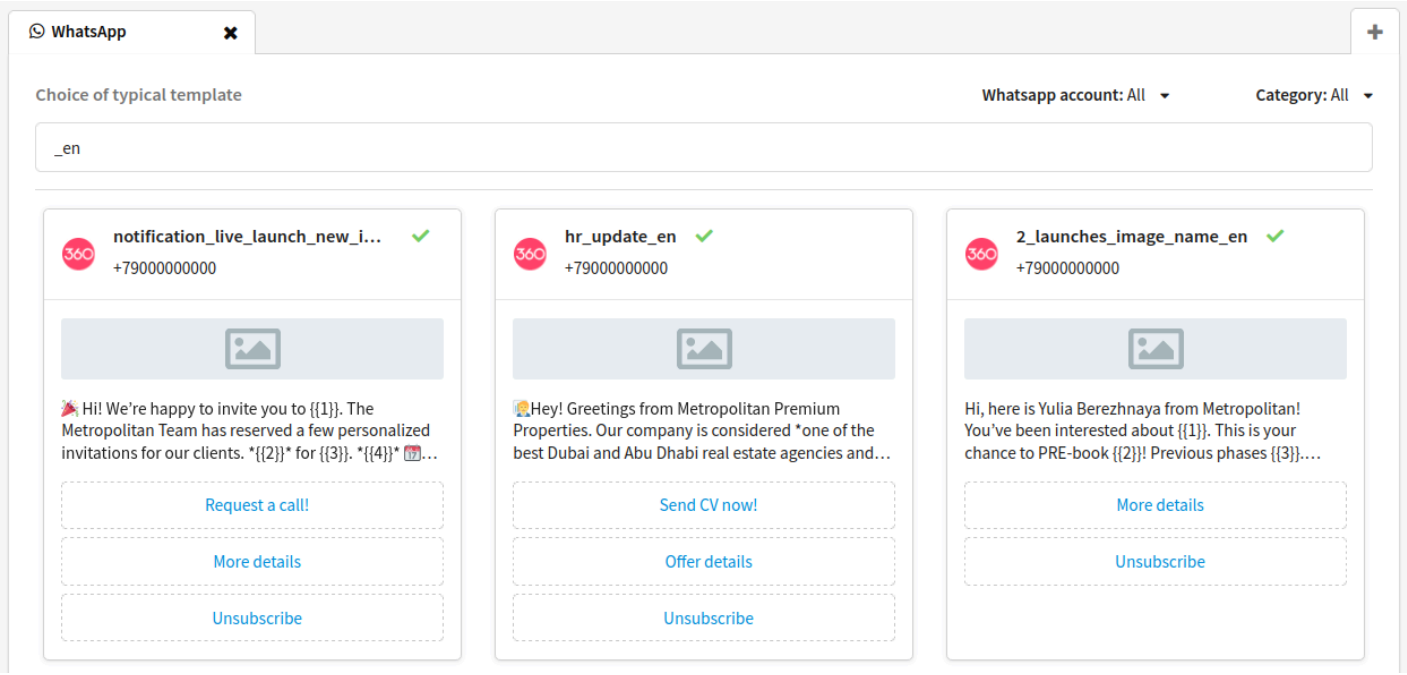
- **Edna: (https://pulse-alpha-six.vercel.app/docs/template/whatsapp/operator-template)**
- **360dialog: https://docs.360dialog.com/docs/360-client-hub/template-management-ui**

CAUTION

A template created in the provider's personal account must be verified by Facebook*. Mailings launched through Altcraft can only use templates approved by Facebook*.

Step 6. Create a message based on the template

Go back to Altcraft Platform, open the Templates section and create a new template for the WhatsApp* channel. All templates available in connected WhatsApp* accounts through edna or 360dialog are automatically available in the platform and displayed in the editor:



Choose a template and create a message based on it. To do this, fill in the variables, configure the buttons and add media files.

TIP

Here is a guide where you will find more details on how to create a template for the WhatsApp* channel.

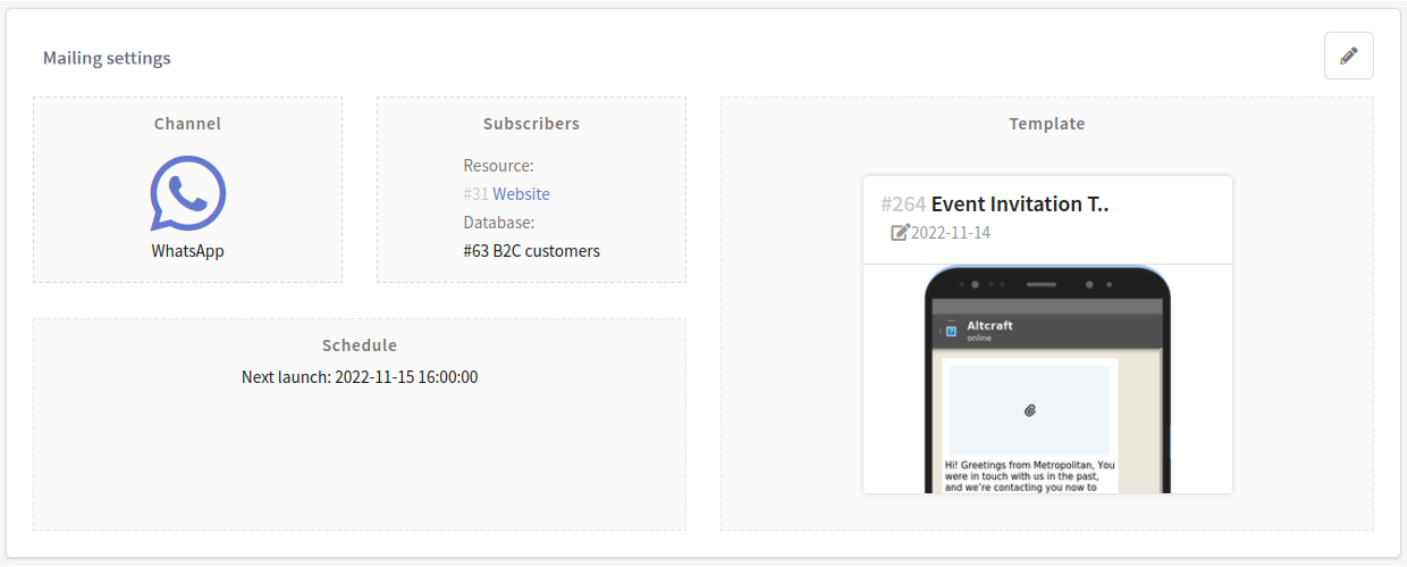
Step 7. Launch the mailing and evaluate its performance

Go to Mailings and select the mailing type:

- **Broadcast is a one-time ending of messages.**
- **Regular mailing isa periodic sending of messages according to a schedule.**

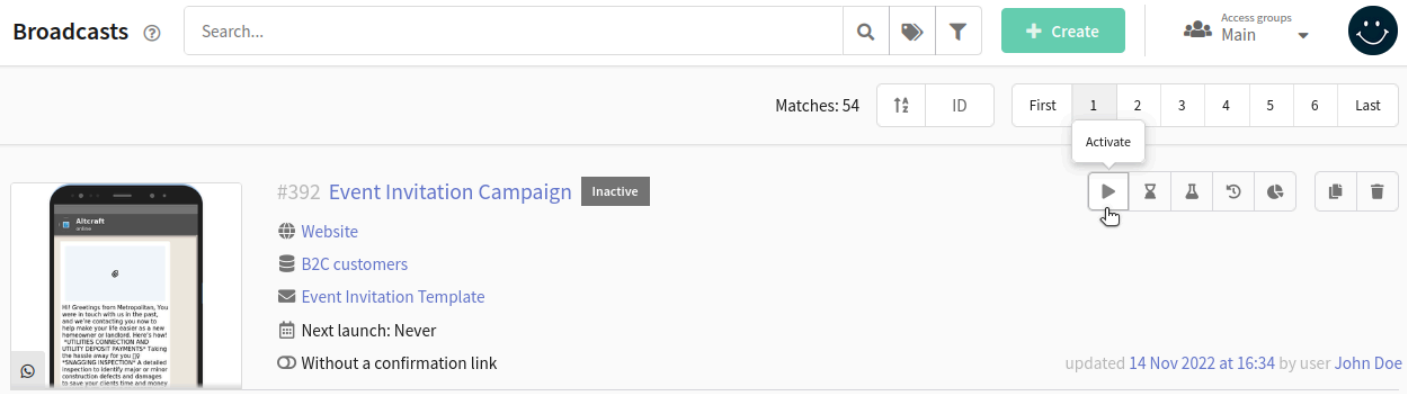
Set up a mailing:

- **select WhatsApp* as a communication channel**
- **select a previously configured resource and a recipient audience (database or segment);**
- **add a message template;**
- **set the schedule:**



Save the mailing. Find it in the list of mailings of the selected type and activate it by clicking

in the quick action bar.The active mailing will start sending messages when the scheduled launch time comes.



Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Description
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**
Audience source	**Source of contacts:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resource**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your mailings go to Analytics. The channel report will provide data for each mailing. You will receive information about the number of sent, delivered and undelivered messages, as well as the number of new subscriptions to the WhatsApp* channel. For the convenience of grouping, you can choose a preset of indicators for WhatsApp*.

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.*

First Viber*™ Mailing

	This guide will help you launch your first Viber*™ mailing. Integrating Altcraft Platform with Viber*™ is done through the **Devino Telecom** through the [**SMS Traffic**](https://www.smstraffic.ru/), [**MTS OmniChannel**](https://omnichannel.mts.ru/) or through the [**Rapporto**](https://rapporto.ru/). If your company doesn't have an account yet, please register one. </td> </tr>

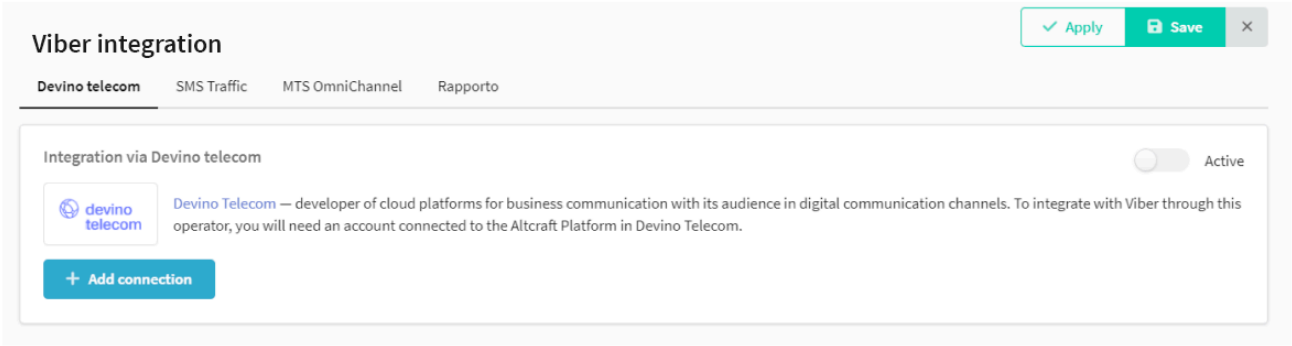
INFO

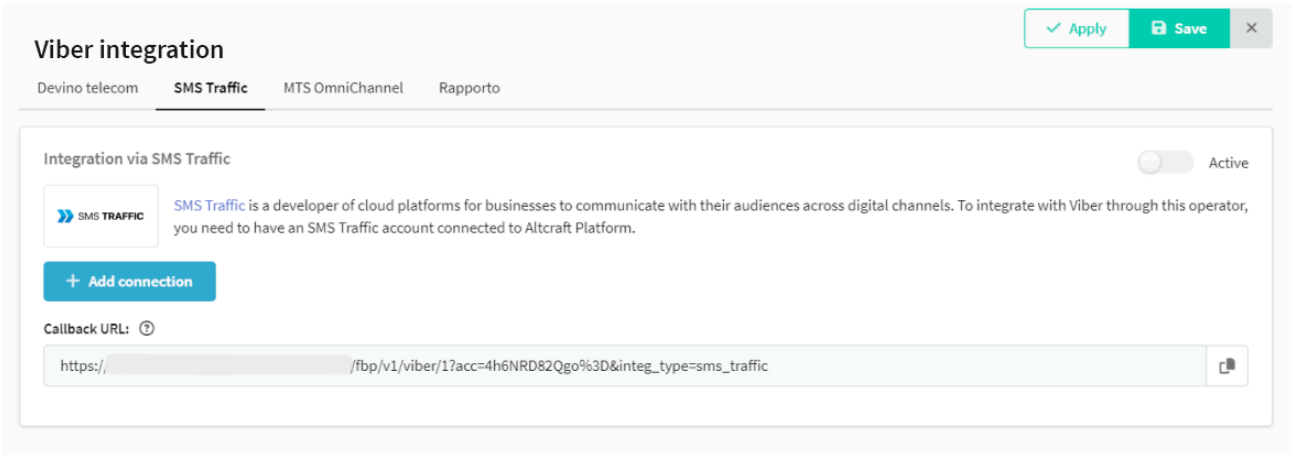
The cost of messages is determined by providers. Altcraft does not charge additional fees for sending messages to Viber*™.

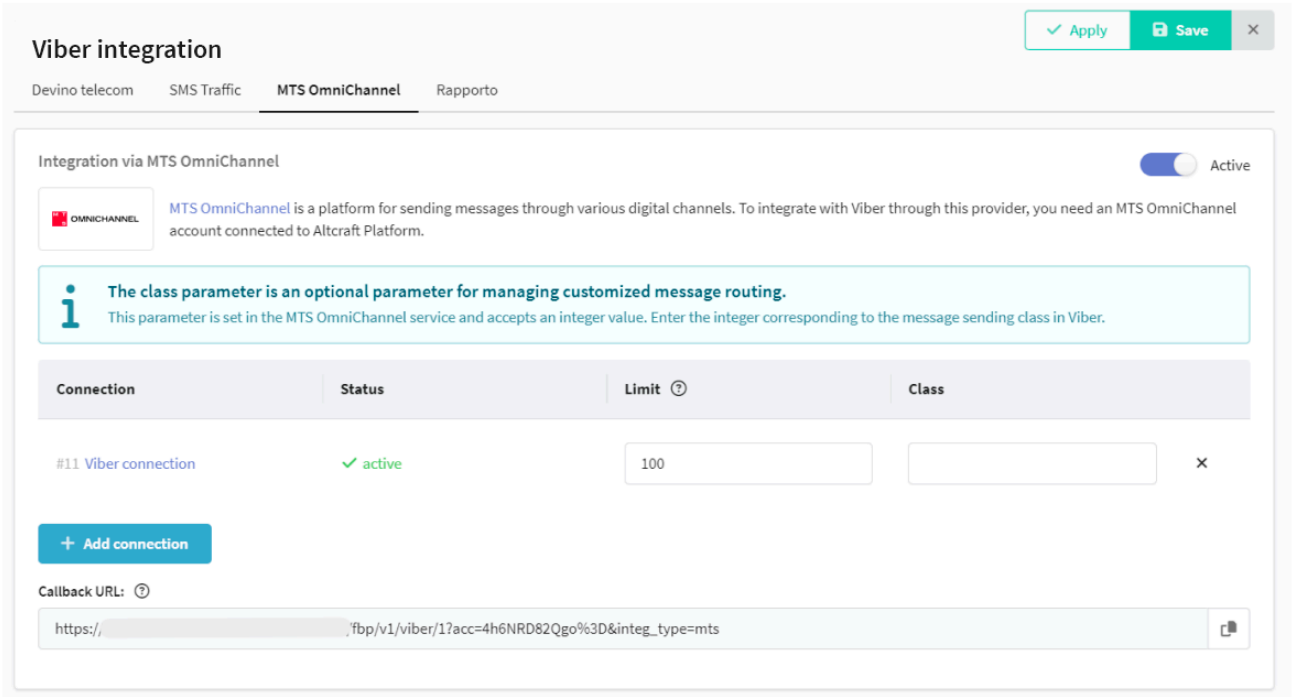
Step 1. Set up integration

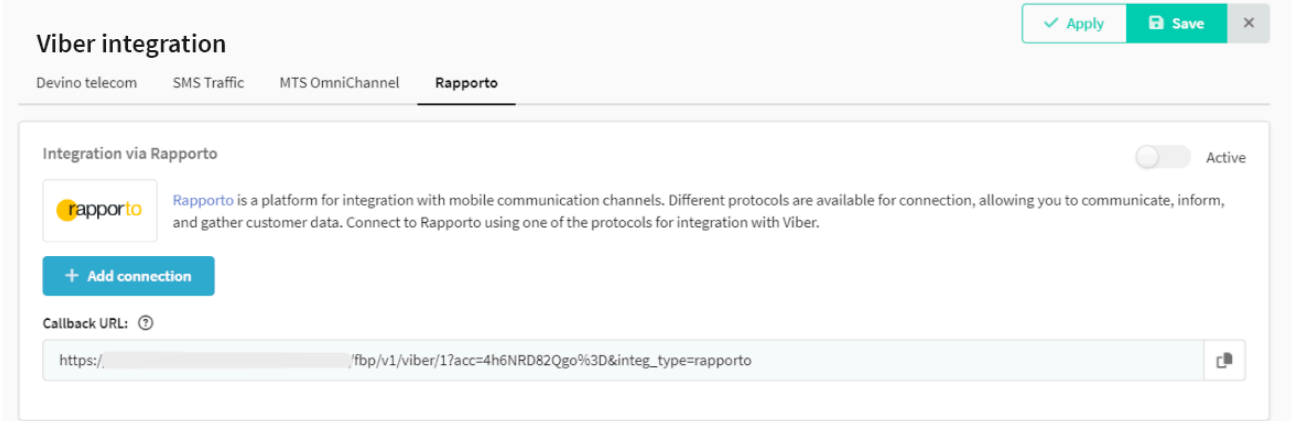
1. In the main menu, go to Integrations. Choose *Viber™* as the integration type.

2. Choose the provider — Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto.


Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via Devino telecom
Active
Devino Telecom — developer of cloud platforms for business communication with its audience in digital communication channels. To integrate with Viber through this operator, you will need an account connected to the Altcraft Platform in Devino Telecom.
Add connection

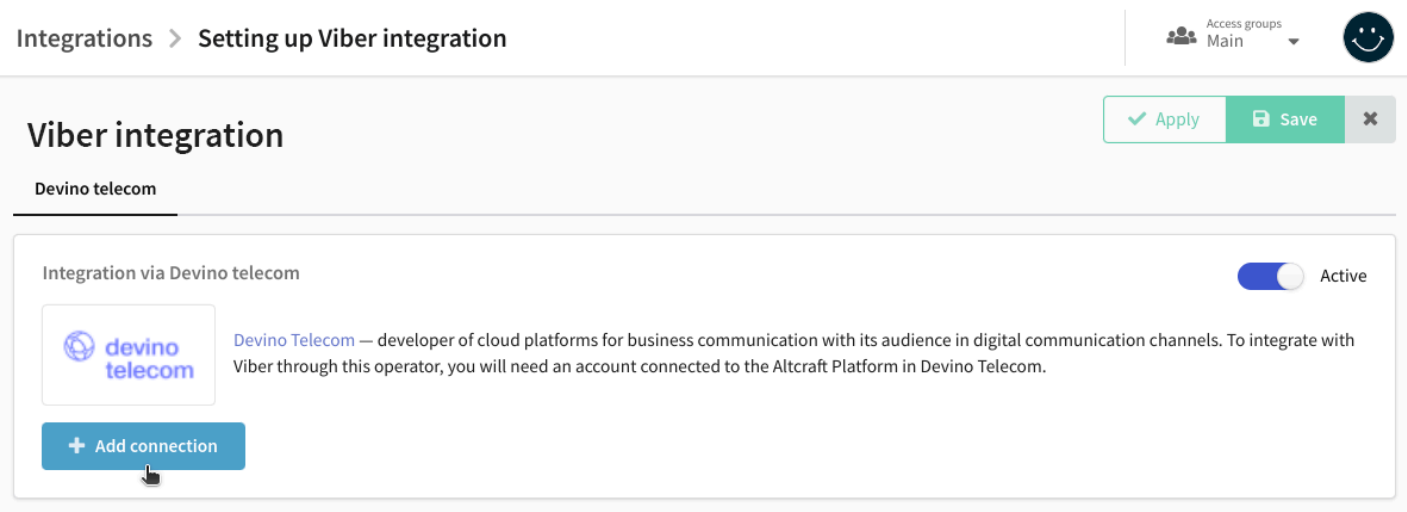

Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via SMS Traffic
Active
SMS Traffic is a developer of cloud platforms for businesses to communicate with their audiences across digital channels. To integrate with Viber through this operator, you need to have an SMS Traffic account connected to Altcraft Platform.
Add connection
Callback URL:
https:/ /fbp/v1/viber/1?acc=4h6NRD82Qgo%3D&integ_type=sms_traffic


Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via MTS OmniChannel
Active
MTS OmniChannel is a platform for sending messages through various digital channels. To integrate with Viber through this provider, you need an MTS OmniChannel account connected to Altcraft Platform.
The class parameter is an optional parameter for managing customized message routing.
This parameter is set in the MTS OmniChannel service and accepts an integer value. Enter the integer corresponding to the message sending class in Viber.
Connection
Status
Limit
Class
#11 Viber connection
active
100
Add connection
Callback URL:
https:/ /fbp/v1/viber/1?acc=4h6NRD82Qgo%3D&integ_type=mts

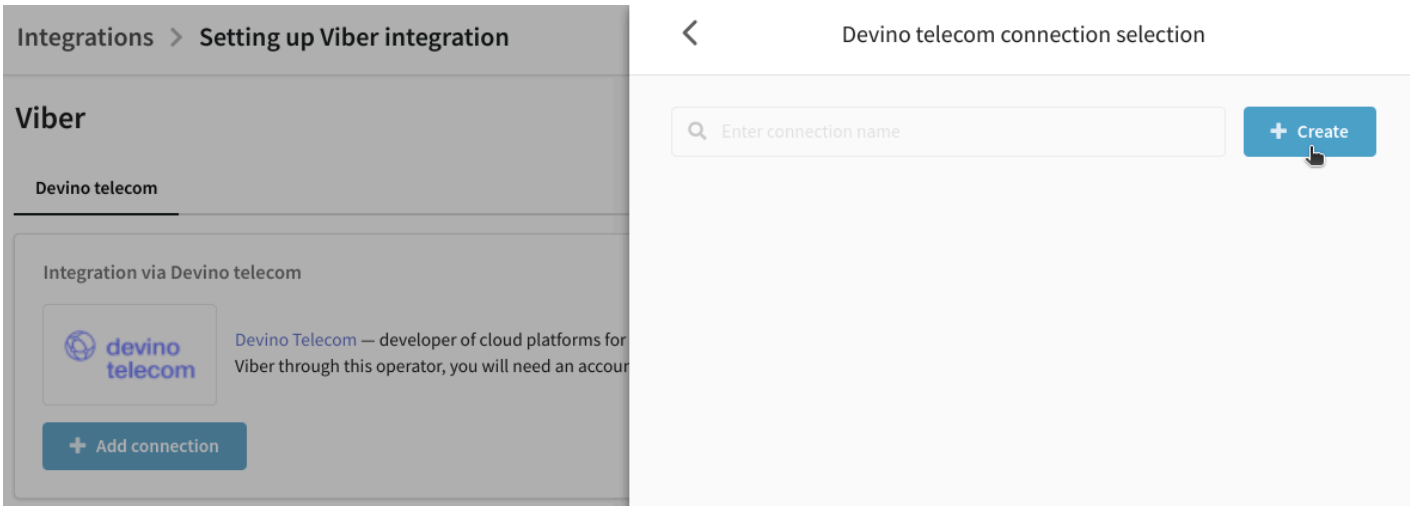

Viber integration
Apply
Save
Devino telecom
SMS Traffic
MTS OmniChannel
Rapporto
Integration via Rapporto
Active
Rapporto is a platform for integration with mobile communication channels. Different protocols are available for connection, allowing you to communicate, inform, and gather customer data. Connect to Rapporto using one of the protocols for integration with Viber.
Add connection
Callback URL:
https:/ /fbp/v1/viber/1?acc=4h6NRD82Qgo%3D&integ_type=rapporto

3. Activate the integration using the toggle switch in the top right corner.

4. Click Add connection:



5. A new window will appear on the right, showing available connections. A connection links your provider account with Altcraft Platform. Click “Create” to set up a new connection:

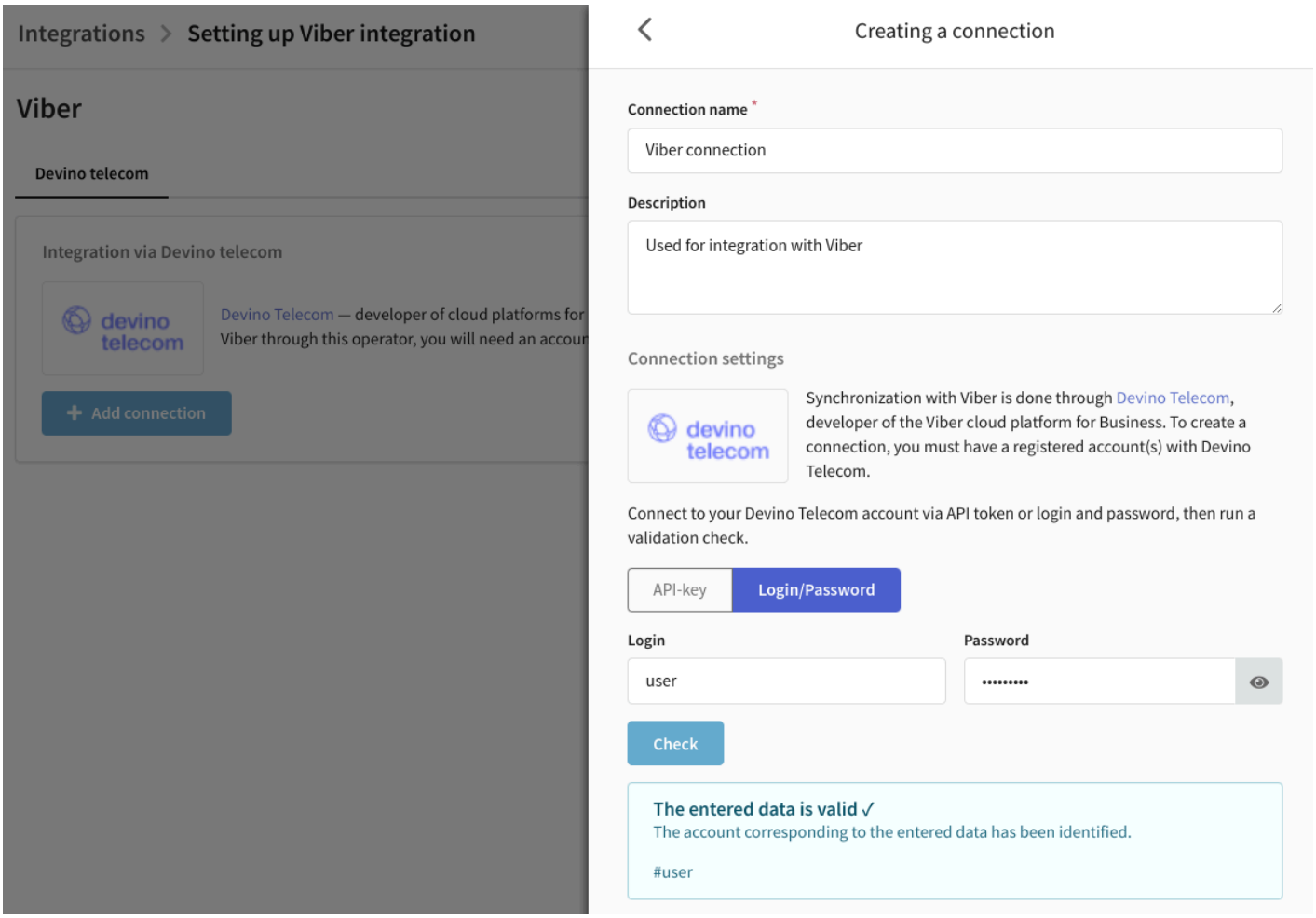


6. In the new window, enter the name and description of the connection.

7. Authorize in your provider account.

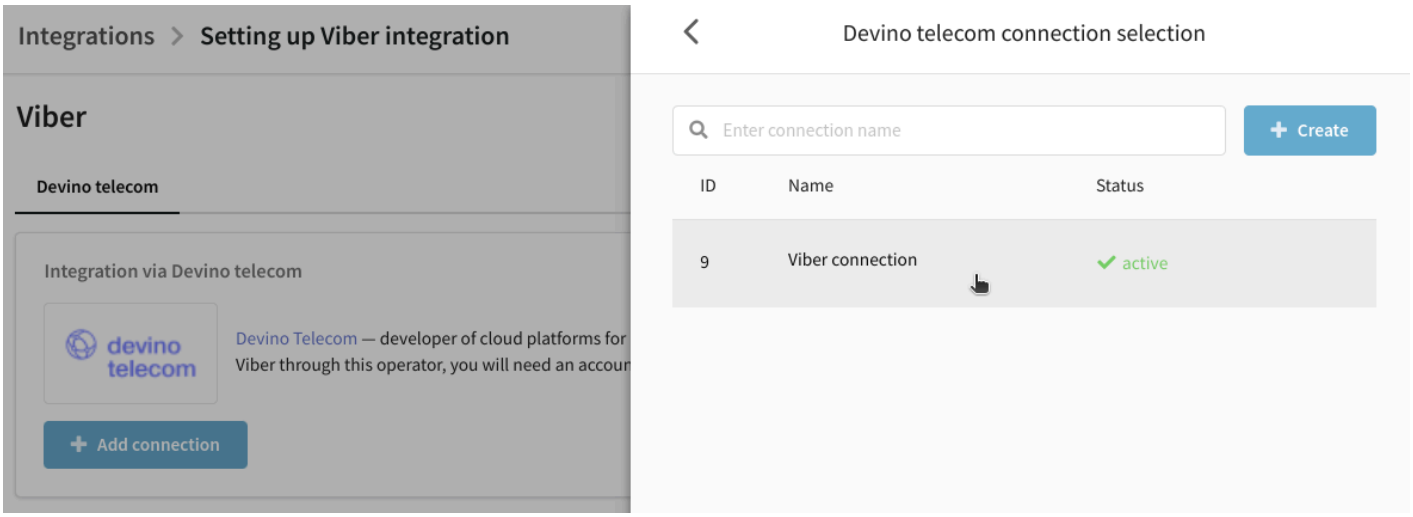
> TIP
>
> **For Devino Telecom there are two ways of authorization. You can learn more about creating of connection in this article.**

8. Enter the authentication details and click “Check” to verify their validity. If the data is correct, information about the successfully identified account will appear below.

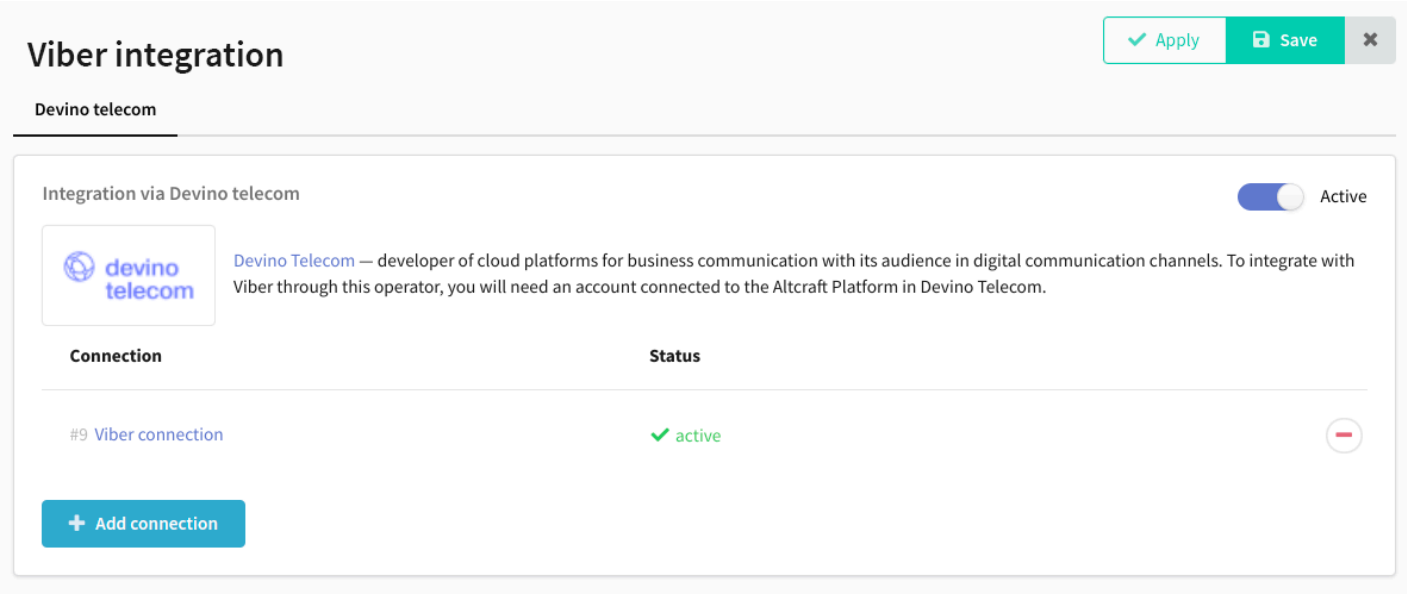


9. Click “Create” in the bottom right corner to save the connection.

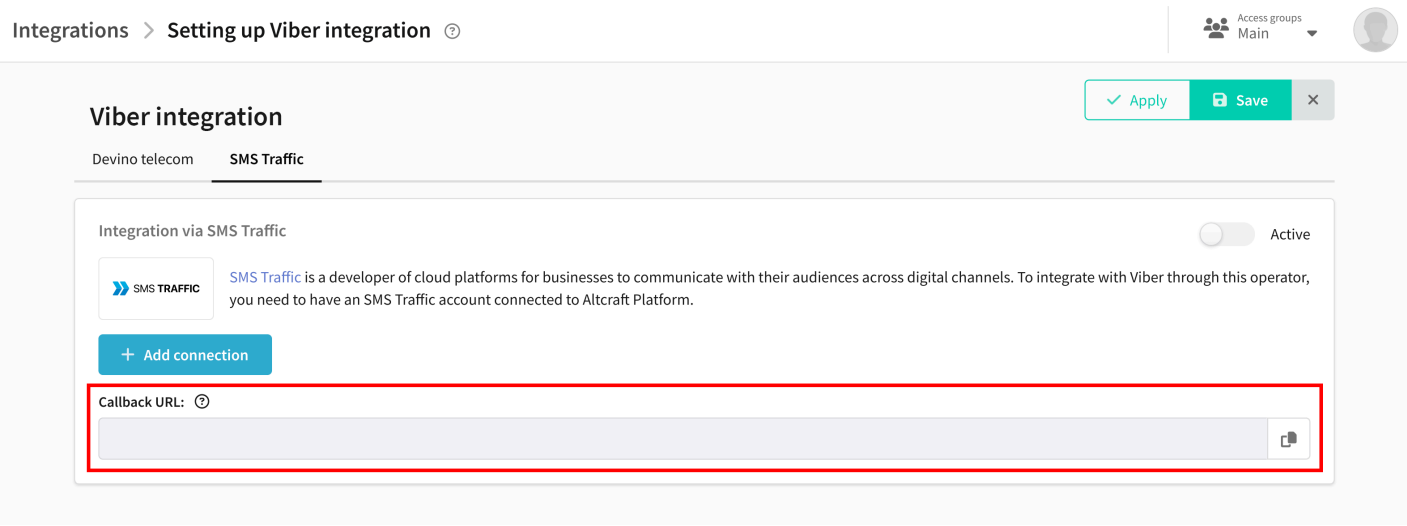
10. Go back to the list of connections, and select the one you just created.



11. Information about the added connection will be displayed in the integration section. Ensure that the connection status is “active.” If not, check your account status in the provider dashboard. Note that your account might be blocked if the subscription fee is not paid.



12. If you use SMS Traffic as a provider, you have to send your Callback URL from the integration to technical support



13. Save the integration settings using the buttons in the top right corner.

Step 2. Subscribe customers to the Viber*™ channel

To send a message to a user on Viber*™, you need:

- **The user's phone number in the database.**
- **The profile must have a subscription to the Viber*™ channel of the resource.**

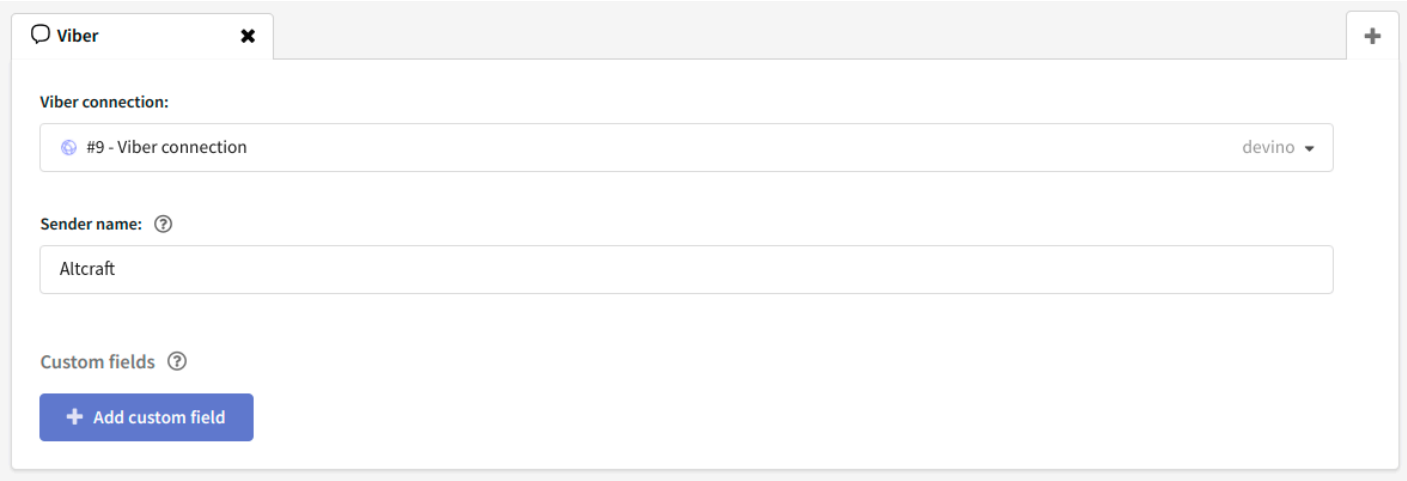
INFO

The profile phone number to which you send messages must be recorded in international format. Please use the following format: +79001234567.

To subscribe clients to a Viber*™ channel, follow these steps:

1. Choose or create a profile database where subscriber data, including phone numbers, will be stored.

2. Create a new resource or edit an existing one, adding a Viber*™ channel. To configure the channel, select the previously created Viber*™ connection and specify the sender's name. Subscribers will see this name when they receive a message. You can find the sender's name in your provider account, and it should be verified by your provider.

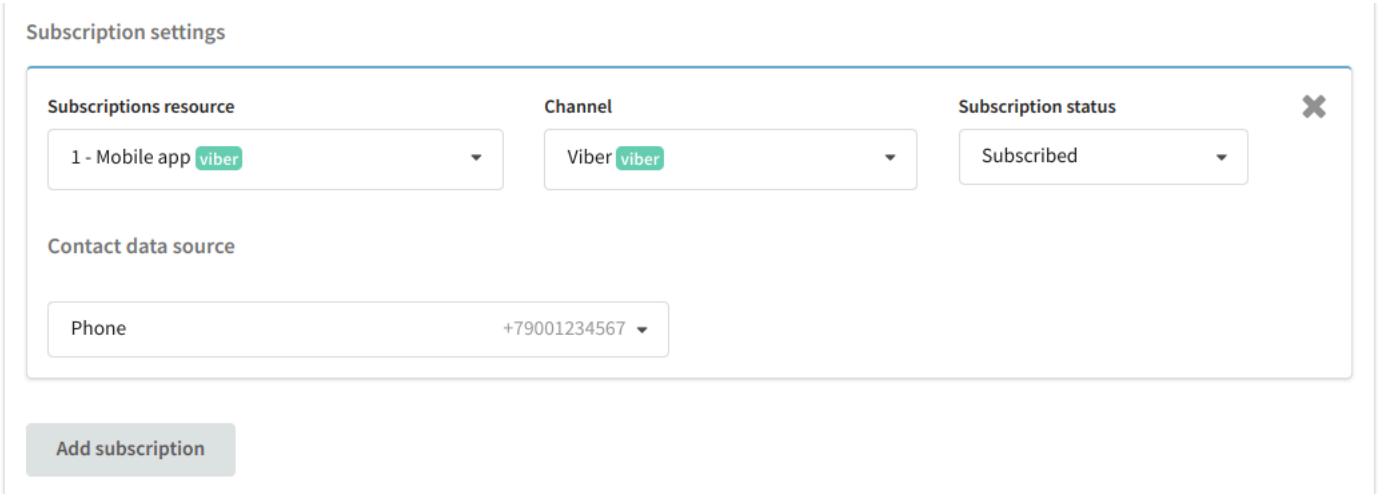


INFO

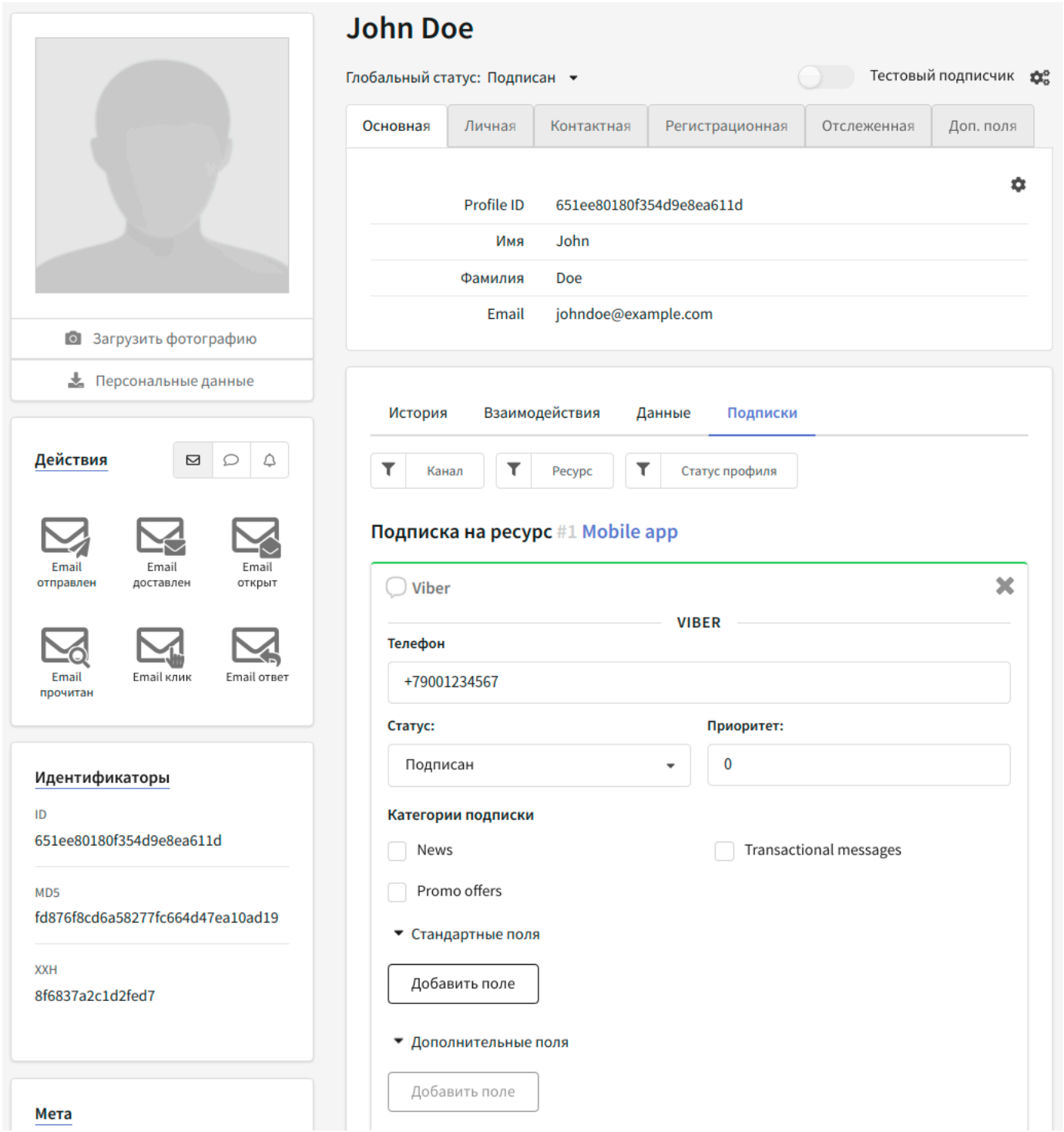
To set up callbacks with message statuses, please contact your account manager or [technical support] of provider:

- **Devino Telecom tech support**
- **SMS Traffic tech support**
- **MTS OmniChannel.**
- **Rapporto.**

3. Import subscriber data into the platform, for example, by using a file. During the import process, make sure to add a subscription to the Viber*™ channel in the resource for each subscriber. You should specify the column in your file that contains phone numbers as the source for subscriptions:



For each customer, a digital profile is created, which compiles their personal information, activity history, and subscriptions to various resources. If a client's subscription to the Viber*™ channel was successfully added during import, this data will be displayed in their history:



Step 3. Create a message template

1. In the main menu, open the Templates section and create a new template using the Create button on the top panel.

2. Choose the creation method — Editor, and the sending channel — *Viber™*.

3. Fill in the general template settings.

4. A Viber*™ template consists of a series of messages. To add the first message to the series, click the + Message button.

5. Select the message type:

Message type	Devino	SMSTraffic	MTS OmniChannel	Rapporto
Text	+	+	+	+
Text + button	+	+	+	+
Text + image + button	+	+	+	+
Image	+	+	+	+
Video	+	+	-	-
Video + text	+	+	-	-
Video + text + button	+	-	-	-
File	+	-	-	-

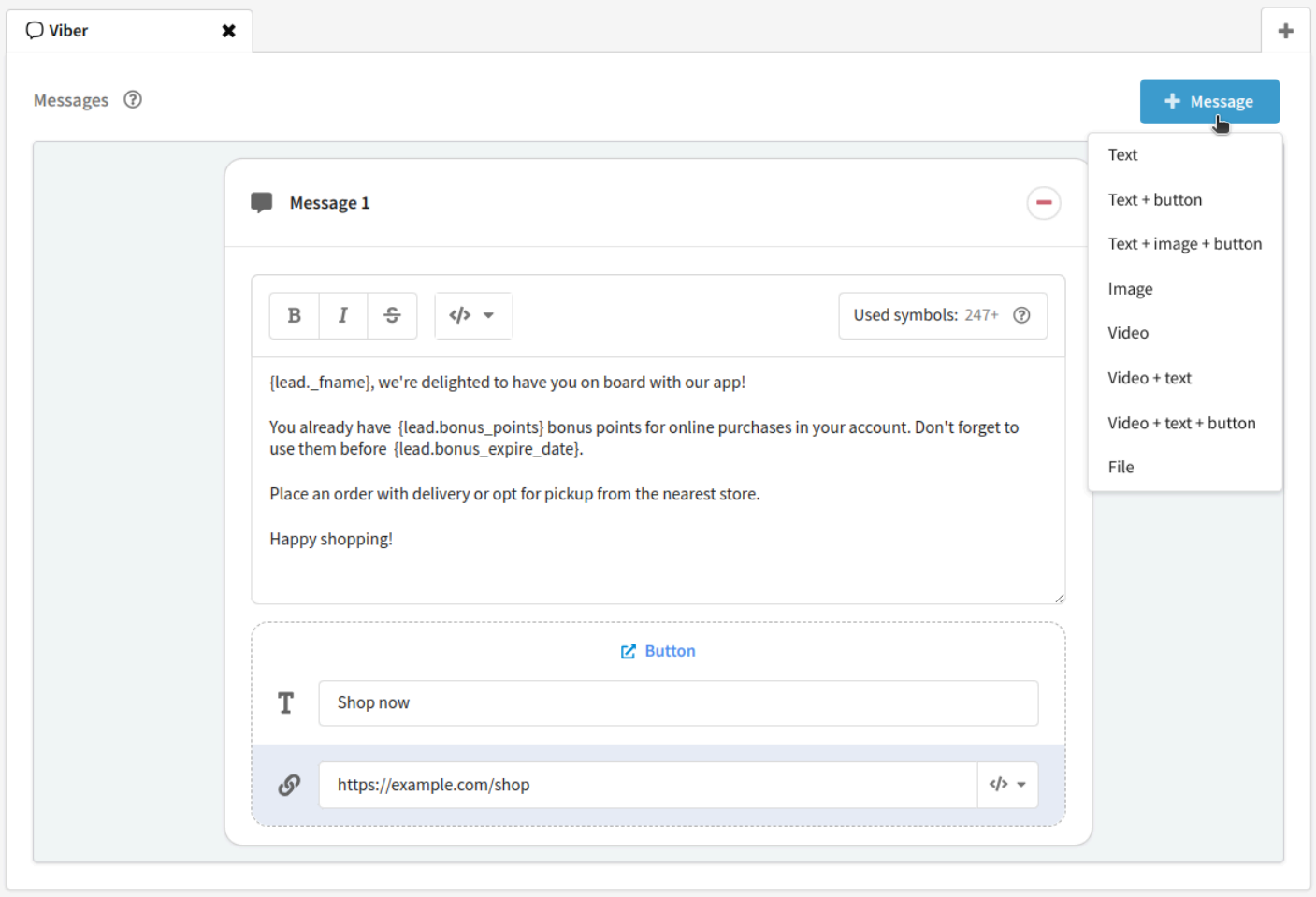
6. Fill in the content. To personalize the message, use variables in your messages

</>

.

INFO

The maximum number of messages in a series is 5. If multiple messages are used in the template, each will be sent with a 2-second interval.



7. Check how your recipients will see the message. To do this, click on the preview button on the top panel.

8. Save your message template.

TIP

You will find more information about setting a template for a Viber*™ channel in this article.

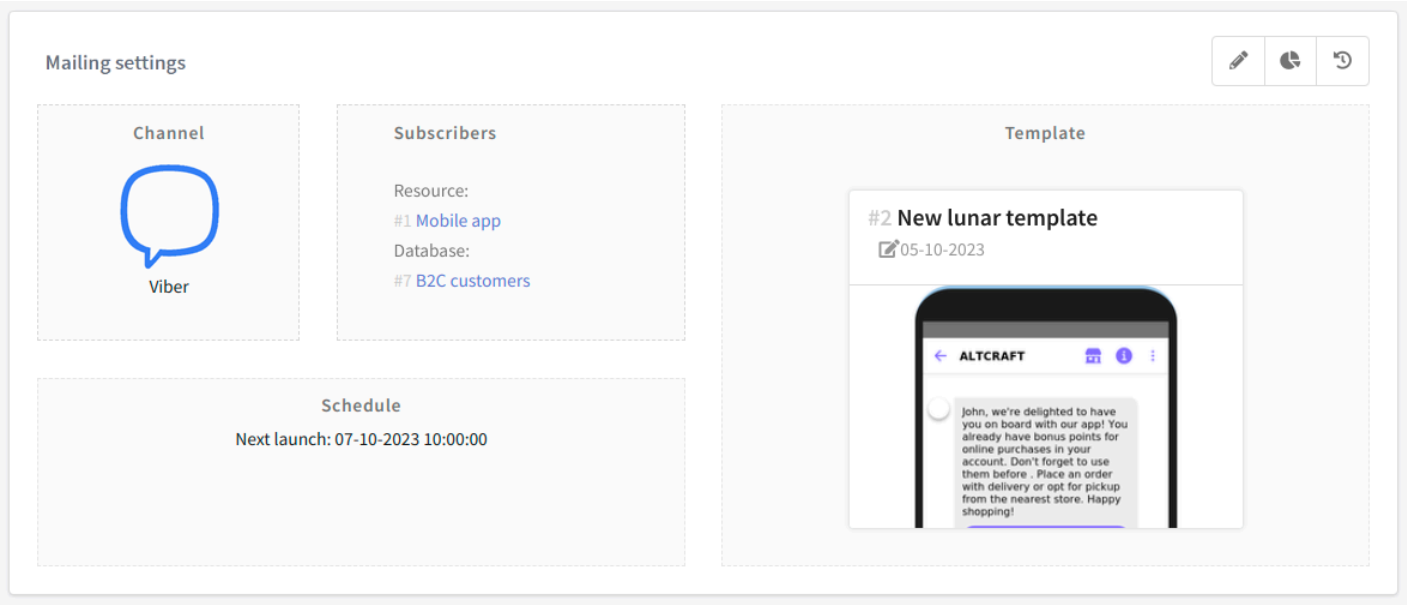
Step 4. Launch the mailing and evaluate its performance

1. Go to the Mailings section and choose the type of mailing:

- **Broadcast is a one-time ending of messages.**
- **Regular mailing is a periodic sending of messages according to a schedule.**

2. Configure the mailing:

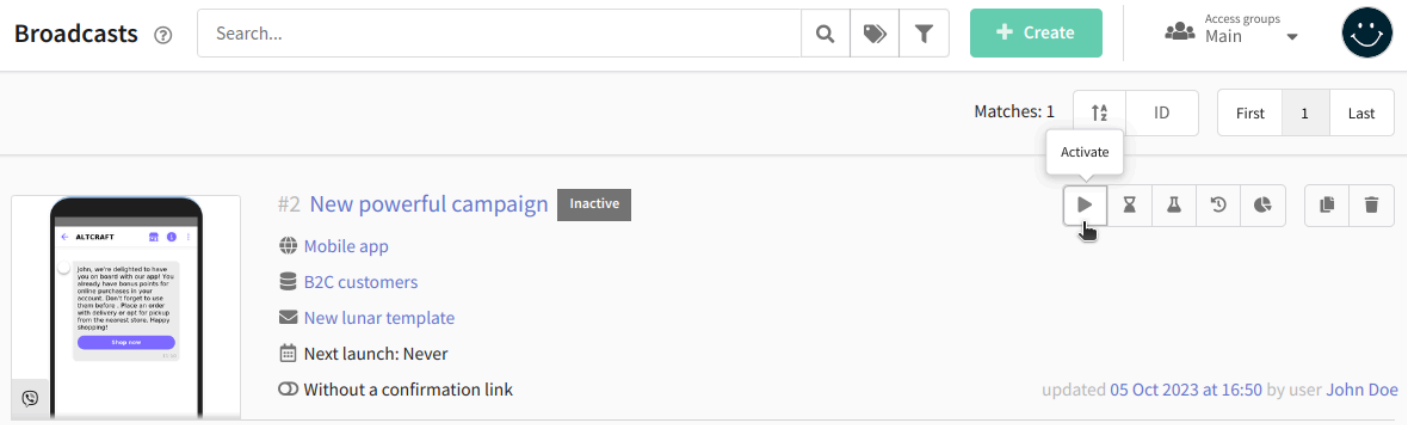
- **Select Viber*™ as the communication channel.**
- **Choose the previously configured resource and the recipient audience (database or segment).**
- **Add the message template to the mailing.**
- **Set the schedule.**



3. Save the mailing.

4. Find it in the list of mailings of the selected type and activate it by clicking the

button in the quick actions panel. An active mailing will start sending messages when the scheduled launch time arrives.



Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Description
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**
Audience source	**Source of contacts: List — Database Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resource**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your push mailings use the channel report. To access the report, click on the analytics button

in the quick actions panel. The report will provide the following information about your mailing:

- **The number of messages sent, delivered, and undelivered.**
- **The number of reads and clicks on links within the messages.**
- **The count of unsubscribes and subscriptions to the Viber*™ channel.**

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

First Notify mailing

INFO

According to the service policy, new access to the Notify service personal account is currently not provided.

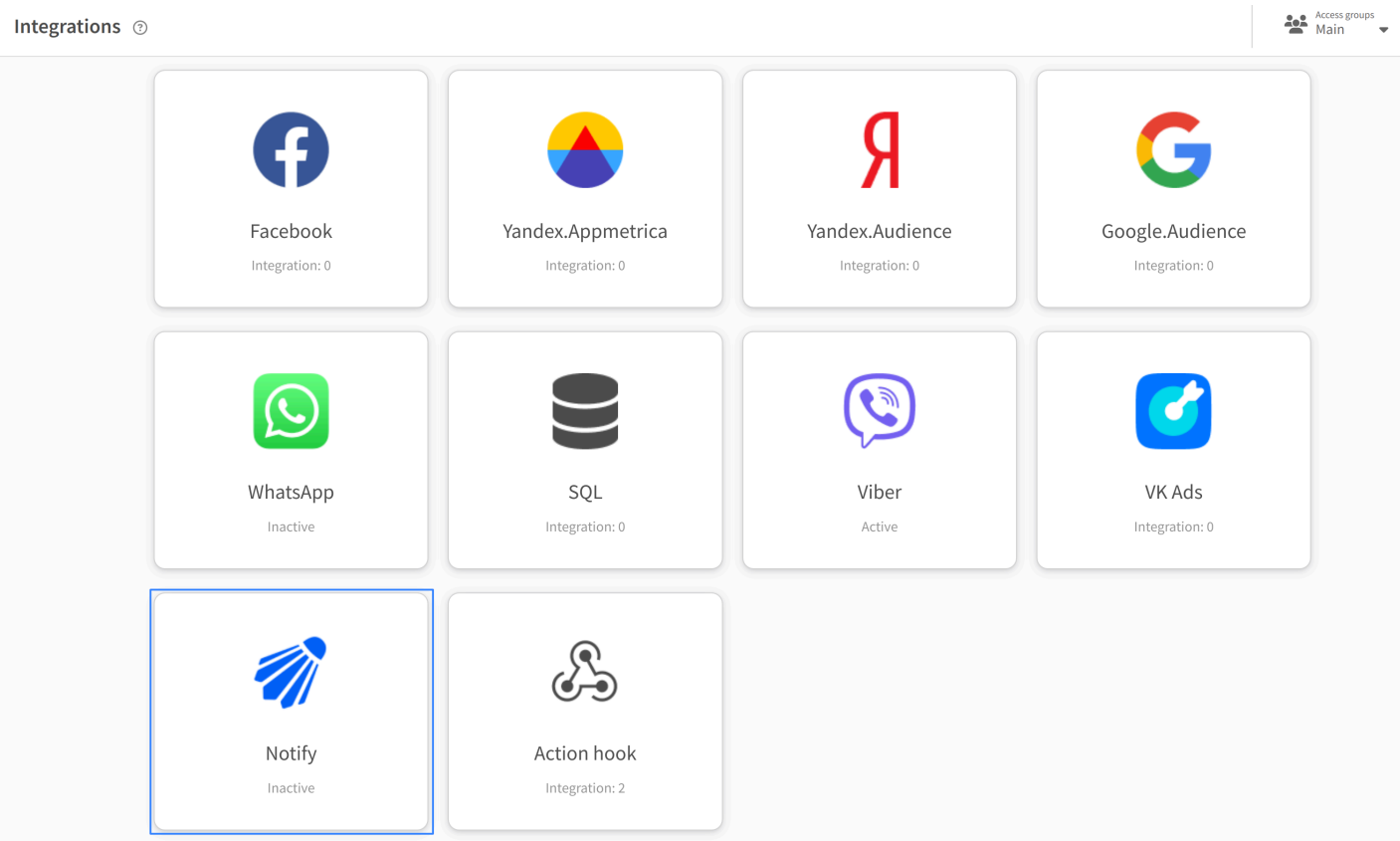
Support for current users is provided by MAX Business tech support.

Notify is a service that allows companies to send service and transactional notifications to VKontakte and Odnoklassniki users. Notifications are delivered from the company's official group, even if the user is not subscribed to it, and can be sent in the form of service messages that can be replied to, or business notifications with no reply option.

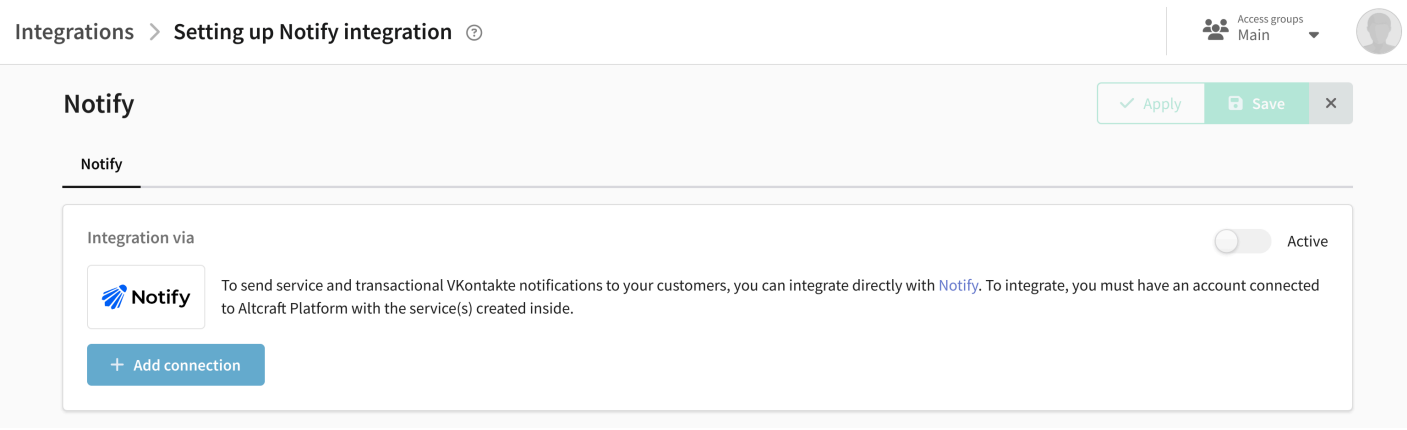
The service allows you to significantly reduce the cost of mailings, ensuring high delivery speed and a high percentage of messages read. In addition to business notifications for VKontakte and Odnoklassniki users, the Notify channel also supports sending SMS reminders.

Step 1: Configure the integration

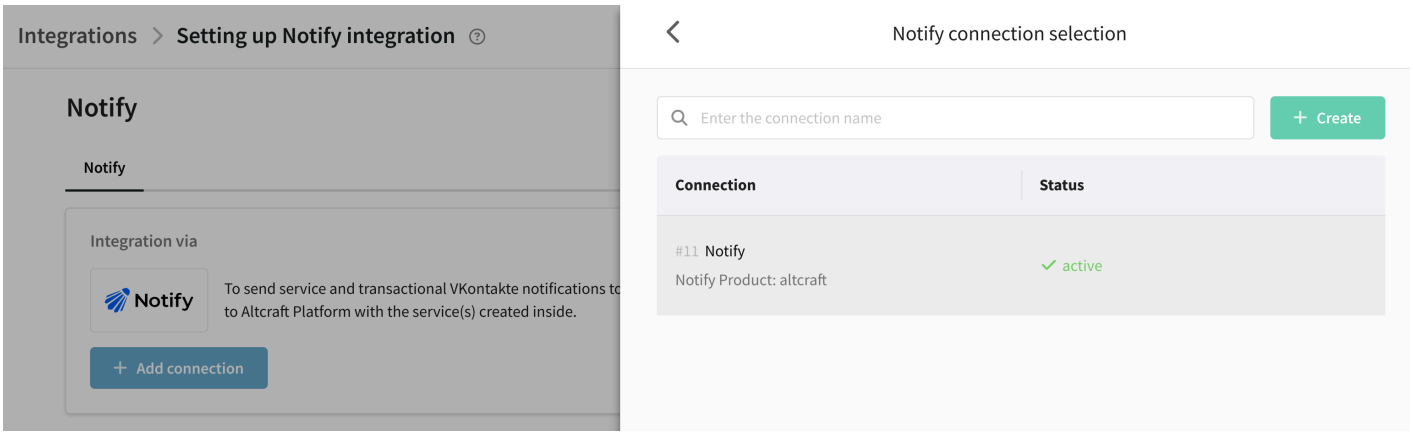
1. Open the Integrations section and select the integration type - Notify:



2. Click Add Connection:



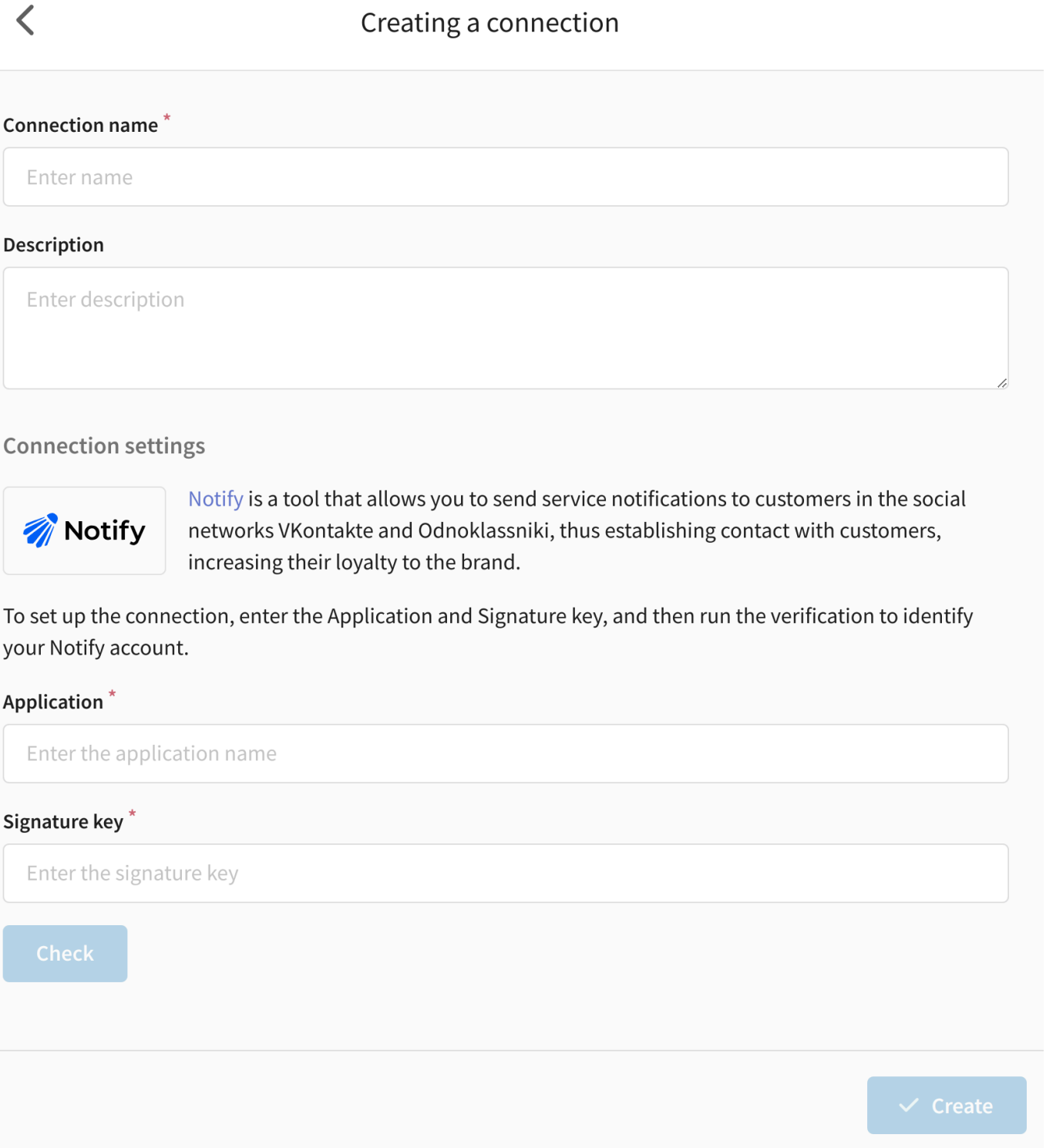
3. Select an available connection to Notify:



No available connections

If you haven’t set up a connection to Notify yet, you can create one right in this window.

Click Create. Enter a name and description for the connection. Next, enter the Application and Signature key parameters and click Check to verify that your data is correct. If the data has been verified, click the Create button in the lower right corner:

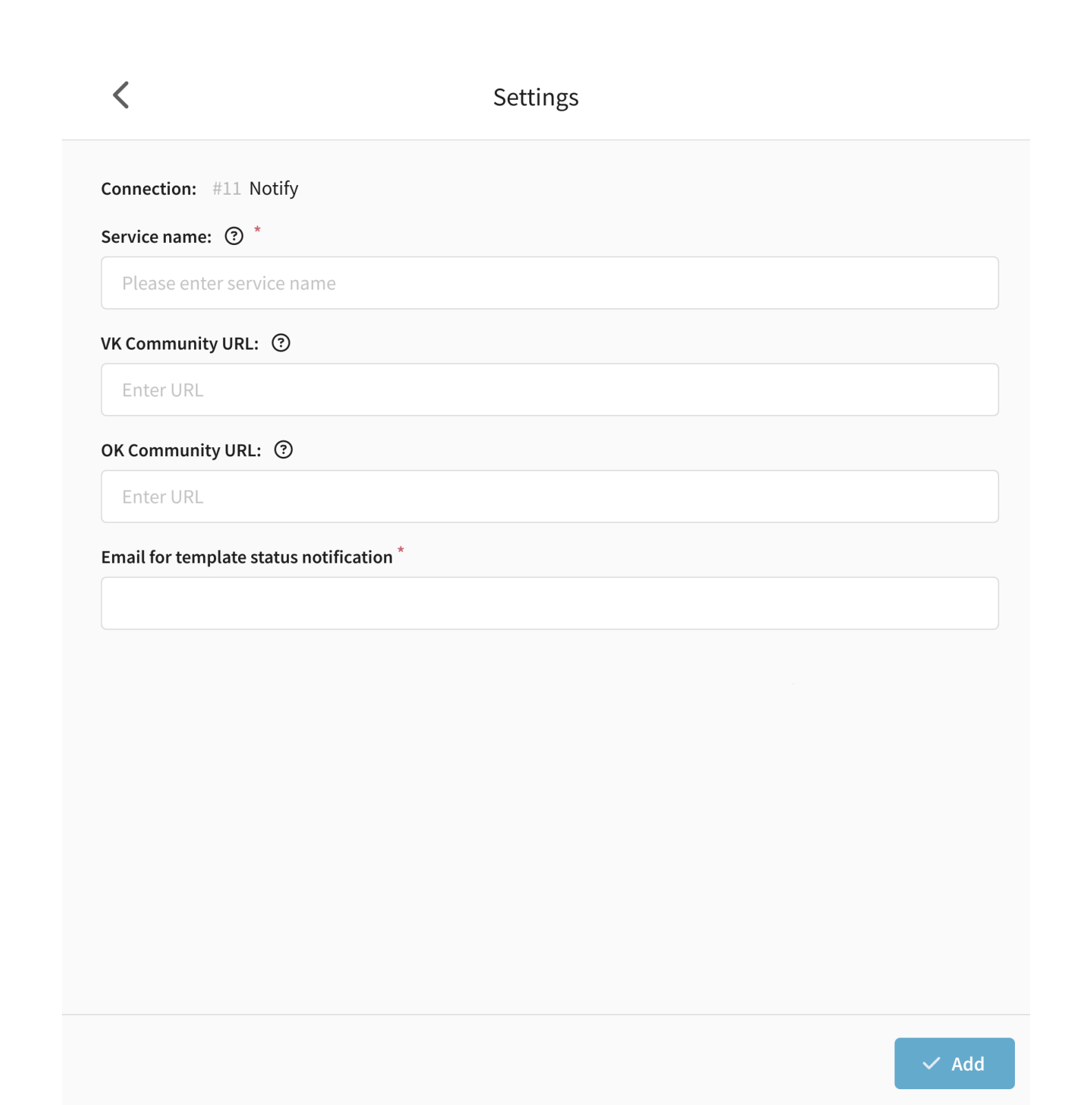


TIP

You can read more about creating a connection ih this article.

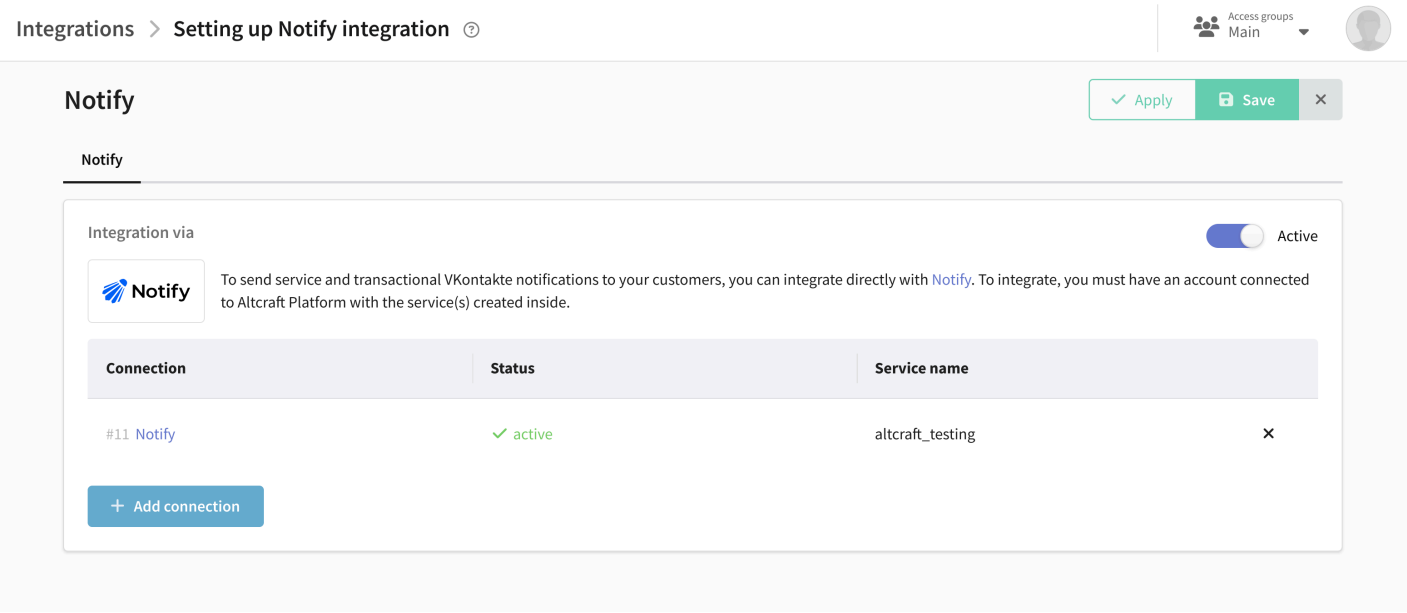
4. Next, specify integration parameters, for correct work they must exactly match the corresponding parameters from Notify account:

- **Service name - the name of the service for mailing, mandatory field.**
- **VK Community URL - link to the VK community on behalf of which the mailing will be made.**
- **OK Community URL - link to the community in OK on behalf of which the mailing will be sent.**
- **Email for template status notification - e-mail address where reports on template statuses will be sent. Required field.**



5. **After filling in the fields, click the Add button. If the entered data is incorrect, the platform will ask you to verify its correctness. If the data is correct, the parameters will be saved.**

6. **Activate the integration using the toggle and save it.**



.

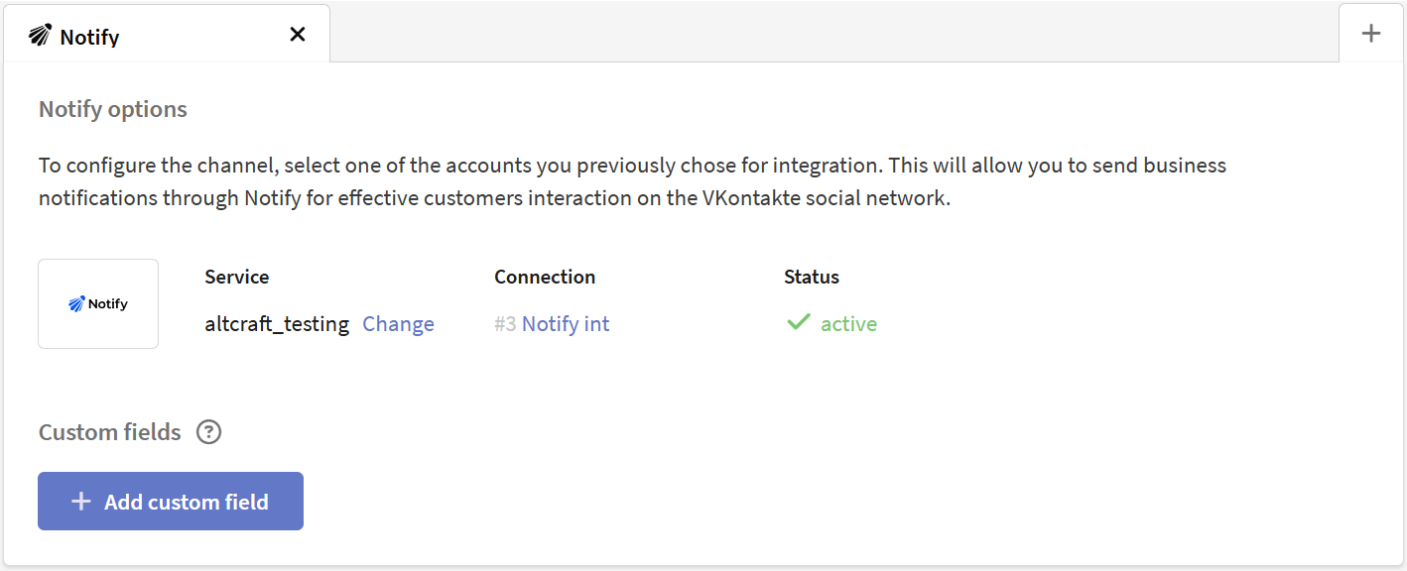
After customizing the integration, you can use it to add a Notify feed to resource.

Step 2: Sign customers up for Notify channel

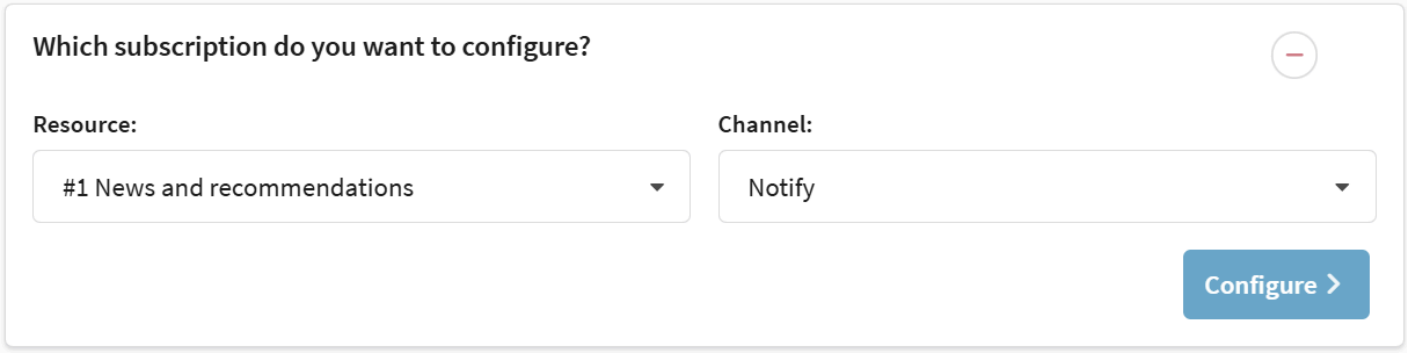
To subscribe clients to the Notify channel, follow the steps below:

1. Select or create a profile database where subscribers' data will be stored.

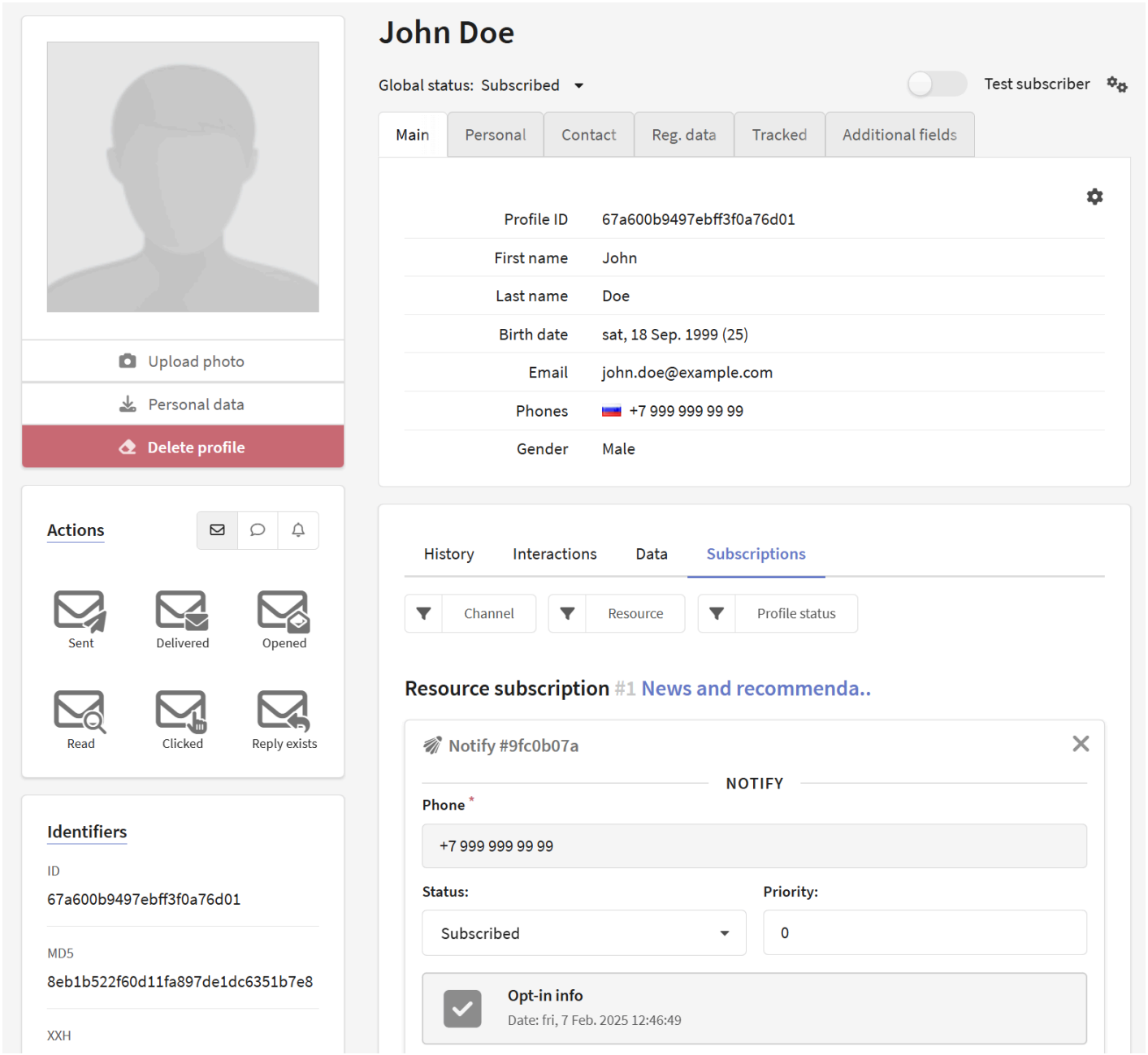
2. Create a new resource or edit an existing one by adding a Notify channel. To customize the channel, select one of the accounts you previously selected for integration.



3. Import subscriber data into the platform - e.g. via a file. When importing, add them a subscription to the Notify channel in the resource. In this case the phone number is the source of the subscription.



A digital profile will be created for each customer, combining personal data, activity history, and resource subscriptions. If a Notify channel subscription was successfully added to a customer during import, this data will be displayed in their history:



Step 3: Create a message template

INFO

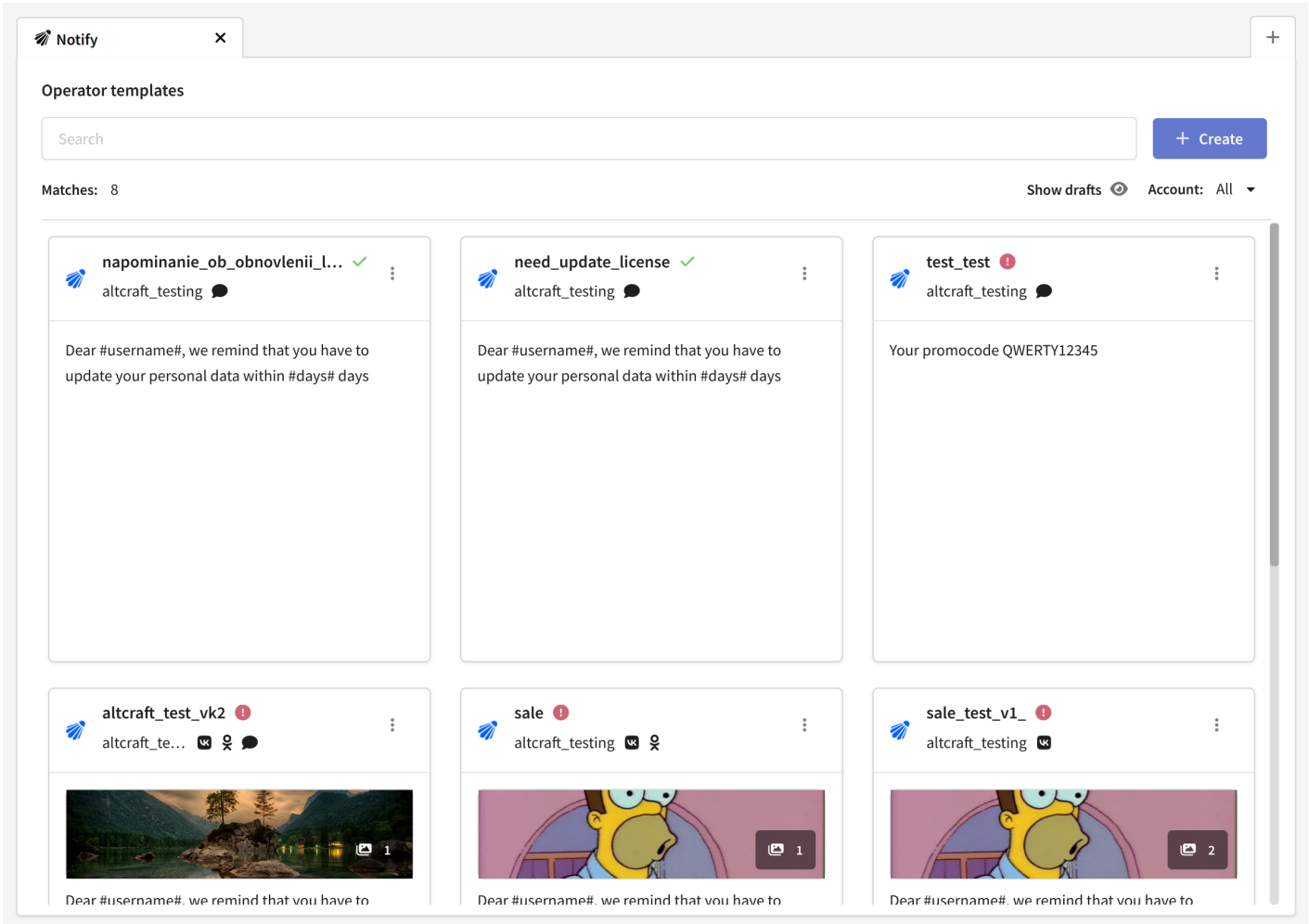
Only templates that have been moderated by Notify can be used to send messages.

1. In the main menu, open the Templates section and create a new template using the Create button on the top panel.

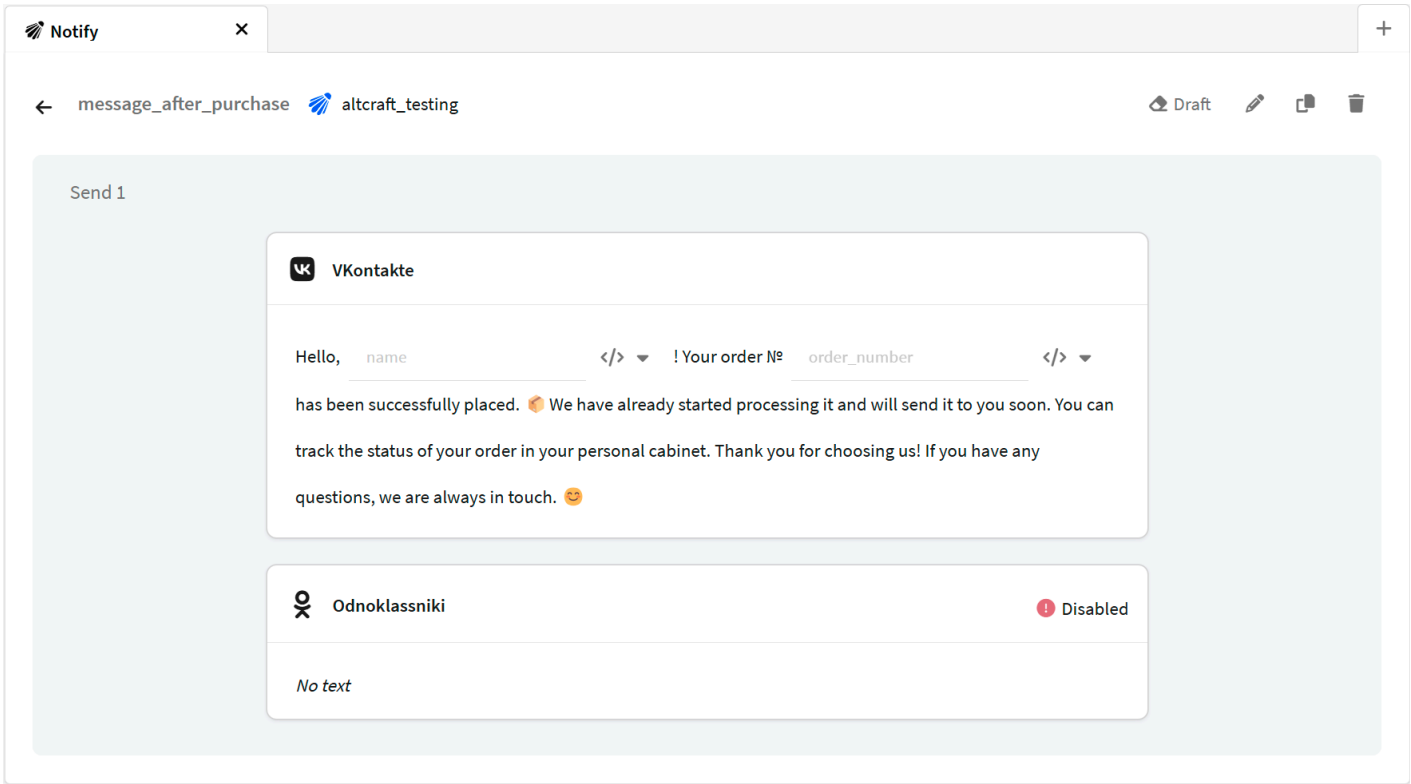
2. Select Editor as the creation method and Notify as the sending channel.

3. Fill in the basic settings of the template.

4. When you open the message editor for Notify, you will be shown the generic templates created in your Notify account. You can also create a new template by clicking the button.



5. Fill in the parameters inside the template. Parameters are text blocks that change for each new message (name, address, event name, date and time, link, etc.). You can manually prescribe parameter values or add Altcraft variables.



6. Check to see how your recipients will see the message. To do this, click the preview button on the top bar.

7. Save the message template.

TIP

You can also create a Notify template while in the platform. To create a template, click the corresponding button. The template creation window will open, in this window you should specify the name of the template and the service configured in your personal cabinet. More information about creating a template for Notify channel can be found in [this article](#).

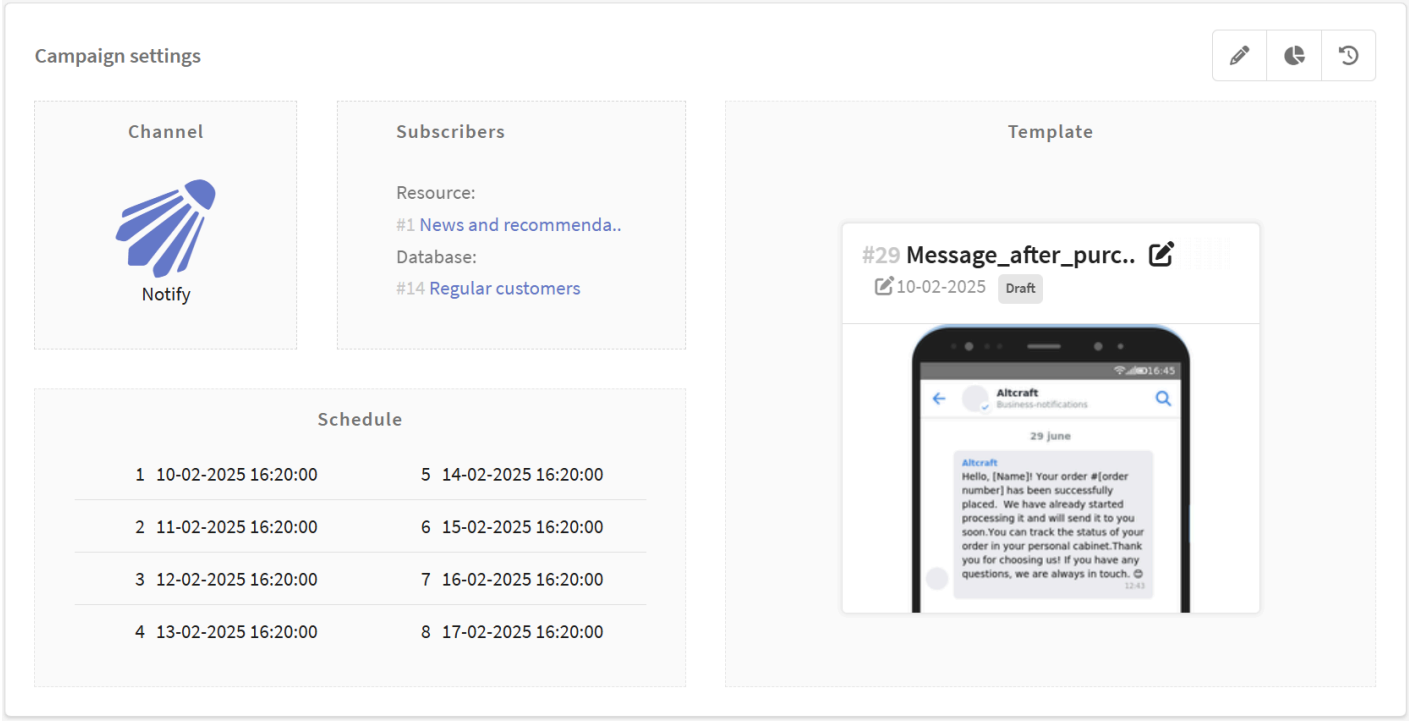
Step 4: Launch the mailing and evaluate the effectiveness of the mailings

1. Go to "Mailings" and select the type of mailing:

- **Broadcast - one-time mailing of messages.**
- **Regular Mailing is a periodic mailing.**

2. Customize the mailing:

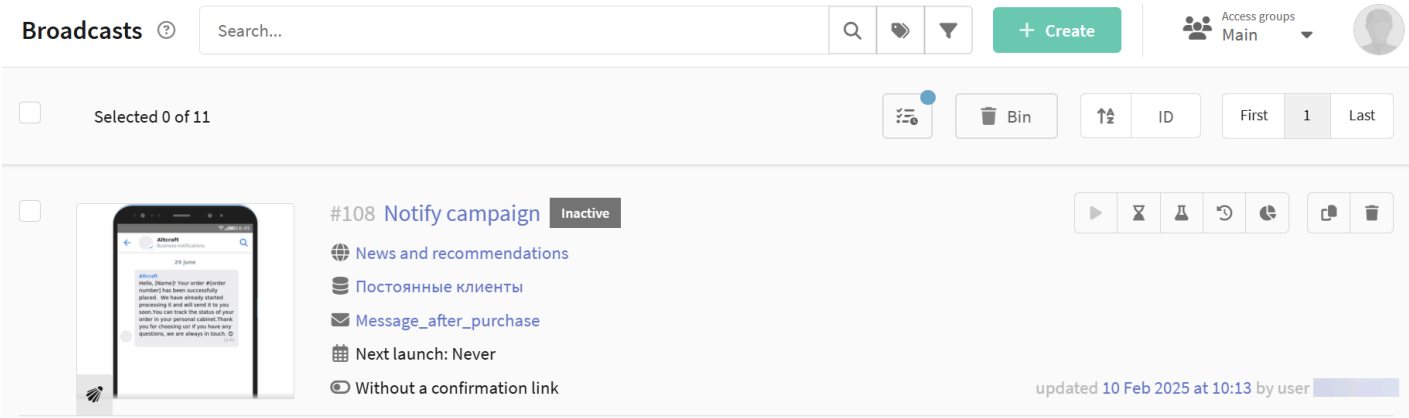
- **select Notify as the communication channel,**
- **select the previously configured resource and recipient audience (base or segment),**
- **add a message template to the mailing,**
- **set the schedule.**



3. Save the mailing.

4. Find it in the list of mailings of the selected type and activate it by clicking

in the Quick Actions panel. The active mailing will start sending messages when it's time to run on the schedule.



Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Description
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**
Audience source	**Source of contacts:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resource**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your push mailings use the channel report to evaluate the results of your mailing. To access the report, click on the analytics button

in the quick action bar. The report will provide the following information about the mailing:

- **number of messages sent, delivered and undelivered;**
- **number of reads and clicks on links in messages;**
- **the number of unsubscribes and subscriptions to the Notify channel.**

Communication Channels Workflow

General Structure

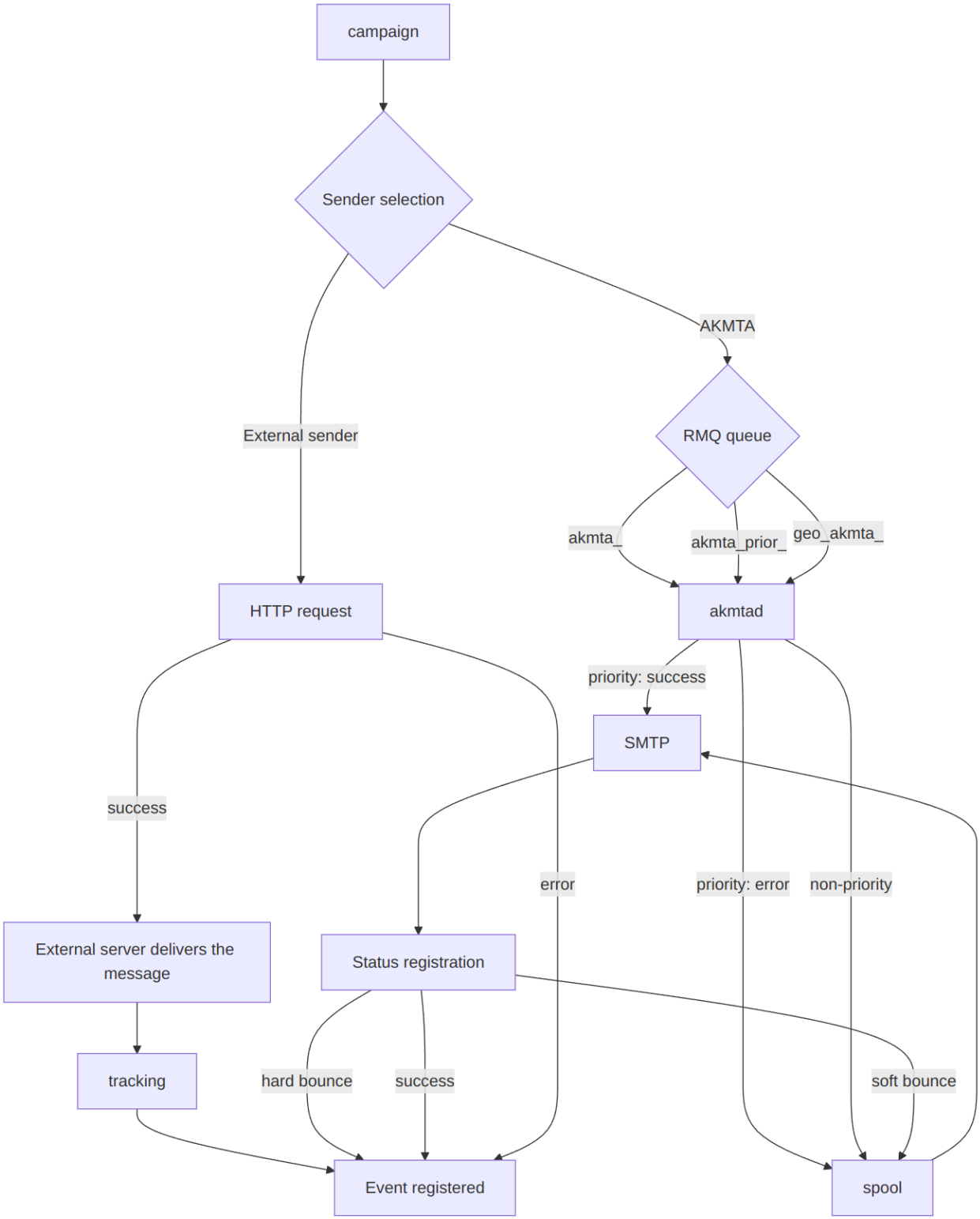
When a mailing is launched, the `mailing` binary is executed. It:

- **processes the template and business logic (blocklists, limits, subscriptions, filters, etc.);**
- **builds the message structure for delivery;**
- **sends the prepared messages to RabbitMQ queues;**
- **records errors in the mailing log.**

Each message then goes to the corresponding processing service depending on the channel.

Email

Several types of senders are supported for the email channel: AKMTA (the Altcraft sender) and external senders. The workflow may vary depending on the sender used.



Sending Process

AKMTA

The mailing sends messages to the following queues:

- `akmta_<sender_id>` — **standard queue;**
- `akmta_prior_<sender_id>` — **priority queue.**

Messages from the queues are processed by the `akmtad` **service. Its behavior depends on the queue type:**

- **For a non-priority queue, it writes messages to disk, periodically reads them, and sends them according to the sending strategy configured for the specific** `sender_id`**;**
- **For a priority queue, it makes an attempt to send immediately; if it fails, it writes the message to disk.**

The sending flow looks like this:

`mailing` → **RMQ Queue (** `akmta` / `akmta_prior` **)** → `akmtad` → `SMTP`

Other Senders

External senders receive messages via an HTTP request from the `mailing` **binary. If the platform cannot send a message, a mailing error is logged. Note that such an error is not considered a "Non-Delivery" and is not included in the non-delivery report.**

Status Registration

AKMTA

Delivery statuses are registered immediately using:

- **the SMTP server response code;**
- **bounce patterns (analysis of bounce templates).**

Other Senders

Tracking service of the platform handles event status.

Retry Logic

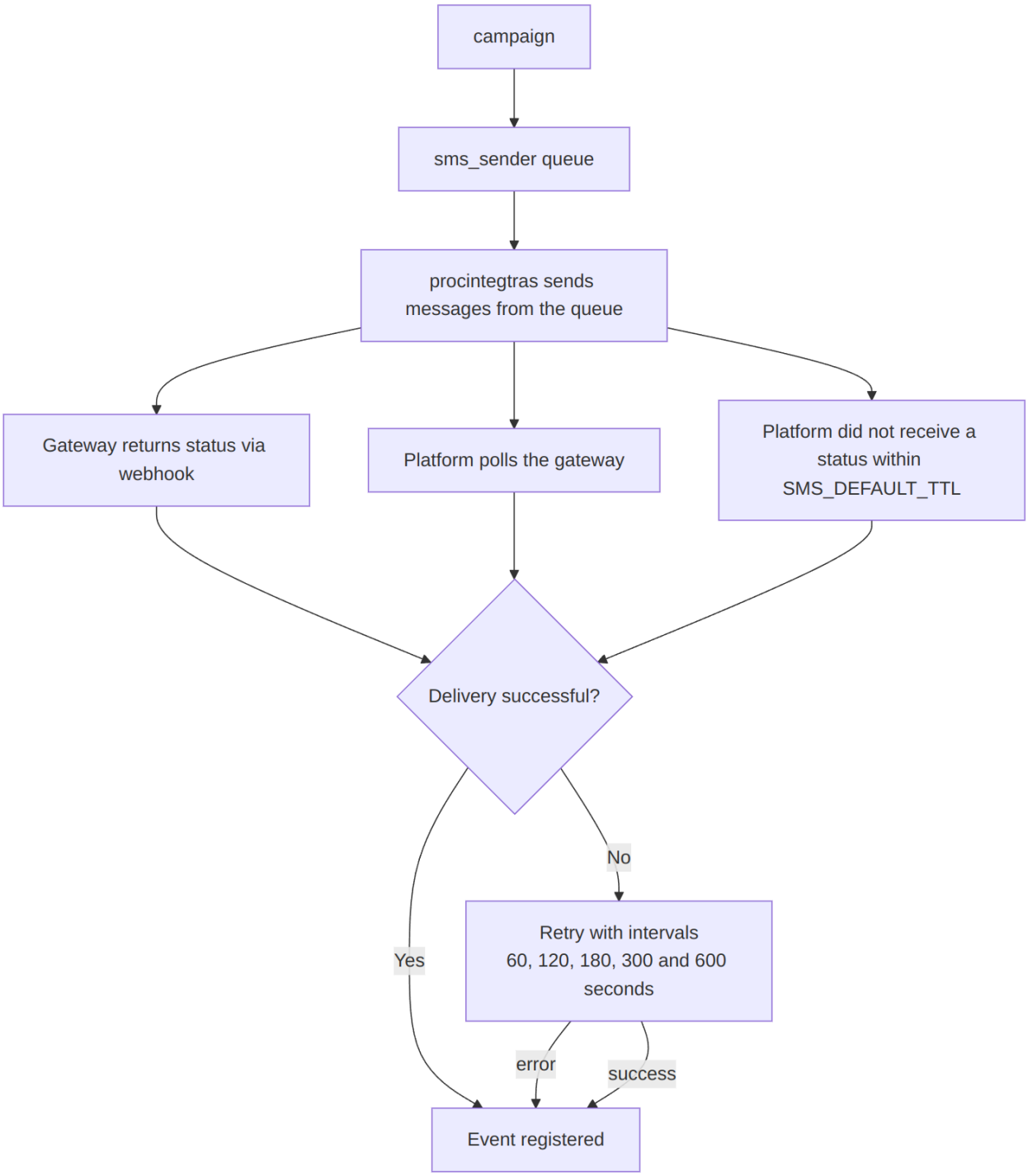
AKMTA

- **Retry rules are defined separately in the Admin Panel. Details are available in the administrator documentation;**
- **Based on the SMTP code, hard bounce or soft bounce statuses are registered;**
- **Retry logic differs for soft and hard bounce cases.**

Other Senders

External senders do not support retry attempts. If an HTTP request fails, the platform does nothing further.

SMS



Sending Process

The mailing sends messages to a single queue — `sms_sender`. The `procintegras` service then sends messages from the queue. Some gateways support batch sending.

Status Registration

Status registration works in one of the following ways:

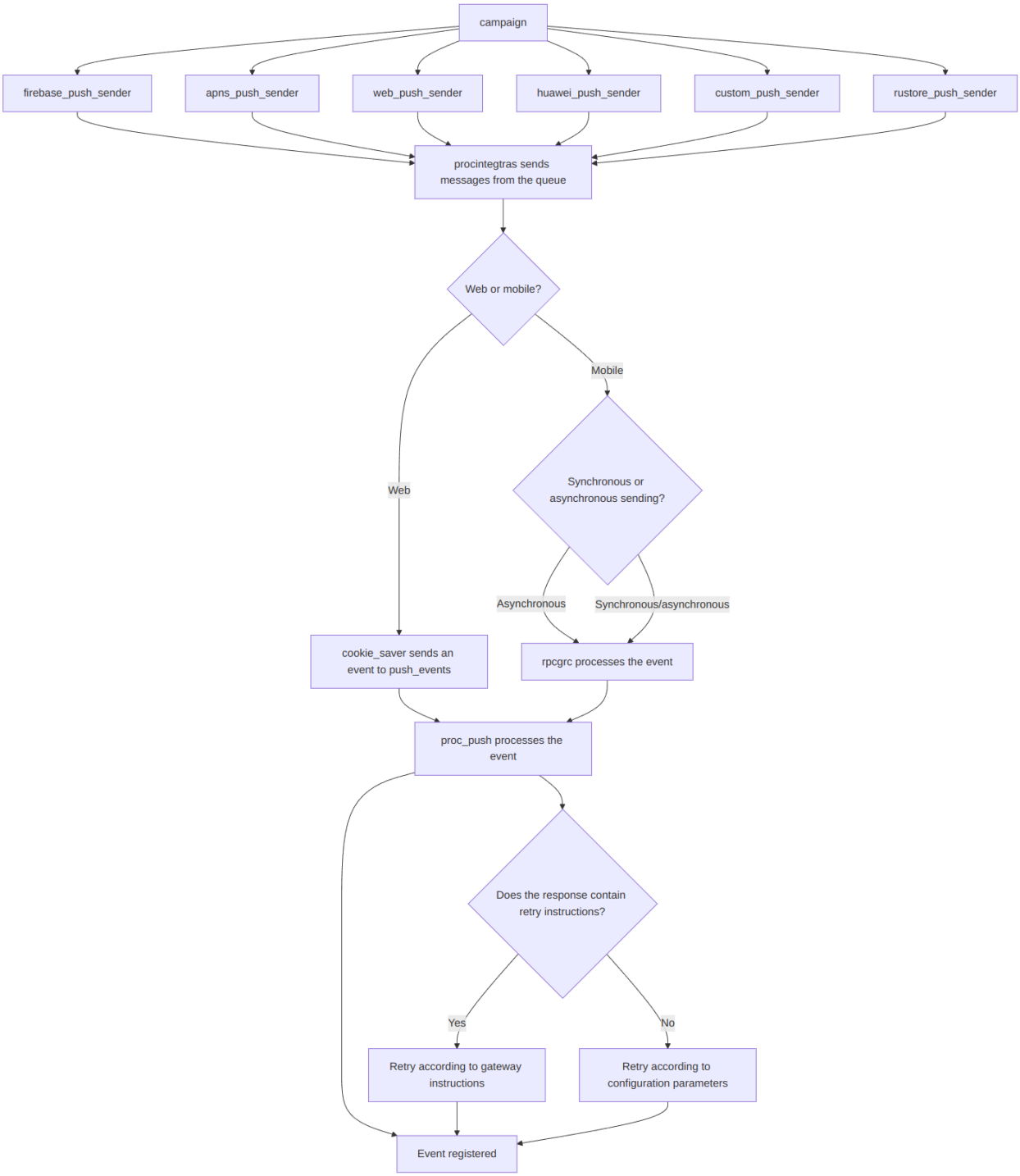
- **The gateway sends a message status via a webhook. The status is processed by `procsmslisten`;**
- **The platform polls the gateway. The status is processed by `procsmsev`.**

The lifetime of an SMS message is defined by the configuration parameter `SMS_DEFAULT_TTL` (default — 1 day). The platform registers a non-delivery if no status is received within this time.

Retry Logic

SMS retries happen after 60, 120, 180, 300, and 600 seconds from the last failed attempt. The maximum number of attempts is 5.

Push



Sending Process

The mailing sends messages to a queue. The specific queue depends on the provider:

- **APNs** — `apns_push_sender`
- **Firebase** — `firebase_push_sender`
- **Huawei** — `huawei_push_sender`
- **RuStore** — `rustore_push_sender`
- **Browser push (Firefox, Safari)** — `web_push_sender`
- **Other providers** — `custom_push_sender`

The `procintegras` service then sends messages from the queue. Some gateways support batch sending.

Status Registration

For browser push, the `cookie_saver` service handles status updates and sends events to `push_events`. Further processing occurs in `procpush`.

For mobile push, the logic is similar, but asynchronous sends are processed via `push_events`, while synchronous ones are immediately handled by `procrpc`.

The lifetime of a push message is defined by `MTA_DEFAULT_TTL` (default — 3 days). A non-delivery is registered if no status is received within this time.

> TIP: Synchronous and Asynchronous Requests
>
> **Synchronous requests — the platform registers the result based on the response, meaning it waits for a reply.**
> **Asynchronous requests — the platform registers the result when putting the message into the queue, meaning it does not wait for a reply.**

Retry Logic

The sender checks the gateway's response and headers to apply retry rules. If the gateway explicitly specifies retry instructions (for example, "try again in 5 minutes"), retries follow them. If not, the platform uses its configured parameters:

- **`PUSH_SENDER_RETRY_MAX_AMOUNT` — maximum retry count (default — 3);**
- **`PUSH_SENDER_RETRY_INTERVAL` — interval between retries (seconds) (default — 1800);**
- **`PUSH_SENDER_RETRY_SCAN_INTERVAL` — scan interval for retry records (seconds) (default — 60).**

Detailed information is available in the administrator documentation.

Messenger Channels

Sending Process

The mailing sends messages to a queue. RabbitMQ queues are processed depending on the channel:

- **Telegram Group, Telegram Bot, MAX Group, MAX Bot — queue `piper_messages`, processed by `procpiper`;**
- **Viber, WhatsApp, Notify — processed by `procintegras`.**

Status Registration

- **Telegram — status is determined based on the HTTP response from Telegram. Delivery is registered if no error occurs during the send request. A non-delivery is registered if an error is returned.**
- **Viber, WhatsApp, Notify — statuses arrive via webhooks. If an error occurs during sending, a non-delivery event is registered. Statuses are handled by `trklisten`.**

Retry Logic

Retries for messages in messengers other than MAX are not supported.
For MAX, retries occur after 15, 30, 60, 120, and 300 seconds; after that, an undelivered event is registered.

> INFO: Sending limits
>
> **Telegram imposes a limit of 28 requests per second per token.**
> **For Notify, the default limit is 20 requests per second, configurable via `VKNOTIFY_SEND_LIMIT_PER_SECOND`, with a maximum of 50 requests per second.**
> **For Viber, the default limit is 50 requests per second, configurable via `VIBER_DEVINO_SEND_LIMIT_PER_SECOND`.**
> **For WhatsApp, the default limit is 20 requests per second (`WHATSAPP_SEND_LIMIT_PER_SECOND`); individual providers may apply their own request limits.**

Email: sending domain configuration

SPF configuration

Sender Policy Framework (SPF) technology allows to verify that sender domain has not been faked. To make it work you need to add a special TXT record to your sending domain. It will allow only specified hosts to send messages using your domain.

We recommend to add records for different SPF versions:

Domain	Record type	Contents
example.com.	**TXT**	**v=spf1 include:spf.akmta.net -all**
example.com.	**TXT**	**spf2.0/pra include:spf.akmta.net -all**

The domain spf.akmta.net here contains the list of IP addresses Altcraft Cloud MTA uses.

If you have already used your domain for email, add to the existing SPF records “include:spf.akmta.net”.

> CAUTION
>
> **Note that we recommend to use restricting rules where only the enlisted resources are allowed!**

DKIM configuration

DomainKeys Identified Mail (DKIM) technology adds to your message a digital signature of your from-domain. The signature is automatically verified on the recipient side and hen it is used to manage sender reputation. The signature is technically an RSA key pair: the private part is built in sender infrastructure and the public part should be added to your domain as a special subdomain TXT record:

Domain	Record type	Contents
`ak._domainkey.example.com.`	**TXT**	**v=DKIM1; k=rsa; p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCldFYh3Rfrmeov+WqYYpwfW2bVUzxXPy9dSVoUCGLKCn+vgY/pdKIIBFitkvZJXGLnHqreqwt**
`_domainkey.example.com.`	**TXT**	`o=-;`

> TIP
>
> **To get a personal DKIM for Altcraft Marketing contact support@altcraft.com and provide us with your domain name.**

DMARC settings

DMARC technology allows mailing servers to decide what to do if SPF or DKIM verification fails.

Here we recommend to restrict messages sent from unknown IP addresses or signed incorrectly.

Domain	Record type	Contents
`_dmarc.example.com.`	**TXT**	`v=DMARC1; p=reject; sp=reject; mailto:report@example.com`

BIMI settings

BIMI technology (Brand Indicators for Message Identification) is used along with SPF, DKIM and DMARC or identifying email sender by company brand logo, placed near the message subject in recipient’s inbox. Email providers thus can additionally verify the sender.

To start using BIMI you will need an svg image, that is rectangular and has no extra layers.

> NOTE
>
> **BIMI authentication is supported by email services such as Fastmail, Gmail, and Yahoo!**

Add the following record to your domain settings, specifying the svg file path:

Domain	Record type	Contents
`default._bimi.example.com`	**TXT**	`v=BIMI1; l=https://example.com/bimi/bimi.svg`

You can also use BIMI record generator at https://bimigroup.org/bimi-generator/

Tracking domain settings

Tracking domains are used to collect your customers’ behavior information. For any subdomain you use add this record:

Domain	Record type	Contents
`trk.example.com.`	**CNAME**	`trk.aksend.net`

You can use this record for all of your tracking domains.

CNAME settings for processing bounces

For cloud clients, it is also necessary to configure an additional CNAME record directed to the domain `akmta.net`:

Record example:

```
ac-bounces.example.com IN CNAME akmta.net
```

In this example, `ac-bounces` is a subdomain, and `example.com` is your from-domain. This subdomain will be used in the `Return-Path`, enabling SPF authentication and alignment to improve email deliverability.

Email: setting up and using postmasters

##Description {#id-postmasters-description}

Postmaster is a service of an email provider that tracks email distrubution statistics: how many messages were delivered, how many ended up in spam, sending failures, how many emails were complained about, etc.

Postmaster registration is required if you plan to deliver large volumes of messages, more than 10 thousand per day. A postmaster is also required when registering an FBL.

> TIP
>
> **FBL/CFL (FeedBack Loop) is a standard for providing information about spam complaints from an email service provider to the sender of emails. It is highly recommended to use this technology on all ISPs where it is available in one form or another. You can learn more about FBL in this article.**

The choice of postmaster depends on which email domains are most commonly used in your database. Each postmaster provides statistics only for their domains.

The most popular postmaster services in Russia:

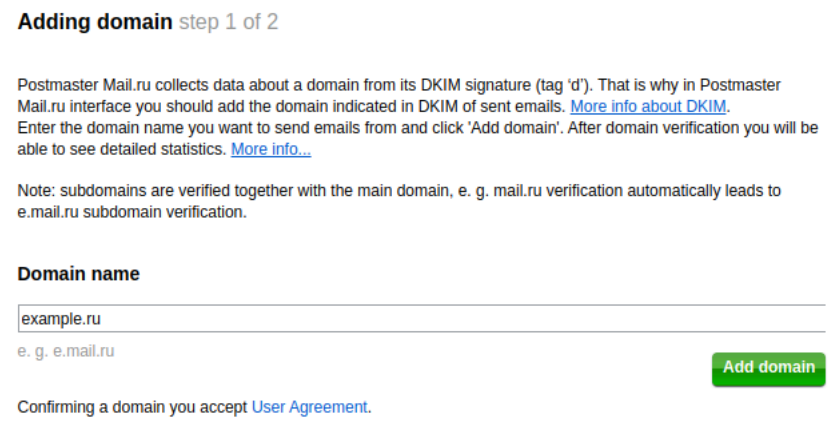
- **Mail.ru Postmaster**
- **Gmail Postmaster Tools**
- **Outlook.com Postmaster**
- **Yahoo Complaint Feedback Loop Service**

How to set up Postmaster Mail.ru

Step 1. Create a Mail.ru account or log in using one of the specified methods.

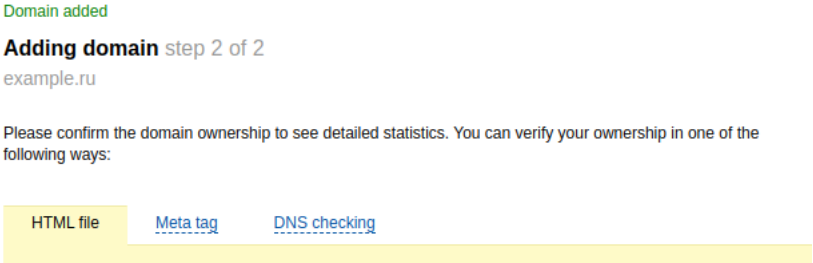
Step 2. Go to the domain adding page - https://postmaster.mail.ru/add/.

Step 3. Enter the name of the domain about which you want to collect statistics and click “Add Domain”:

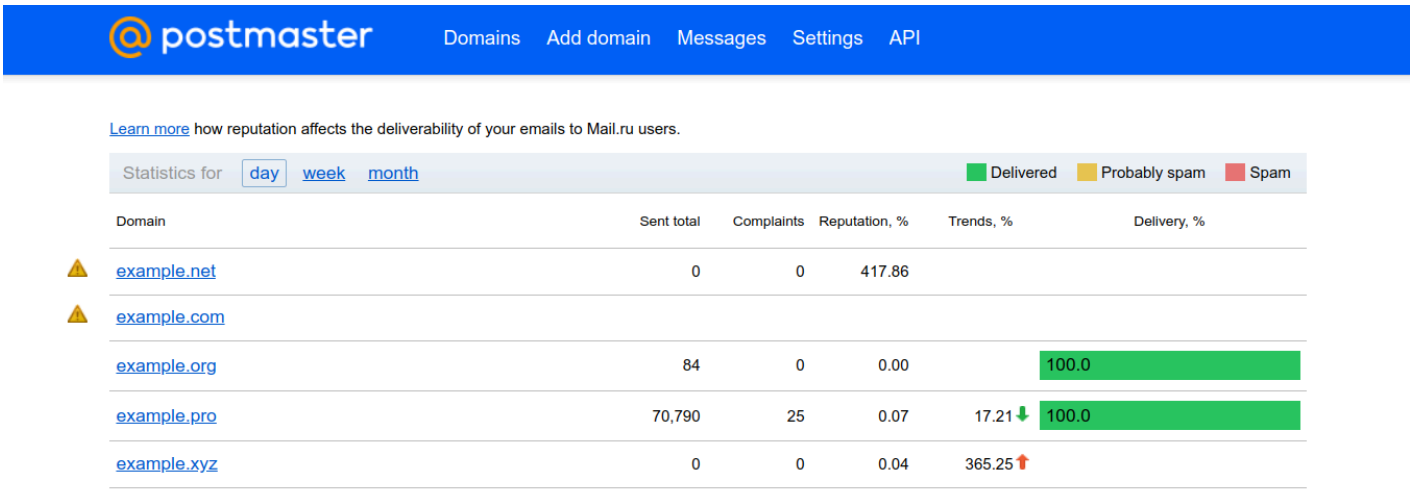


Step 4. Select one of the domain verification methods:

- **HTML file - this method can only be used if the domain has a web server. Follow the instructions described on the tab and click “Confirm”**
- **Meta tag - used only if you have your own website. Add a tag with a personal code to the `head` section of the main page of the site and click “Confirm”**
- **DNS check - allows you to use the postmaster even if you don’t have your own website. The DNS record value must be placed in the domain’s DNS records using the interface of the hosting provider.**



If all instructions are completed correctly, the domain will be added to the postmaster. In the described way, you can add an unlimited number of domains:

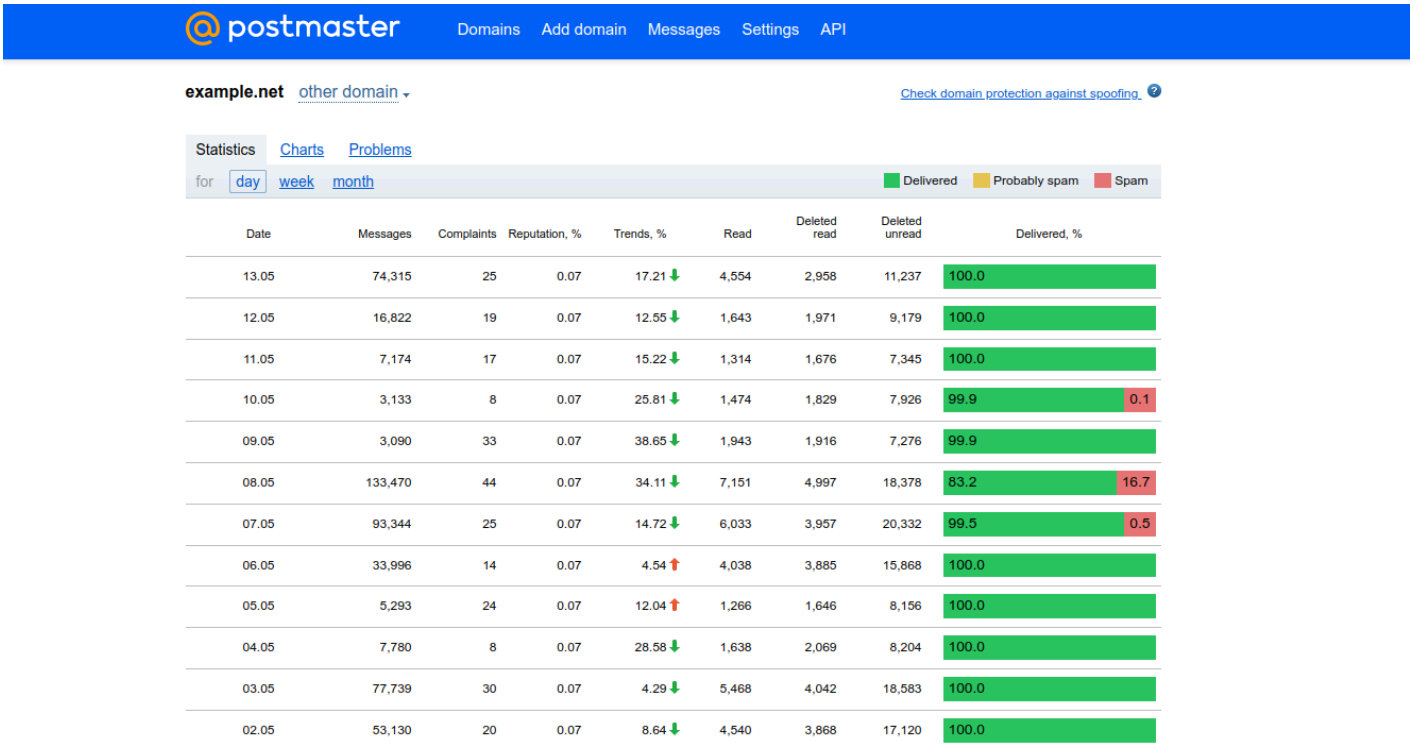


Setting up FBL forwarding

Postmaster Mail.ru supports FeedBack Loop technology. FeedBack Loop provides access to mailing statistics, grouping it by day, week and month.

This statistics allows you to track the following indicators:

- **Messages — number of messages sent;**
- **Complaints — number of clicks on the “Spam” button;**
- **Reputation, % - average percentage of complaints over 30 days. It is calculated as the ratio of the number of complaints over 30 days to the total number of submissions during this period of time and is expressed as a percentage;**
- **Trends, % - change in the percentage of complaints for 7 days in relation to the percentage for 30 days;**
- **Read — number of open letters;**
- **Deleted read — number of messages deleted after reading;**
- **Deleted unread — number of messages deleted before being read;**
- **Delivered, % — percentage of the following types of messages:**
  - **Delivered — the number of emails delivered to the Inbox;**
  - **Probably spam — the number of emails delivered to the Spam folder;**
  - **Definitly spam — the number of messages that were not accepted by Mail.ru servers. In response to such emails, you always receive a messages with the error “550 spam message discarded”.**



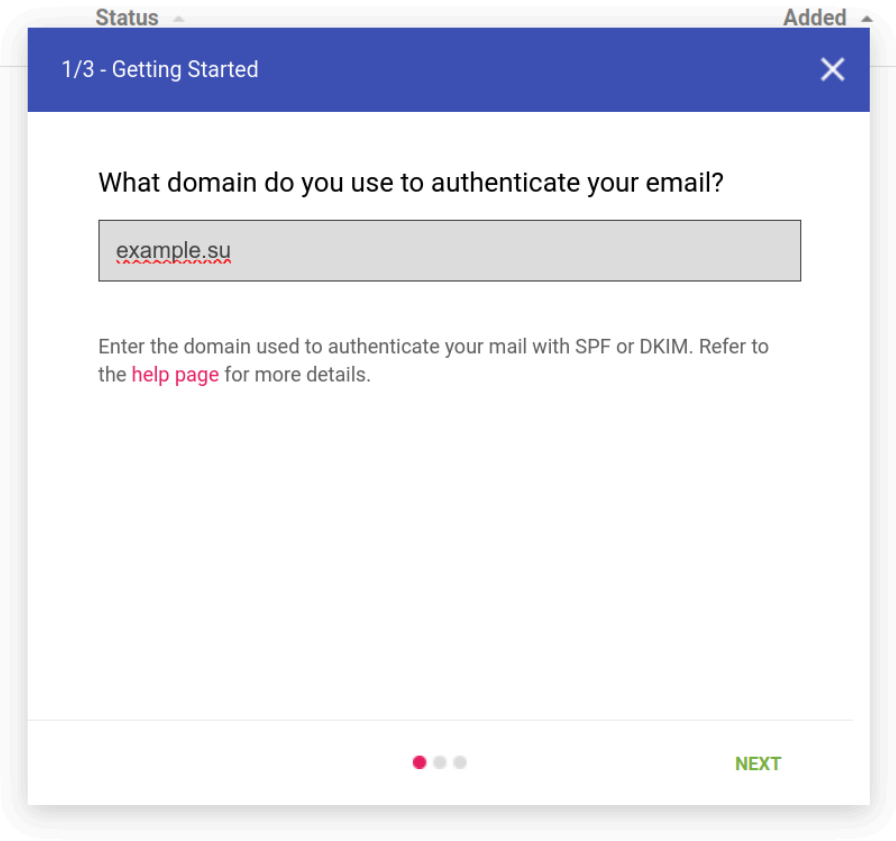
On the settings page for all domains, enter the FBL address. It will receive information about which recipients clicked the “Spam” button after receiving the message. After each mailing, Mail.ru Postmaster will display statistics on the number of emails sent and their status: Delivered, Probably spam, Definitely spam.

How to set up Postmaster Tools by Gmail

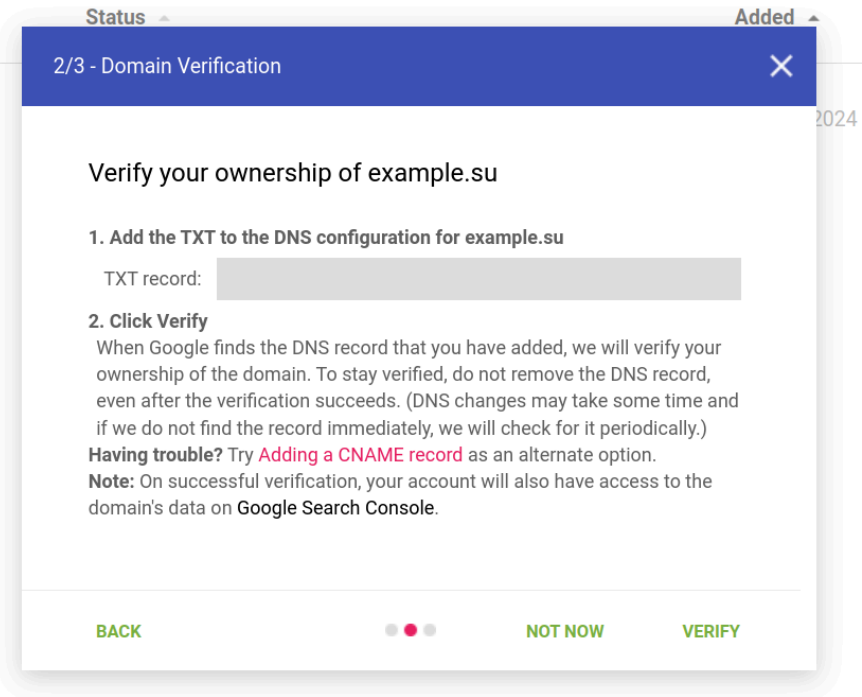
Step 1. Sign in to [Postmaster Tools] using your Google account.

Step 2. In the lower right corner, click the “Add” button.

Step 3. Enter your authentication domain (DKIM or SPF) and click “Next”:



Step 4. Confirm domain ownership. To do this, access your DNS record and add the TXT record provided by Google. Once complete, return to Google Postmaster and select “Verify.” If the entry is not found, check that the TXT was added correctly and try again. If you have trouble adding a TXT record, Google offers a second verification method that involves adding a “CNAME” record:

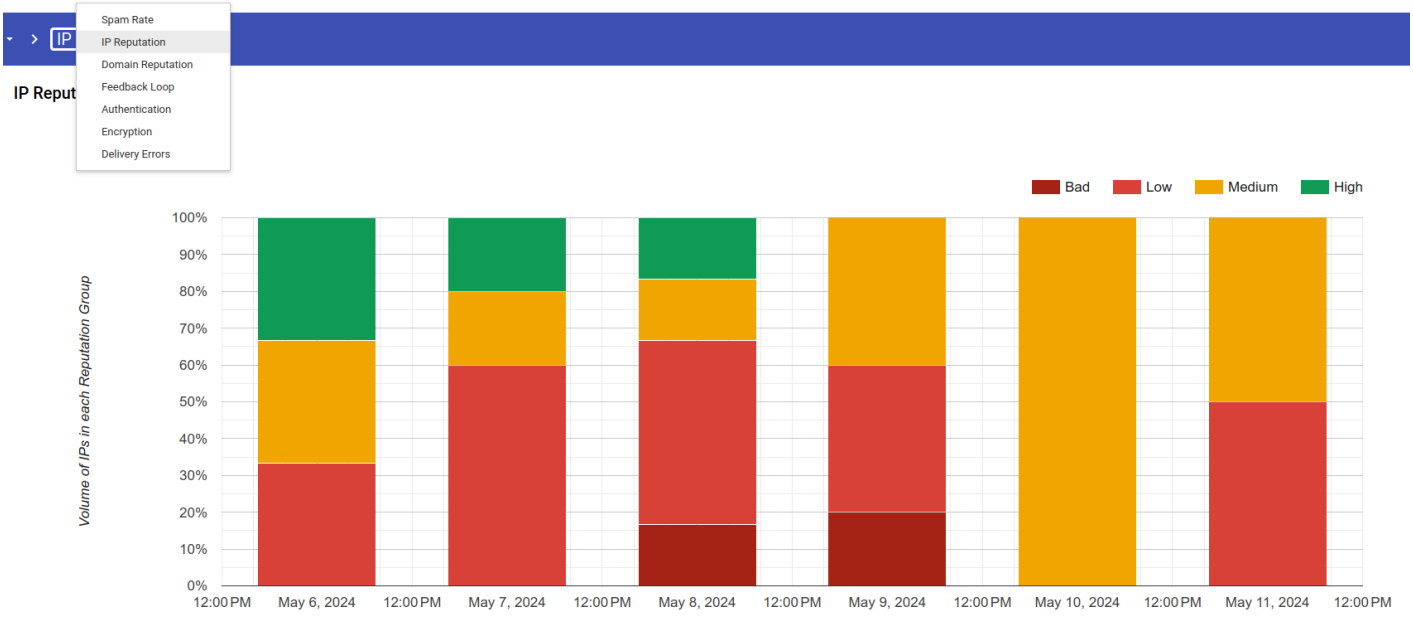


After confirming the rights to the domain, it will be displayed in the Postmaster Tools interface with the status “Verified”:

Domain	Status	Added
example.net	Verified	Aug 25, 2017
example.com	Verified	Aug 25, 2017
example.org	Verified	Dec 29, 2020
example.pro	Verified	Mar 2, 2023
example.xyz	Verified	Mar 14, 2023
example.io	Verified	Sep 25, 2023
example.su	Verified	Feb 26, 2024

Gmail Postmaster allows you to collect information about your mailings. The following data is tracked:

- **percentage of getting into spam;**
- **IP and domain reputation;**
- **information about authentication (passing DKIM, SPF and DMARC checks);**
- **information about traffic encryption (TLS share);**
- **information about delivery failures.**



All reports in Gmail Postmaster are generated in the form of graphs and tables with a selection of data for a period of 7 to 120 days. Information in reports may be missing if there is a small number of sends to Gmail mailboxes. Data will begin to be displayed when there is a sufficient amount of data (a hundred or more records) about daily traffic.

Setting up FBL forwarding

To set up FBL, the sender must add a new Feedback-ID header, consisting of parameters that uniquely identify the mailing. Such parameters are called identifiers. IDs whose messages often end up in spam or have delivery problems are displayed in the FBL panel in Postmaster Tools.

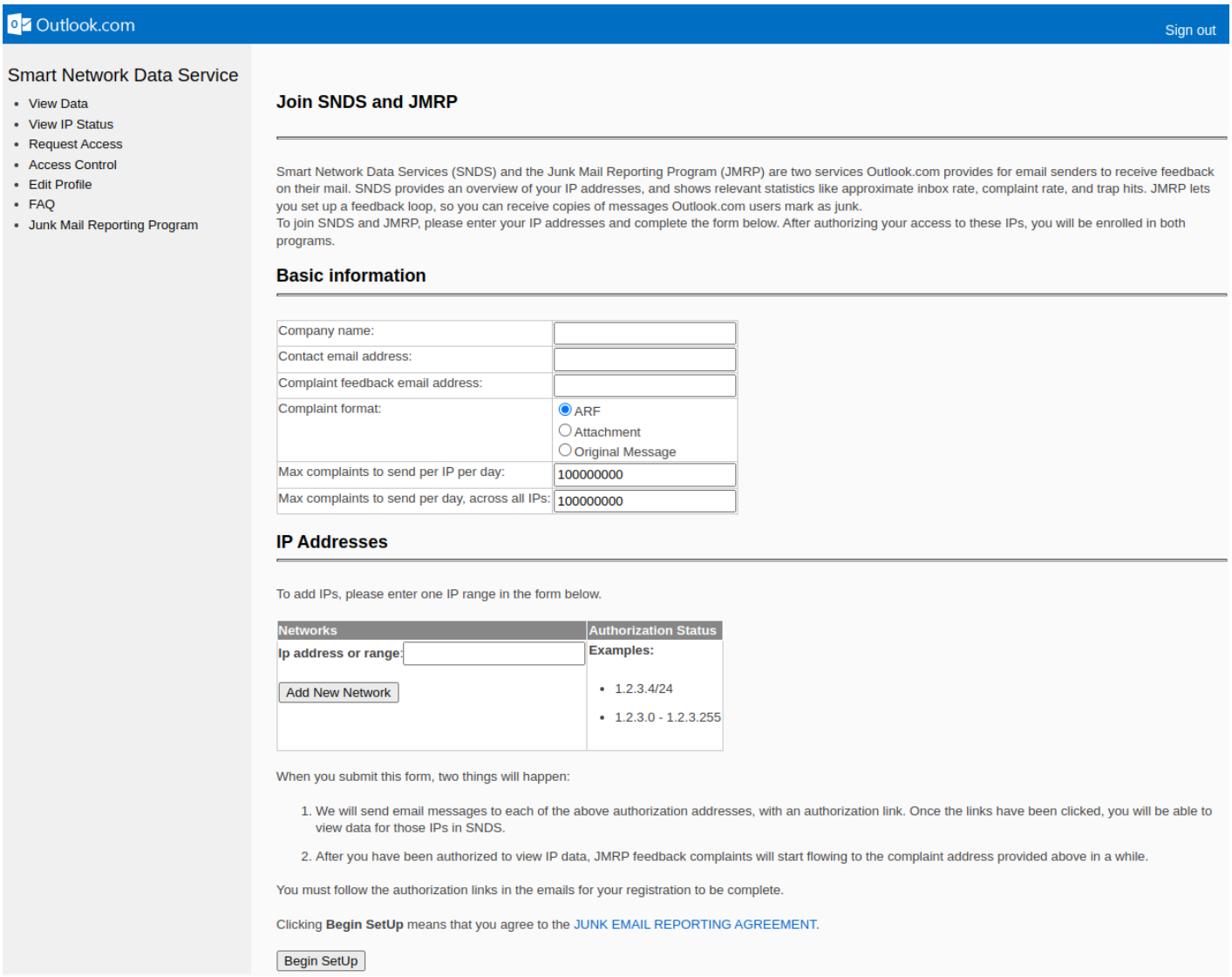
The header format is `Feedback-ID: a:b:c:SenderId`**,**

where `Feedback-ID:` **is the name of the header to add, and** `a, b, c:` **are optional fields with which the sender can specify up to three identifiers (for example, with the name of a company or client).** `SenderId` **– mandatory unique identifier selected by the sender (5–15 characters). It must be the same in all emails.**

How to set up Outlook.com Postmaster

Step 1. Sign in to your Microsoft account and go to service.

Step 2. Complete and submit the form on the [Junk Mail Reporting Program] page. Here you need to enter all IP addresses for mailings:



Step 3. After entering the IP address, the service asks you to select mailboxes to which, after submitting the form, you will receive a message with a confirmation link. You will need to open this link.

After all requirements are met, notifications about complaints about mailings will begin to arrive at the specified mailbox.

[Outlook.com] Postmaster Services allows the user to:

- **Receive reports on spam incidents reported by [Outlook.com] users**
- **recognize potential problems with email distribution methods and content**
- **improve sender reputation by removing unwanted subscribers from lists**
- **provide general information about how users rate the email they receive and the reliability of a range of sender IP addresses based on [Outlook.com] system metrics**
- **see how your email is rated by spam filters**
- **track the number of users complaining about your email**

How to set up Yahoo FBL

Yahoo provides a tool for senders to unsubscribe users who complain about spam - Yahoo Complaint Feedback Loop Service (YFBL).

Step 1. For YFBL to work, you will need a mailbox like postmaster@(your_domain). Create it on the service where your domain mail is configured.

Step 2. Follow the link and fill out the required fields:

- **Company name — name of the organization;**
- **Contact Name — name of the contact person;**
- **Contact Email — email of the contact person;**
- **Reporting Email - address for receiving FBL reports;**
- **Selector — specify “*”;**
- **Request type — request type: adding, changing or deleting. Specify “Add”;**
- **Domain — the domain for which you are setting up FBL. After filling out the required fields, click the “Get Verification Code” button.**

Step 3. Find the email from the service in the created mailbox and copy the confirmation code. Paste the copied code into the appropriate field on the YFBL service page and click “Create Request”.

Please note that it takes up to 48 hours to confirm your settings and enable reporting. After successful setup, the service will work as follows: if the recipient clicks the “Spam” button in a letter from your mailing list, an FBL report will be sent to your account, and the recipient will be automatically unsubscribed from mailing lists.

Quick Start

How to send your first email mailing

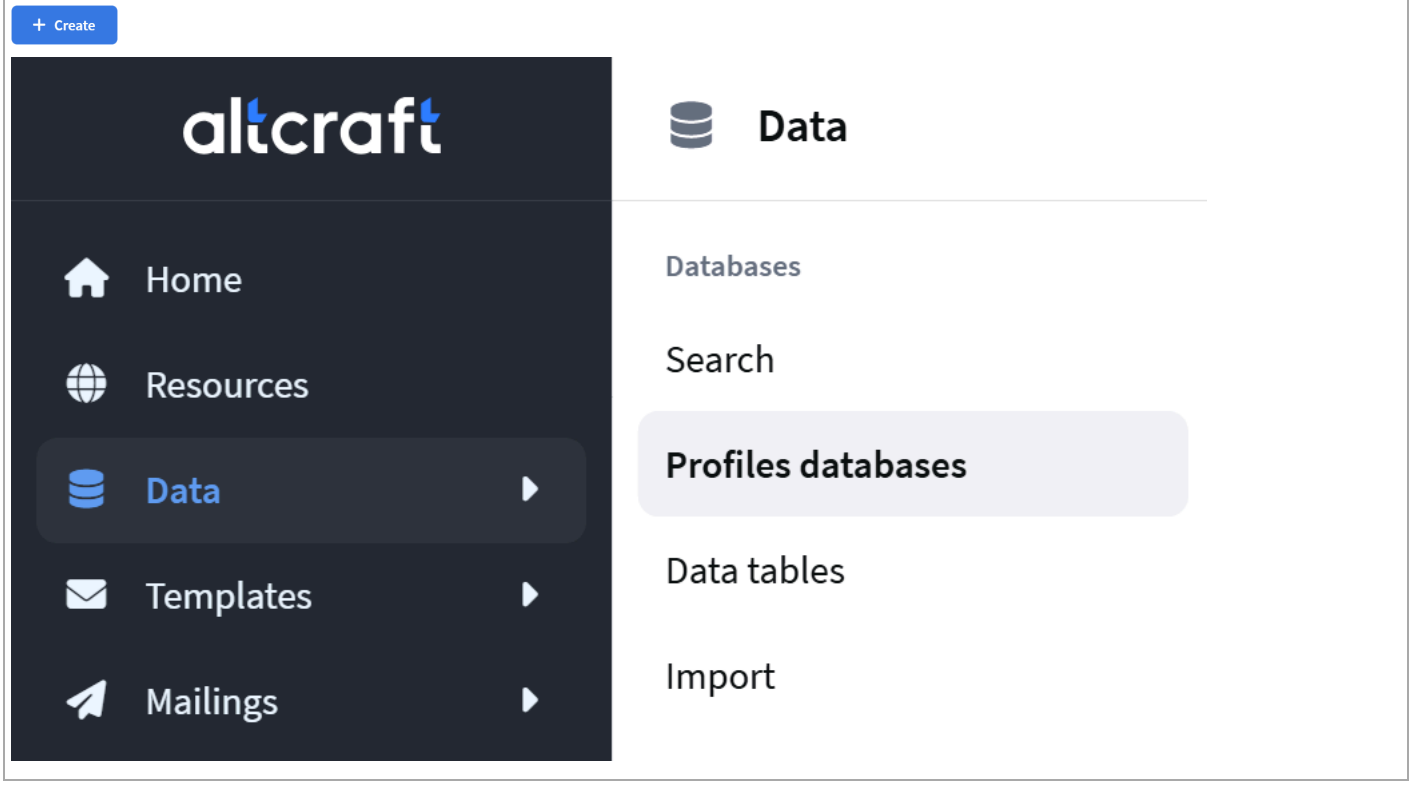
This article shows the basic workflow in Altcraft Platform. The goal is to send your first email and understand how the platform works.

Sequence of steps: Profile Database — Resource — Profile — Template — Mailing — Reports

Step 1. Create a profile database

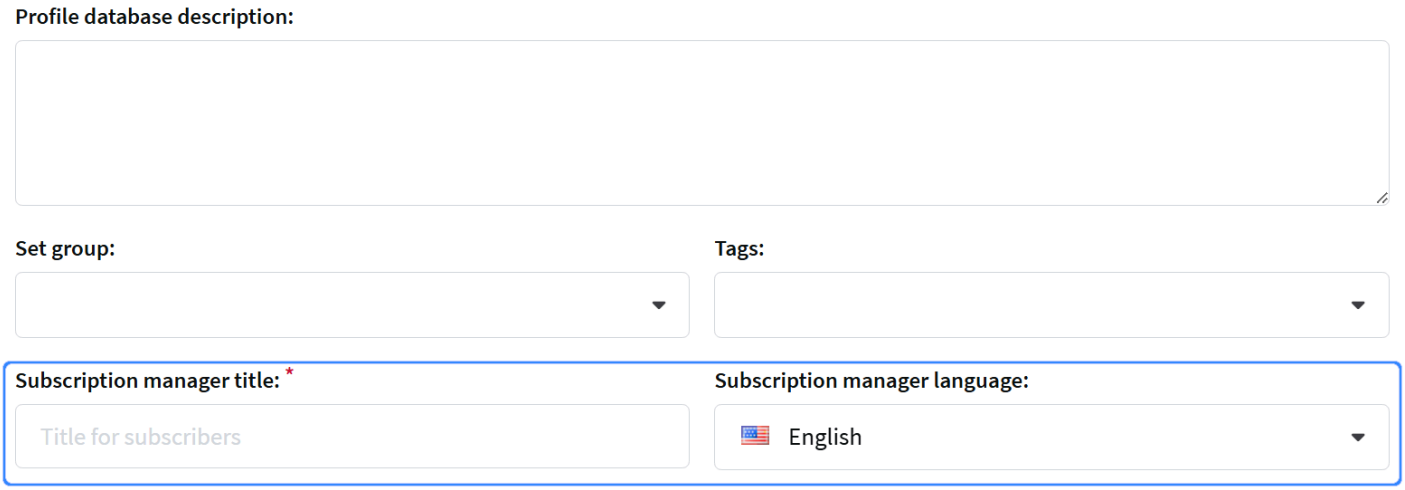
Profile Database is where customer data is stored. In the documentation, you may see terms like “customer”, “subscriber”, or “user”, but in all cases this refers to a profile.

1. Go to **Data — Profile Databases** and click:



---

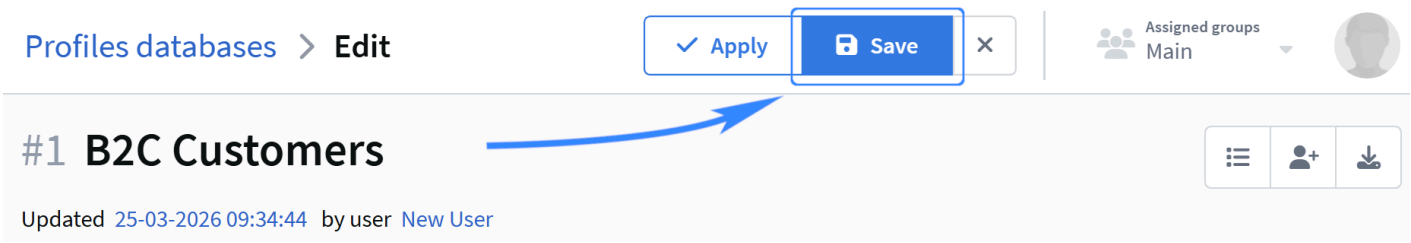
2. Enter the **Profile Database Name**



Subscription Manager Title — text on the subscription management page (for example, your company name).

---

4. Save the database



For more details on working with profile databases, see this article.

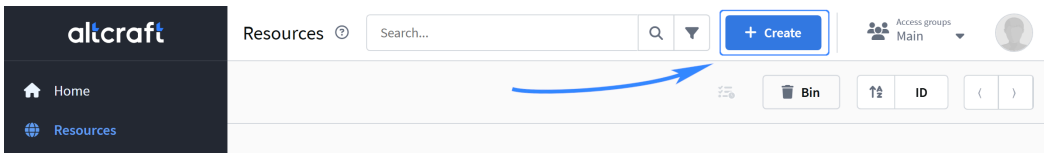
Step 2. Create a resource

Resource is the source of communications: a website, service, brand, or project. Subscriptions are collected and messages are sent through a resource. You cannot create a mailing without a resource.

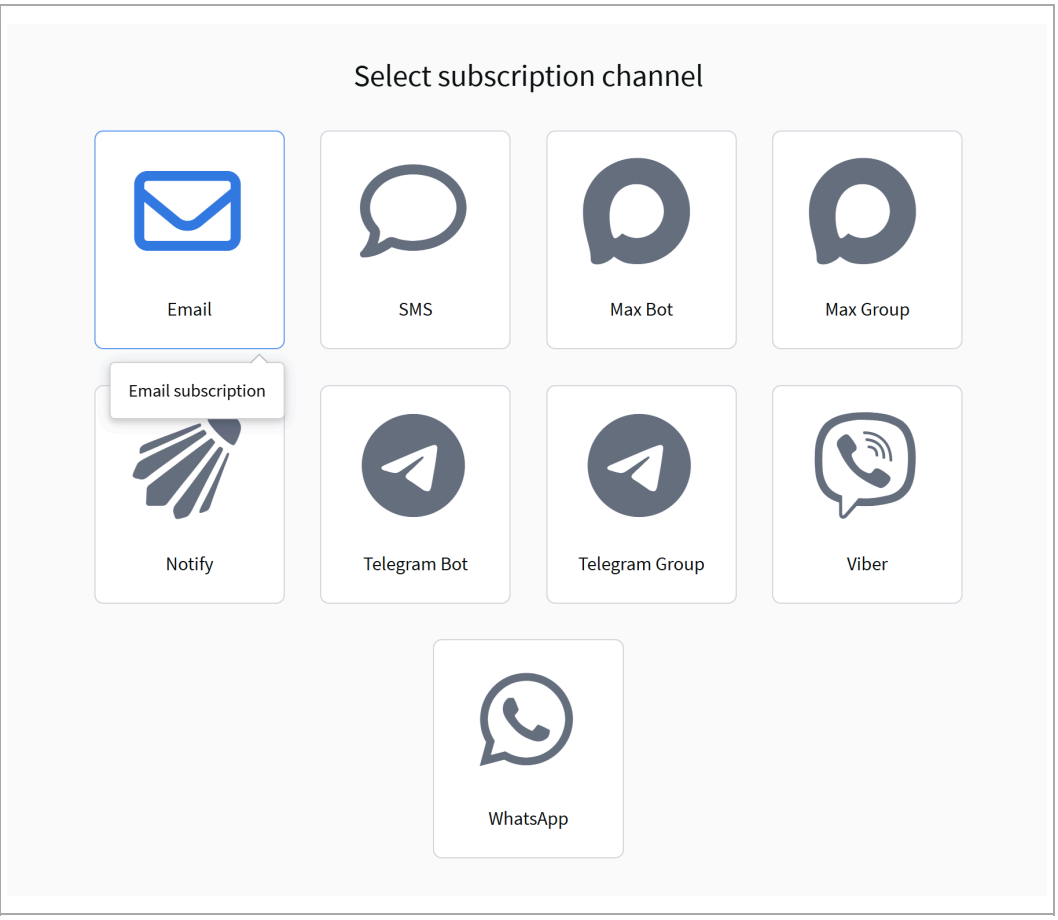
A resource must be linked to a profile database.

INFO

For email sending, DKIM must be configured for the sender domain. It is not provided automatically. Contact technical support chat to enable it.

1. Go to the **Resources** section	

2. Select a subscription channel. **For your first mailing, we recommend using Email**



3. Enter the **Resource Name**

4. Select the previously created Profile Database

5. Enter the Website URL and Tracking Domain

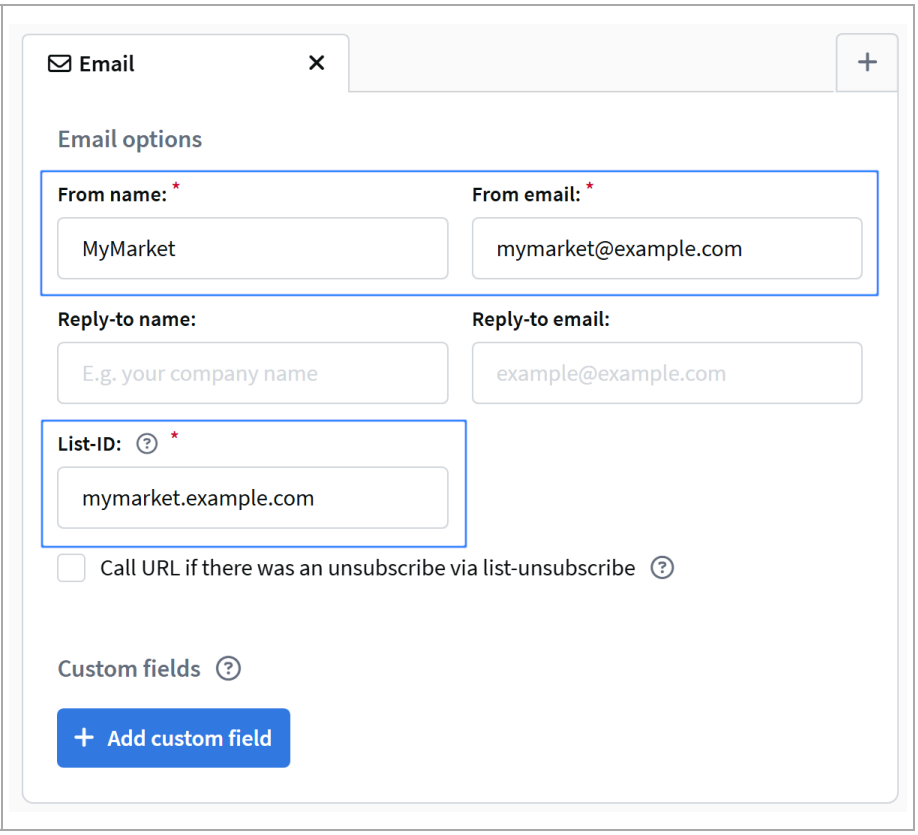
```
</td>
  <td width="55%px">
  <img src="/img/attachments/quick-start/en/resource-main-settings.png" class="img"> </img>
  </td>
```

Website / form / application URL — the address where users submit their contacts. You can specify your company's main website

Tracking Domain — technical domain for tracking link clicks from emails. If the field is already filled, leave the value unchanged

6. Enter the **Sender Name** and **Sender Email**

7. Enter the List ID

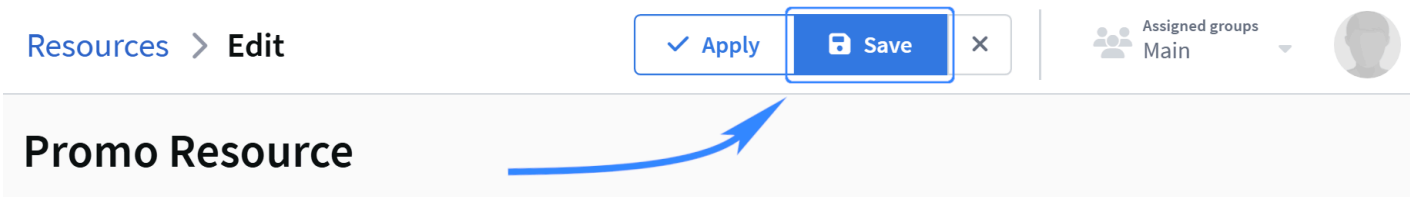


- **Sender Name — the name shown in emails - **Sender Email** — the sender address - **List ID** — identifier of the subscription list within the resource. Specify any clear value you understand, you can use the sender email in format `example.domain.com`**

INFO

Sender Name and Sender Email were already specified in the profile database. In resource settings, these values take priority.

8. Save the resource

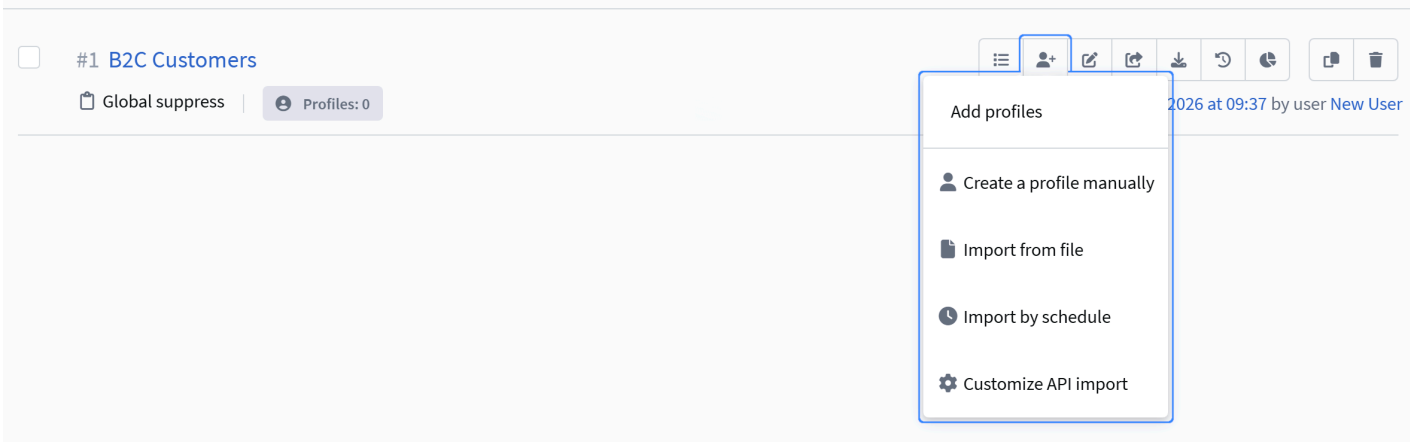


For more details on working with resources, see this article.

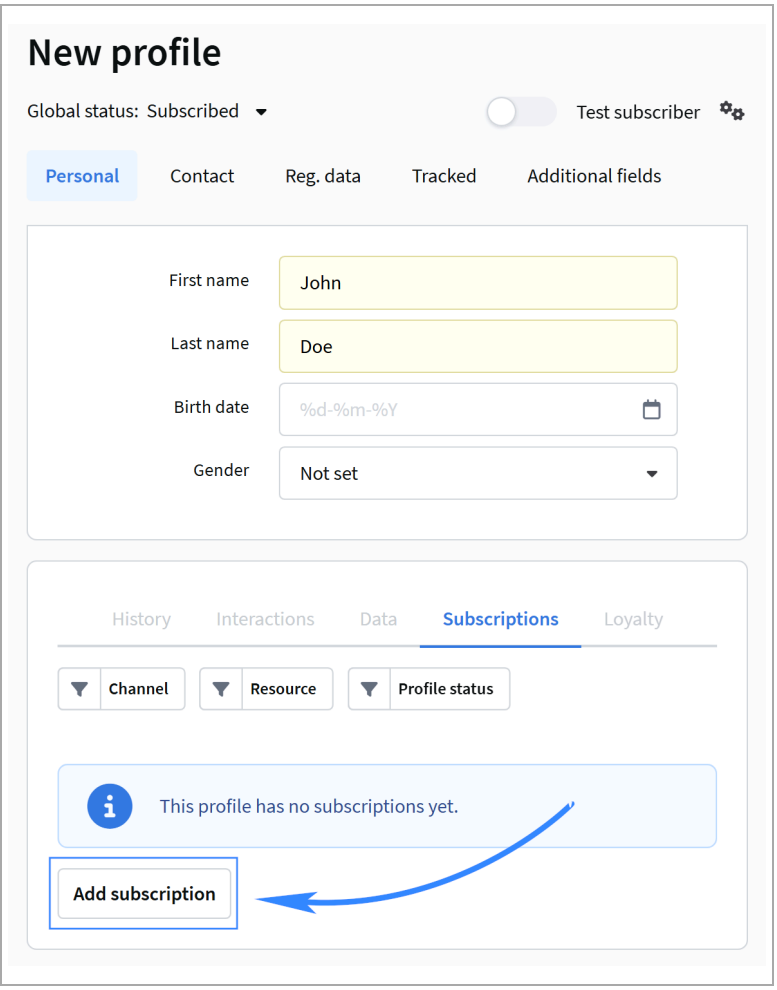
Step 3. Add a profile

Create a profile to send the email to. To do this, go back to Data — Profile Databases.

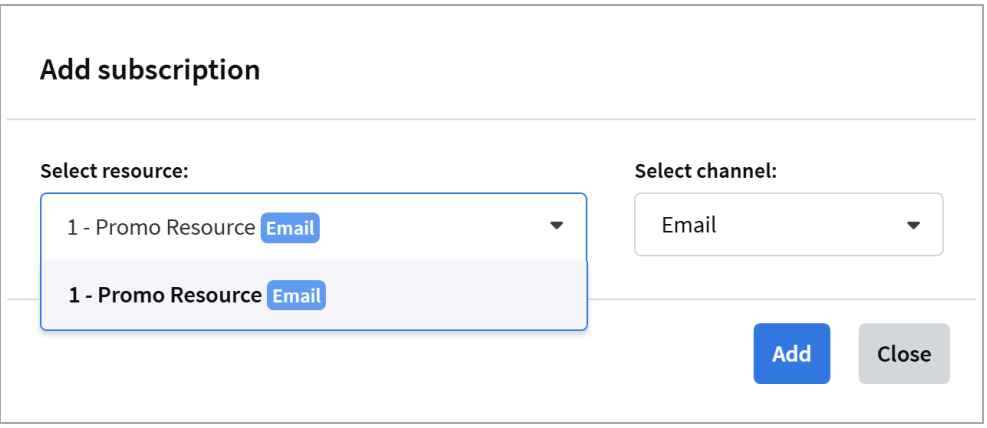
1. Click Import customer profiles — Create profile manually



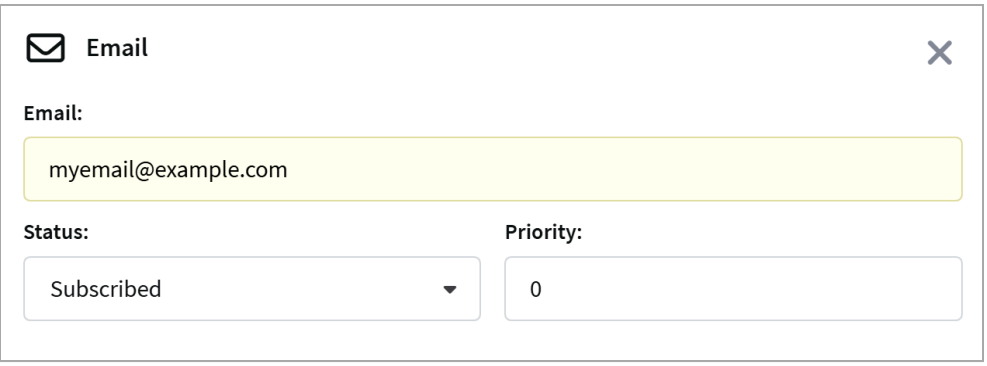
2. Enter the profile name and click **Add subscription**



3. Select the created resource and the Email channel



4. Enter the email address to send the message to



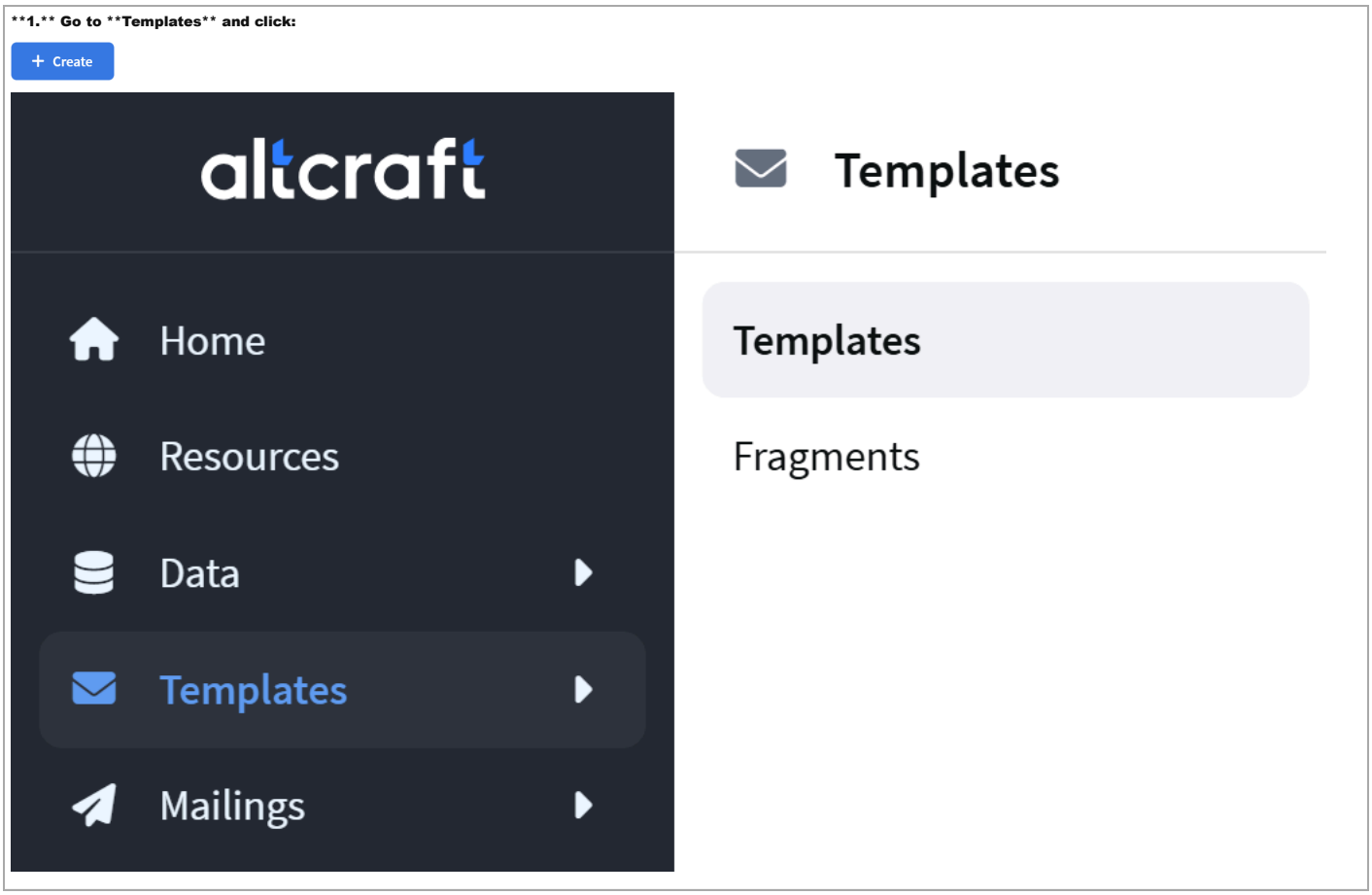
5. Save the profile

For more details on profiles, see [this article](#).

Step 4. Create a message template

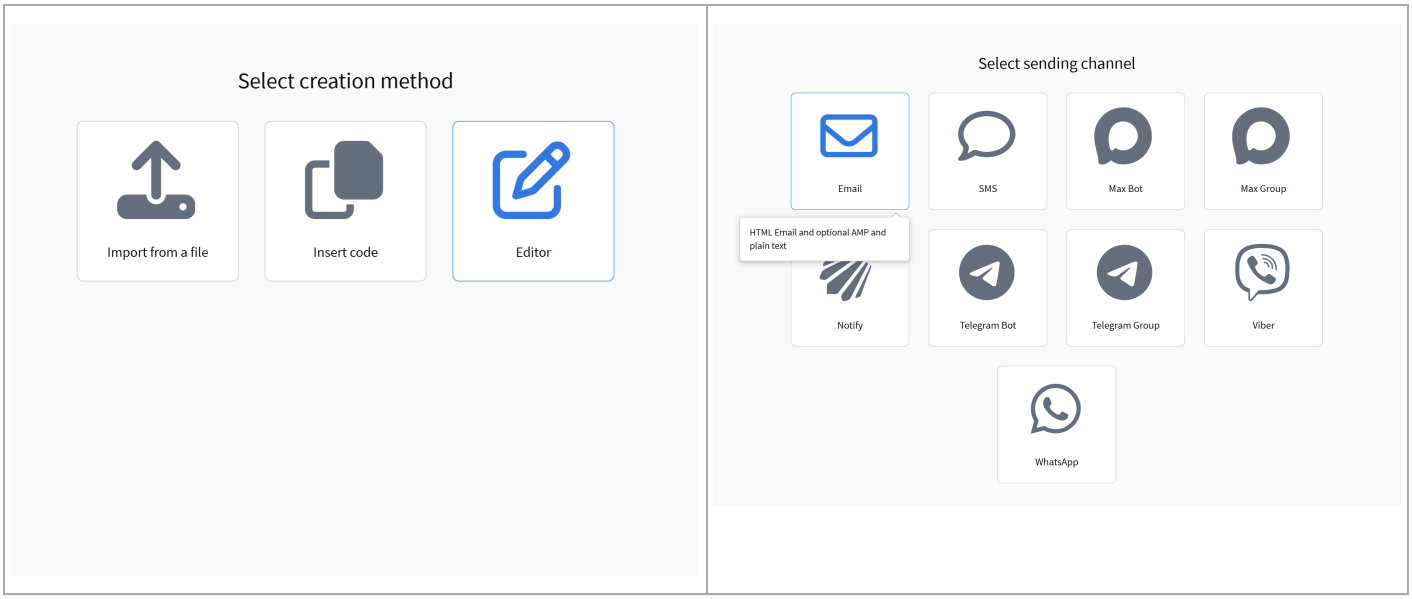
Template is the email content.

1. Go to **Templates** and click:



---

2. Select creation method Editor and channel Email

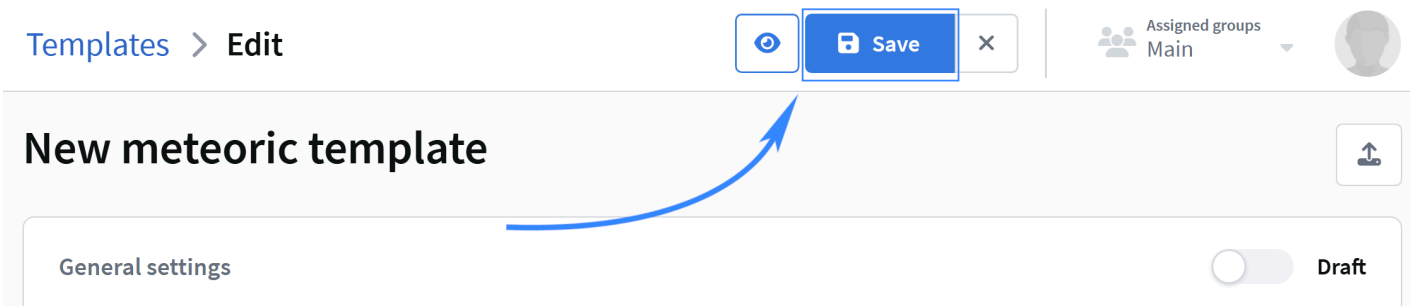


---

3. Open the Code Editor:

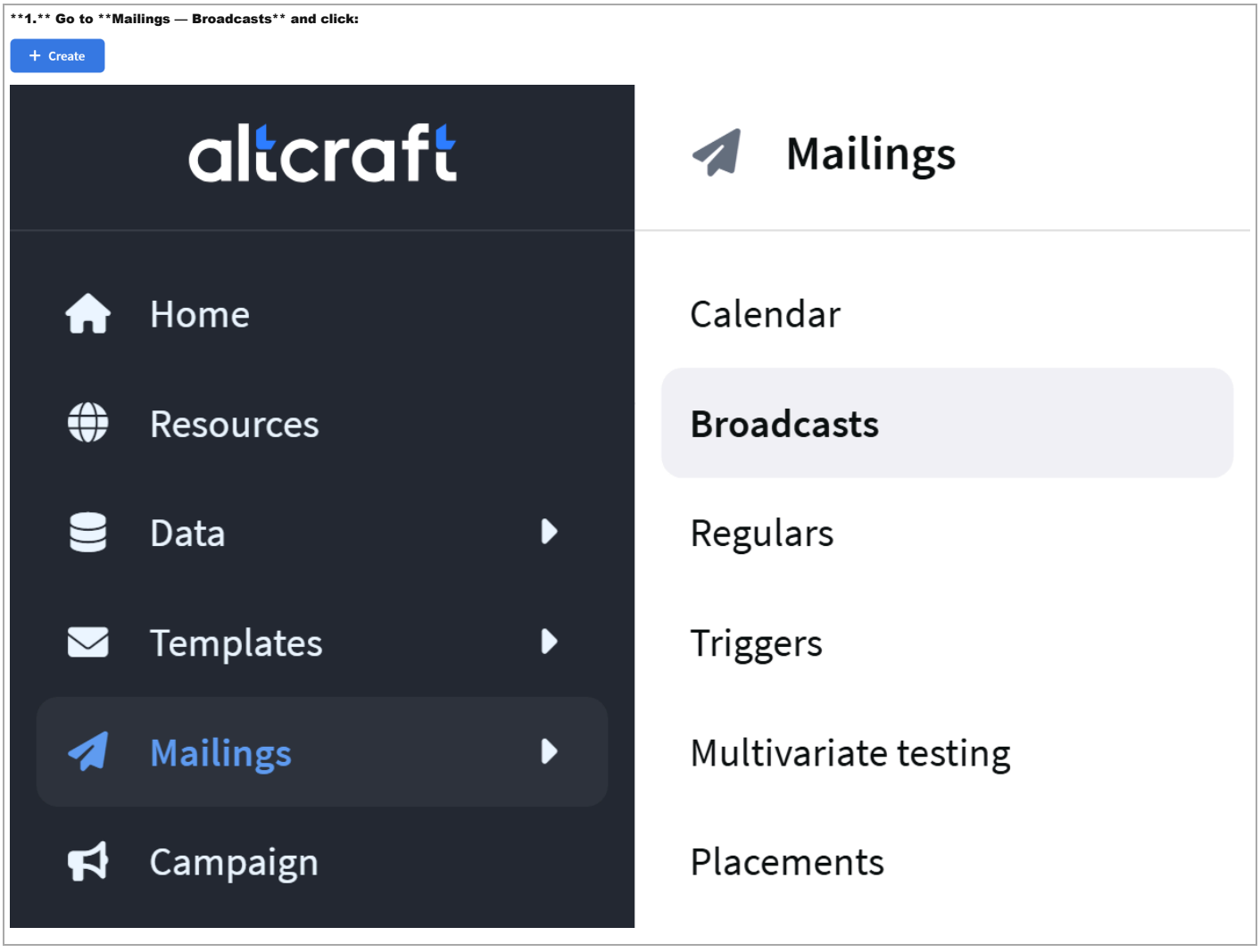
<img src='/img/attachments/quick-start/en/code-icon.png' style="width: 200px class="img">

and save the template:



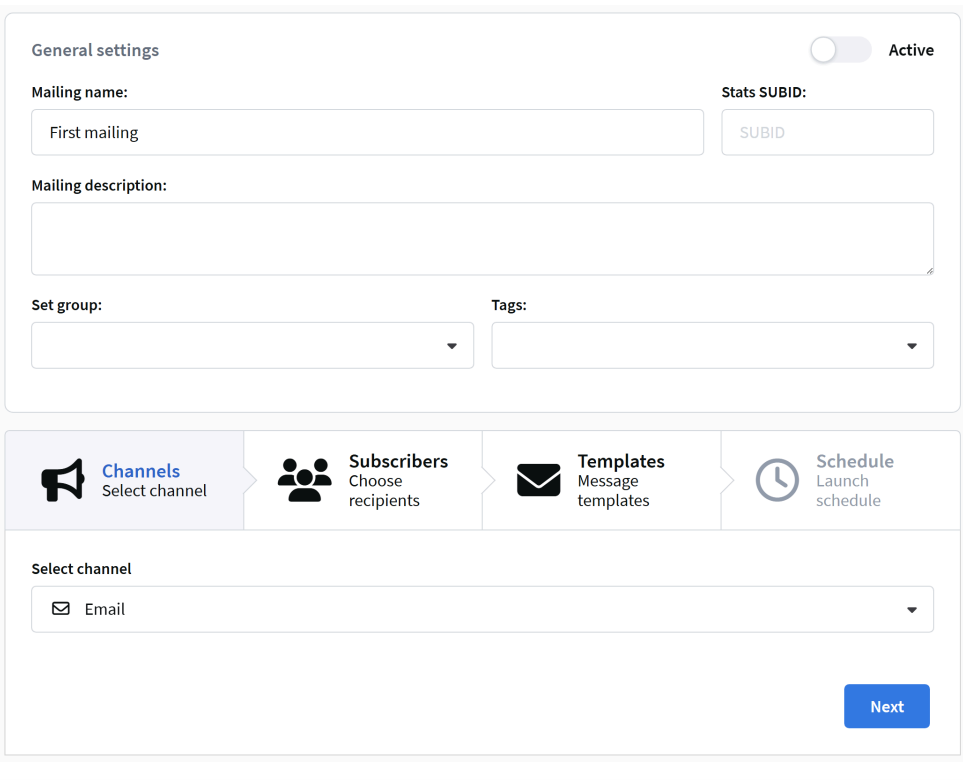
In this case, the default (placeholder) message template will be used. To configure or import your own templates, we recommend reading the [Message Templates] section.

Step 5. Create a mailing



---

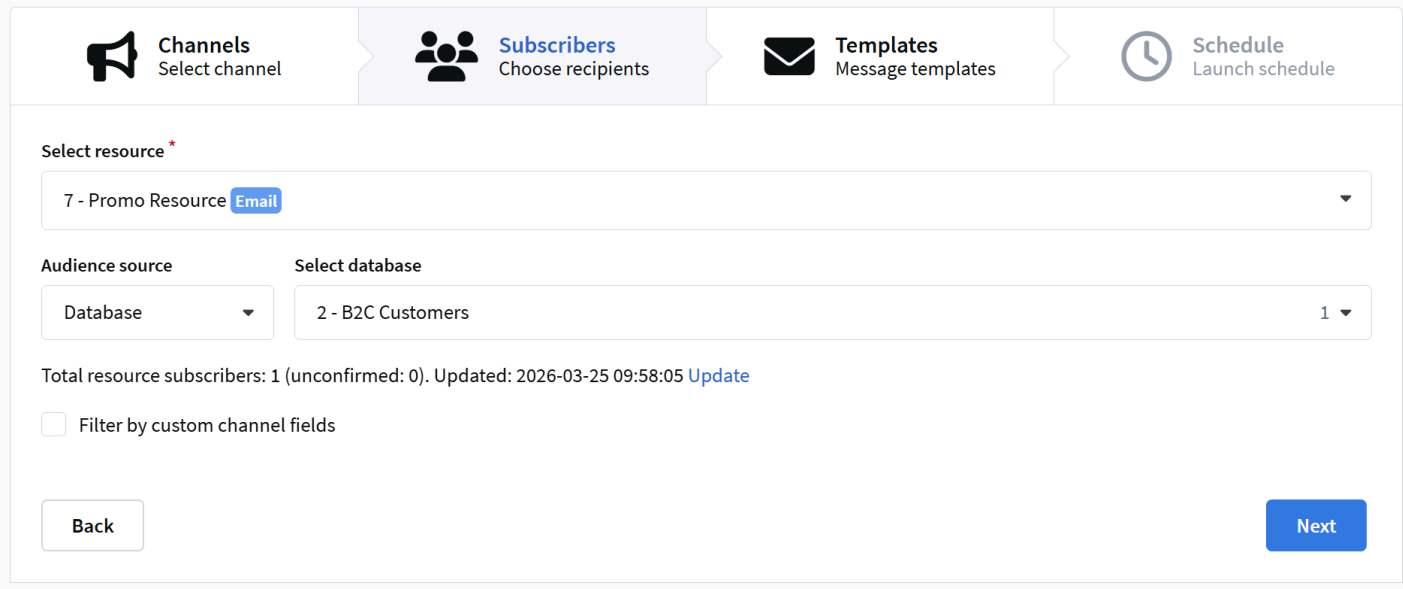
2. Enter the **Mailing Name** and select the **Email** channel



3. Configure recipients

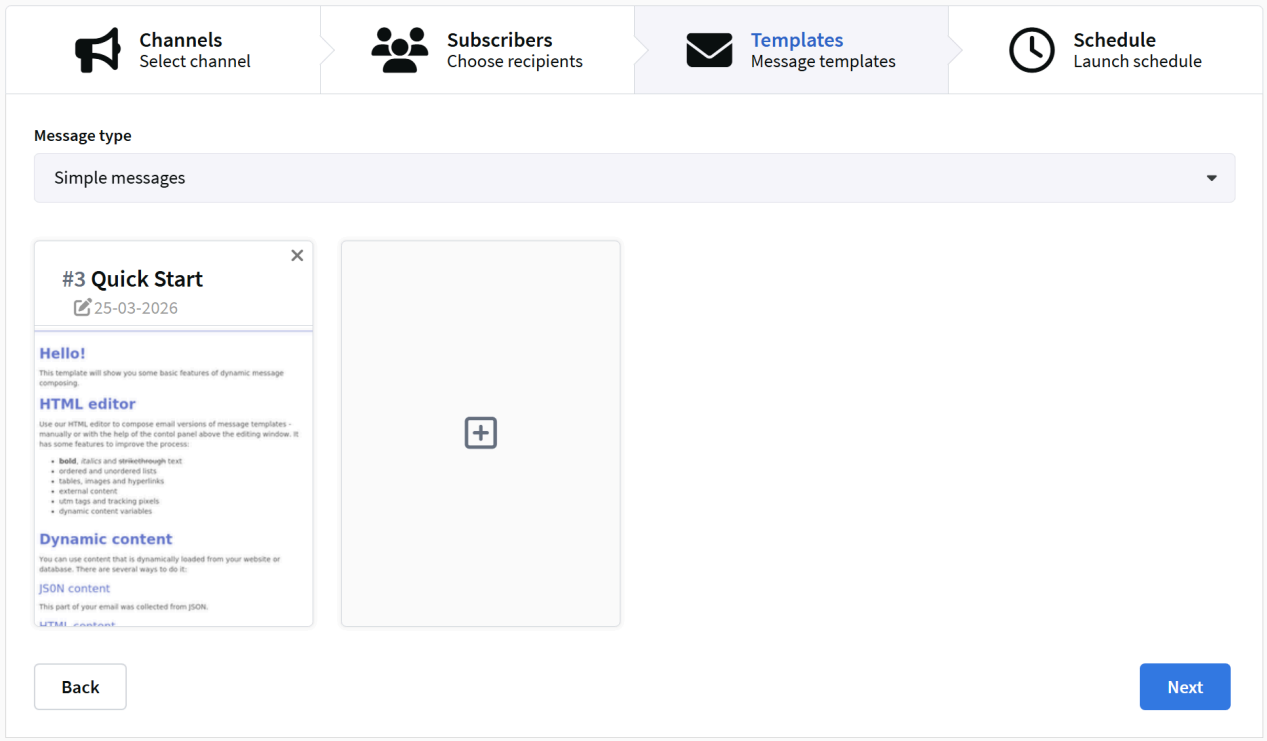
Go to the Subscribers tab and specify:

- **Resource — select the created resource**
- **Subscriber source — leave Database**
- **Profile Database — select the database created in step 1**



4. Select content

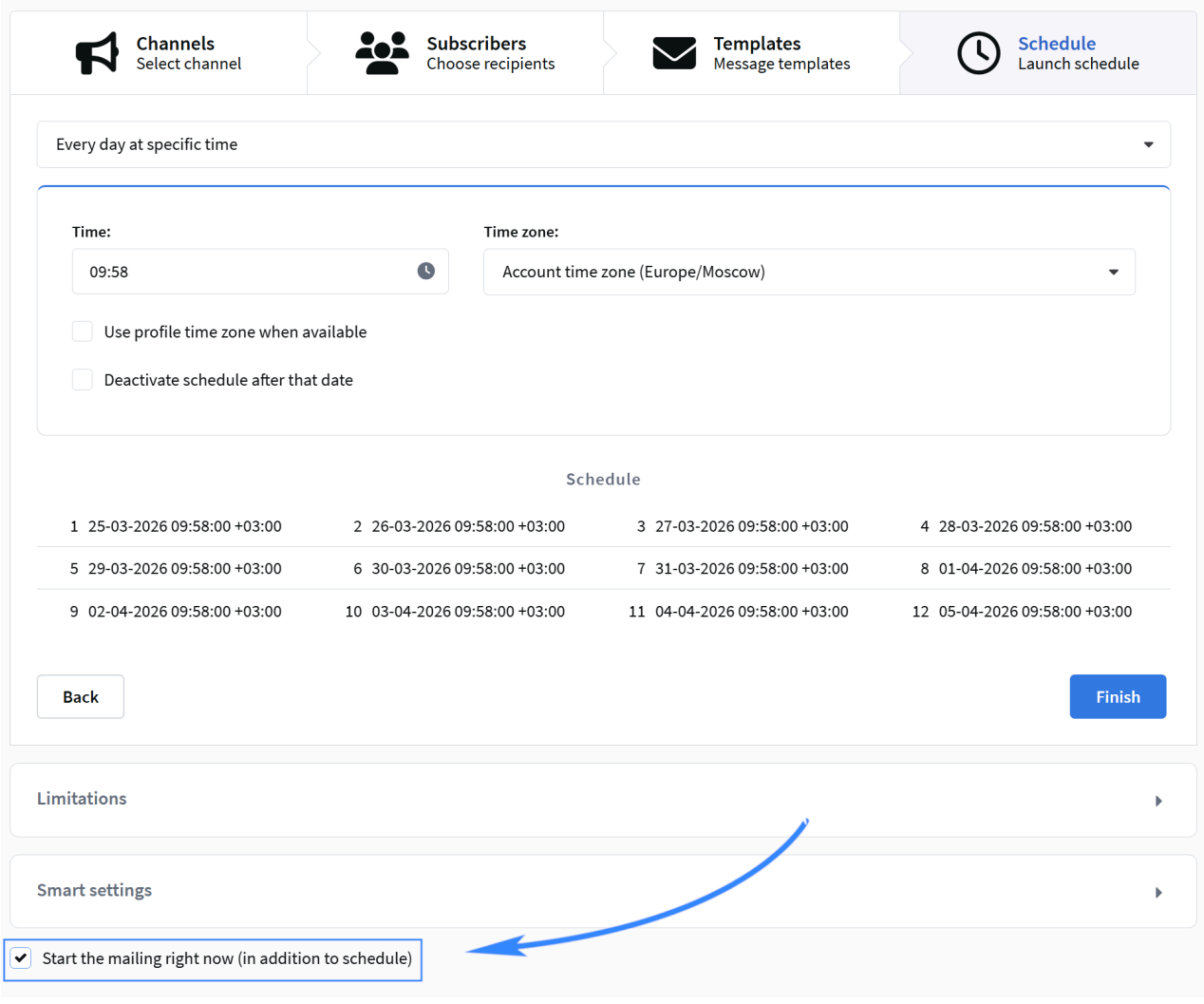
In the Templates tab, select the created Message Template.



5. Set the sending time

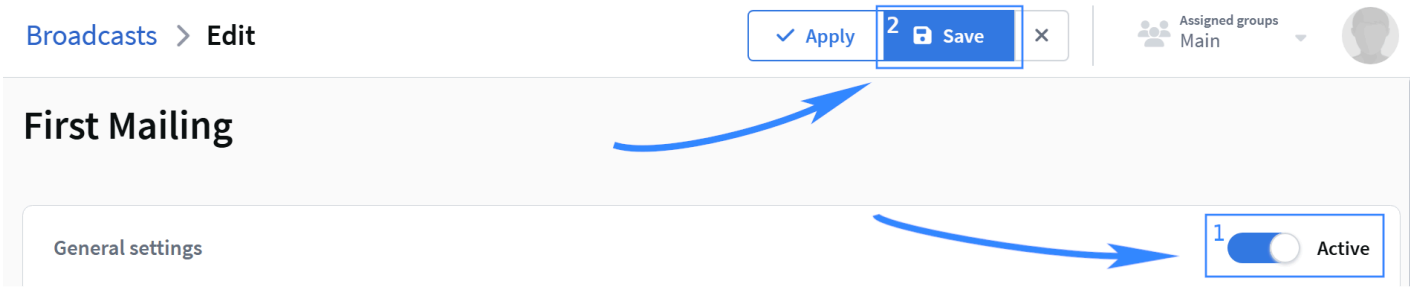
Go to the Schedule tab:

- **You can set a specific start time**
- **To send immediately after saving, enable Start mailing now (in addition to schedule)**



6. Save and start

Activate the mailing and click Save. After saving, the mailing will start sending according to your settings.



The email will be sent to the created test profile.

INFO

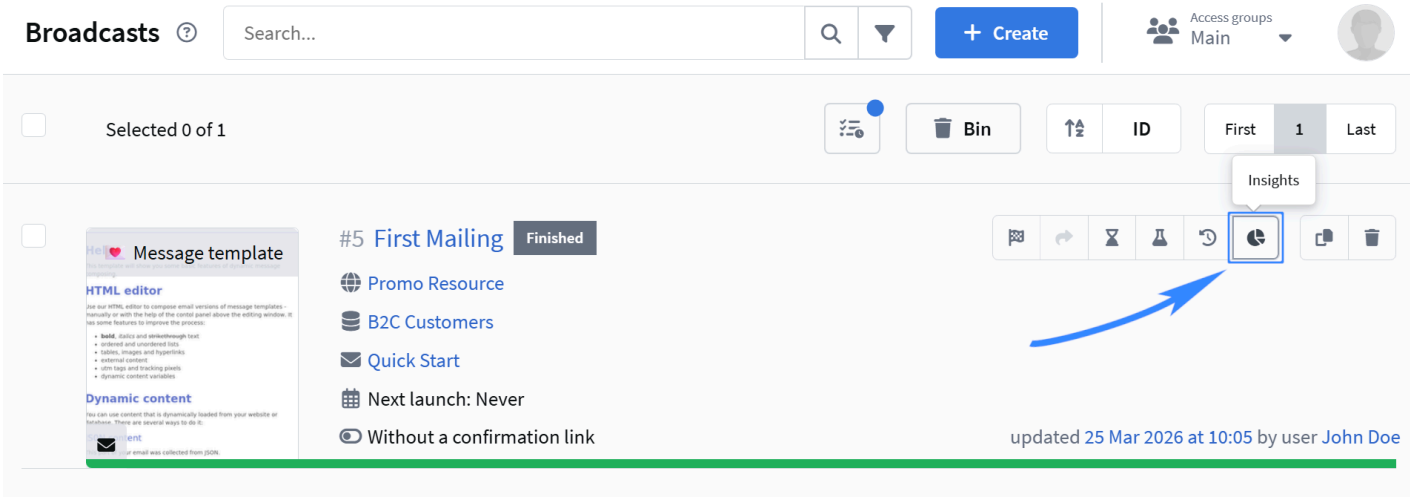
For more detailed information on sending email mailings, you can refer to this article.

Step 6. Check the result

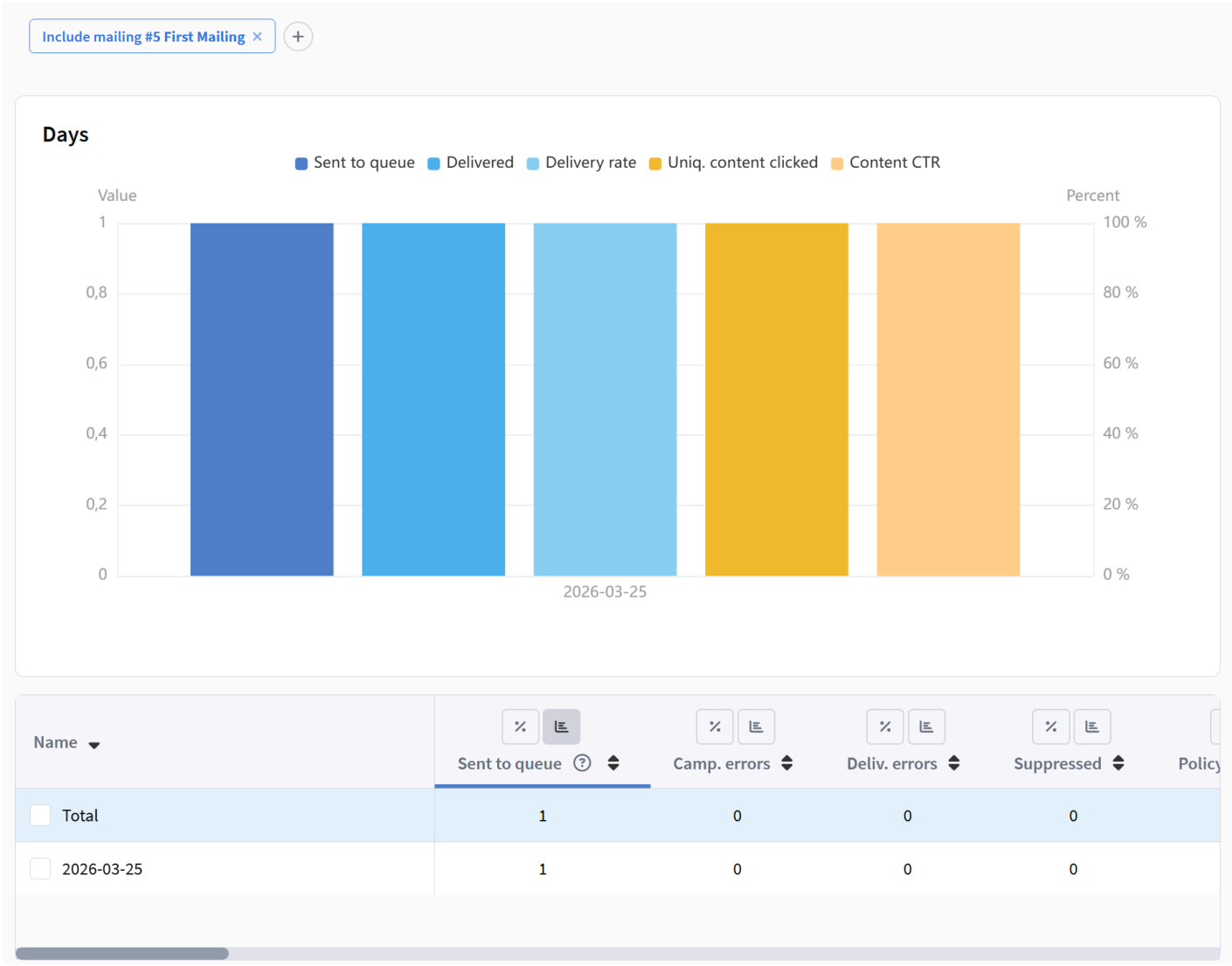
You can check the results in two ways: in the analytics section or in the profile card.

Method 1. In the “Insights” section

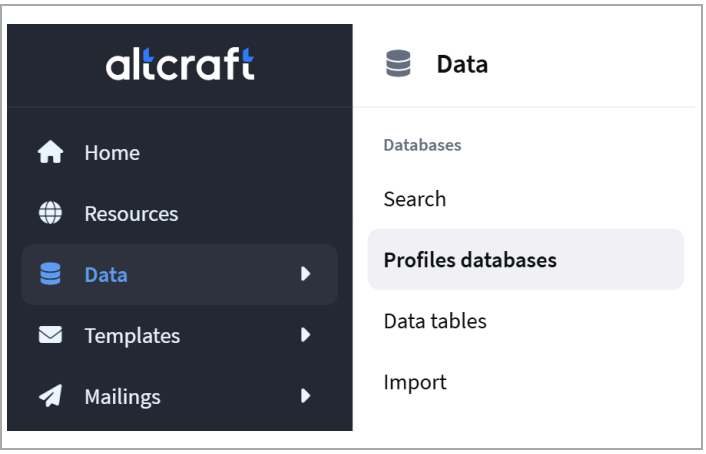
**1. Select the completed mailing and click **



2. Review the statistics



Method 2. In the profile

1. Go to **Data — Profile Databases**	

2. Open the created profile card and go to the History tab — it contains all information about the profile:

History | Interactions | Data | Subscriptions | Loyalty

Mailing actions | Last 7 days

1 - 4 from 4

Email message clicked

wed, 25 Mar. 2026 10:05:42

Template: #3 Quick Start

Mailing: #5 First Mailing

Resource: #7 Promo Resource

Subscription hash ID: #7f7d460e

Email: myemail@example.com

Address: 120.1612°, 30.2994°, China, Zhejiang, Hangzhou

UTM Content: 3

UTM Source: email

UTM mailing: 5

UTM Term: Quick Start

Email message opened

wed, 25 Mar. 2026 10:05:42

Template: #3 Quick Start

Mailing: #5 First Mailing

Resource: #7 Promo Resource

Subscription hash ID: #7f7d460e

Email: myemail@example.com

Address: 120.1612°, 30.2994°, China, Zhejiang, Hangzhou

INFO

For more detailed information on reports, you can refer to this article.

First Email Mailing

	This tutorial will help you to create and launch your **Email mailing from scratch: from setting up the sending domain to viewing mailing statistics.** **Email** remains one of the most effective marketing communications with customers. You can send more content to your subscribers by using email channel than by using other channels: images, links to various products and services, interactive elements that attract attention and increase customer engagement. Altcraft Marketing monitors the actions of subscribers in an Email message, such as opening, reading, clicking on a link. </td> </tr>

Step 1: Configure Email Sender

Altcraft Marketing platform sends email using SMTP protocol. This is the reason why we have developed our own transport module AKMTA (Altcraft Mail Transfer Agent), which allows you to optimize the delivery of letters. Even if the mailing audience is tens of millions of recipients!

Sending and tracking domains

You need the sending domain to send email messages. The domain will be used in the sender's address (from email), which the recipients see in the mail client. Using SPF and DKIM technologies, the domain is connected to the SMTP sender, which prevents your mailings from being spoofed.

You need the tracking domain to track email messages. This can be a subdomain of your main domain, or a separate domain. Clicks on links in the message are redirected to the tracking domain — this is how the platform tracks clicks. In addition, through it images are loaded into messages for registering letter openings.

When you have purchased a domain — on the side of the domain name registrar that provided it, you need to make configuration records to ensure efficient delivery.

> TIP
>
> **Custom domain settings are described in another article. If you use an Altcraft Marketing cloud account, please give the information to the support team at support@altcraft.com.**
>
> **When installing the platform on your equipment, contact us and we will help you set up the sending infrastructure.**

Installing transport module AKMTA

> CAUTION
>
> **If you use a cloud service, we provide our own sending infrastructure with all the necessary settings. The only thing you need to start sending is your configured domain (see the previous point).**

If the Altcraft Marketing platform is installed on your equipment (On-Premise), you can configure one or several sending servers with AKMTA transport modules. In the platform administrative panel, you need to configure:

- **Nodes are network dispatch nodes. The module uses the sending IP addresses available to you.**
- **Senders are configuration records that combine parameters for sender identification, sending speed, and settings for the sender's behavior in case of sending errors.**
- **Virtual senders are the rules for sending from multiple senders. Can be configured both in the Administration Panel and in the User Interface.**

Step 2: Subscribe customers to the Email channel

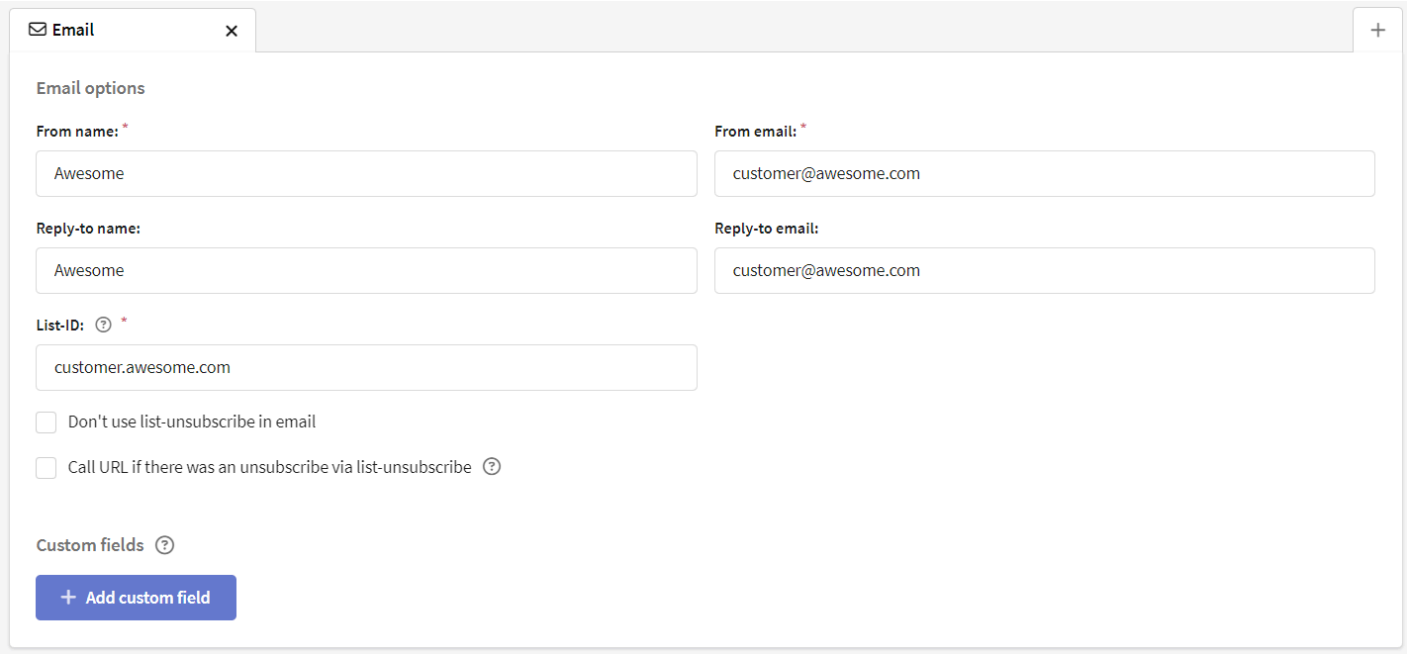
Profiles Databases

Altcraft Marketing emails are received by your subscribers. Subscriber profiles are stored in profiles databases.

Each customer profile has a system field "Email": it is used to identify profiles and in cases where your customers only have one email address.

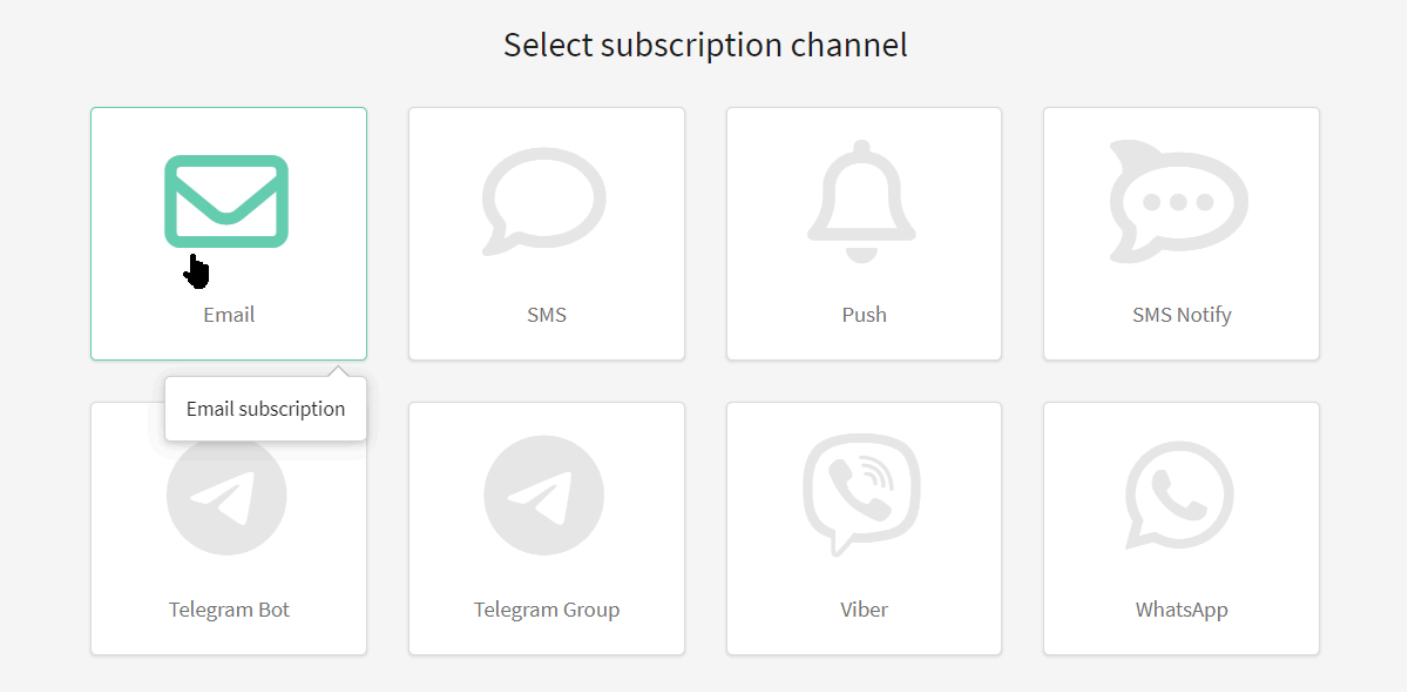
> TIP
>
> **If you are using different contact information for the same customer or submitting messages with different content - set up multiple subscription resources.**

When you create a database, it's necessary to enter the sender's email (from name). On his behalf, the sending will be made without using resources. Enter the sending address in the corresponding field and don't forget to enter the sender's name (from name). If it's necessary, specify the data for response to the letter (Reply-to name & Reply-to email):

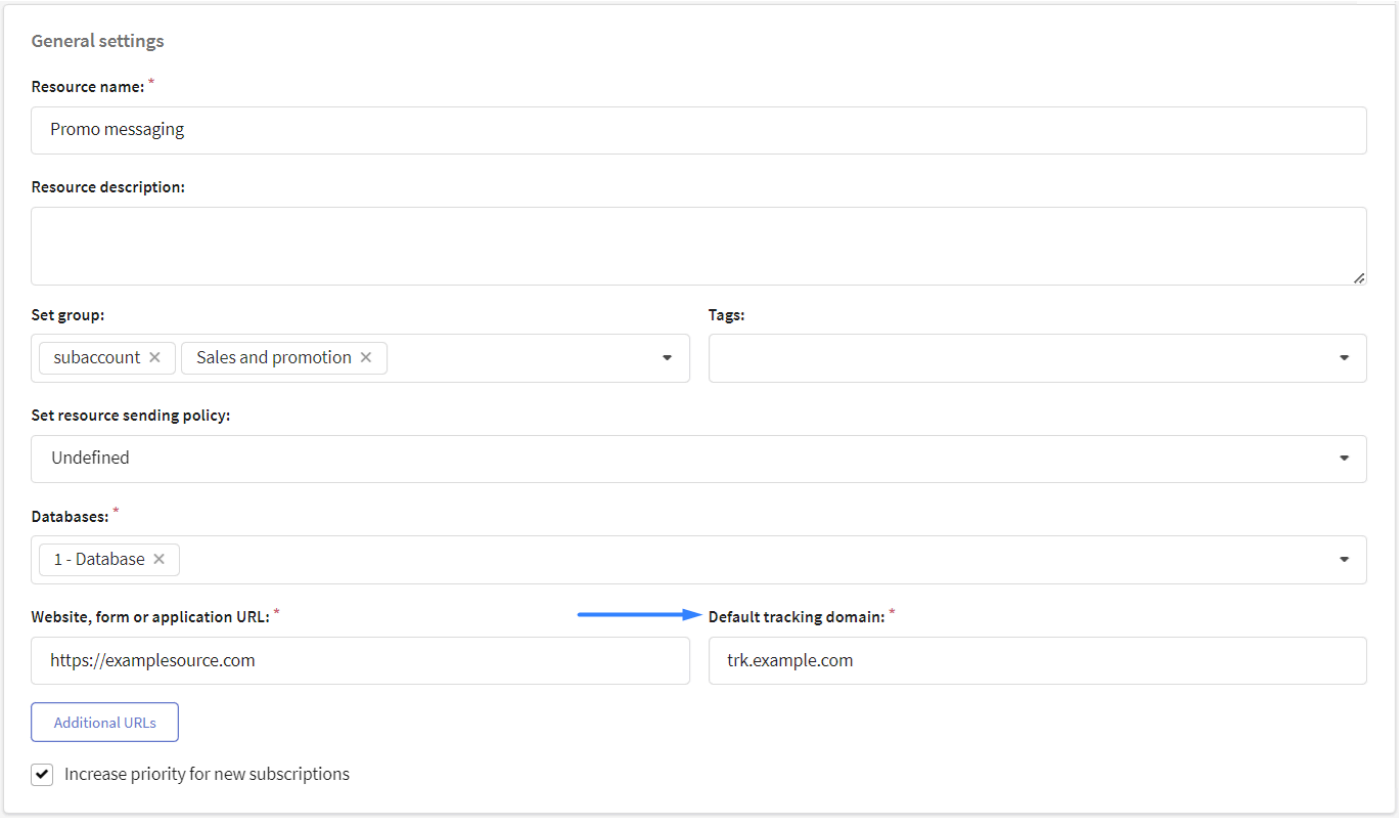


Subscription resources

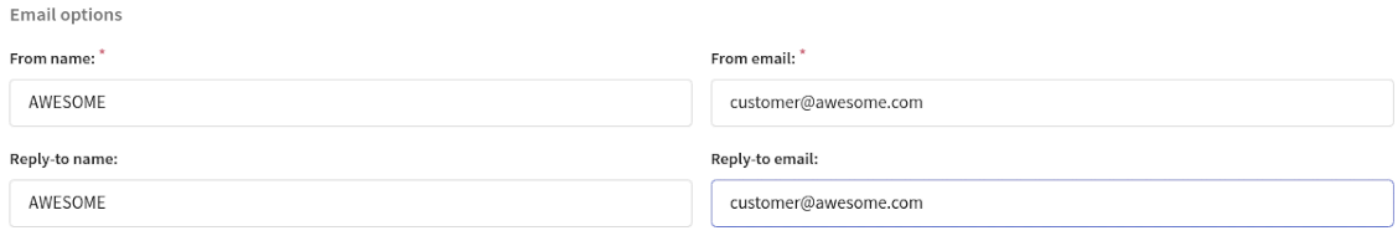
Each client profile can be subscribed to one or more resources. What can be a resource? A specific marketing area, a site, an application, or the type of content. Select the corresponding item in the main menu, then create a new resource or select an existing one. Select an Email channel for a new resource:



In the general settings, add the necessary databases. If it's necessary, assign a sending policy and specify the Default tracking domain configured earlier:



In the Email options of the resource channel, specify the name and email of the sender (From name & From email).If it's necessary, enter the name and address for receiving responses. These settings will override database settings before sending.



Customer profiles and subscription to the resource

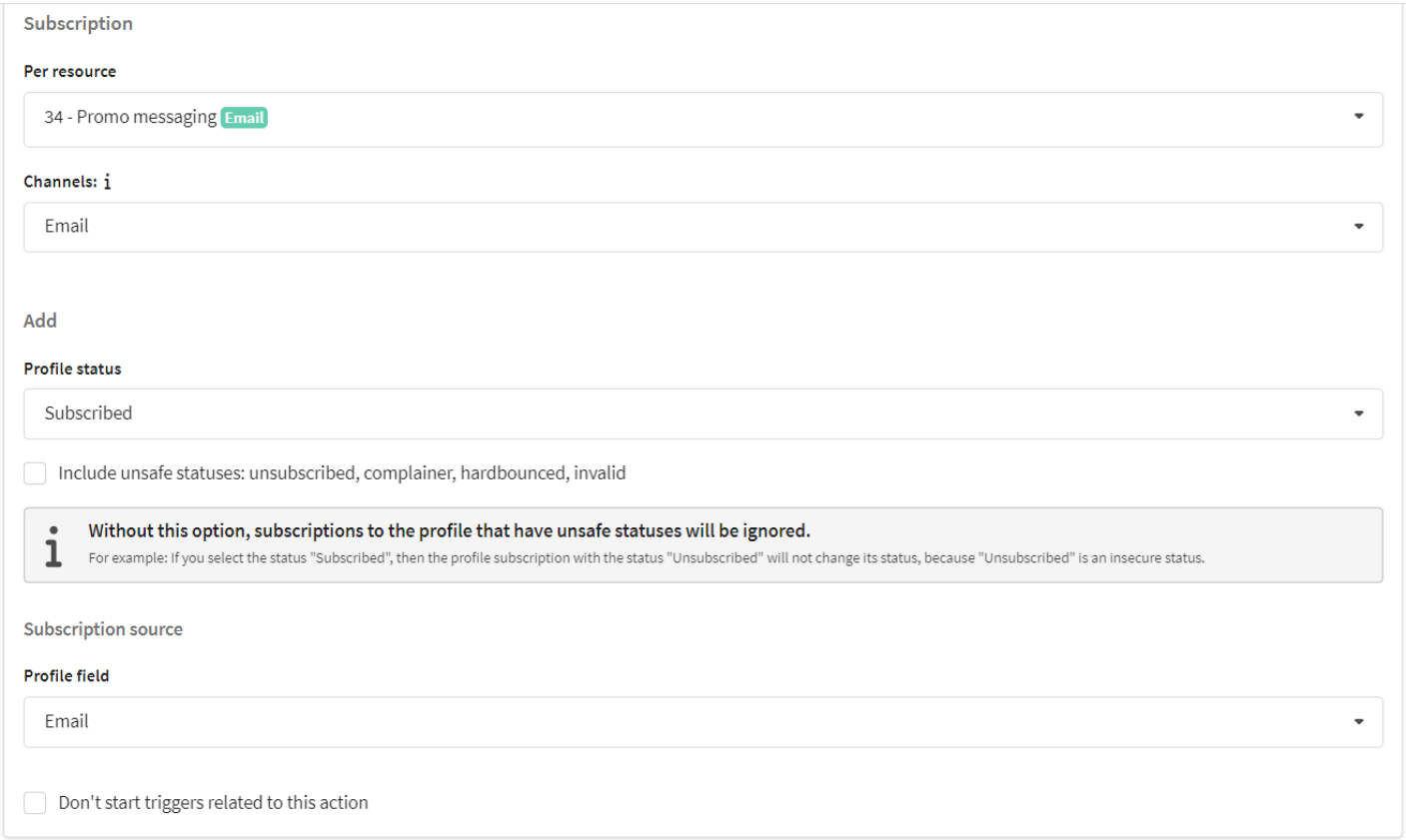
So, we've just configured the profiles database and subscription resource. Now it's time to import customer data into the platform. For each client, Altcraft Marketing creates a single digital profile that includes personal data, history of actions and subscriptions to your resources.

You can upload customer data manually: one at a time, in a list, or from a table file. Different import methods are described in this article.

You can also set up automatic import of client profiles:

- **on schedule from your SQL database**
- **through the use of the integration with platform API**

When you import data, sign clients to the previously prepared resource. Select Email as the communication channel and specify the column with subscribers' addresses:



TIP

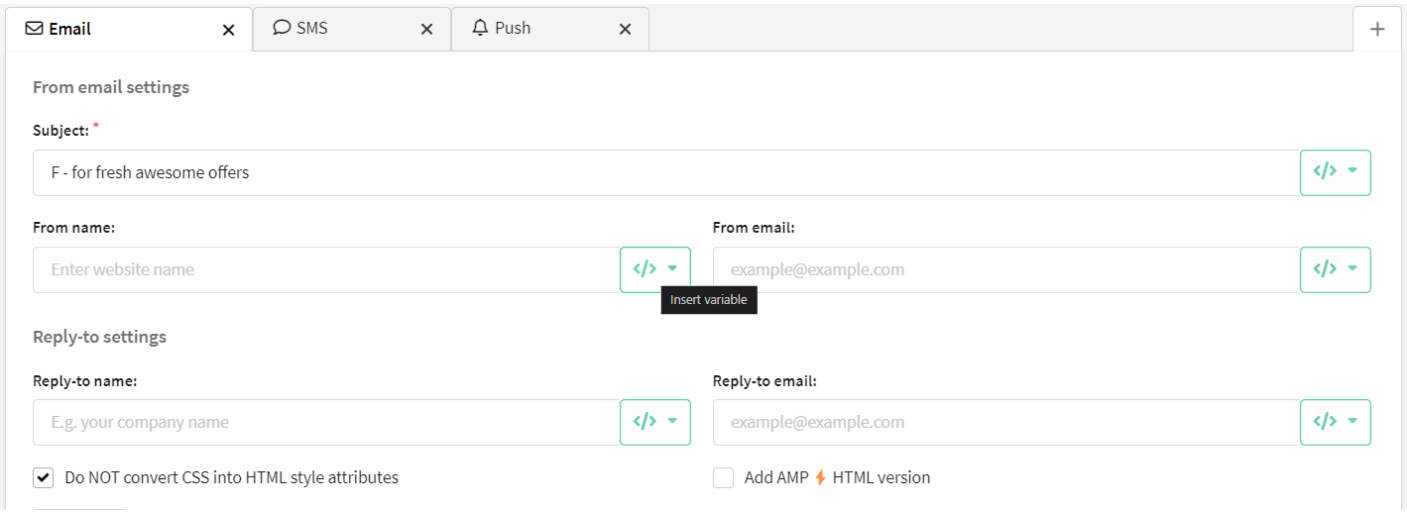
If you already have a profiles database, add them a subscription to a new resource. There are two options:

- **if customers have subscriptions to another resource with up-to-date email addresses — use bulk profile update**
- **if you need to upload to the platform email addresses for subscription, use the profile import in the "update only" mode**

Step 3: Prepare content for submission

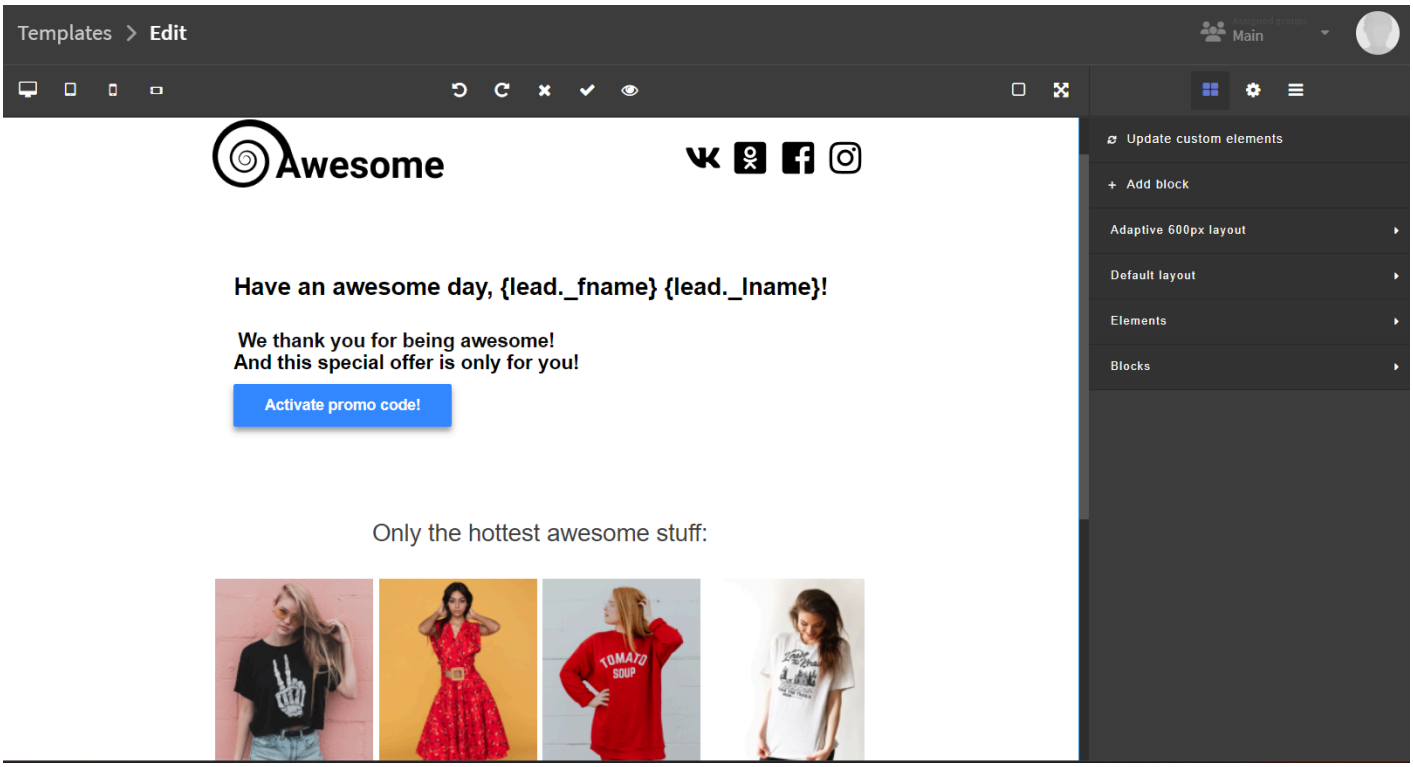
The content for your Email mailings is created in Message Templates. The templates contain content for all available channels. When you create a new template, select the Email channel or add it for an existing one.

For the Email version of the template, you can set a Subject — it will be displayed in the correspondence. You can also enter the name and the address of the sender (From name & From email).The settings in the template will override the settings of both the resource and the database.



Templates in Altcraft Marketing offer many options for personalizing content for a specific recipient. Different types of variables, boolean expressions, loops, functions and dynamic content from various external sources — all this can be used to personalize content.

Use the HTML code editor or the Visual editor to create email templates:

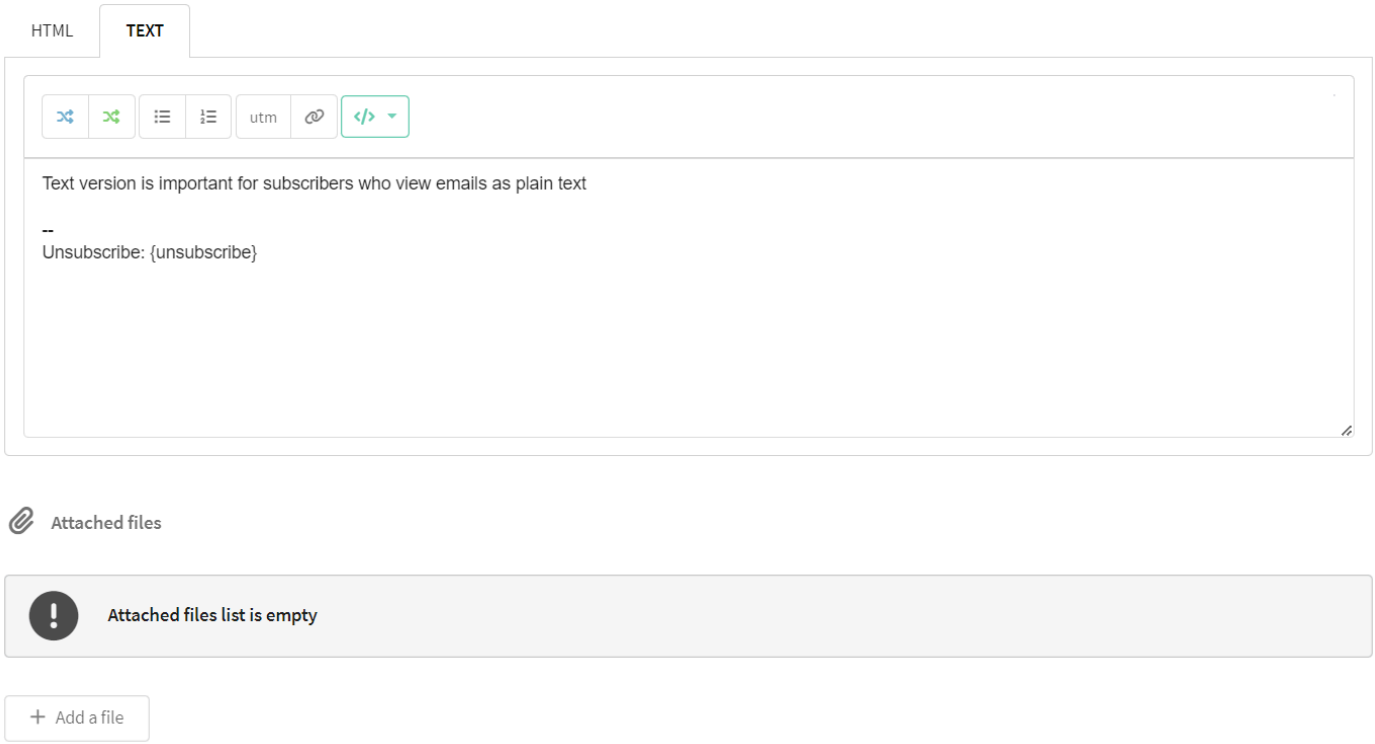


Some of your subscribers may use a mail client in which it is not available to upload images, for example, according to corporate policy. For such cases, the template has a Text version.

CAUTION

The text version is sent along with the HTML version of the letter. If it is not filled out — or does not match HTML — it could have a slightly negative impact on the sender's reputation.

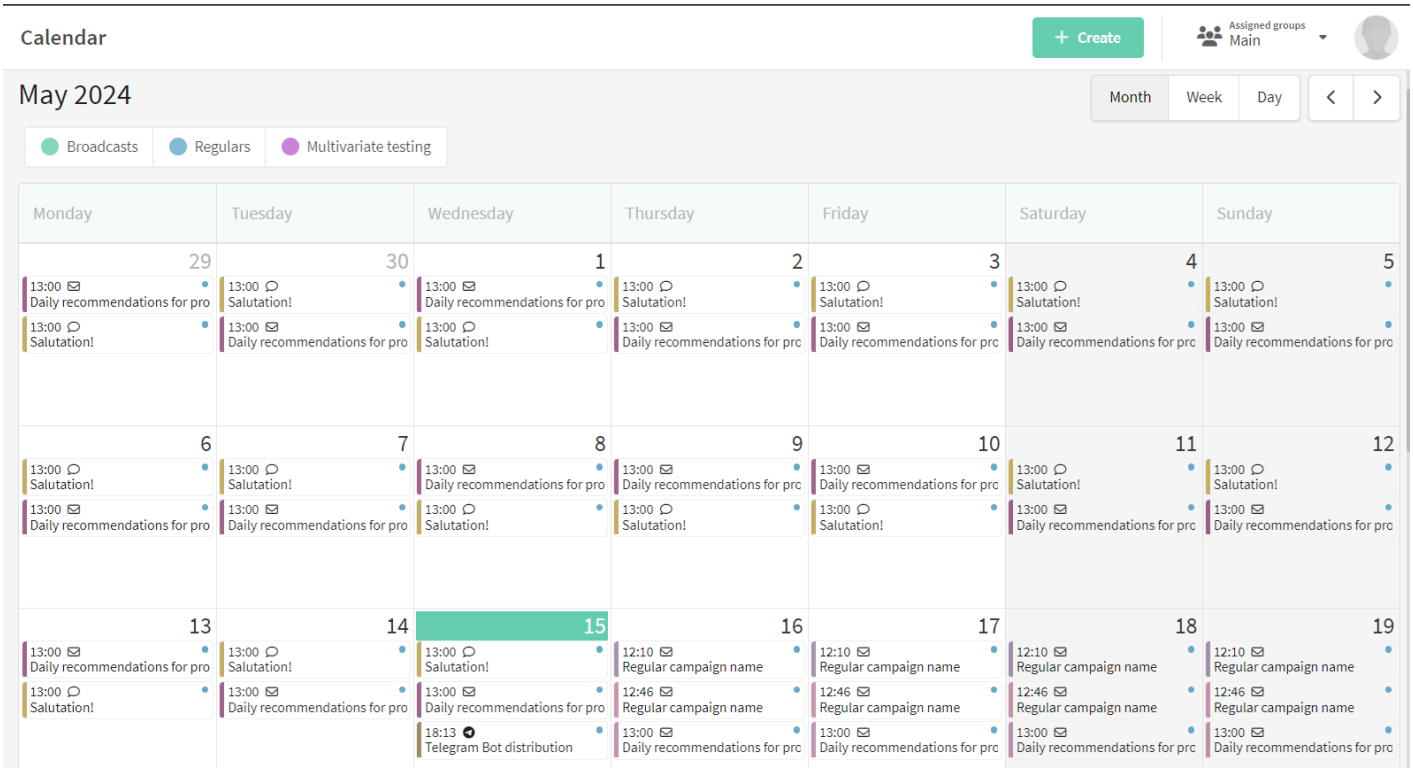
So it's better to have the text version. This way, the indicators of confidence in the mailing will significantly increase.



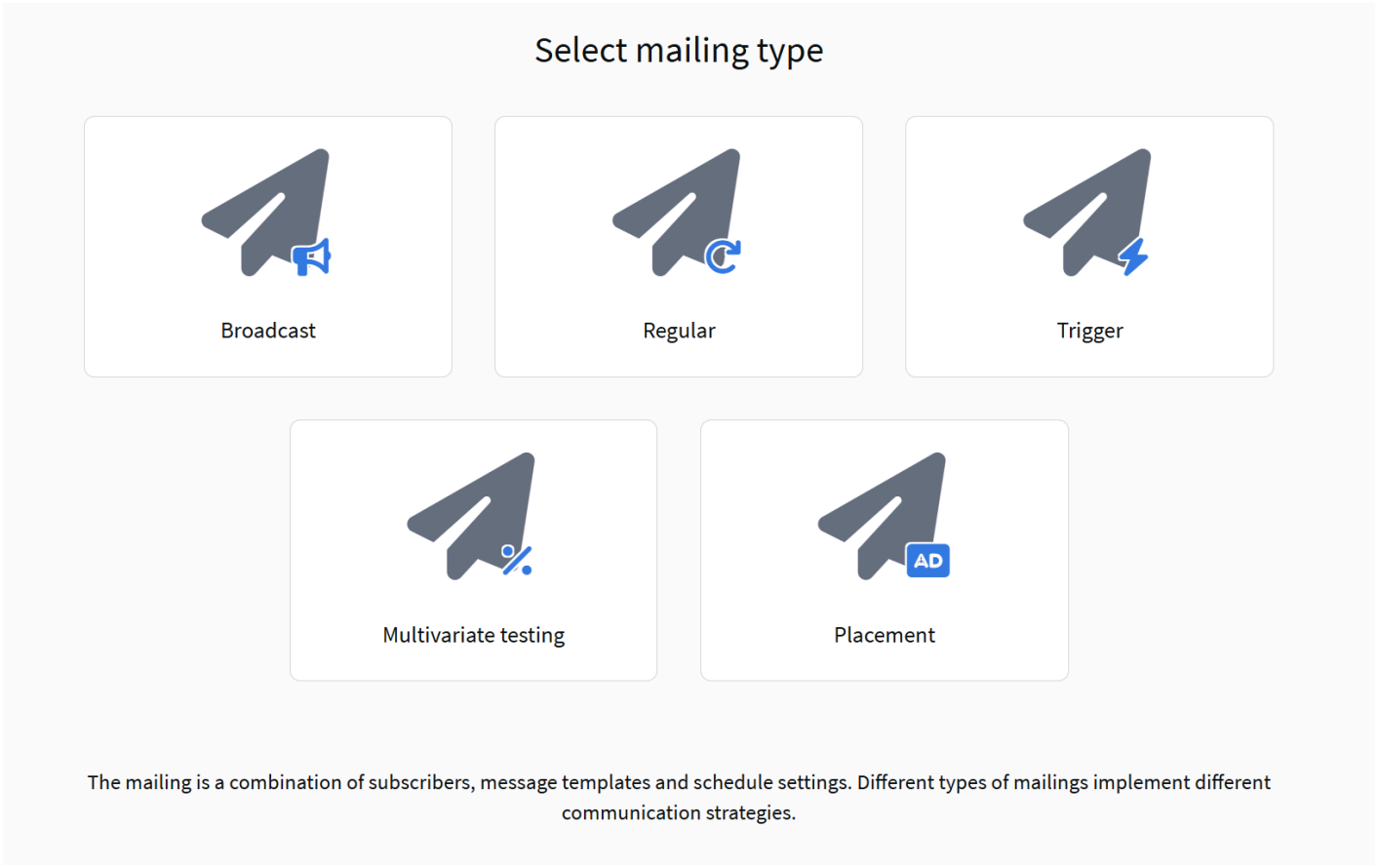
Below the editor of the text version is a form you can use to attach files in any format to the message.

Step 4: Select the type of email mailing

	If you want to send an Email message, you need to use such a tool as a [Mailing**]. Do you want to send out a **one-time mailing** to all profiles of the database or to the audience of the specific segment? Choose **Broadcasts**. If the mailing is periodic, your choice is a **Regular** mailing. In the main menu select **Mailing** and open **Calendar**. This is where you'll find all scheduled mailings.**



Decide on a date and create a mailing of the required type:



TIP

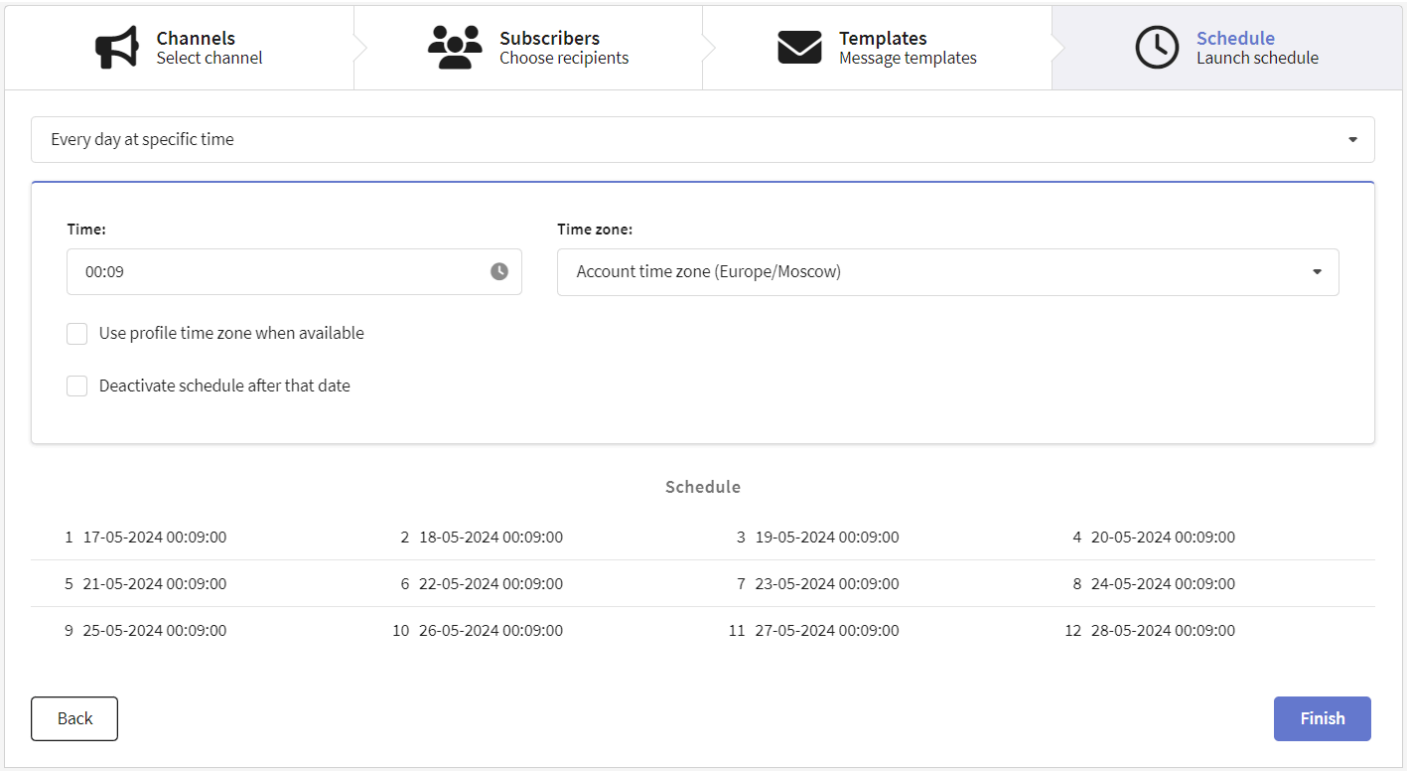
You can find details about how to set up Broadcast on this page.

TIP

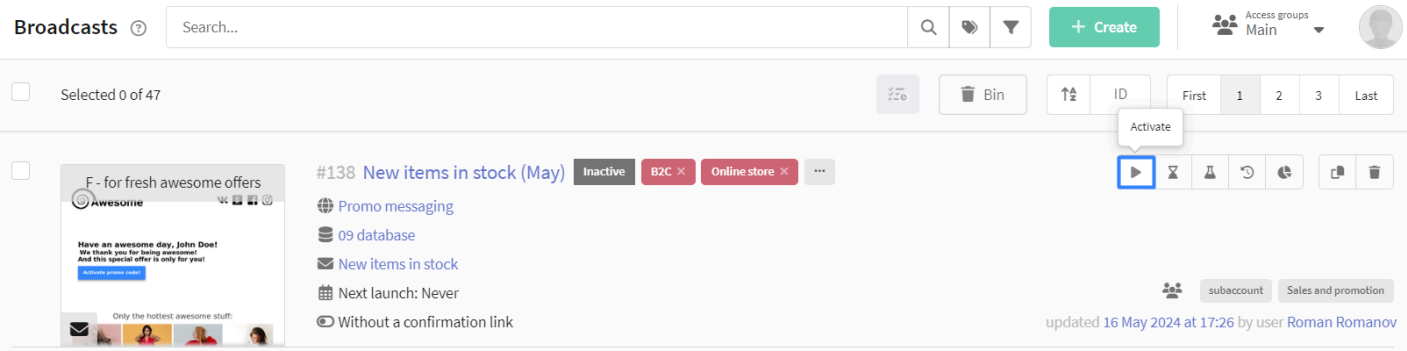
You can find details about how to set up Regular mailing on this page.

Once you're done with the general mailing settings, you need to:

1. **Select Email as the communication channel;**
2. **Select the subscription resource you've just configured and the audience of recipients;**
3. **Add a message template to your message;**
4. **Set a schedule. For example, for a broadcast the option "On specific date at specific time" is available**



Once the mailing is set up, save it. Later you can find it in the list of mailings of the corresponding type. There is a quick action menu on the right side of each mailing. Use it to [test the mailing] — and if everything is successful, activate it:



CAUTION

The active mailing will start sending emails as soon as it will be time to launch as scheduled. Messages are sent to the sender. You can also open the sender queue from the quick action menu.

The sending speed depends on [the reputation of the sender's domain].

Step 5: Assess the effectiveness of the email mailing

Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Field Name	Description
Mailing started	**The start time of the email mailing execution**
Limit not set	**Indicates whether a sending limit is set**
Audience source	**The source of the subscribers:** **List — Database** **Segment — Segment**

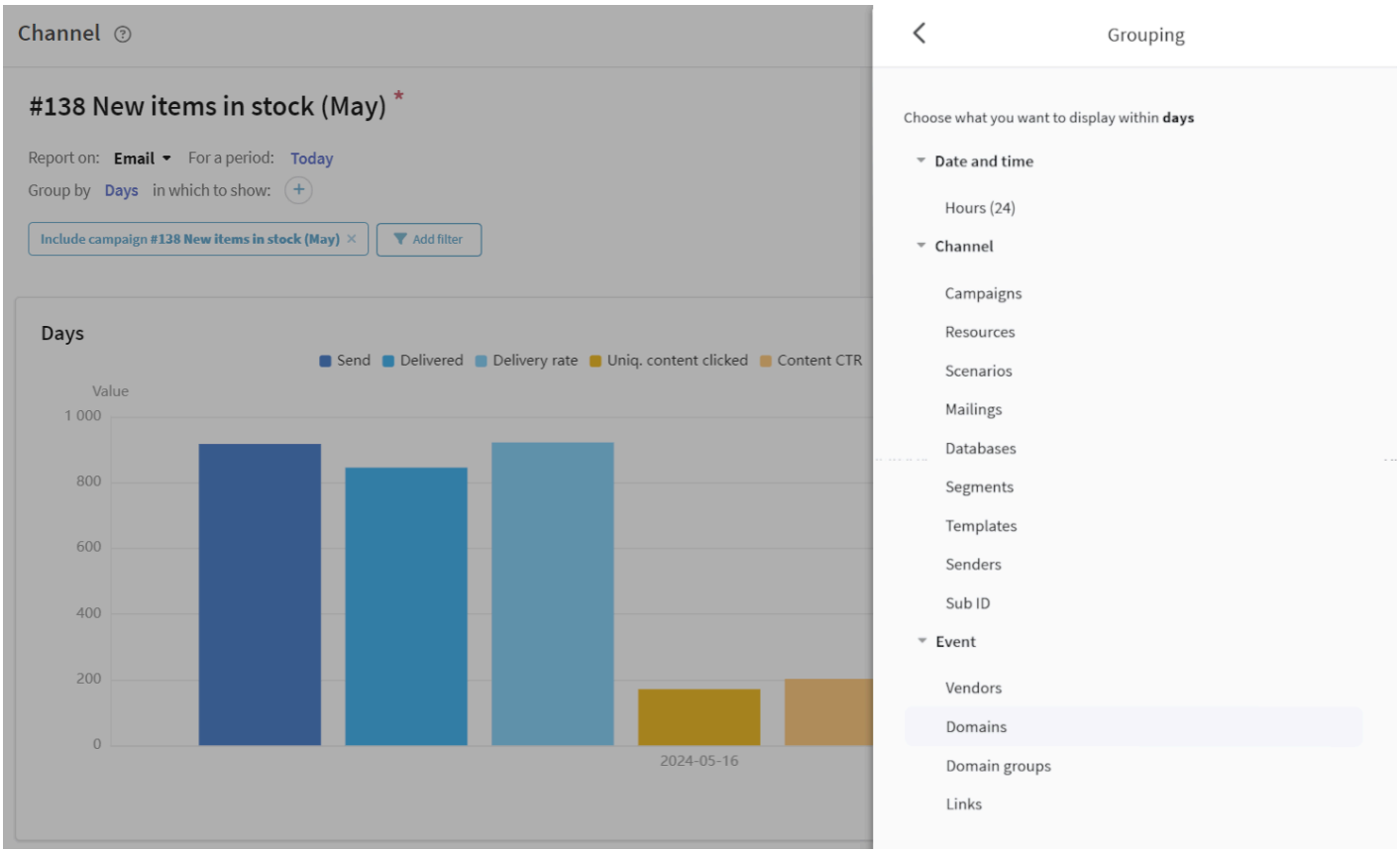
Field Name	Description
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent retrieving segment data from the database (in ms)**
Number of subscribers received from the source	**Total number of profiles found in the selected database**
Number of subscribers found	**Number of profiles in the database subscribed to the selected resource**
Number of subscribers without email	**Profiles that couldn't receive emails (missing/invalid addresses)**
Number of email recipients	**Final number of successful email sends**
Mailing finished	**Successful completion of the mailing execution**
Total time spent	**Total duration from mailing start to completion**

Reports and Analytics

To evaluate the effectiveness of your email mailings then go to the section Analytics in the main menu of the platform.

The channel report provides analytics for one or several mailings, for one or several communication channels. For the convenience of grouping, you can select a preset of indicators for Email:

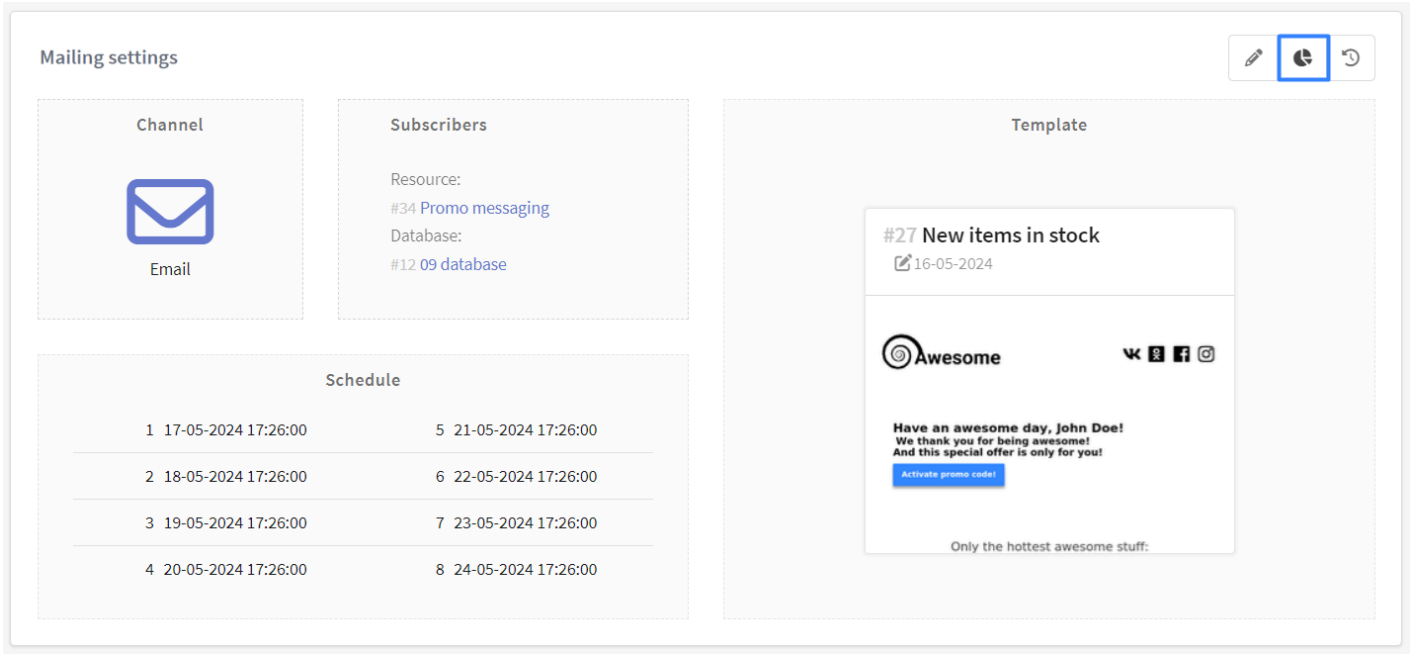
You can group the report by domain group of recipients to analyze the effectiveness of message delivery to email boxes of different providers:



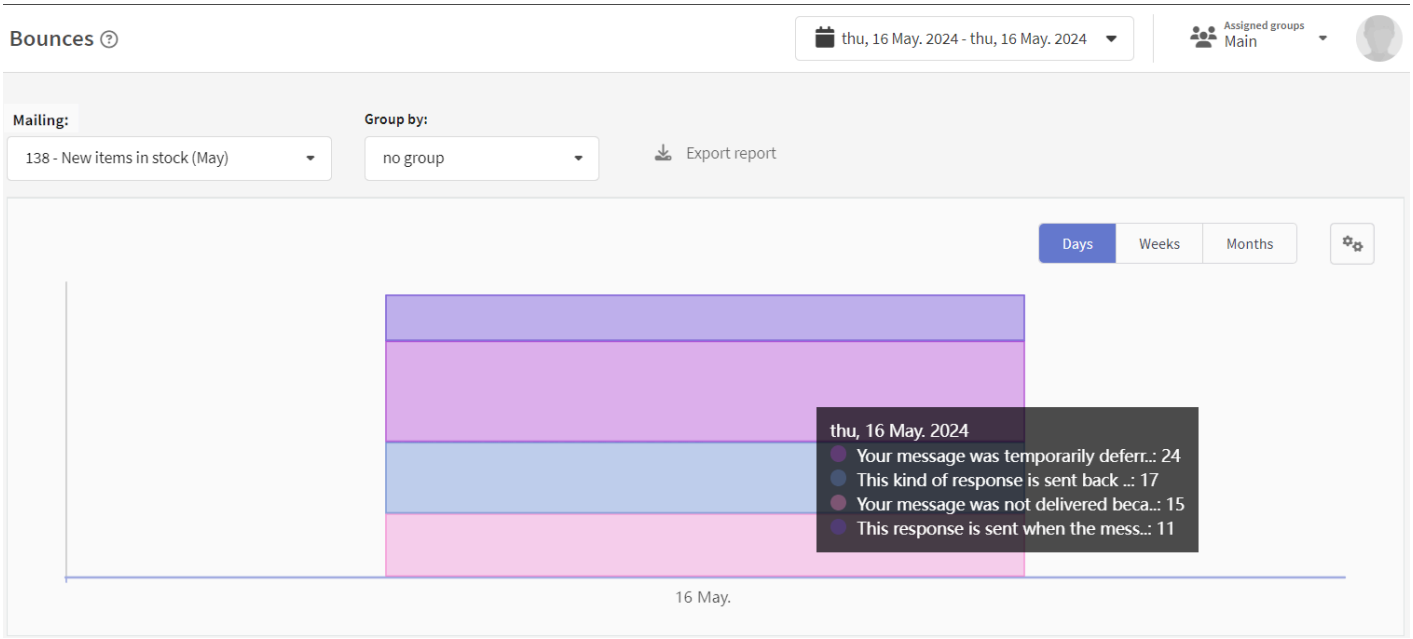
You'll see information on providers in the table below the graph:

Name	Send	Delivered	Delivery rate	Suppressed	Opened	Read	Content clicked	Uniq. content clicked
Total	917	845	92,15 %	583	412	326	855	171
2024-05-16	917	845	92,15 %	583	412	326	855	171

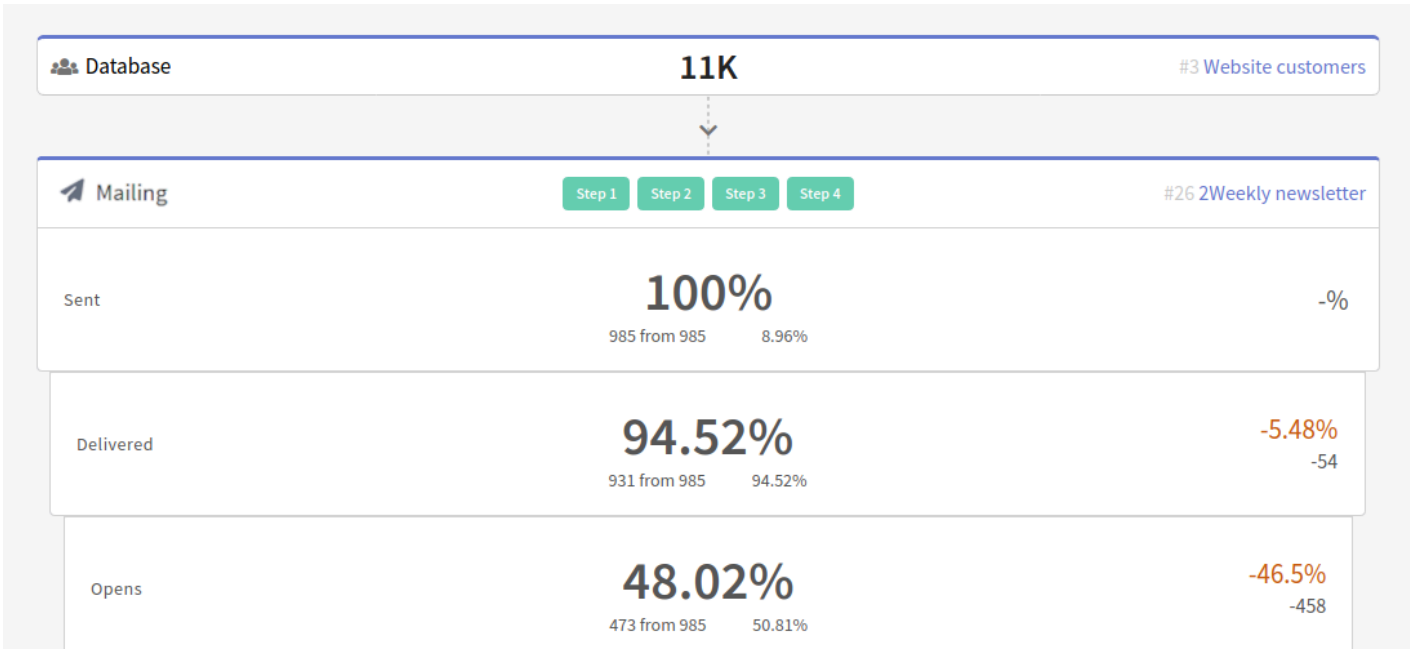
You can also go to the channel report for a specific mailing by following the link in the mailing editor:



To analyze the problems of email message delivery, use Bounces and Undeliveries reports:



**Actions in the Email mailing or in the template can turn into steps in the [interactive conversion funnel](): **



Another kind of report that is available for email messages is [Click map](). Click map is compiled for templates that includes an Email version. This template must be sent at least once.

When you make changes to the template and send again, a new version of the click map is created. It allows you to evaluate the dynamics of subscriber behavior:

Click map

Assigned groups
Main

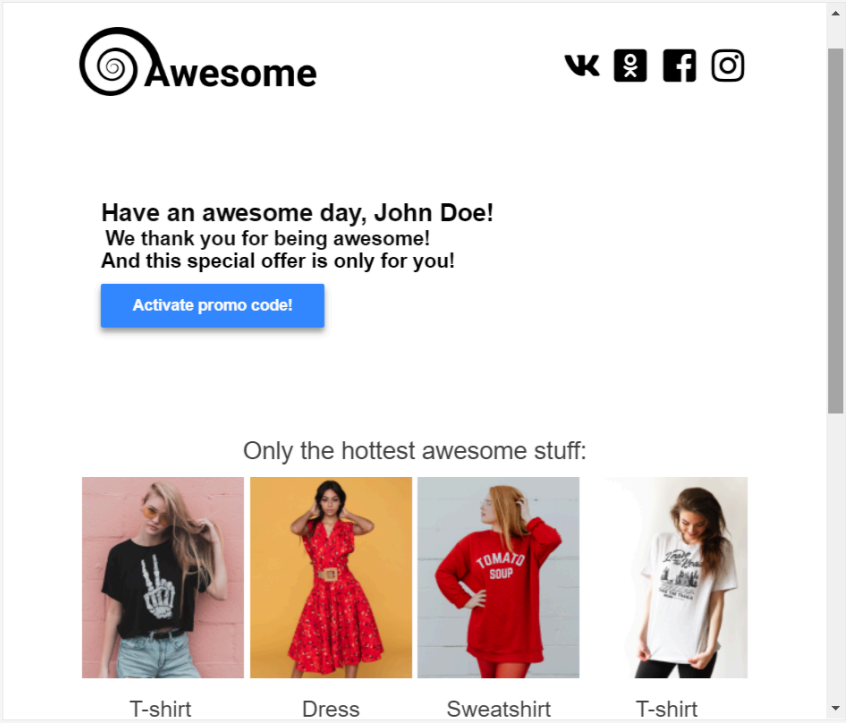
Choose message template

27 - New items in stock

Date range

16-05-2024 17:29:06 - now

Export report



412 OPENS

862 CLICKS

7 UNSUBSCRIBES

Name	Clicks	%	CTR
loyalty_program	171	19.84%	41.5%
buy_1	171	19.84%	41.5%
buy_2	171	19.84%	41.5%
buy_3	171	19.84%	41.5%
buy_4	171	19.84%	41.5%
unsubscribe	7	0.81%	1.7%
link8	0	0%	0%
link9	0	0%	0%

Push Notifications

Push Notifications — a powerful tool for instant audience engagement. Short personalized messages sent directly to the screens of smartphones or computers enable effective reminders about promotions, new products, or important events, even if the user is not active in the app or on the website.

Mobile Push

Mobile Push Notifications are short messages that appear on a mobile device's screen, even if the app is inactive. Thanks to personalization and targeting capabilities, these notifications help retain users' attention, increase engagement, and stimulate repeated interaction with the app.

Web Push

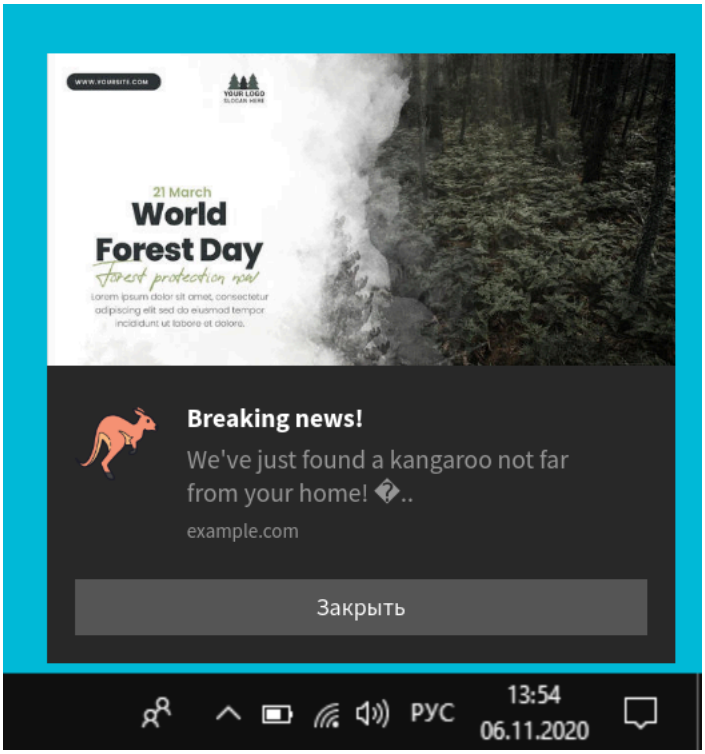
Web Push Notifications are messages that appear on the screen of a computer or mobile device via a web browser, even if the website is not open. They operate using browser technologies (such as Chrome, Safari, Firefox) and do not require an app to be installed. The main difference from Mobile Push is that Web Push does not depend on having an app. Web Push notifications can appear on both desktop and mobile browsers, making them a universal tool for engaging with your audience.

Altcraft Platform also supports sending push notifications from PWA applications.

Rich Push

The Altcraft platform supports the creation of Rich Push Notifications. They offer unique advantages by utilizing visually rich content. This not only informs the user but also engages them in interactive communication, increasing conversion and maintaining their attention. You can add to a Rich Push notification:

- **an image (banner), video, or audio;**
- **interactive elements (buttons);**
- **external links.**



To create a template for such notifications, use the Rich Push editor.

Mobile Push: Setup and Connection

TIP

The setup of browser push notifications is described in a separate article.

	Mobile Push are short messages that appear on the screen of a mobile device, even if the application is inactive. Unlike Web Push, **Mobile Push notifications are delivered not to a browser but to a mobile application.**

A push notification can contain the following data:

- **Title — similar to the subject of an email message;**
- **Body — any text for the message;**
- **Icon — an image within the notification;**
- **Redirect link — directs the user when clicking on the push notification.**

The following providers are used to send these notifications:

Service	Supported OS
Google Firebase Cloud Messaging	**Android, iOS**
Apple Push Notification Service	**iOS**
Yandex.AppMetrica	**Android, iOS**
Huawei Mobile Services	**Android, iOS**
RuStore	**Android**

How to Set Up and Connect Mobile Push Notifications

Mobile push: settings & implementation

TIP

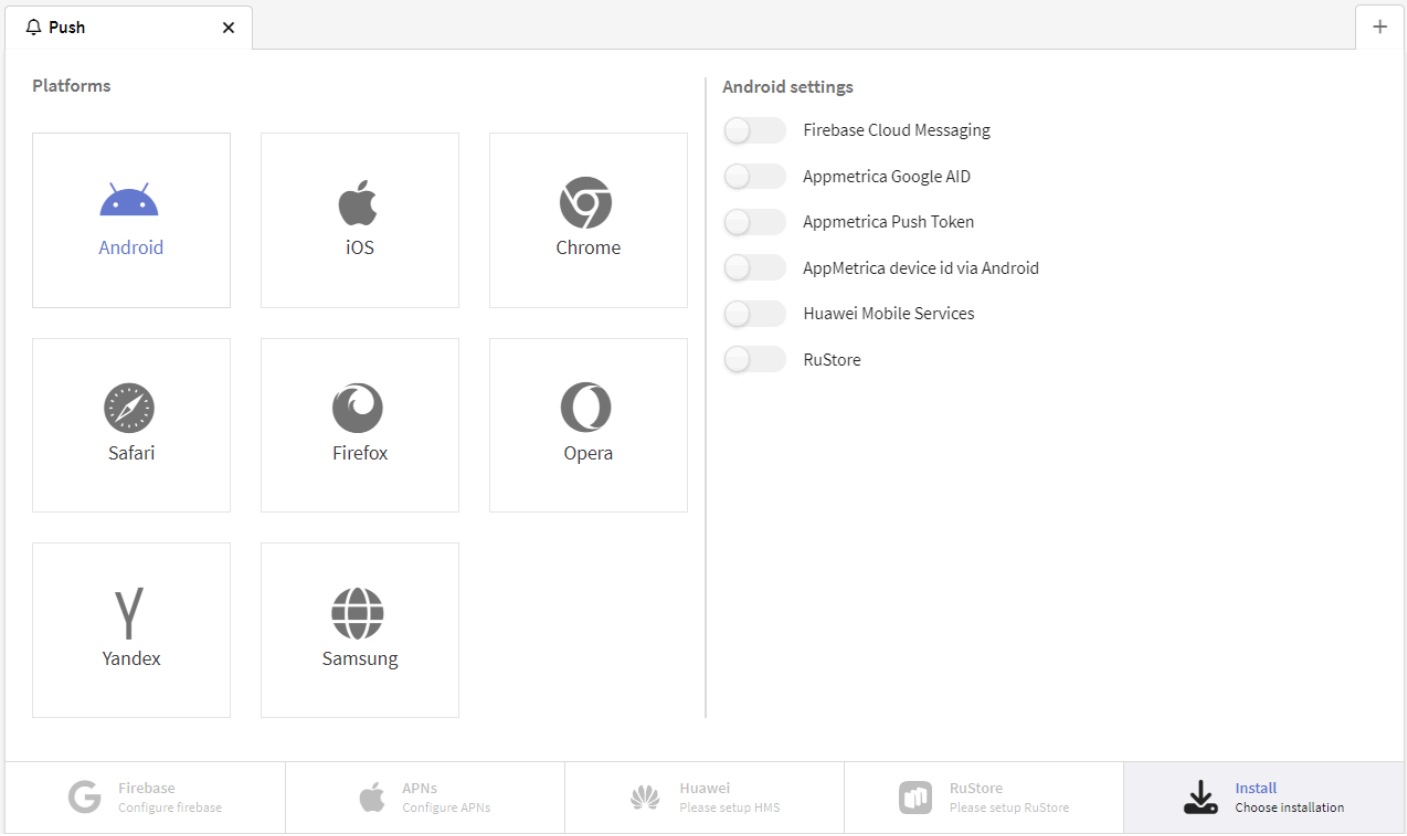
If you want to set up web browser push, see this guide!

Description

Mobile push notifications are displayed in your customers' mobile device message center, in the upper part of the screen most commonly. They are allowed by default on application installation — thus you can start engaging your customers right away and increase your app conversion.

In Altcraft Platform, there are 5 mobile push options:

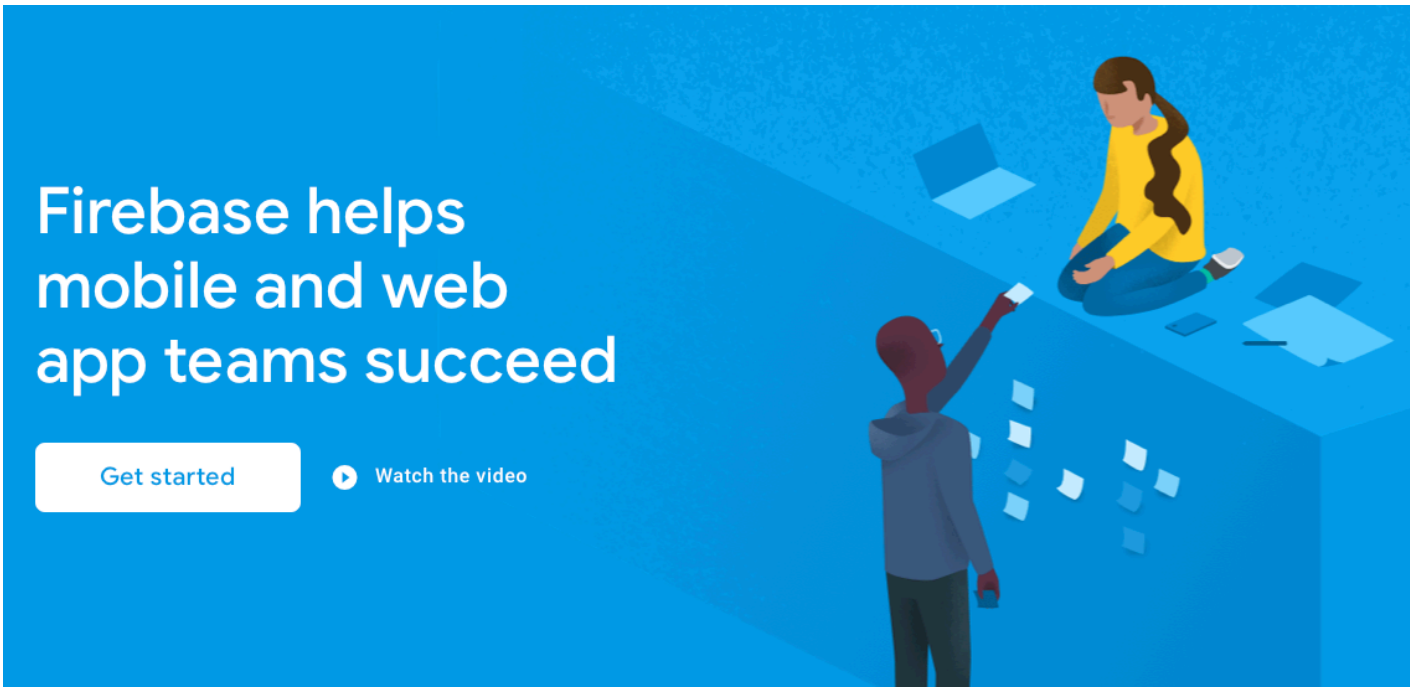
- **Google Firebase Cloud Messaging — for both Android & iOS.**
- **Apple Push Notification Service — iOS apps only.**
- **Yandex.AppMetrica — for both Android & iOS.**
- **Huawei Mobile Services — for both Android & iOS.**
- **RuStore — Android apps only.**



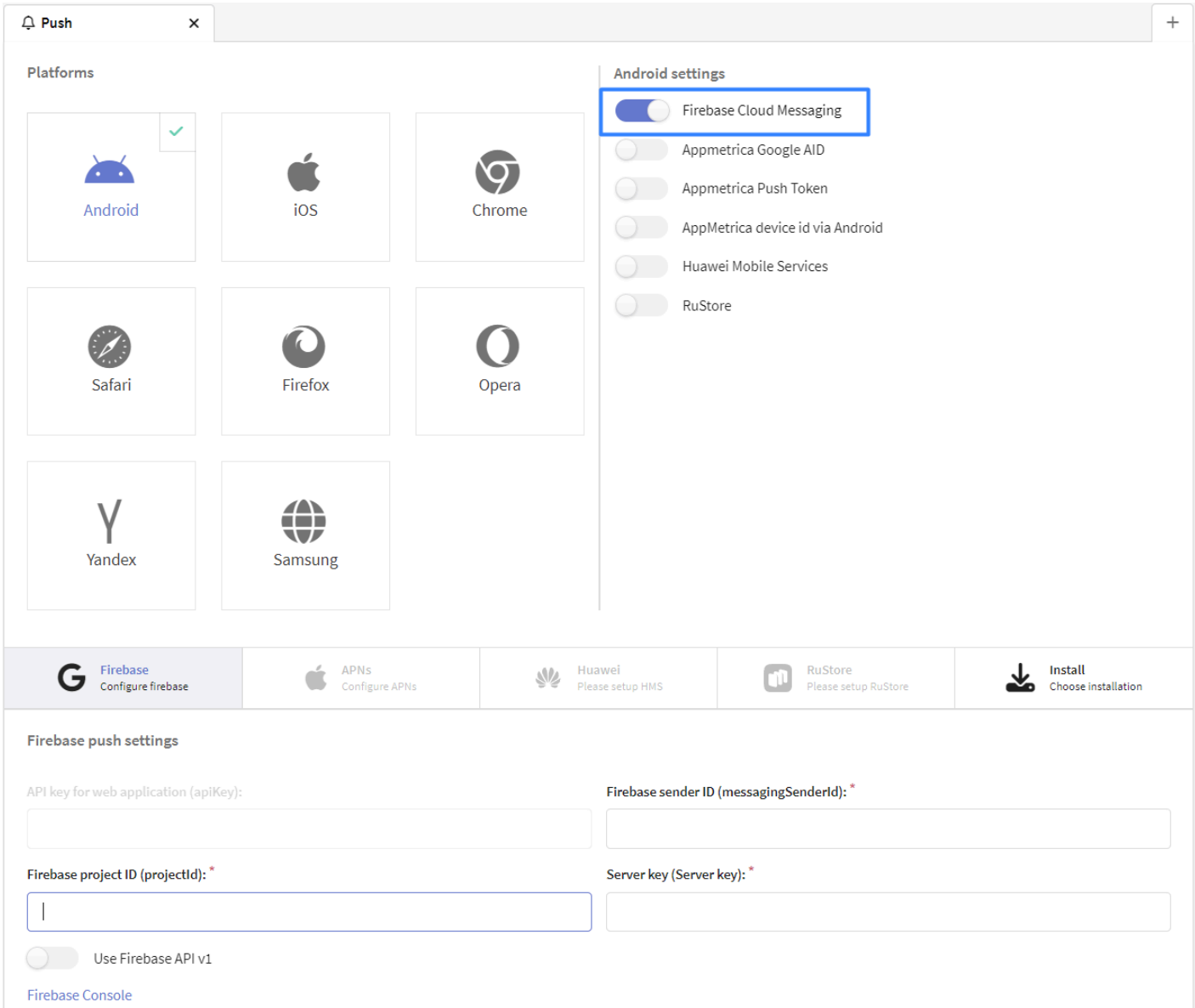
Configuring push for Android and iOS

Firebase Cloud messaging

	Firebase cloud messaging is used as a cross-platform and a cross-browser solution to activate push notifications for Mozilla Firefox, Opera, Google Chrome and both iOS and Android applications. You can configure both platforms within one Resource. The first step is to create a project at Firebase Console.



In Altcraft Marketing select or create a [Resource], where you will activate Firebase Cloud Messaging for supported mobile platforms. On activation Firebase section becomes active in the lower part of the form:

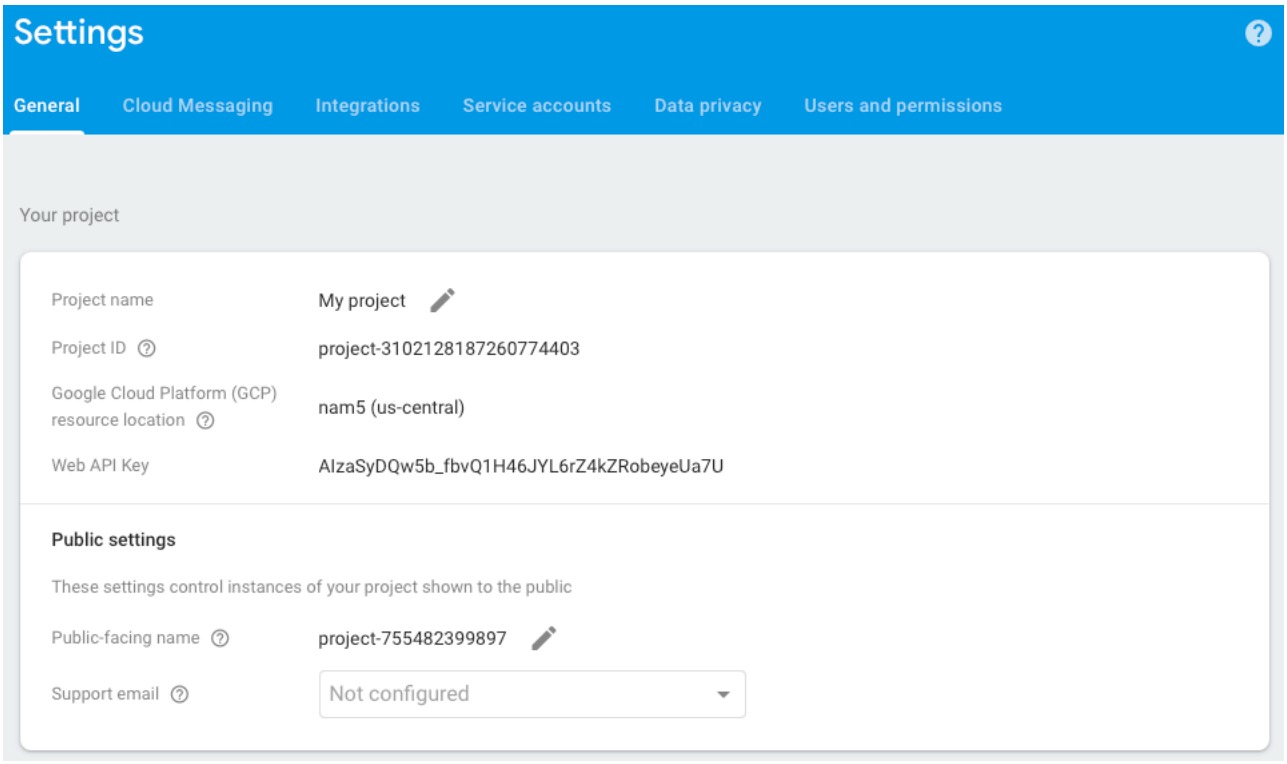


You would need the following information, retrieved from your Firebase project:

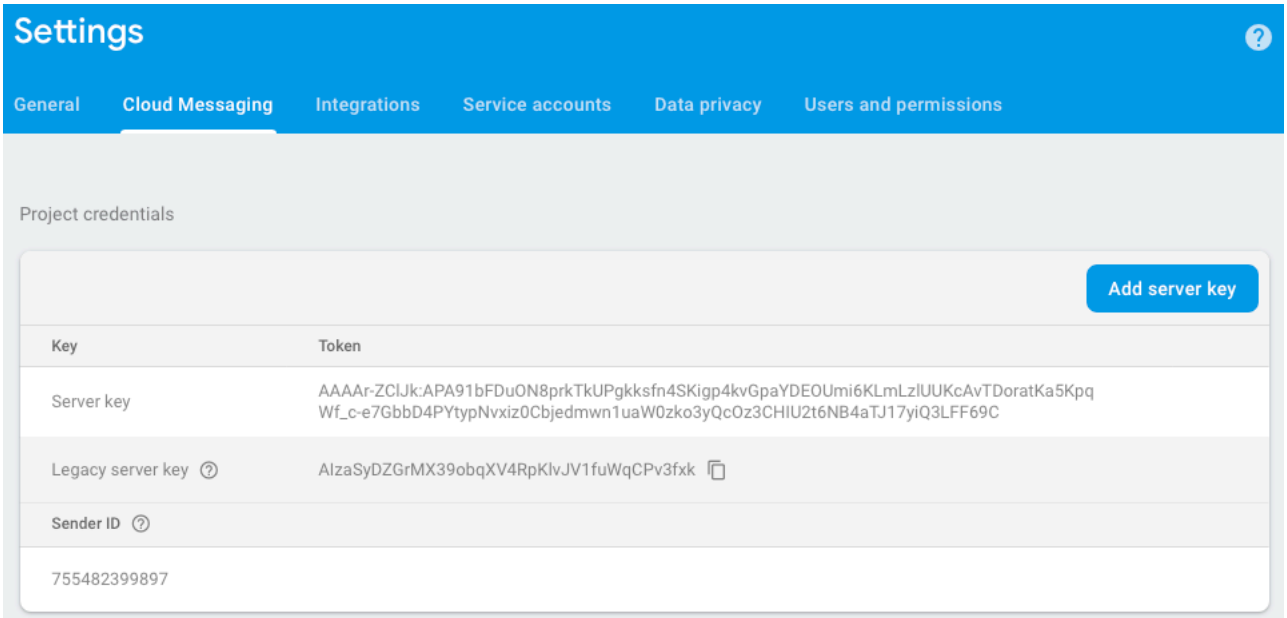
- **Firebase API key (Web API key)**

- **Firebase project ID**
- **Server key**
- **Firebase sender ID**

Project ID and Web API key can be found in Project settings in General tab.



Server key and Sender ID are found in the Cloud messaging tab also in Project settings.



Apple Push Notification service

	Apple Push Notification service allows you to send push notifications from mobile apps to iOS devices. You need an Apple Developer account and a macOS device to set up the service.

Go [to the Apple Developer website](#) to **Account ** — Certificates, Identifiers & Profiles — Identifiers. Create a new identifier. Select the App IDs option, then App. Enter a title in the Description field, specify the Bundle ID of the application in the appropriate field, select the Explicit option, then select Push Notifications from the list below and register the ID:

Developer

Certificates, Identifiers & Profiles

‹ All Identifiers

Register an App ID

Back Continue

Platform
iOS, iPadOS, macOS, tvOS, watchOS, visionOS

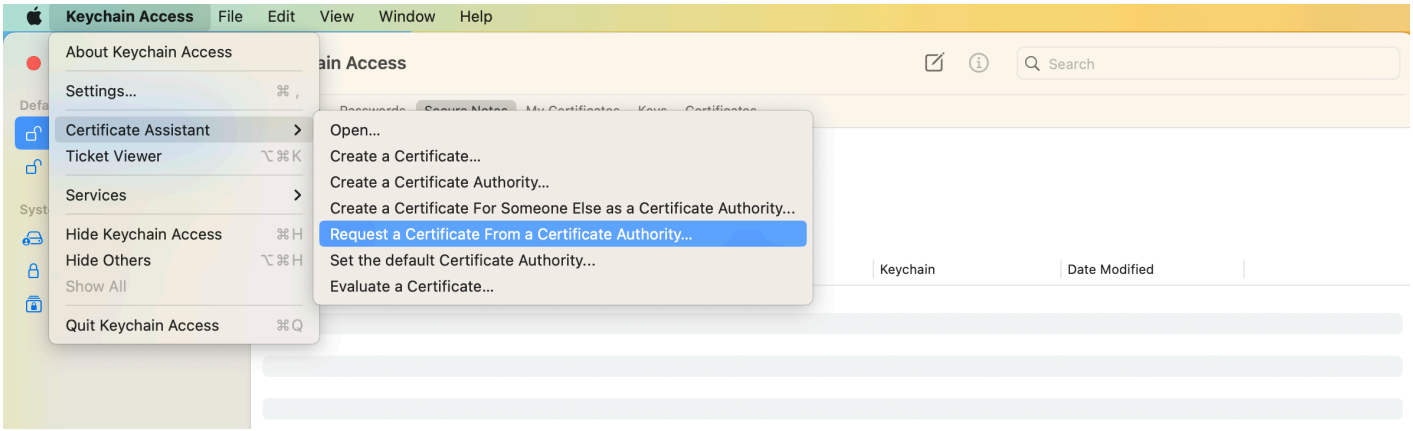
App ID Prefix

Description
Altcraft Apple notifications
You cannot use special characters such as @, &, *, "

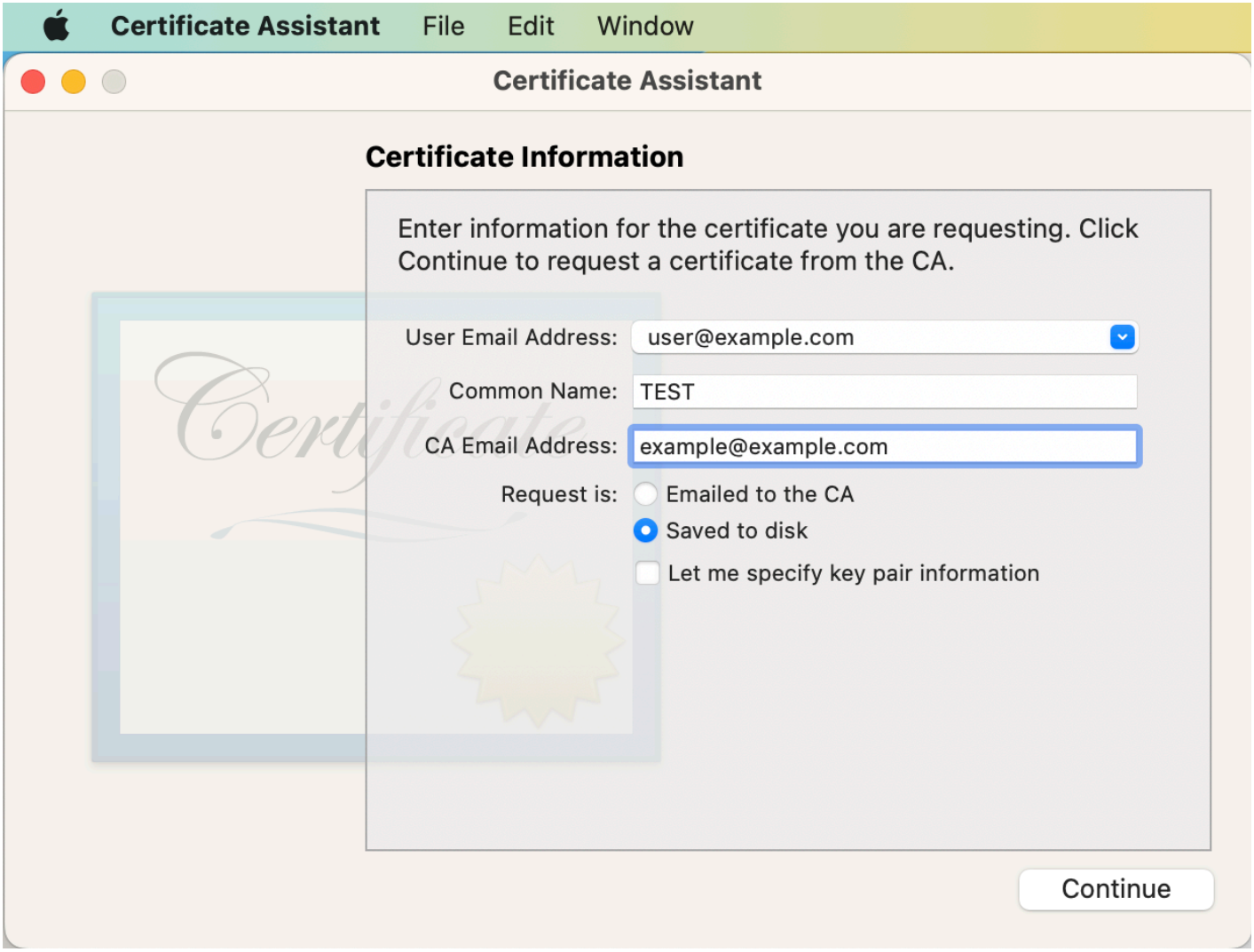
Bundle ID Explicit Wildcard
bundle-id
We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

- Messages Collaboration
- Multipath
- Network Extensions
- NFC Tag Reading
- On Demand Install Capable for App Clip Extensions
- Personal VPN
- Push Notifications
  - Broadcast Capability

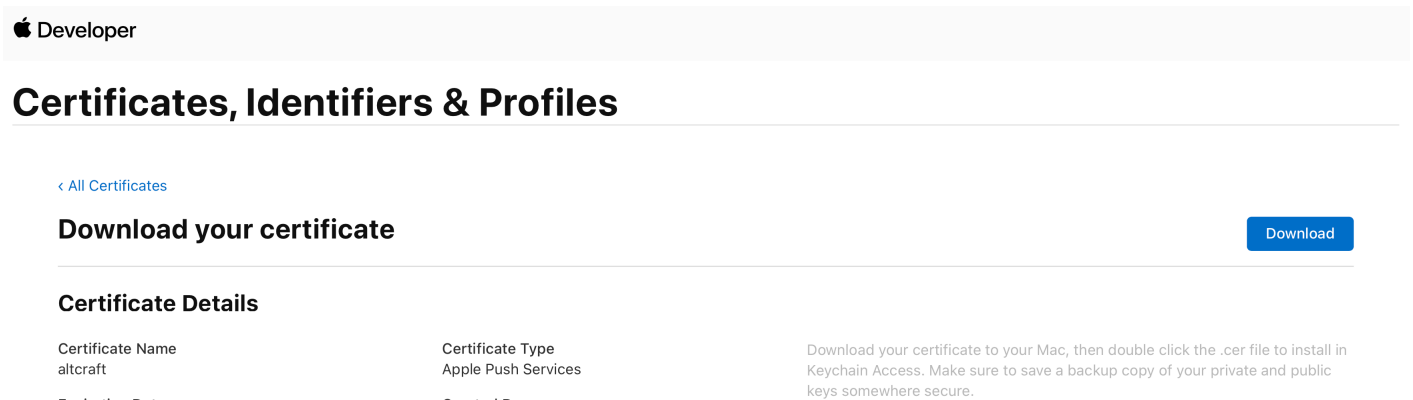
Open the Keychain application on your device. Go to Default Keychains — Login, select **Certificate Assistant ** — Request a Certificate From a Certificate Authority from the menu:



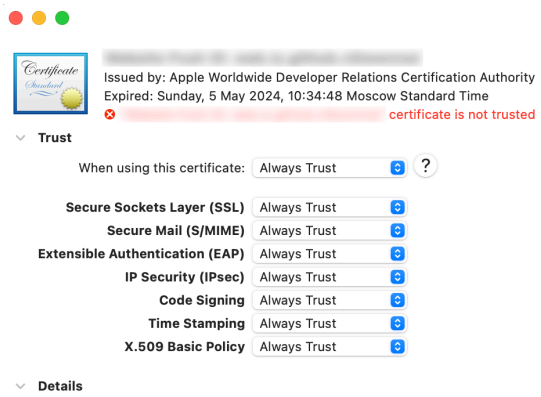
In the window that opens, fill in the required fields and select Saved to disk. Save the certificate on the device:



Go back to **Certificates, Identifiers & Profiles ** — **Certificates ** on the website. Create a new certificate. In the Services list, select Apple Push Notification service SSL (Sandbox & Production). In the next item, find your application in the AppID drop-down list. Upload the certificate you created earlier into the appropriate field. Save the resulting Apple certificate to your device:



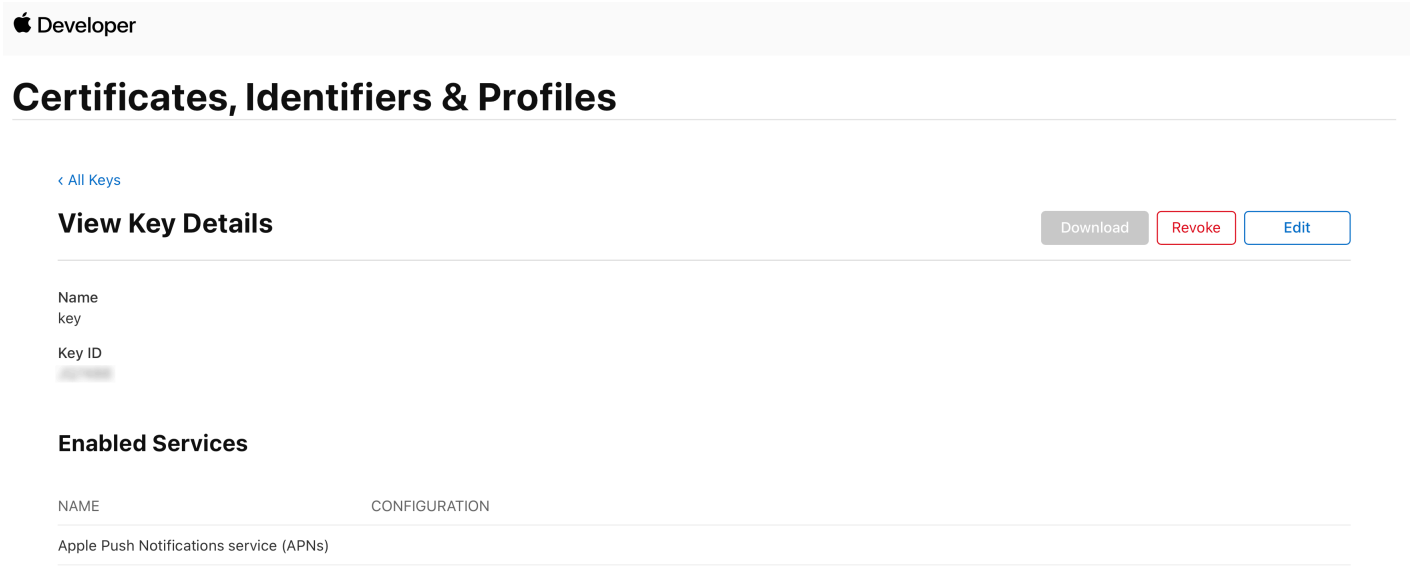
Double-click it to make it appear in the Keychain application. Open it and select Always Trust in the certificate usage options:



The next steps depend on whether you want to use token or certificate authentication.

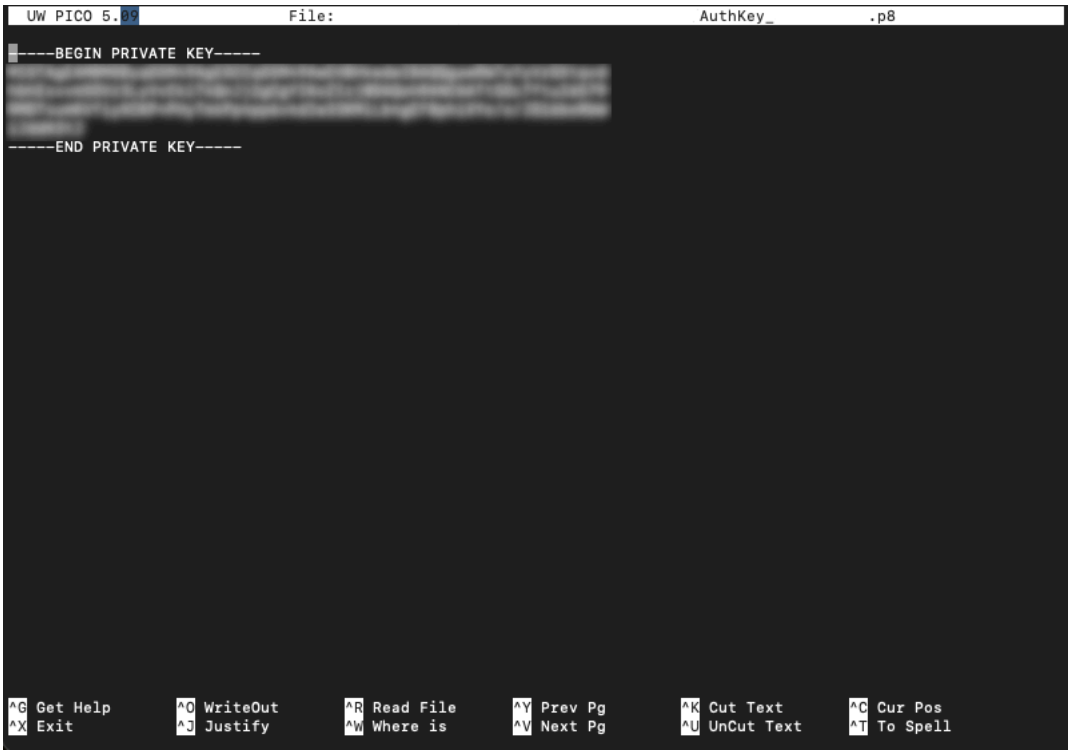
Token authentication

On the Apple Developer site, go to Certificates, Identifiers & Profiles — Keys. Create a new key. Specify the name of the key and its type — APNs, click Continue, then Register. Make a note of Key ID, it will be required to configure the resource on the platform:

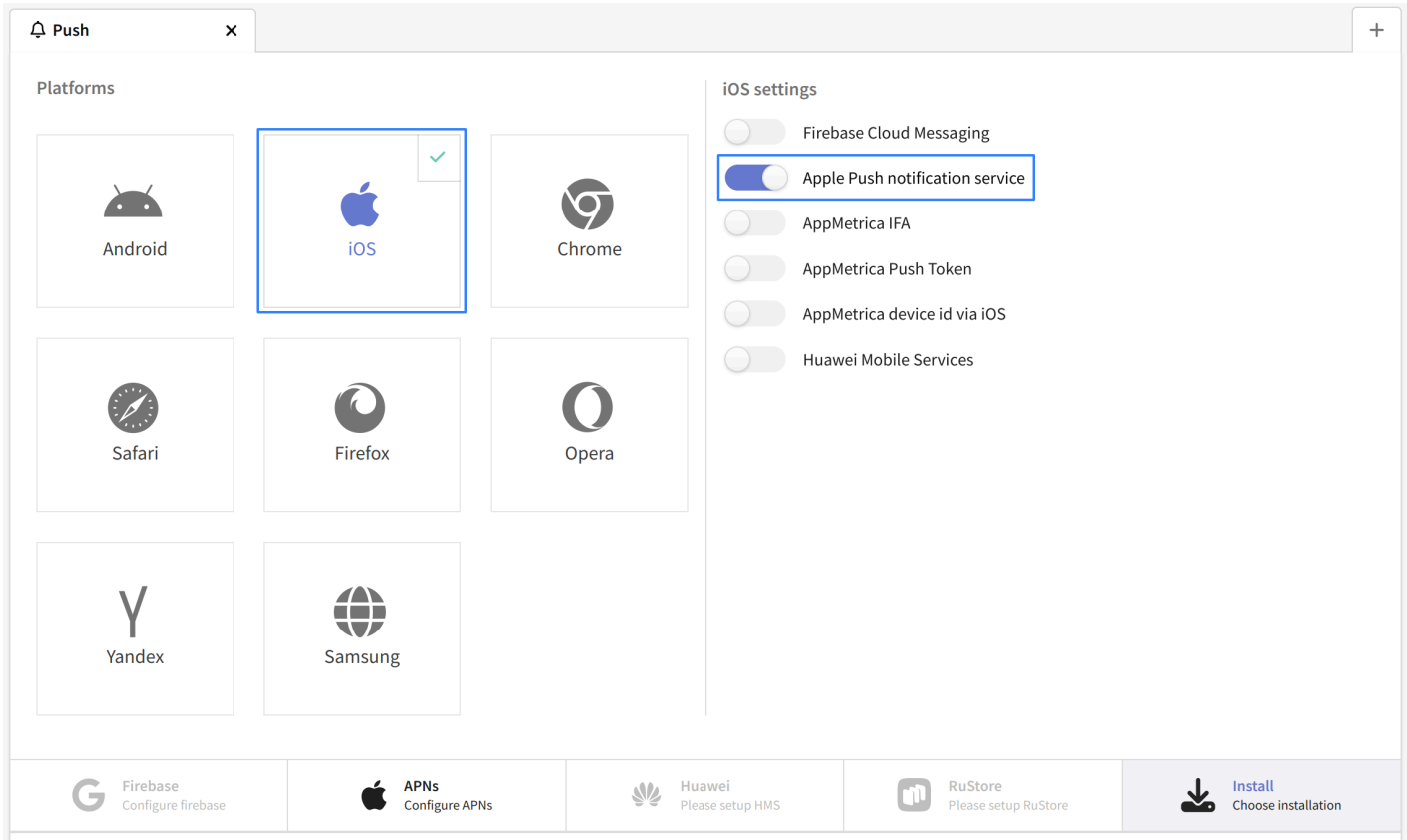


Download the key to your device. You need to open it with any text editor. The signature key is contained inside. You will need to specify it in the appropriate field when configuring the resource.

> **CAUTION**
>
> **The key can only be downloaded one time. If you can't download it now, click Done and come back when you can.**

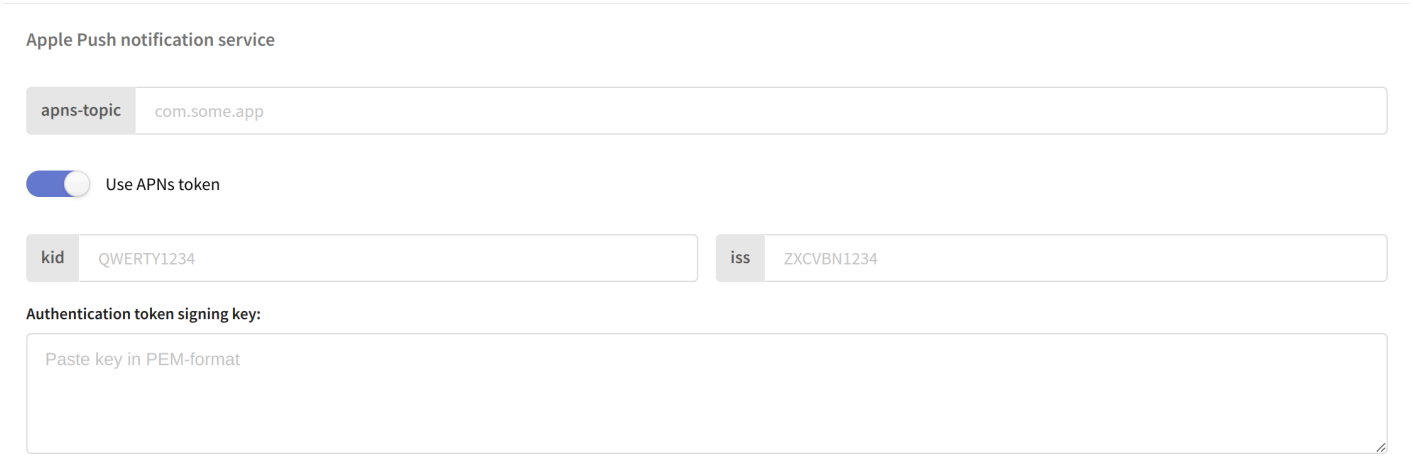


On the Altcraft platform, create a resource for the push channel, select the iOS platform and enable the Apple Push Notification service switch:



Activate the Use APNs token switch in the settings below. Fill in the following fields:

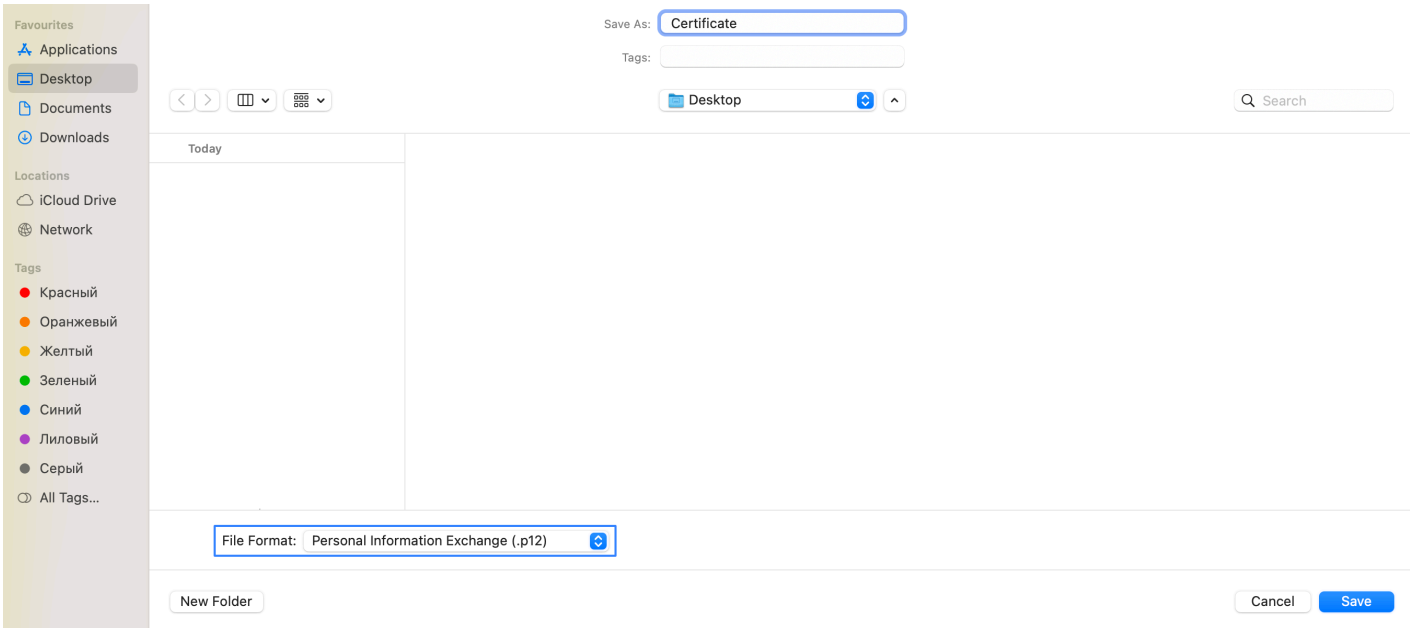
- **apns-topic — Bundle ID of the application.**
- **kid — KeyID value of the key.**
- **iss — Team ID, usually found in the developer account on the Apple Developer site.**
- **Signature Key — The contents of the key file downloaded earlier.**



Save the configured resource using the buttons at the top of the page. The platform is ready to send notifications using APNs.

Certificate authentication

In the Keychain application, right-click the created certificate and export it. Select the .p12 format. If necessary, set a password for the certificate:

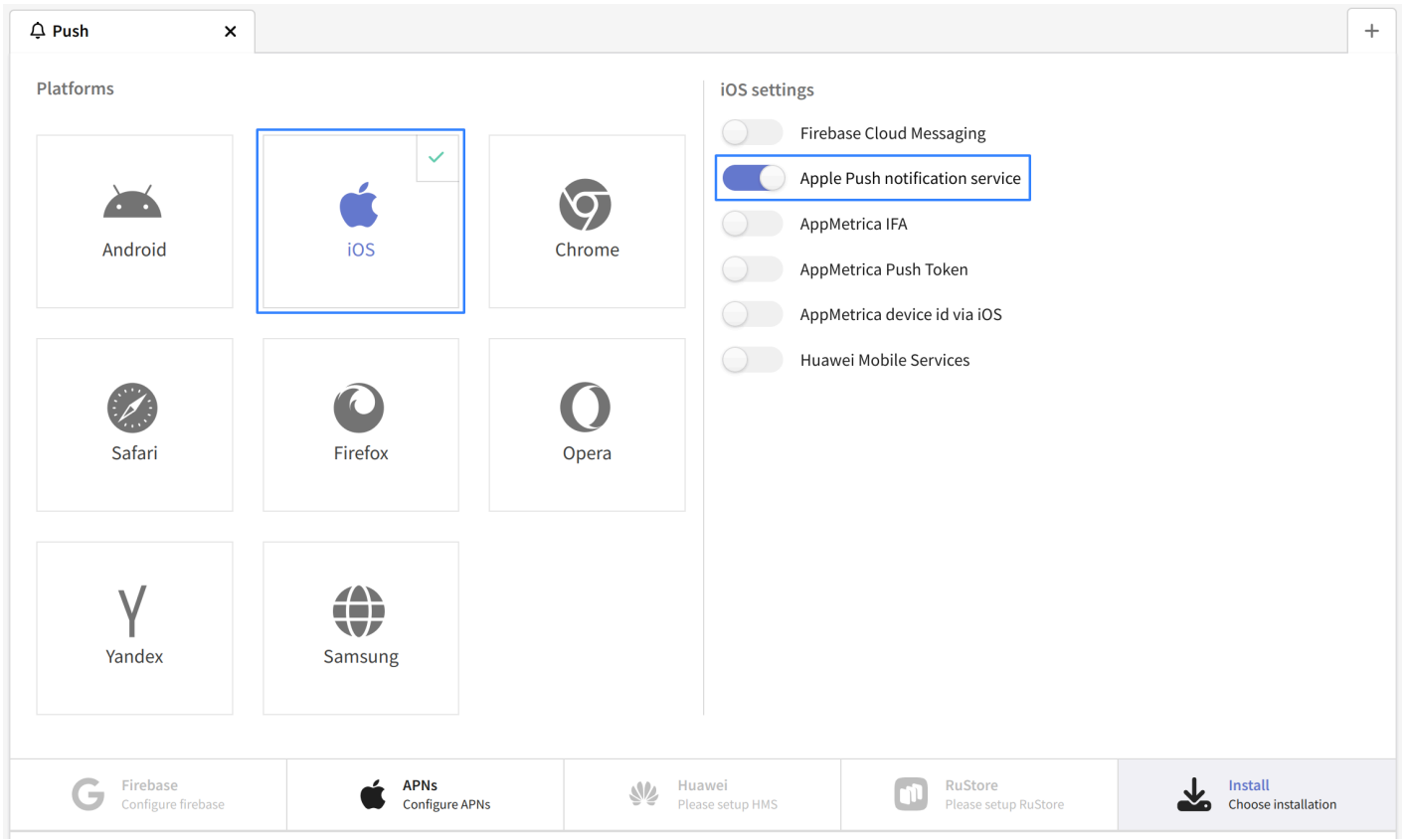


Using terminal navigate to the folder where you exported the certificate. Create a pem key using the command `openssl pkcs12 -in filename.p12 -out filename.key.pem -nocerts -nodes`, where `filename.p12` is the name of the exported file and `filename.key.pem` is the name of the key pem file to be created.

Without closing the terminal or navigating to other folders, create a pem certificate using the `openssl pkcs12 -in filename.p12 -out filename.crt.pem -clcerts -nokeys` command similar to the command above (`filename.p12` is the source file, `filename.crt.pem` is the destination file).

Open these files with the nano editor and copy the key and certificate value:

On the Altcraft platform, create a resource for the push channel, select the iOS platform and enable the Apple Push Notification service switch:



Activate the **Use APNs certificate ** switch in the settings below. Fill in the following fields:

- **Certificate — the value of the pem certificate you created.**
- **Private Key — the value of the pem key you created.**

Use APNs certificate

Certificate: *

Paste cert in PEM-format

Private key: *

Paste private key in PEM-format

Save the configured resource using the buttons at the top of the page. The platform is ready to send notifications using APNs.

Yandex.AppMetrica push SDK

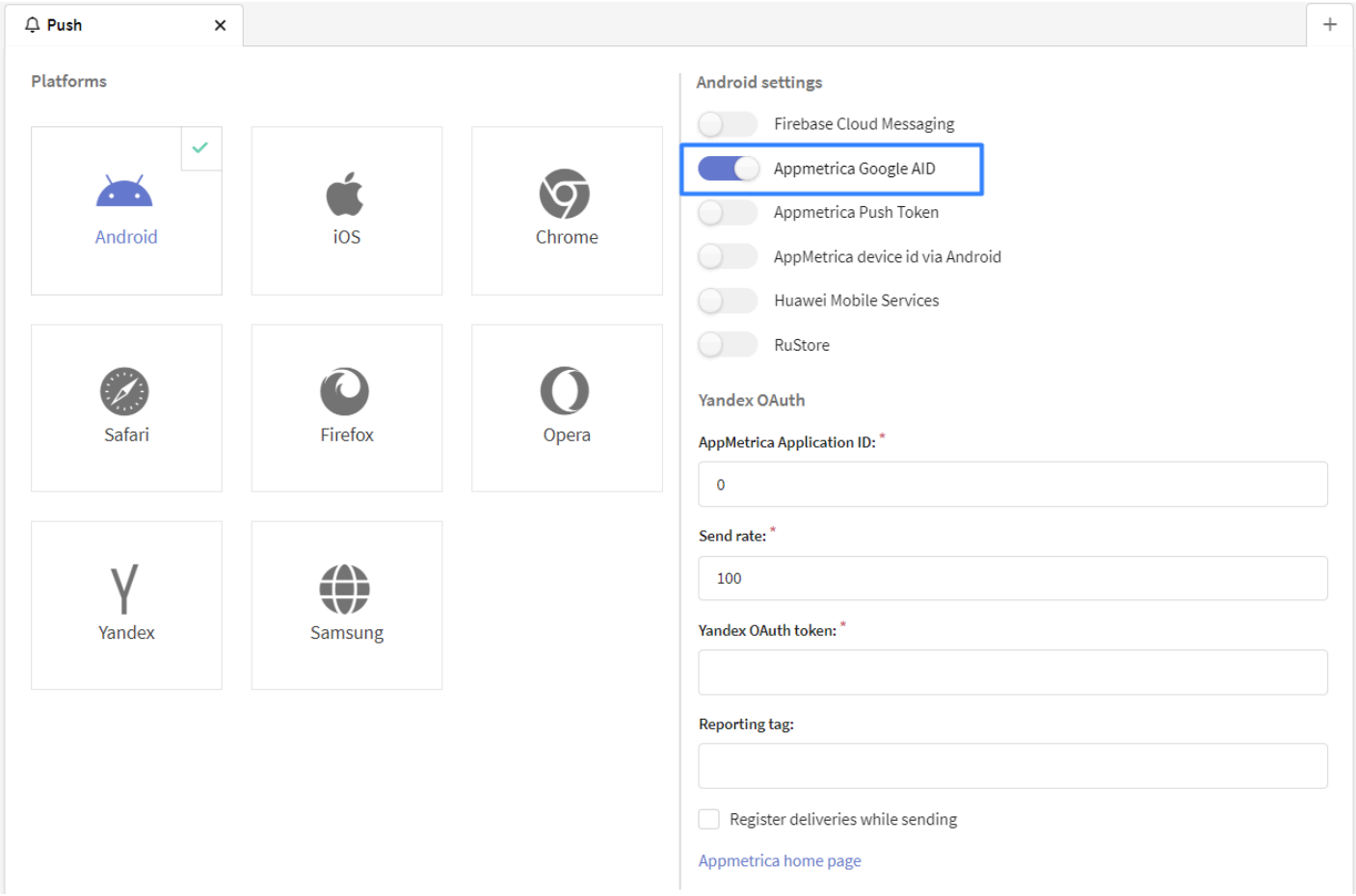
AppMetrica	**Yandex AppMetrica platform can be used as well to send mobile push notifications. AppMetrica push SDK must be integrated into your applications to do this. Yandex documentation is available here: Yandex AppMetrica documentation.**

CAUTION

Note that AppMetrica uses Google Firebase push SDK to deliver messages. You will want to implement it in your application.

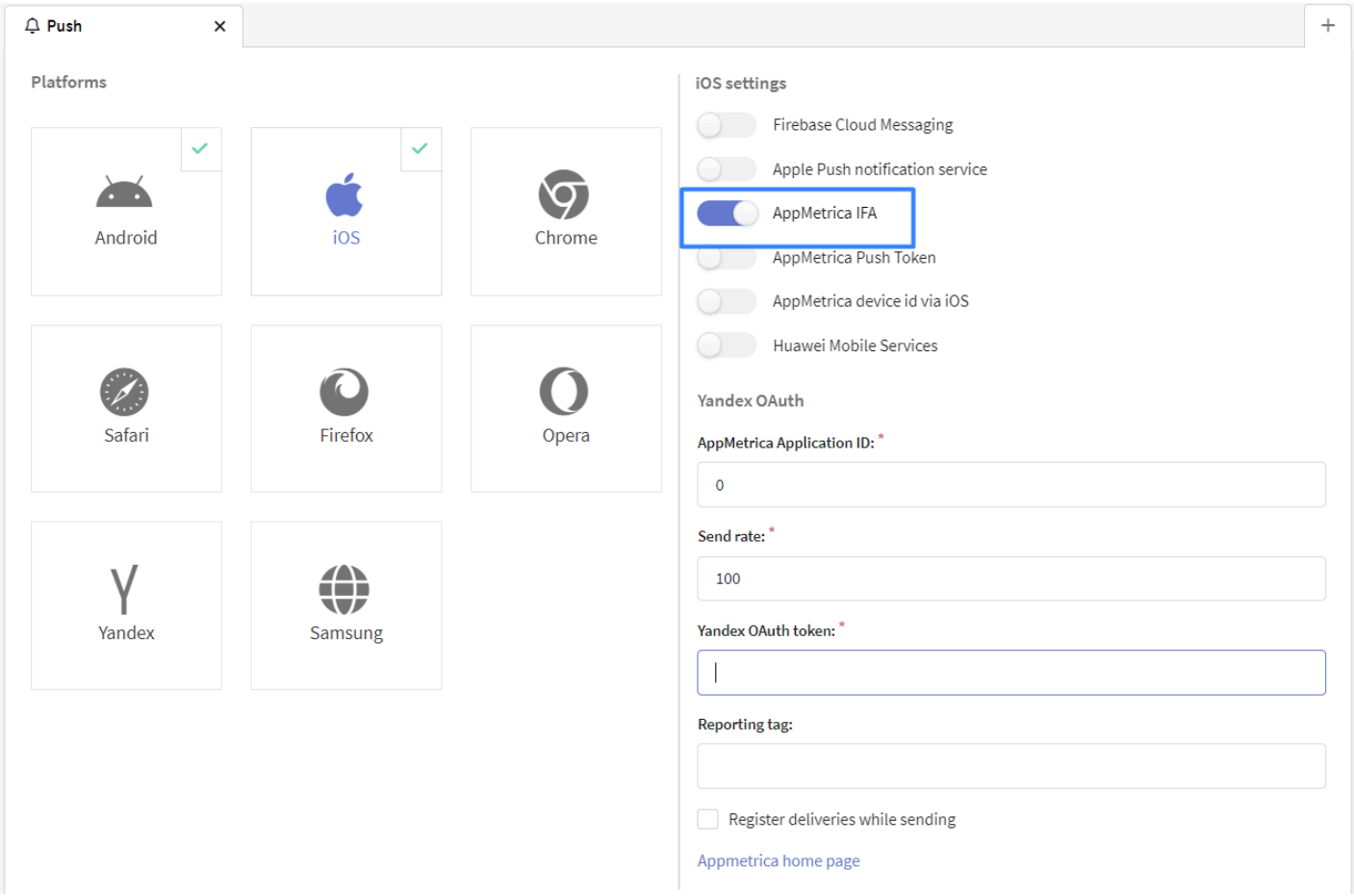
The following data can be used as push subscription source:

- **AppMetrica device ID — unique AppMetrica device identifier for both OS.**
- **iOS IFA — Apple advertising ID only for iOS.**
- **Google AID — Google advertising ID only for Android.**
- **Android Push token and iOS Push token — AppMetrica SDK tokens for both OS.**



To access AppMetrica, you will need AppMetrica API token. For each application you will also get a unique AppMetrica Application ID. You can get this data in AppMetrica application settings.

Send rate parameter limits maximum sending speed per second. You can specify a custom Reporting tag that will be used in AppMetrica analytic reports:



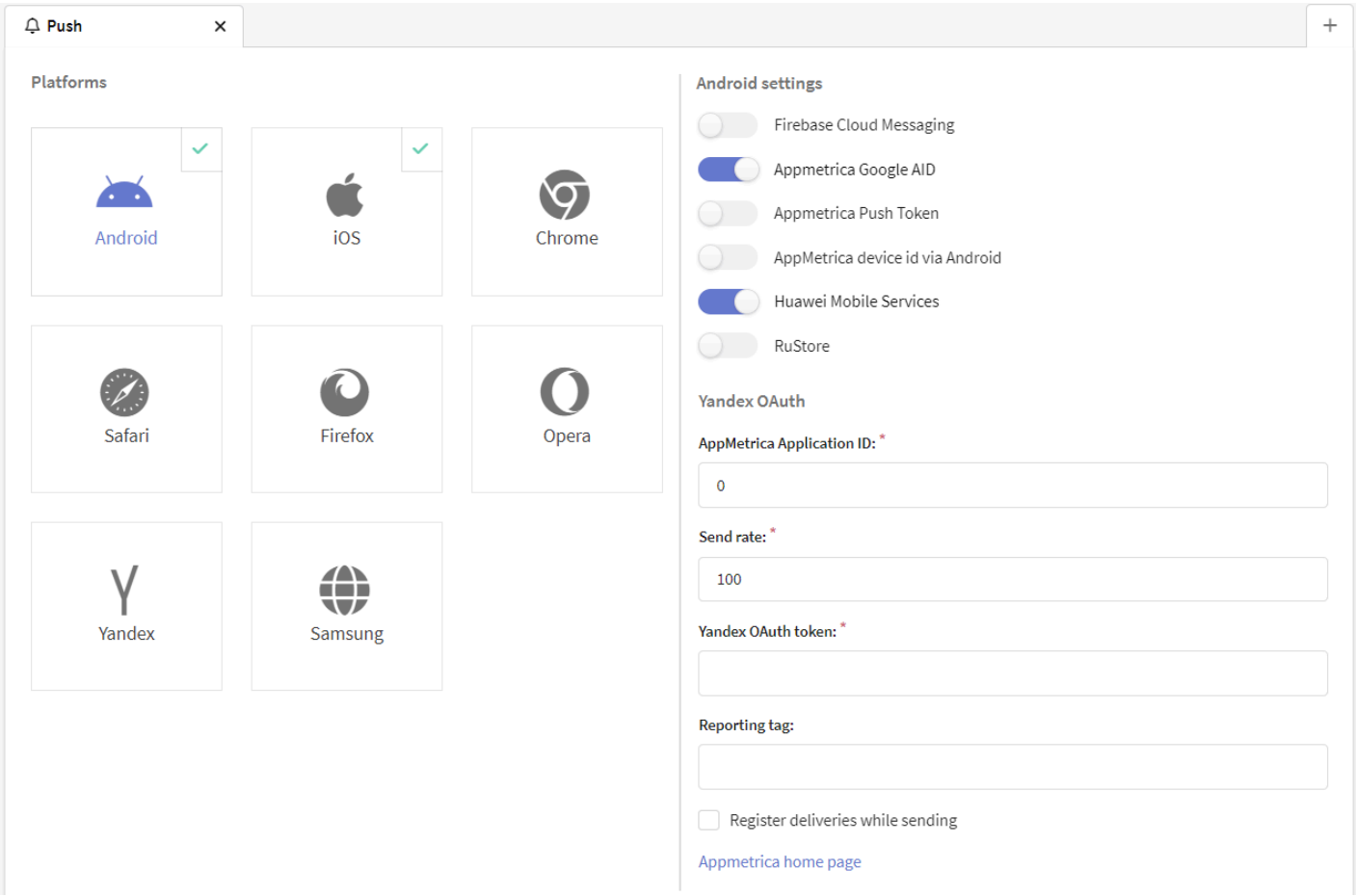
TIP

AppMetrica push subscriptions can be formed automatically. Yandex.AppMetrica events integration can manage this and much more. Contact us to get more information.

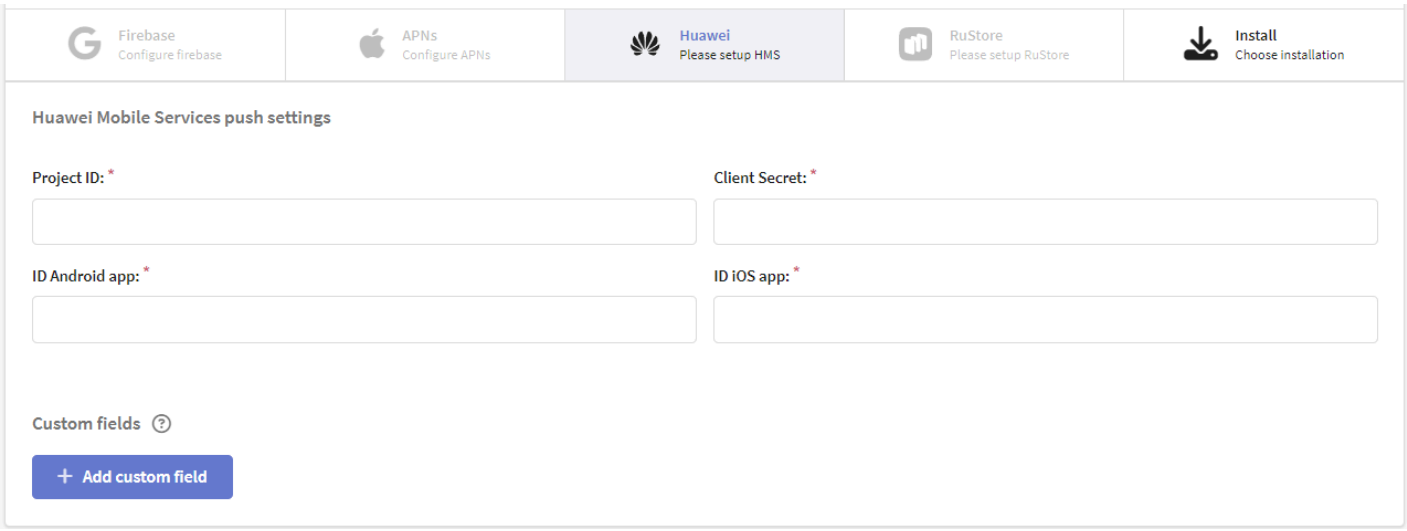
Huawei Mobile Services

HUAWEI	**Huawei Mobile Services allows you to send push notifications from Huawei-supported apps to Android and iOS devices.**

Go to resource editor and activate Huawei Mobile Services for Android and iOS apps:

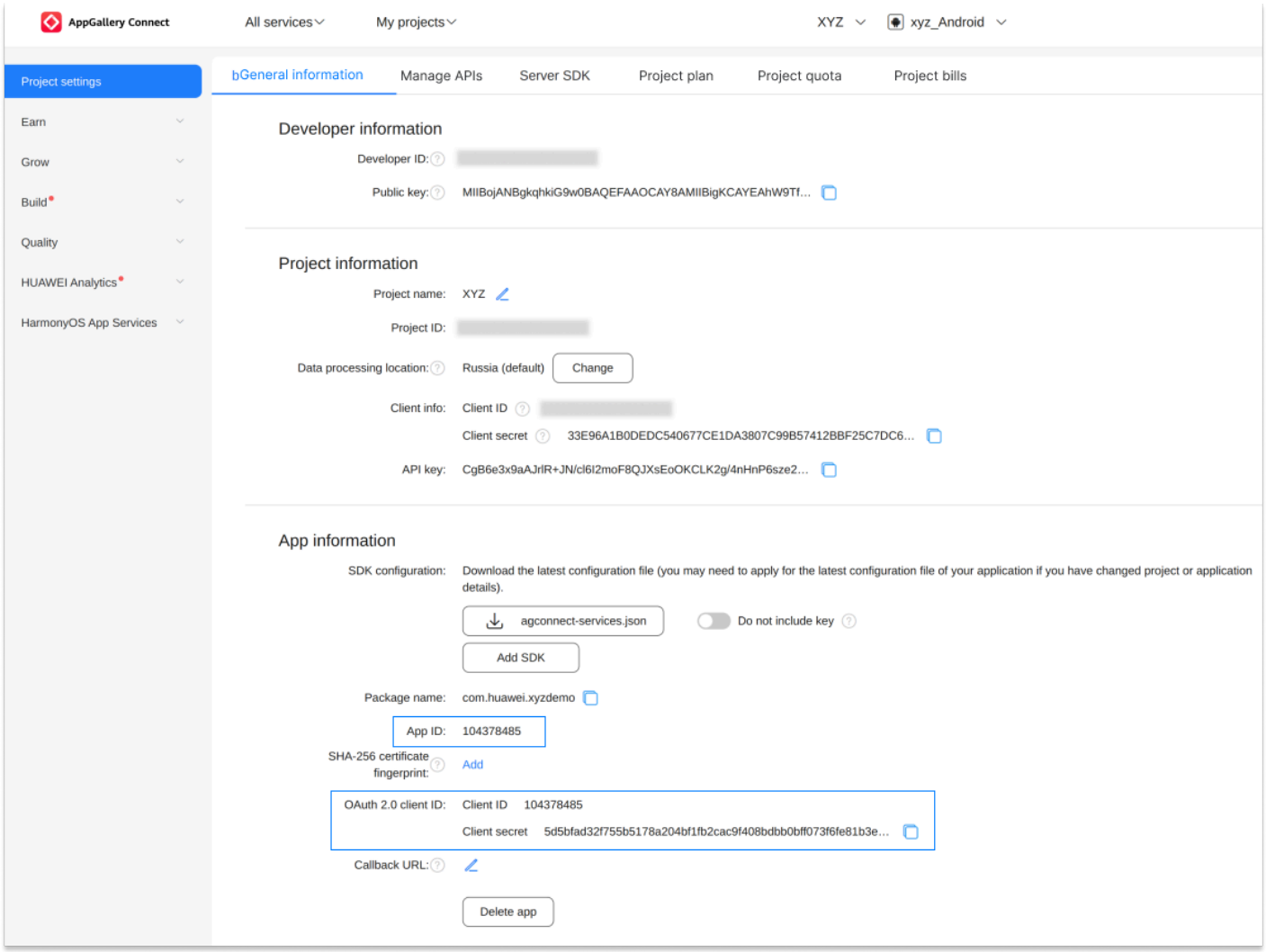


In the panel below, select Huawei. Enter app information for setup:



You can find the Project ID, Client Secret and IDs of Android or iOS apps in your project settings. To do this, open the console in your Huawei developer account and go to AppGallery Connect —> My projects. In the top panel, select the application for which you want to configure the sending of push notifications.

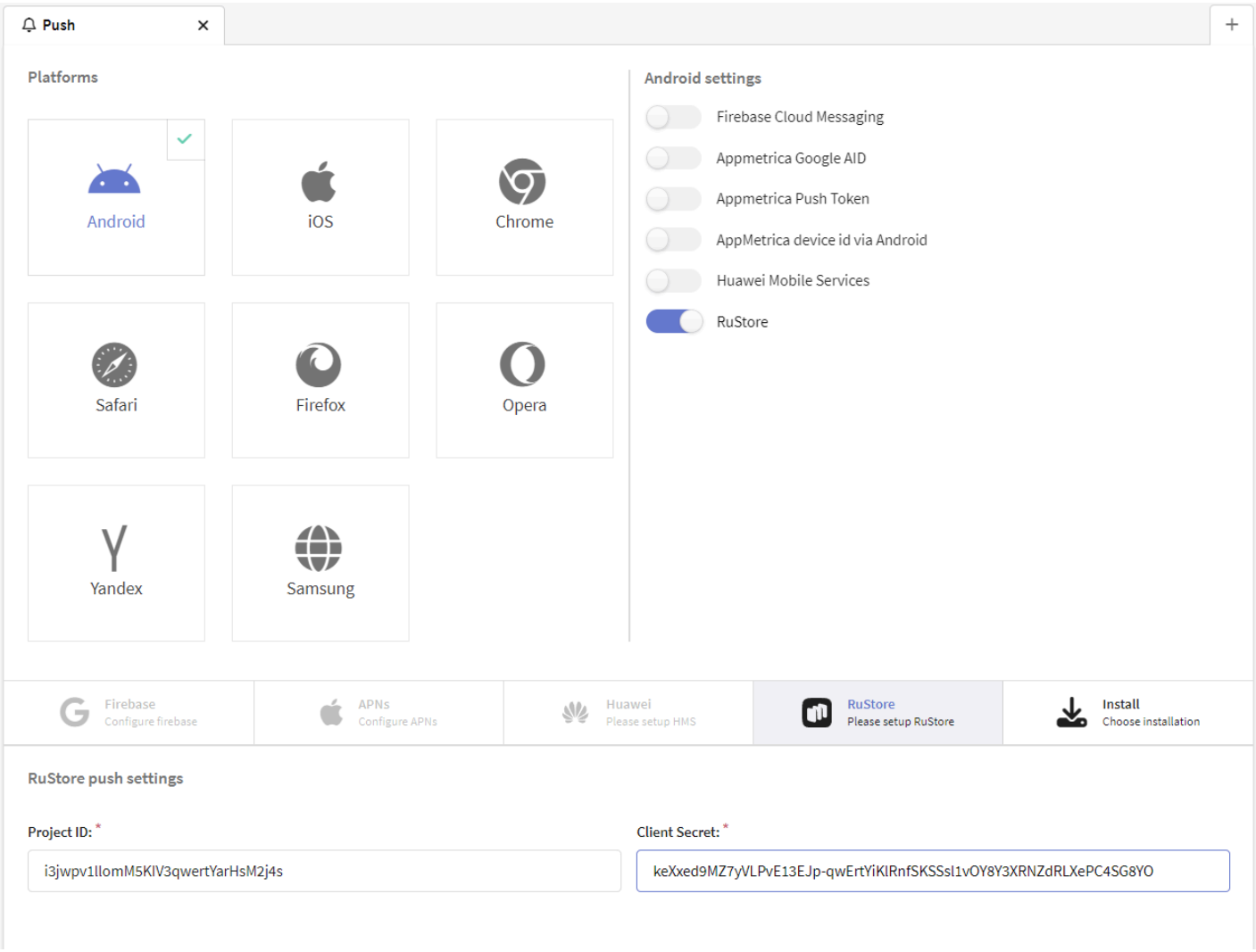
Use app information. Note that App ID = Client ID = Project ID.



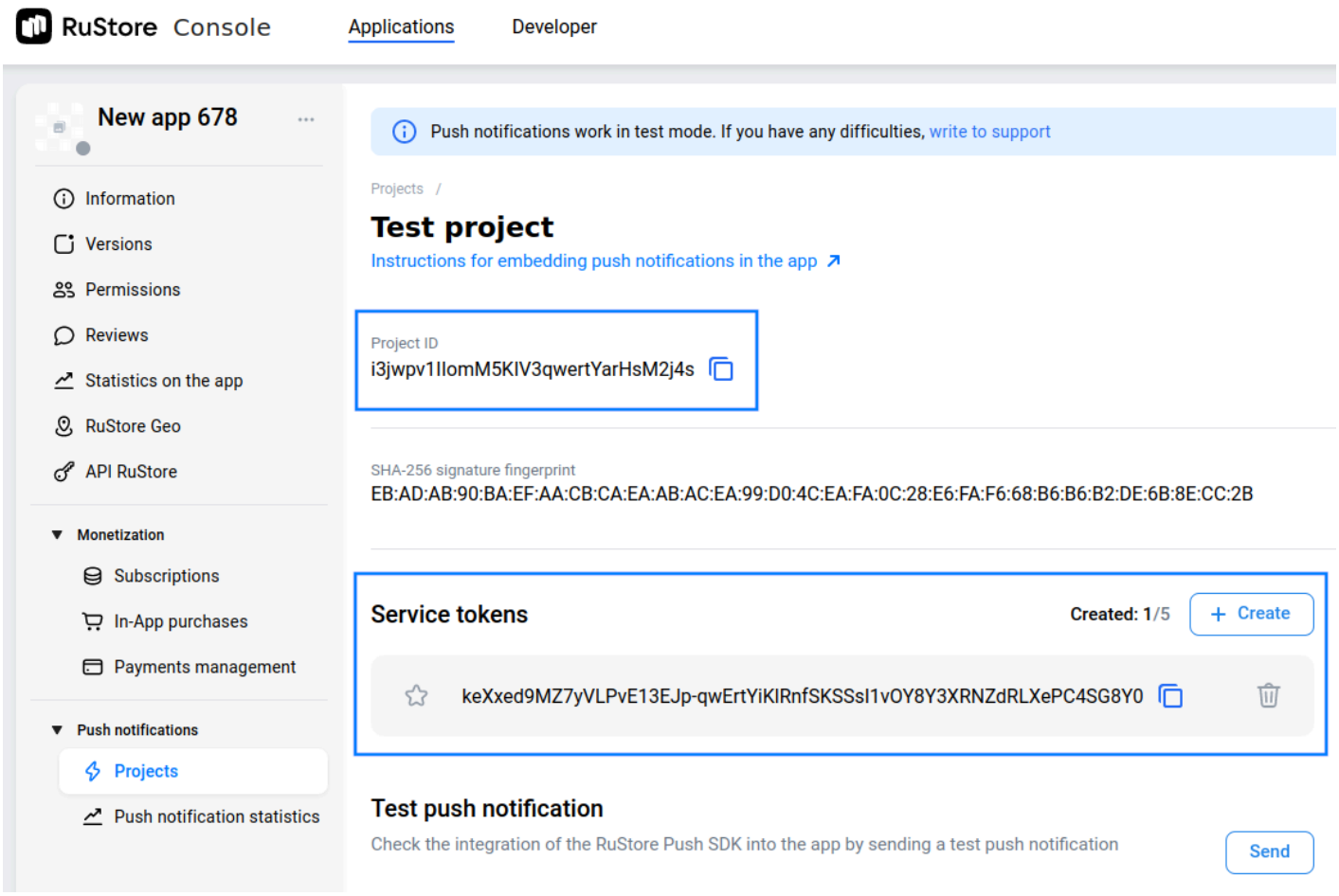
RuStore

RuStore	**The RuStore service allows you to send push notifications from applications published in RuStore. First you need to create a push notification project for your application.**

Go to resource editor and activate RuStore for Android apps. On the panel below, select RuStore and enter the project details for configuration:



The details can be obtained in the [RuStore Console] system. To do this, on the application page, go to the “Push notifications” —> “Projects” section and open your project. Use a service token as a Client secret.



App push subscription and tracking

To add an application push subscription via API please use the following request: [Add push subscription to database]

App push event tracking

App push notifications contain `ack` and `open` links.

GET HTTP request to these URLs generate “push delivered” and “push opened” events to be tracked by Altcraft platform.

You can then use these events in segmentation, profile action history and channel reports.

```
{
        "hub_link": {
          "ack":"http://your.domian.com/api/v1.1/pushes/crypto_post_events?/2ab9162159...",
          "open":"http://your.domian.com/api/v1.1/pushes/crypto_post_events?/555fbe0b9..."
        }
}
```

Add app push events

You can also add events via API requests. Here is the link: [Add app push events].

First Mobile Push Mailing

	This tutorial will help you to create and launch your Mobile Push mailing in the Altcraft Platform from scratch. Mobile push notifications are displayed in your customers' mobile device message centre – in the upper part of the screen most commonly. They are allowed by default on application installation. That's why you can start engaging your customers right away and increase your app conversion. Push notifications are much more profitable than SMS, because sending them practically costs nothing. </td> </tr>

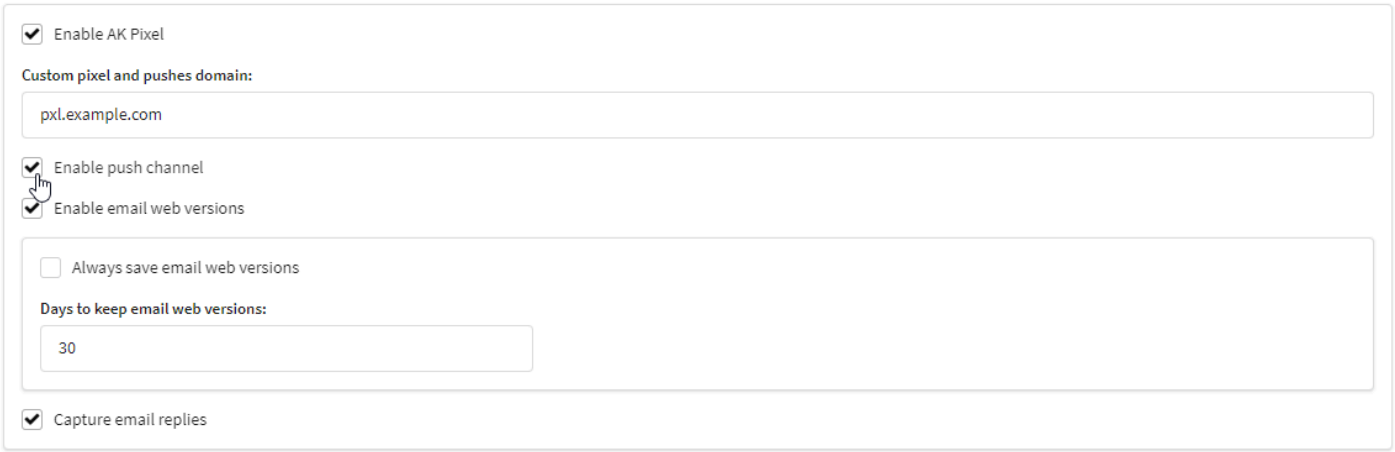
Step 1: Configure a Domain to Work with Push

CAUTION

If you use an Altcraft Marketing cloud account, please give the information to the support team at support@altcraft.com. They will help you to connect the integration.

If the Altcraft Marketing platform is installed on your equipment (On-Premise), contact your platform administrators. Or just contact our support team!

Make sure your website is running over HTTPS. In addition, web pixels must be configured in the Altcraft Marketing system.



Step 2: Subscribe customers to the Push channel

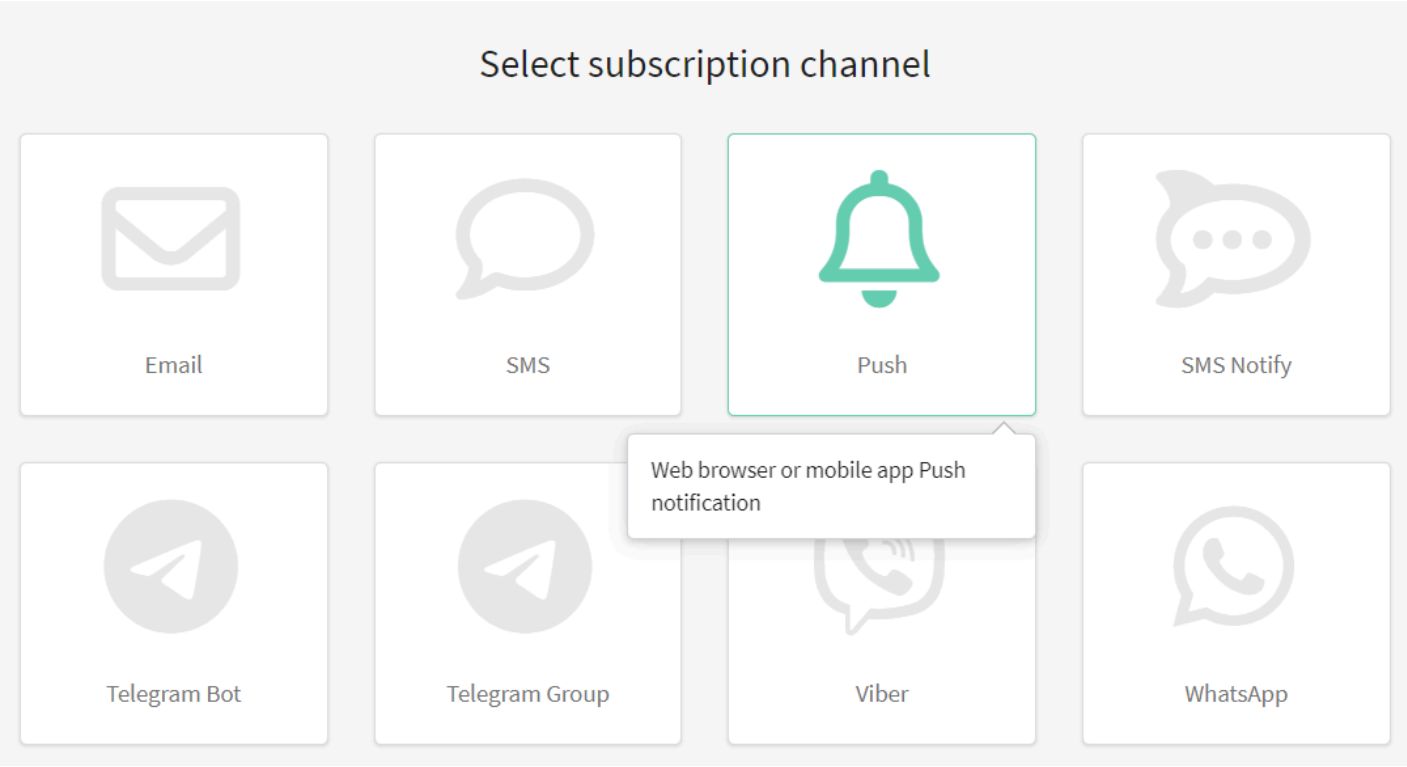
Profile databases and subscription resources

Altcraft Marketing mailings are received by your subscribers. Subscriber profiles are stored in profiles databases.

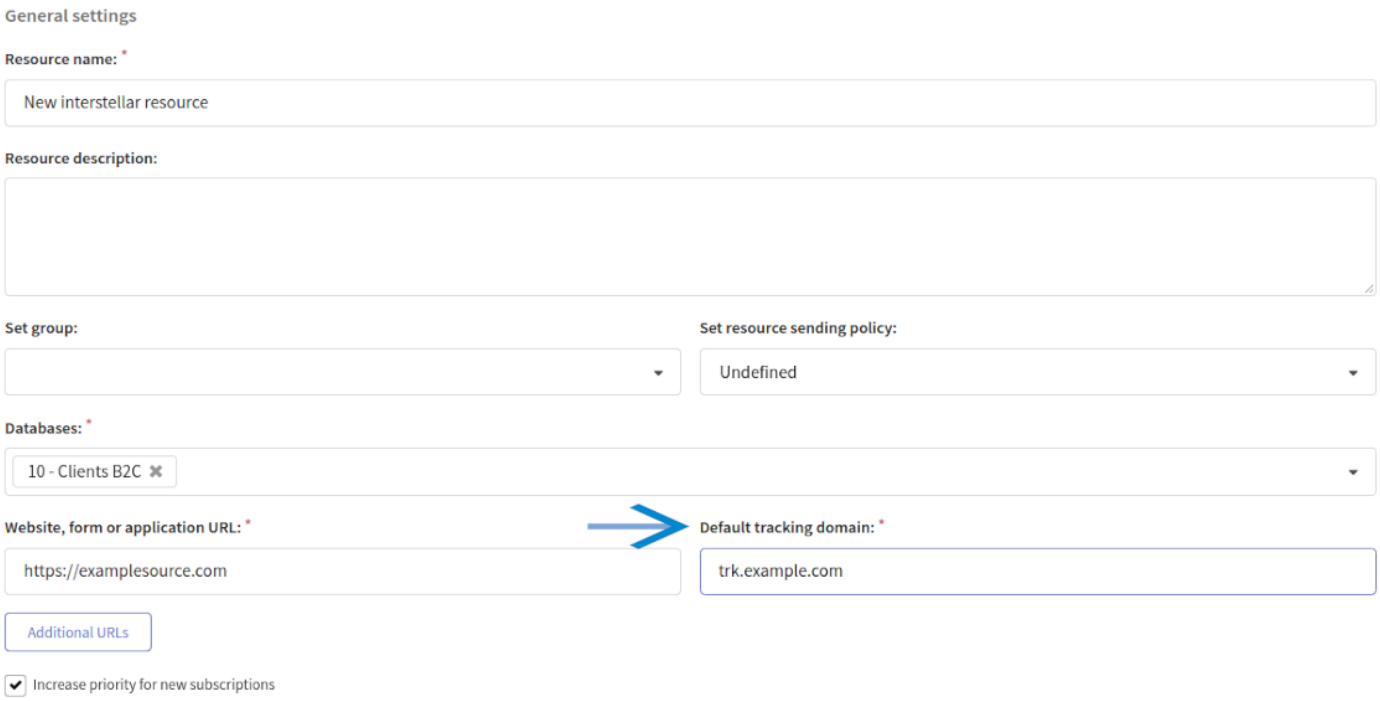
When you start your Push mailing, you use profile subscriptions to resources. This way, you can send notifications to different devices or to different browsers. Each client profile can be subscribed to one or more resources.

What can be a resource? A specific marketing area, a site, an application, or the type of content.

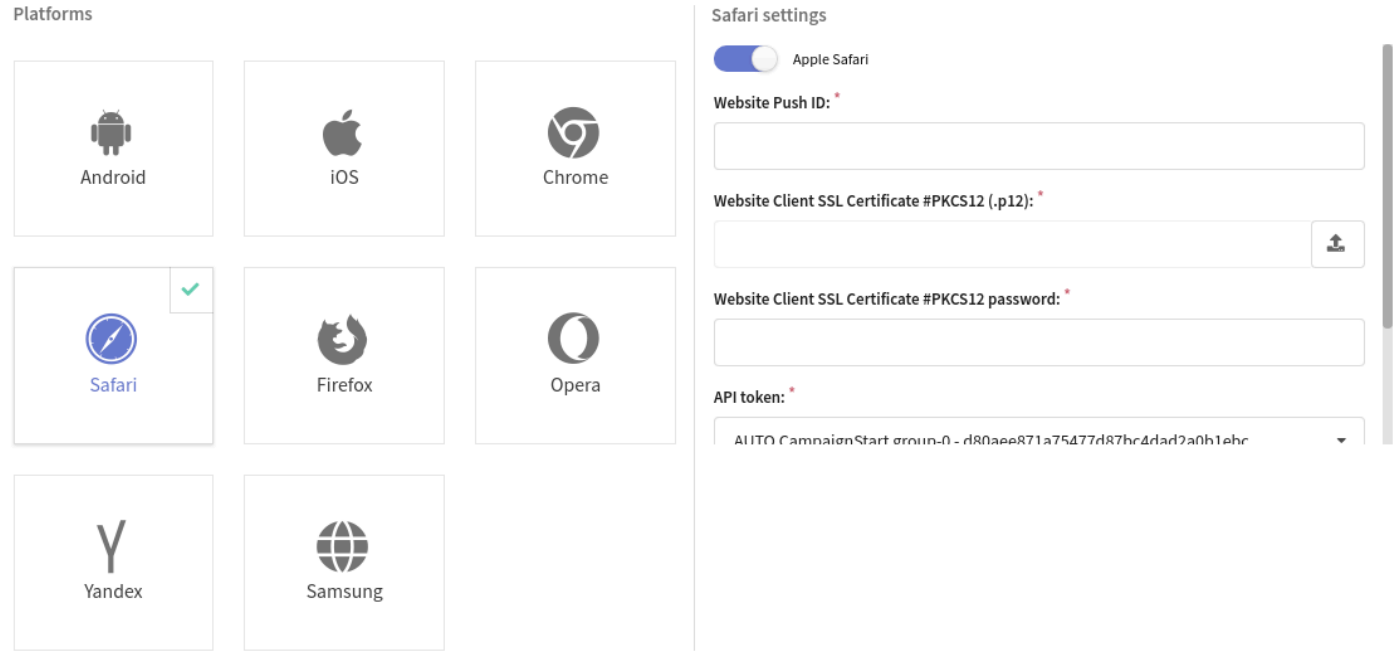
Select the section Resources in the main menu, then create a new resource or select an existing one. Select the Push channel for your new resource:



In the general settings, add the necessary databases. If it's necessary, assign a sending policy and specify the Default tracking domain. This domain will process your customers' clicks on links in messages:



In Push settings, select the browsers to which you are going to send notifications to clients and set them up:



TIP

In Altcraft Marketing there are 5 mobile push options:

- **Google Firebase Cloud Messaging — for both Android & iOS**
- **Apple Push Notification Service — iOS apps only**
- **Yandex.AppMetrica — for both Android & iOS**
- **Huawei Mobile Services — for both Android & iOS.**
- **RuStore — for Android apps only.**

Don't know how to configure each platform? Take a look at this tutorial.

Customer profiles and subscription to the resource

So, we've just configured the profiles database and subscription resource. Now it's time to import customer data into the platform. For each client, Altcraft Marketing creates a single digital profile that includes personal data, history of actions and subscriptions to your resources.

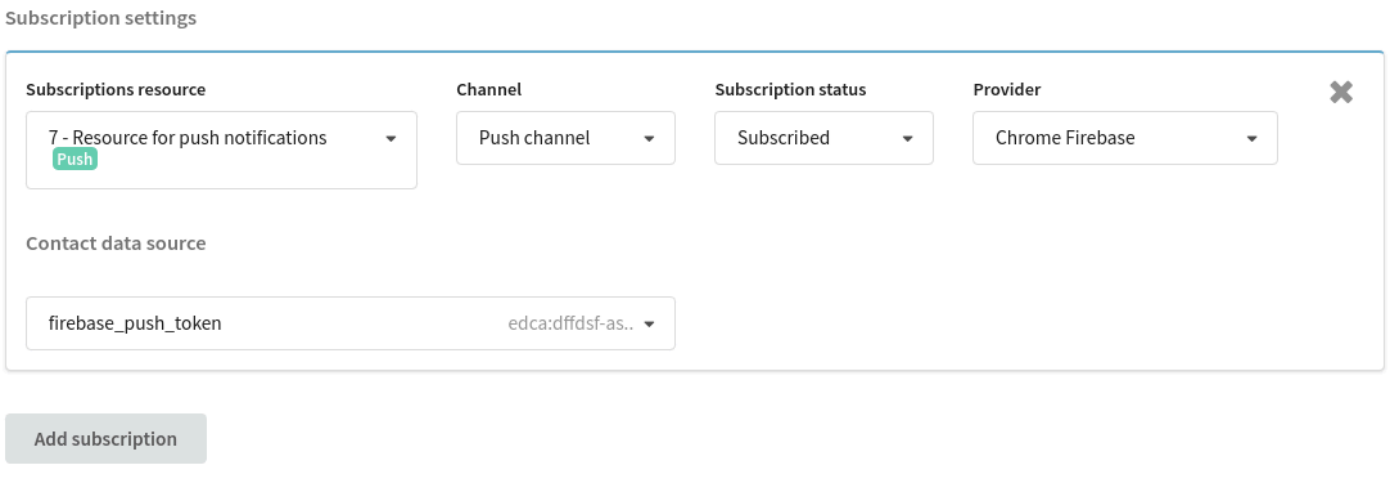
Existing subscriber database

If you already have the database full of tokens for your customers, you can upload customer data to the platform and use it for mailings: one at a time, in a list, or from a table file. Different import methods are described in this article.

You can also set up automatic import of client profiles:

- **on schedule from your SQL database**
- **through the use of the integration with platform API**

When you import data, sign clients to the previously prepared resource. To do this, in the Subscription settings section, select Push as a communication channel, select a notification provider, in the Contact data source field, specify the column with device tokens:



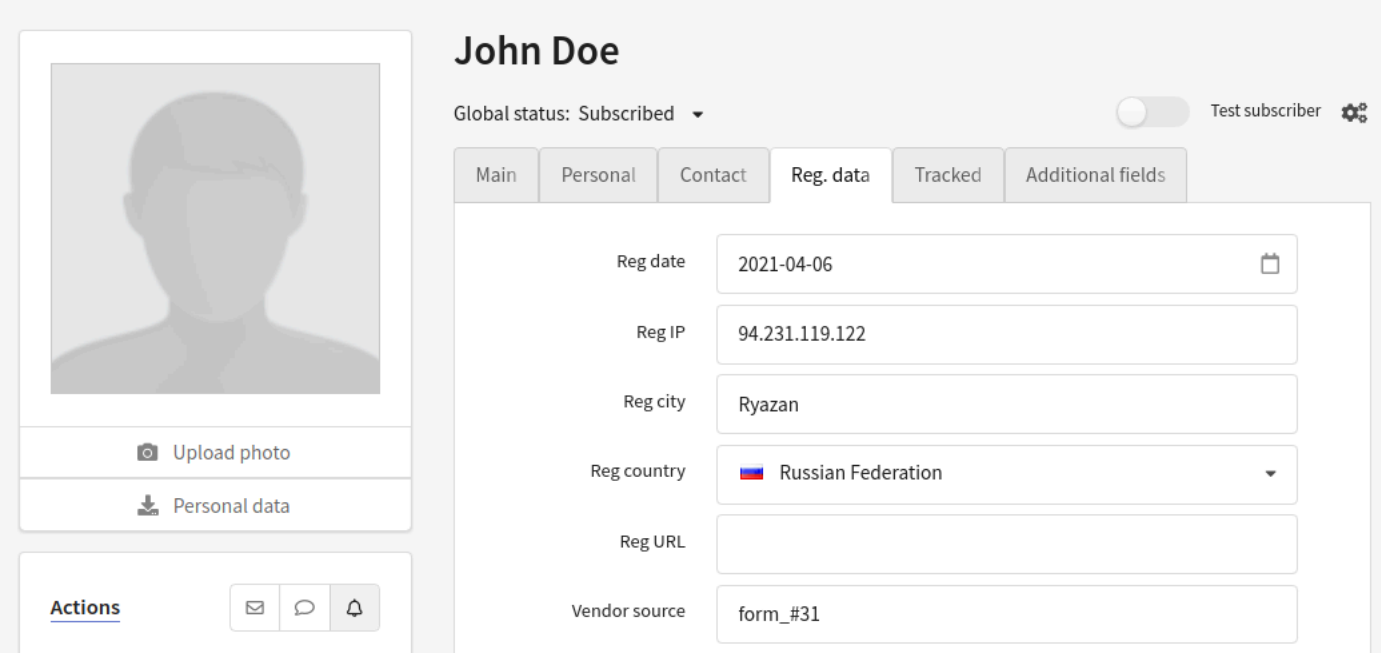
TIP

If you already have a profiles database, add them a subscription to a new resource. There are two options:

- **if customers have subscriptions to another resource with up-to-date email addresses — use bulk profile update**
- **if you need to upload to the platform SMS phones for subscription, use the profile import in the "update only" mode**

New push subscribers of your website

Altcraft Marketing imports the profiles of your website visitors when they allow push notifications. At the same time, information about the subscriber's device, IP address and geolocation determined by it are recorded in the profile:

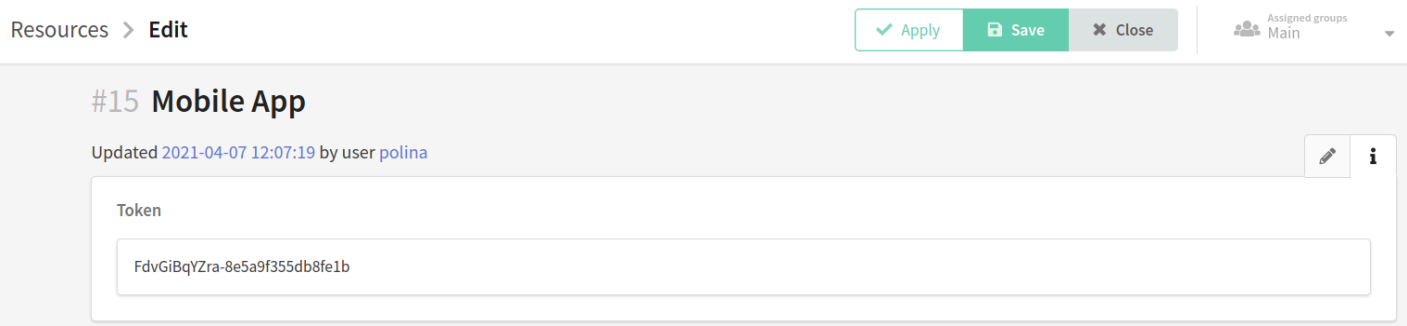


Go to the next step to configure the import of new push subscribers.

Step 3: Add Push SDK to your app

If you're going to launch push mailings, your app must have a toolkit for working with push. This toolkit is SDK of the sending platform you plan to use.

Contact the developers of your app and give them the token of the previously configured resource. How to get it? Go to the resource settings and look at the screenshot below:



Subscriptions for app notifications are assigned to users with API request from the application: Add push subscription to database.

Altcraft Platfrom has its own SDK, Altcraft mSDK. It is implemented on Firebase for the Android platform and on Firebase and APNs for the iOS platform. To access the SDK you can contact our support team. You can learn more about Altcraft mSDK features here.

TIP

A subscription to Yandex.AppMetrica can be added automatically to users of the app that has the connected metric service. Read more in the article "Integration with Yandex.AppMetrica".

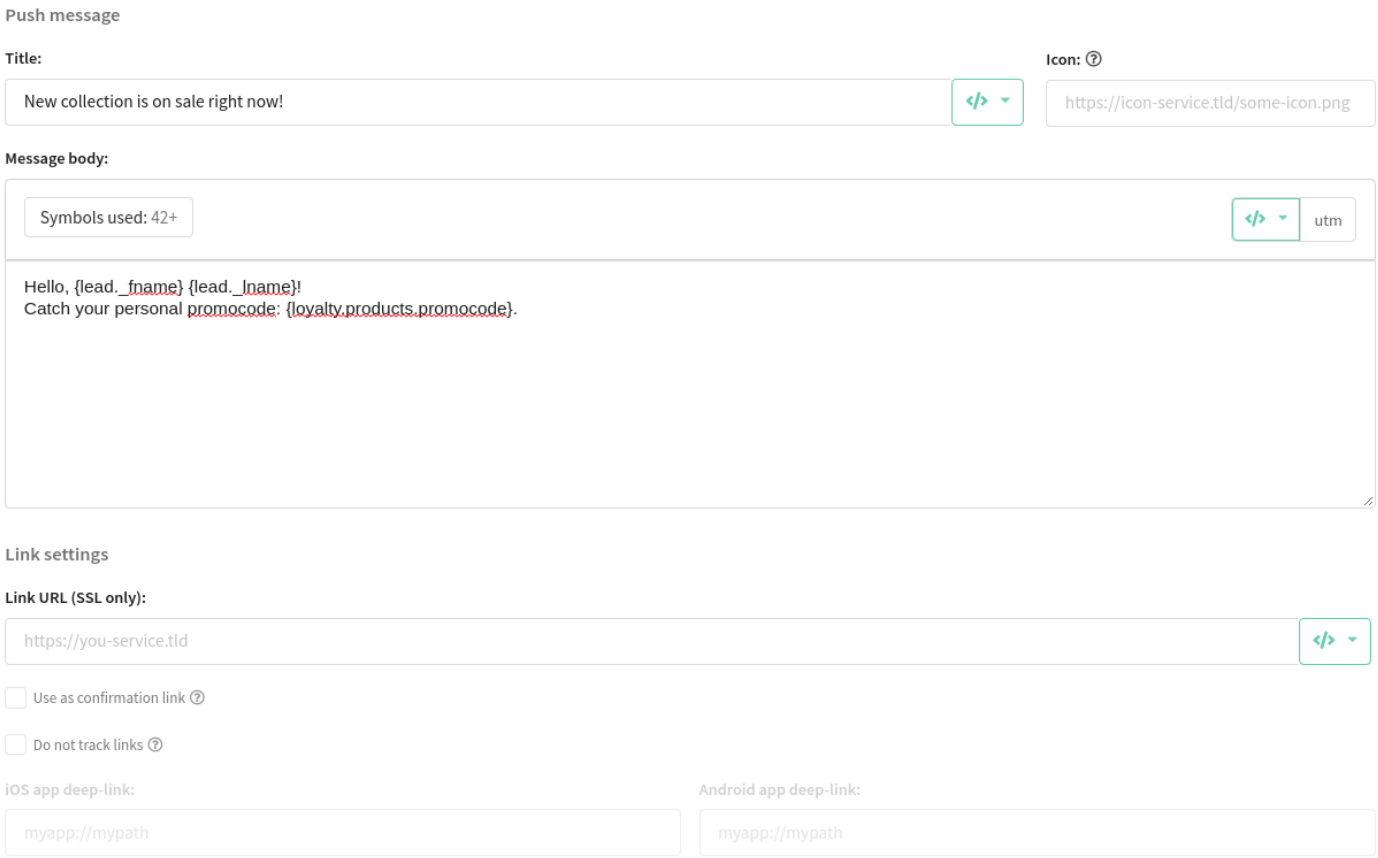
Step 4: Prepare content for sending

The content for your Push mailings is created in Message Templates. The templates contain content for all available channels. When you create a new template, select the Push channel or add it for an existing one.

The template of the push notification can contain:

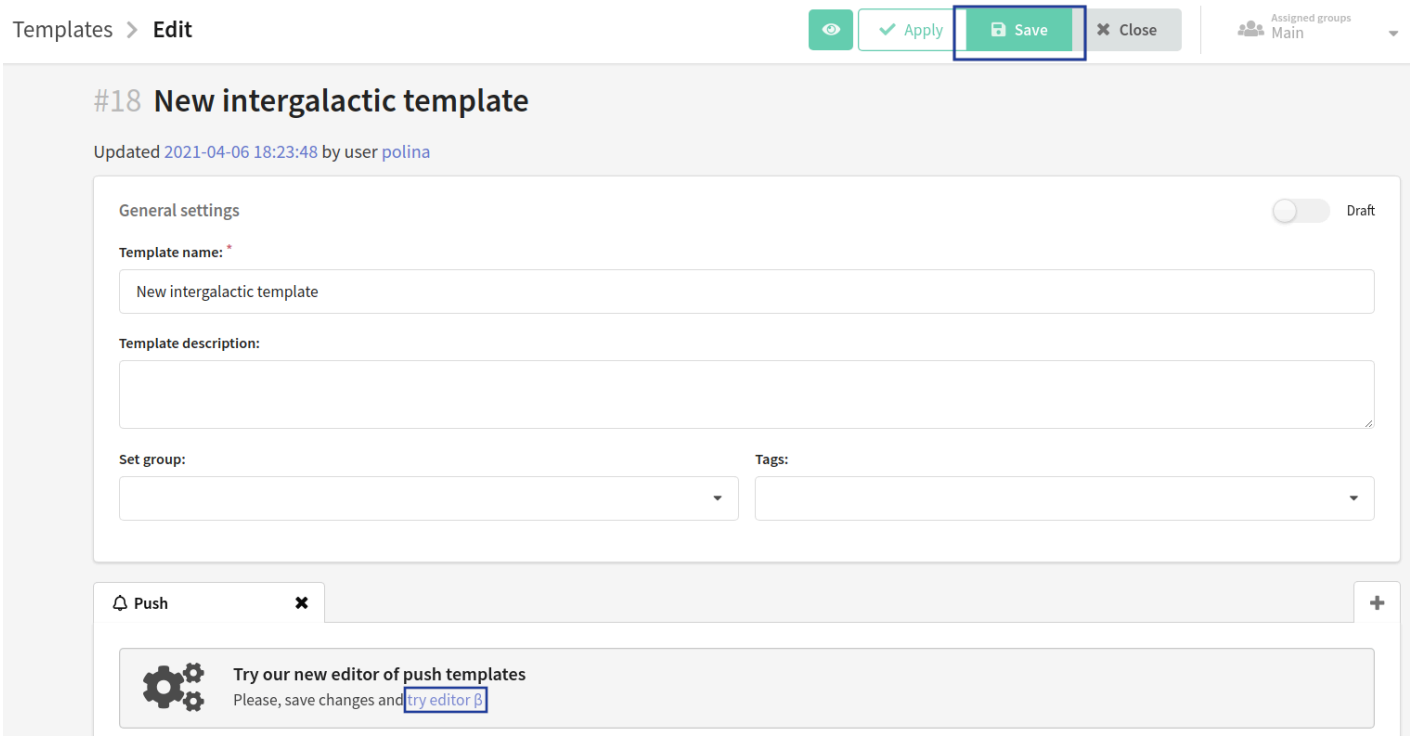
- **A title that is similar to email subject**
- **Notification body is the main text part of the message**
- **An image or an icon that will be displayed in the notification**
- **A clickable link is the link that the subscriber will follow when he clicks on the push notification.**

To the title, link and body, you can add the data of a specific client, as well as dynamic text content from external sources:



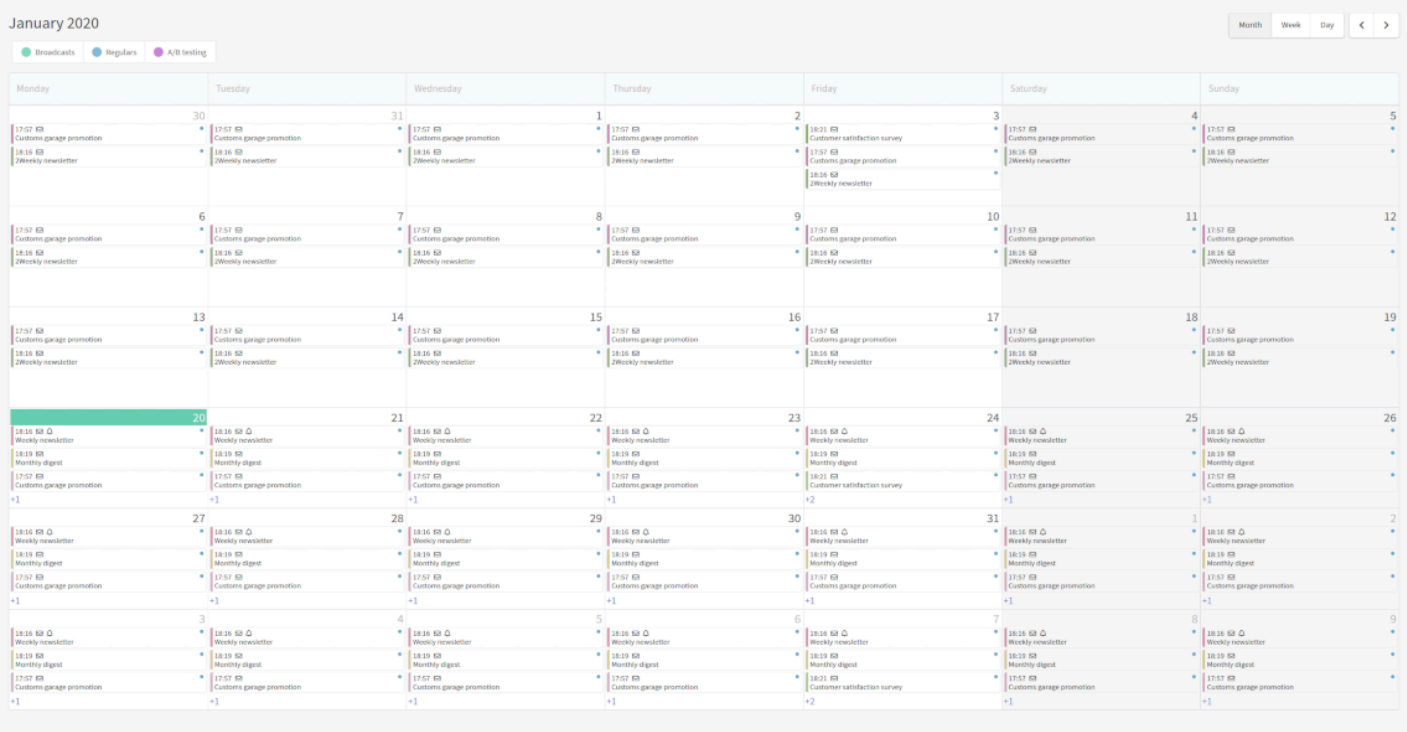
If you send not just push, but Rich push notifications, you'd better take a look at the special editor to create a template. In Rich push notifications, you can add not only a text and an icon, but also buttons, images, banners, and even audio and video files. What's more, the new editor will allow you to individually customize the notification for a specific OS, as well as add adaptive links and deeplinks.

To open the Rich push editor, you need to make an initial version of a push message in the usual template editor at first. We've just described how to do this above. Then click Save. Now go back to the created template and click try editor β to launch the new editor.

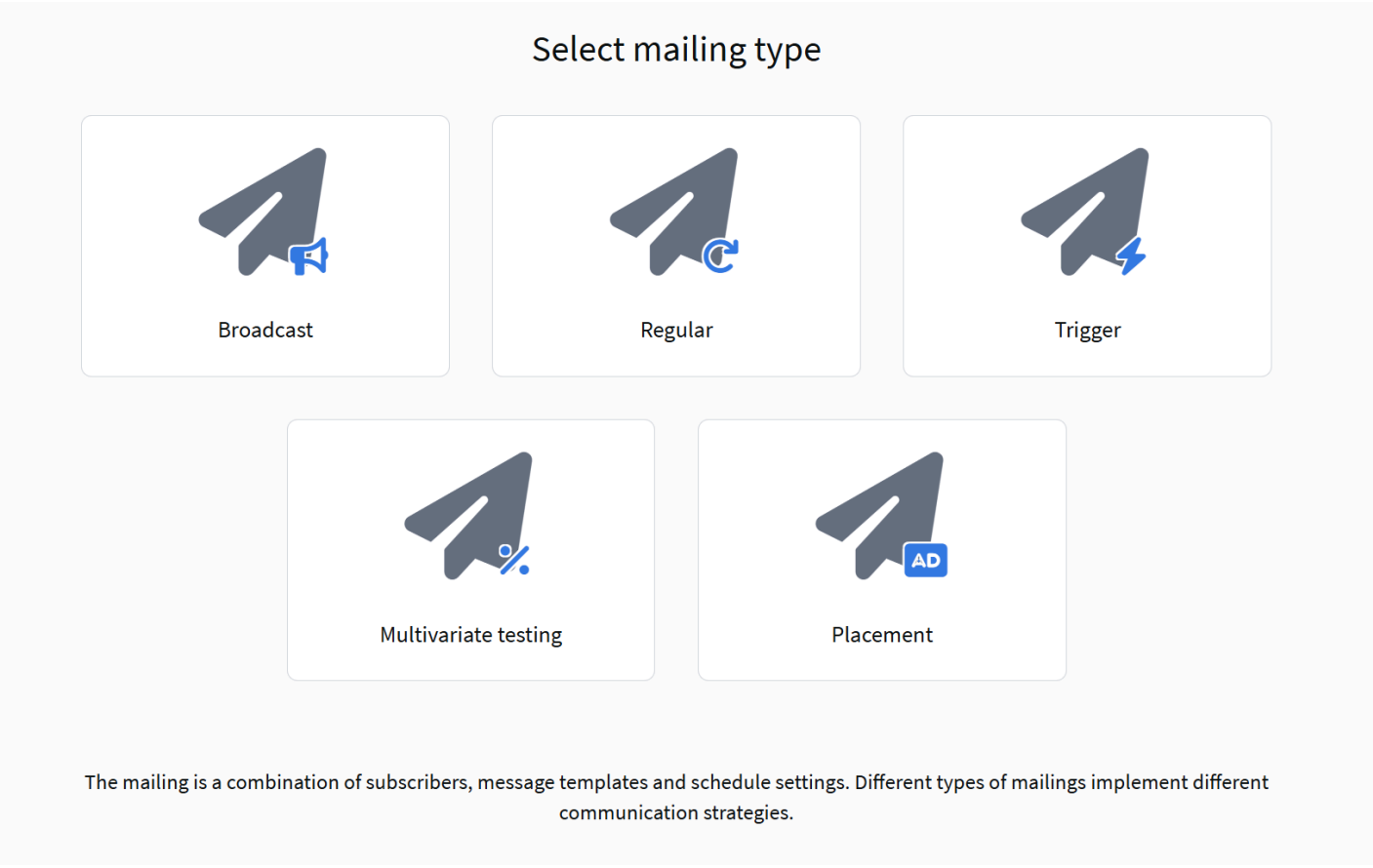


Step 5: Select a mailing type

	If you want to send an SMS message, you need to use such a tool as a [Mailing](#). Do you want to send out a **one-time mailing** to all profiles of the database or to the audience of the specific segment? Choose ** ** Broadcasts. If the mailing is periodic, your choice is a ** ** Regular mailing. In the main menu select Mailing and open ** ** Calendar. This is where you'll find all scheduled mailings. </td> </tr>



Decide on a date and create a mailing of the required type:



TIP

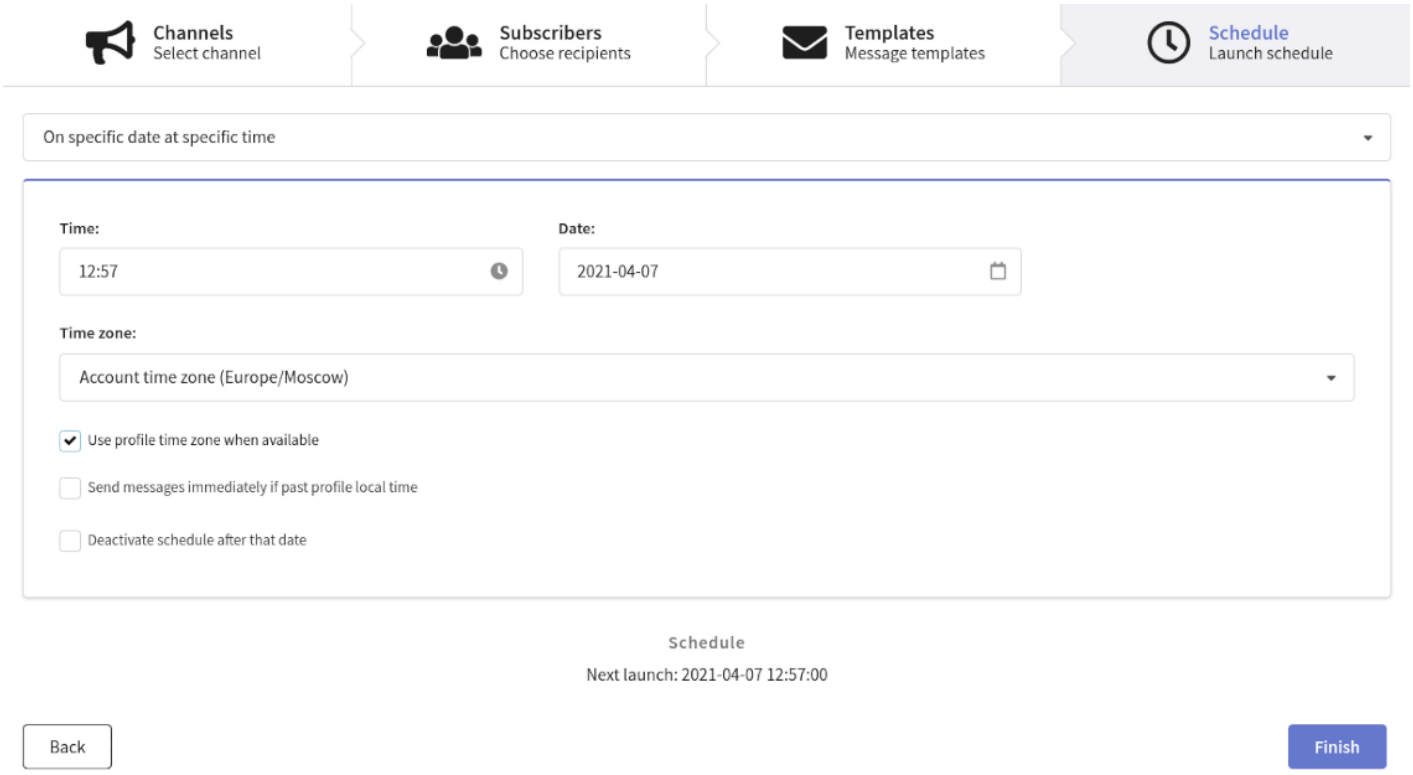
You can find details about how to set up Broadcast on [this page](#).

TIP

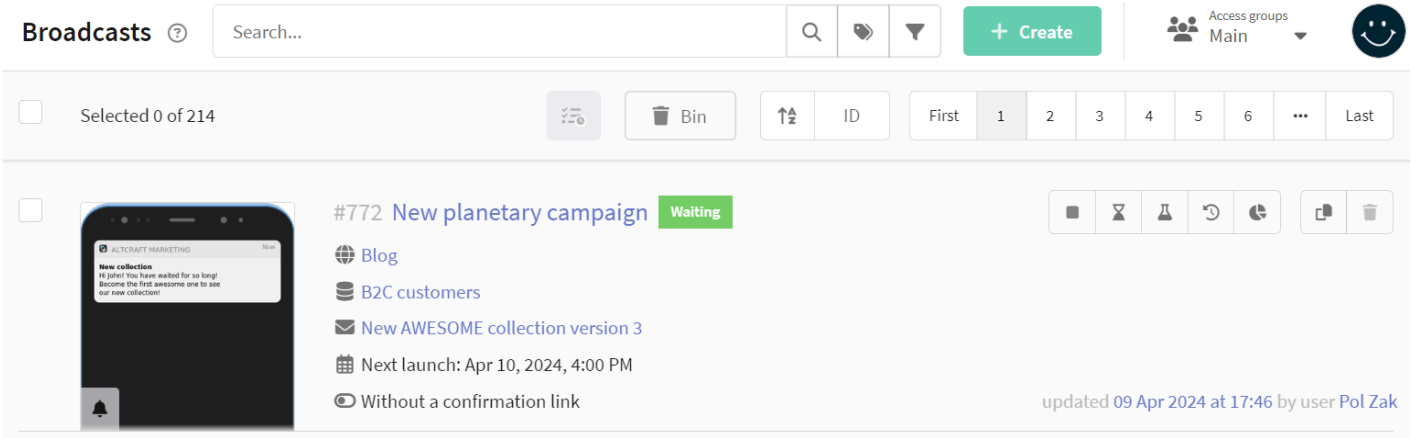
You can find details about how to set up Regular mailing on this page.

Once you're done with the general mailing settings, you need to:

1. **Select Push as the communication channel**
2. **Select providers and sending mode. There are two sending modes - **fanout**, sending to all subscriptions, or **last**, sending to the subscription with the most recent creation date. If there are several of these (created at the same time), then one subscribtion will be selected randomly. If mailings have filters by provider, custom fields or categories, then the filters are first applied, and then the most recent subscription is choosen;**
3. **Select the subscription resource you've just configured and the audience of recipients;**
4. **Add a message template to your message;**
5. **Set a schedule. For example, for a broadcast the option "On specific date at specific time" is available.**



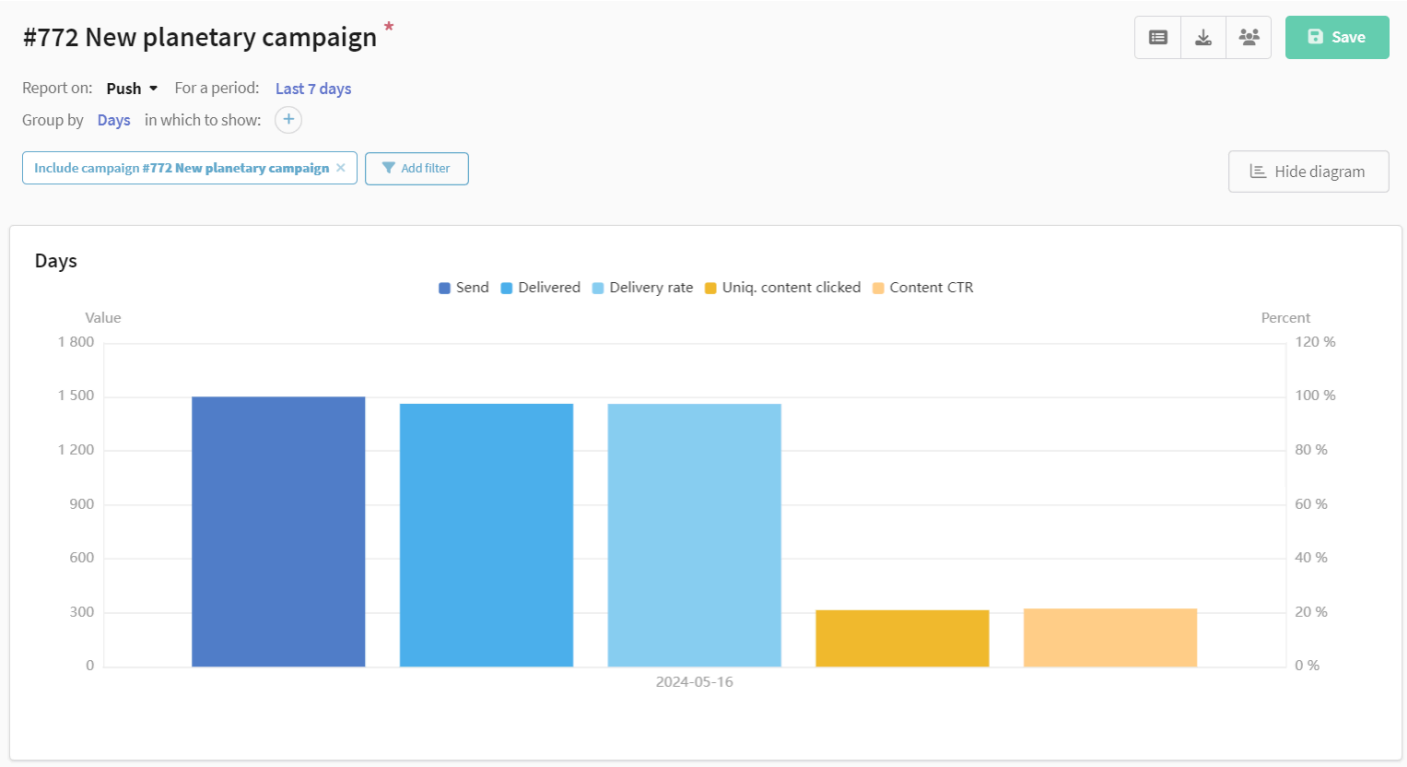
Once the mailing is set up, save it. Later you can find it in the list of mailings of the corresponding type. There is a quick action menu on the right side of each mailing. Use it to test the mailing — and if everything is successful, activate it:



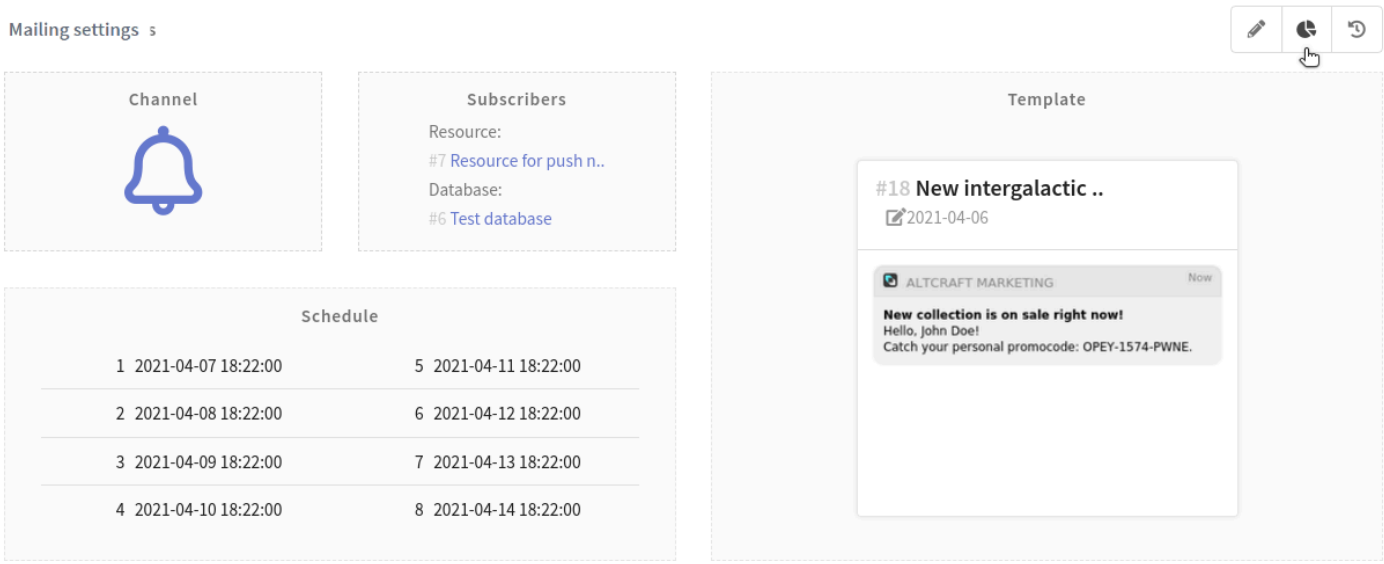
Step 6: Assess the mailing effectiveness

Do you want to assess the effectiveness of your mobile push mailings? Go to the section Analytics in the main menu of the platform.

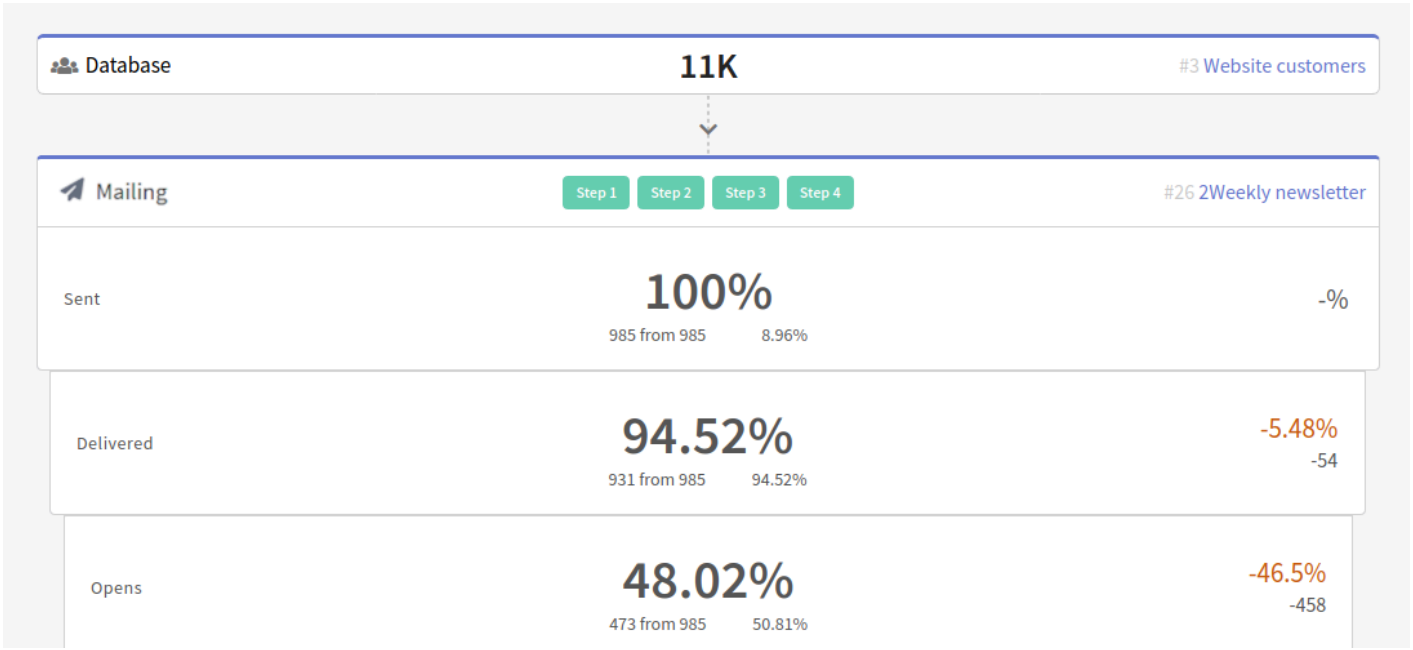
The channel report provides analytics for one or several mailings, for one or several communication channels. For grouping, you can select **Report on ** Push:



You can also go to the channel report for a specific mailing by following the link in the mailing editor:



Actions in the Push mailing or in the template can turn into steps in the interactive conversion funnel:


Database
11K
#3 Website customers
Mailing
Step 1
Step 2
Step 3
Step 4
#26 2Weekly newsletter
Sent
100%
985 from 985
8.96%
-%
Delivered
94.52%
931 from 985
94.52%
-5.48%
-54
Opens
48.02%
473 from 985
50.81%
-46.5%
-458

Mobile push: integrate your app with Altcraft

Here is a list of what the developers must implement themselves to integrate the application with Altcraft Platform:

- **connecting the required SDK push provider**
- **sending new push subscriptions to Altcraft**
- **display push message**
- **registration of events**

Follow the instructions:

Providers: push message structure

Altcraft Platform supports the following services for sending mobile push messages:

- **Google Firebase Cloud Messaging — for Android and iOS apps.**
- **Apple Push Notification Service — for iOS apps only.**
- **Yandex.AppMetrica — for Android and iOS apps.**
- **Huawei Mobile Services — for Android and iOS apps.**
- **RuStore — for Android apps only.**

INFO

You need Connection of the corresponding SDKs in the app.

CAUTION

Yandex.AppMetrica uses Google Firebase SDK to deliver messages. You will need to implement it in your app.

Integration with Yandex.AppMetrica is available in Altcraft Platform for importing customer profiles, registering their actions and value (cost).

At first, it is necessary to verify that the application supports the structure of push notifications in Altcraft Platform. If the format is not supported, Altcraft can adjust the message structure from its side.

Android Firebase (legacy)

INFO

It's possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "to": "SubscriptionID",
        "priority": "Priority",
        "time_to_live": 0-86400,
        "content_available": true,
        "mutable_content": true,
        "data": {
                "uid": "UID",
                "message": "Body",
                "mediaUrl": "Icon",
        "image": "Image",
                "launch_id": "LaunchID",
                "hub_link": {
                        "open": "URLOpenEvent",
                        "ack": "URLDelivEvent"
                }
        },
        "notification": {
                "title": "Title",
                "body": "Body",
                "icon": "Icon",
                "color": "ImageBackgroundColor",
                "click_action": "ClickURL",
                "sound": "Sound"
        }
}
```

TIP

uid — the internal send ID, unique for each message.
launch_id — the internal send ID, unique within the mailing.

Android Firebase (api v1)

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "message": {
                "token": "SubscriptionID",
                "android": {
                        "priority": "Priority",
                        "ttl": TTL,
                        "data": {
                                "uid": "UID",
                                "launch_id": "LaunchID",
                                "hub_link": "{\"open\": \"URLOpenEvent\", \"ack\": \"URLDelivEvent\"}"
                        },
                        "notification": {
                                "title": "Title",
                                "body": "Body",
                                "icon": "Icon",
                                "click_action": "ClickURL",
                                "sound": "Sound",
                                "color": "ImageBackgroundColor",
                                "light_settings": {
                                        "color": {
                                                "red": 0-1,
                                                "green": 0-1,
                                                "blue": 0-1,
                                                "alpha": 0/1
                                        },
                                        "light_on_duration": "LEDOnDuration",
                                        "light_off_duration": "LEDOffDuration"
                                },
                                "image": "Image",
                                "default_vibrate_timings": true/false,
                                "notification_count": 0-999
                        }
                }
        }
}
```

TIP

uid — the internal send ID, unique for each message.
launch_id — the internal send ID, unique within the mailing.

iOS Firebase (legacy)

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "to": "SubscriptionID",
        "priority": "Priority",
        "time_to_live": 0-86400,
        "content_available": true,
        "mutable_content": true,
        "data": {
                "uid": "UID",
                "message": "Body",
                "mediaUrl": "Icon",
        "media": "Media",
                "launch_id": "LaunchID",
                "hub_link": {
                        "open": "URLOpenEvent",
                        "ack": "URLDelivEvent"
                }
        },
        "notification": {
                "title": "Title",
                "body": "Body",
                "icon": "Icon",
                "color": "ImageBackgroundColor",
                "click_action": "ClickURL",
                "sound": "Sound"
        }
}
```

TIP

uid — the internal send ID, unique for each message.
launch_id — the internal send ID, unique within the mailing.

iOS Firebase (api v1)

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "message": {
                "token": "SubscriptionID",
                "apns": {
                        "headers": {
                                "apns-expiration": "TTL"
                        },
                        "payload": {
                                "aps": {
                                        "badge": 0-999,
                                        "sound": "Sound",
                                        "mutable_content": 1,
                                        "alert": {
                                                "title": "Title",
                                                "subtitle": "SubTitle",
                                                "body": "Body"
                                        }
                                },
                                "click-url": "ClickURL",
                                "uid": "UID",
                                "launch_id": "LaunchID",
                                "hub_link": "{\"open\": \"URLOpenEvent\", \"ack\": \"URLDelivEvent\"}",
                                "media": "Media",
                "mediaUrl": "Icon"
                        },
                        "fcm_options": {
                                "image": "Media"
                        }
                }
        }
}
```

TIP

uid — the internal send ID, unique for each message.
launch_id — the internal send ID, unique within the mailing.

iOS Apns

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "aps": {
                "badge": 0-999,
                "sound": "Sound",
                "mutable_content": 1,
                "alert": {
                        "title": "Title",
            "subtitle": "SubTitle",
                        "body": "Body"
                }
        },
        "click-url": "ClickURL",
        "icon": "Icon",
    "media": "Media",
        "hub_link": {
                "open": "URLOpenEvent",
                "ack": "URLDelivEvent"
        }
}
```

Appmetrica iOS

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "push_batch_request": {
                "group_id": 97737,
                "tag": "tag",
                "batch": [
                        {
                                "messages": {
                                        "iOS": {
                                                "silent": true/false,
                                                "content": {
                                                        "title": "Title",
                                                        "text": "Body",
                                                        "badge": 0-999,
                                                        "expiration": 0,
                                                        "data": "{\"hub_link\":
{\"ack\":\"URLDelivEvent\",\"open\":\"URLOpenEvent\"},\"icon\":\"Icon\",\"media\":\"Media\"}"
                                                },
                                                "open_action": {
                                                        "url": "URL"
                                                        "deeplink": "Deeplink"
                                                }
                                        }
                                },
                                "devices": [
                                        {
                                                "id_values": ["SubscriptionID"],
                                                "id_type": "ios_push_token"
                                        }
                                ]
                        }
                ]
        }
}
```

Appmetrica android

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "push_batch_request": {
                "group_id": 97737,
                "tag": "tag",
                "batch": [
                        {
                                "messages": {
                                        "android": {
                                                "silent": true/false,
                                                "content": {
                                                        "title": "Title",
                                                        "text": "Body",
                                                        "time_to_live": 0,
                                                        "image": "Icon",
                                                        "icon_background": "IconBackground",
                                                        "banner": "Banner",
                                                        "led_color": "LEDColor",
                                                        "data": "{\"hub_link\":{\"ack\":\"URLDelivEvent\",\"open\":\"URLOpenEvent\"}}"
                                                },
                                                "open_action": {
                                                        "url": "URL"
                                                        "deeplink": "Deeplink"
                                                }
                                        }
                                },
                                "devices": [
                                        {
                                                "id_values": ["SubscriptionID"],
                                                "id_type": "google_aid"
                                        }
                                ]
                        }
                ]
        }
}
```

HMS Android

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "message":{
                "android":{
                        "notification":{
                                "badges":{
                                        "set_num":2
                                },
                                "body":"Dear user, we offer you our new product, productName!",
                                "click_action":{
                                        "intent":"#Intent;compo=com.example/app2;S.W=U;end",
                                        "type":1
                                },
                                "color":"#cdc167",
                                "icon":"https://i.postimg.cc/2yVZyQMn/demo-Push-Icon.jpg",
                                "importance":"NORMAL",
                                "light_settings":{
                                        "color":{
                                                "alpha":1,
                                                "blue":0.70980394,
                                                "green":0.78039217,
                                                "red":0.5686275
                                        },
                                "light_off_duration":"1.5s",
                                "light_on_duration":"1.5s"
                                },
                                "sound":"tadaam.ogg",
                                "title":"Title of push message"
                        },
                        "params":{
                                "hub_link": "{\"open\": \"URLOpenEvent\", \"ack\": \"URLDelivEvent\"}",
                                "launch_id":"2_6j_4LxYQQ8VECT",
                                "uid":"w4LxYQQsznE1_2_6j_i_o___4.2Pzr5NNonCwUWQwvJ_4u71jH"
                        },
                        "ttl":"604800s"
                },
                "token":[
                        "AGIwonWCgyDXGEG-airg4cUbVxeq0RFDfTuDM2hPWRrxfMociyvLNmn-GWdd4I6ZDrXIee46IrWSjC-
cPDauP3ciS74bVi8wJ7T3PwHhvEY38VdYSu5zeHRXQzBRDgTnxA"
                ]
        }
}
```


TIP

launch_id — internal send ID, unique within the mailing.

HMS iOS

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
        "message":{
                "apns":{
                        "headers":{
                                "apns-expiration":"3600"
                        },
                        "hms_options":{
                                "target_user_type":1
                        },
                        "payload":{
                                "apple":"red",
                                "aps":{
                                        "alert":{
                                                "body":"Dear user, we offer you our new product, productName!",
                                                "subtitle":"sub title",
                                                "title":"Title of push message"
                                        },
                                        "badge":0,
                                        "mutable_content":1,
                                        "sound":"default"
                                },
                                "click-url":"",
                                "hub_link": "{\"open\": \"URLOpenEvent\", \"ack\": \"URLDelivEvent\"}",
                                "launch_id":"2_6j_4LxYKovmDvu",
                                "mediaUrl":"https://i.postimg.cc/2yVZyQMn/demo-Push-Icon.jpg",
                                "uid":"w4LxYKq9nBMc_2_6j_i_o___4.2Pzr5NNonCwUWQwvJ_aDfVX"
                        }
                },
                "token":[
                        "IQAAAACy0huaAQwLF6BO92GYzCAnfqZgaxhD1f7wLWGmOxQ6bFKCQYQQYtTi2-1hXHlsXAnCzLQdX3m4RTCCEI-KYu-
XcaBJYovDZANg2eRTvPv7qA"
                ]
        }
}
```


RuStore Android

INFO

It’s possible to edit the JSON object (on request).

Adding additional data to the `"data"` object is supported. This can be done for each sending individually in the Altcraft MP message settings.

```
{
  "validate_only": false,
  "message": {
    "token":  "{{.SubscriptionID}}",
    "data": {
      "uid": "{{.UID}}",
      "message": "{{.Body}}",
      "mediaUrl": "{{.Icon}}",
      "image":   "{{.Image}}",
      "launch_id": "{{.LaunchID}}",
      "hub_link": "{\"open\": \"{{.URLOpenEvent}}\", \"ack\": \"{{.URLDelivEvent}}\"}"
    },
    "android": {
      "priority": "{{.Priority}}",
      "notification": {
        "title": "{{.Title}}",
        "body": "{{.Body}}",
        "icon": "{{.Icon}}",
        "color": "{{.ImageBackgroundColor}}",
        "image": "{{.Image}}",
        "click_action": "{{.ClickURL}}",
        "sound": "{{.Sound}}",
        "channel_id": "{{.ChannelID}}",
        {{if ne .LEDColorHex ""}}
        "light_settings": {
          "color": {
            "red": {{.LEDColorSRGBA.R}},
            "green": {{.LEDColorSRGBA.G}},
            "blue": {{.LEDColorSRGBA.B}},
            "alpha": {{.LEDColorSRGBA.A}}
          },
          "light_on_duration": "{{.LEDOnDuration}}",
          "light_off_duration": "{{.LEDOffDuration}}"
        },
        {{end}}
        "notification_count": {{.Badges}}
      },
      "ttl": {{.TTL}}
    }
  }
}
```

TIP

uid — the internal send ID, unique for each message.
launch_id — the internal send ID, unique within the mailing.

Processing and adding a subscription

Once the app received a push token, the app must transfer it to Altcraft MP in order to create a subscription. The API request is used for it.

With this request, you can:

- **Add new subscriptions.**
- **Change the status of existing subscriptions.**

The profile search mode can be set using the matching field in the request (see specification below). If the profile is not found, a new one will be created.

Moreover, if several databases are specified in the resource, then all of them will be processed separately. A request to such a resource will create and update profiles in several databases at once. An example of such a case can be seen below.
If there are no databases in the resource, then an error will be returned.

URL

Method: POST

`https://<cookiesaver_host>/push_subscribe`

Request parameters

Parameter	Type	Example	Required	Description
id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**No**	**Profile ID**
resource_token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**Unique resource ID.**
db_ids	**[ ] int**	**[2,3]**	**No**	**Resource database filter.**
matching	**string**	**“custom”**	**No**	**Profile search mode.** **If the parameter is not passed or is empty, profile search by ID will be used. If ID is also not passed, search by push subscriptions will be used.** **For the `push_subscribe` method, this parameter can only accept the following values:** • **custom** • **profile_id**
subscriptions	**JSON array SubscriptionObject**	`[{ “provider” : “android-firebase”, “subscription_id” : “%subscriptionid%” }]`	**Yes**	**Push subscriptions that should be added to profile or should be updated.** **“subscription_id” is unique for each device, Firebase provides it.**

SubscriptionObject

Parameter	Type	Example	Required	Description
provider	**string**	**“android-firebase”**	**Yes**	**Altcraft MP Provider ID.** **The available values:** • **android-firebase** • **ios-firebase** • **ios-apns** • **yandex-Appmetrica-device-id-ios** • **yandex-Appmetrica-device-id-android** • **yandex-Appmetrica-ios-ifa** • **yandex-Appmetrica-google-aid** • **yandex-Appmetrica-android-token** • **yandex-Appmetrica-ios-token** • **android-huawei** • **ios-huawei** **Each token type in AppMetrica has its own provider ID.**
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**	**Yes**	**Subscription push token**
unsubscribe	**bool**	**false**	**No**	**Unsubscribe flag. Dismiss subscription if its value — true.**

Response parameters

Parameter	Type	Description
data	**JSON array DataObject**	**Information about the created subscriptions.** **The array can contain several objects in the following cases:** • **Several SubscriptionObject were passed in the request.** • **ID field was not used in the request, and the Altcraft MP resource uses several databases. In this case, a new profile with subscriptions will be added for each database from the resource.**
error	**int**	**Error code**
error_text	**string**	**Error text**

DataObject

Parameter	Type	Description	
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**	**Subscription push token**
unsubscribe	**bool**	**false**	**Flag whether the subscription was added or canceled**
profile_id	**string**	**“5fb62e815ae47a4aa04dfadd”**	**ID of the profile (transferred or newly created) for which the subscription was added.**

Parameter	Type	Description	
provider	**string**	**“android-firebase”**	**Altcraft MP Provider ID.** **The available values:** • **android-firebase** • **ios-firebase** • **ios-apns** • **yandex-Appmetrica-device-id-ios** • **yandex-Appmetrica-device-id-android** • **yandex-Appmetrica-ios-ifa** • **yandex-Appmetrica-google-aid** • **yandex-Appmetrica-android-token** • **yandex-Appmetrica-ios-token** • **android-huawei** • **ios-huawei** **Each token type in AppMetrica has its own provider ID.**
db_id	**int**	**2**	**Altcraft MP Database ID. The profile is stored in this database.**
resource_id	**int**	**1**	**Altcraft MP resource ID.**
subscription_hash_id	**string**	**“92807e89”**	**Subscription hash**
status	**string**	**“error”**	**Operation status** • **ok** • **error**
description	**string**	**“Duplicate push subscription”**	**Error description**

Examples

Request with matching by system profile ID

Request

```
{
  "id": "59db33d269d426509e9c4bf9",
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bf9",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

Request with matching by a custom profile field

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfc",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdfffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

Request with matching by push subscriptions

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfa",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdfffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

Request to a resource with several databases

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "provider": "android-firebase",
      "resource_id": 22,
      "status": "ok",
      "subscription_hash_id": "340ca49c",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    },
    {
      "db_id": 2,
      "description": "matching error: unknown field",
      "resource_id": 22,
      "status": "error"
    }
  ]
}
```

Request with matching by a custom profile field in a certain database

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "db_ids": [1],
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 1,
      "description": "",
      "profile_id": "637f1a284fda7c5c08be62f",
      "provider": "android-firebase",
      "resource_id": 31,
      "status": "ok",
      "subscription_hash_id": "f8af057de",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    }
  ]
}
```


Request did not update or add any profile subscriptions

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "resource_id": 22,
      "status": "ok"
    }
  ]
}
```

Event registration

There are two ways to register delivery events and open push messages in Altcraft MP:

1. **Directly from the app.**
2. **Through a third-party service.**

Registering events directly from the app

By default, when sending a push message, the `"hub_link"` object is sent in the payload:

```
{
     "hub_link" : {
         "ack" :  "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/2ab9162159..." ,
         "open" :  "http://your.domain.com/api/v1.1/pushes/crypto_post_events?/555fbe0b9..."
     }
}
```

The application must execute a GET request using the rack and open links to register the delivery and open the push, respectively.

Registering events through a third-party service

CAUTION

This method requires the presence of Extra UID in the user data of the push message (configurable on request).

The application informs the third-party service about the event and at the same time transfers Extra UID from the push.

The gateway must send this event to Altcraft MP via API.

URL

Method: POST

`https://example.com/api/v1.1/pushes/post_events`

Request parameters

Parameter	Type	Example	Required	Description
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**API token**
events	**int**	**1 — delivered** **2 — opened** **3 — delivered and opened**	**Yes**	**Event**
uid	**string**	**“uid”**	**Yes**	**Comes along with a push message from Altcraft in the Data payload.**

Request example

```
{
     "token" :  "abcdefghijklmnqrstuvwxyz" ,
     "events" : 1,
     "uid" :  "e|2_4S_4HFn8RDhhmA|w4HFn8RPDFUi_2_4S_S_3___6.2NvyN8Hb67wmzJshM_6BuFTt|ios-apns"
}
```

Response example

```
{
     "error" : 0,
     "error_text" :  "Successful operation"
}
```

Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**

Web push: settings & implementation

TIP

For application push see this article!

	Web push notifications (also known as browser push notifications) are small pop-ups in the browser. These messages are contextual, timely, personalized, and best used to engage, re-engage, and retain website visitors. You can send such notifications to your clients' desktop and mobile browsers. When a user subscribes to Web push notifications, every time you send a push notification it is displayed on the user's device at the top of the screen. Moreover, it is displayed even if the window is inactive. </td> </tr>

TIP

What is the difference between Web push and Mobile push?

Web Push notifications are sent to a visitor's device from a website via a browser. It can be both mobile and desktop browsers. At the same time, Mobile Push notifications come from mobile applications.

A push notification can contain:

- **A title that is similar to email subject**
- **Notification text**
- **An image or an icon**
- **A clickable link**

Altcraft Marketing uses different push platforms for the supported browsers:

Service	Which browsers use this service to send Web push notifications?
Google Firebase (ex. Google Cloud Messaging)	**Google Chrome, Mozilla Firefox, Opera, Яндекс.Браузер, Samsung Internet Browser**
Apple Push Notification Service	**Apple Safari (incl. PWA-apps)**
Mozilla Services	**Mozilla Firefox**

TIP

Please note that sending notifications via Mozilla Firefox is possible both directly - Mozilla Services (SDK is built into the browser), and through the Google Firebase service.

NOTE

Google Cloud Messaging is no longer available due to migration to Firebase Cloud Messaging!

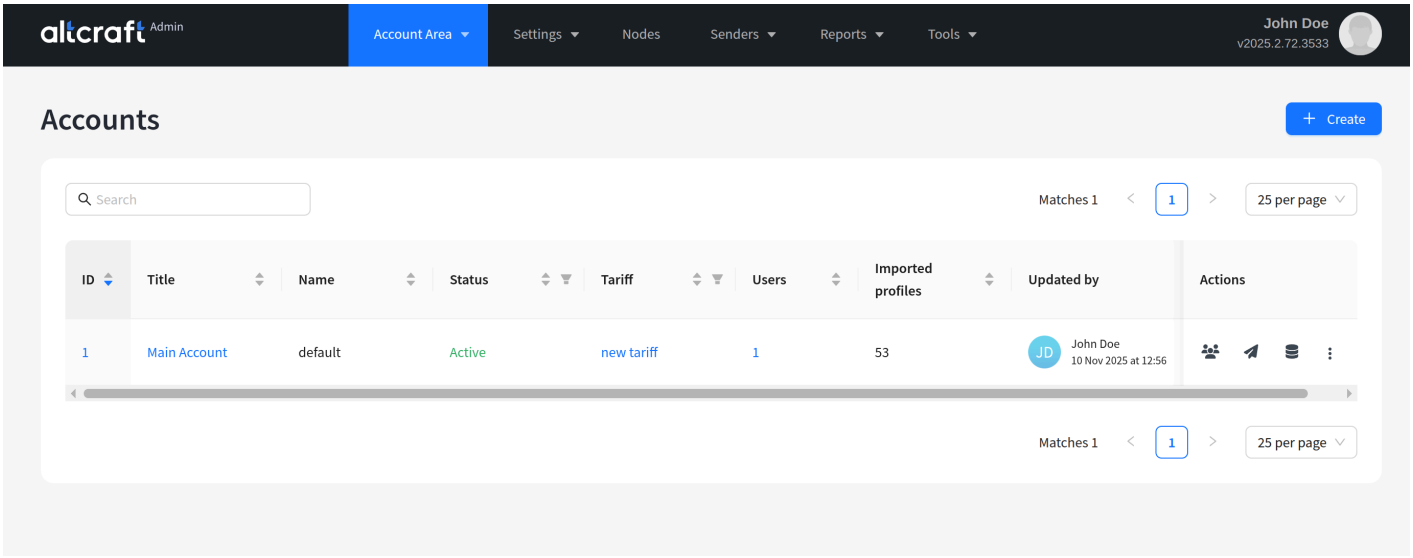
How to set up and enable Web Push notifications?

Preliminary Settings

Technical Settings

Make sure your website is running over the HTTPS protocol.

Additionally, web pixels must be configured in the Altcraft Platform. Go to Admin Panel → Account Management. Click to your account:



Activate push notifications and pixel for the accounts from which sending is planned:

Configuring a Resource in the Platform

> **TIP**
>
> **Each website must have a separate resource created.**

1. In the Altcraft platform, create or select Databases where push notification subscribers will be stored.

2. Select a Resource for Web Push subscriptions or create a new one.

3. Enter the Name and Description of the resource.

4. In the Website, form, or application URL field, specify the address of the site where push subscription occurs and from which notifications will be sent. This site is later added to the Toolkit files, so if an incorrect address is specified, push notifications will not work.

5. In the Default Tracking Domain field, specify the domain where Altcraft tracking is running. This domain will process your customers' clicks on links in messages.

General settings

Resource name: *

Resource for push notifications

Resource description:

Set group:

Set resource sending policy:

Undefined

Databases: *

6 - Test database

Website, form or application URL: *

https://polzar.github.io

Default tracking domain: *

click.polina.zakaryan.dev.altkraft.com

Additional URLs

Increase priority for new subscriptions

Connecting Web Push to a Website

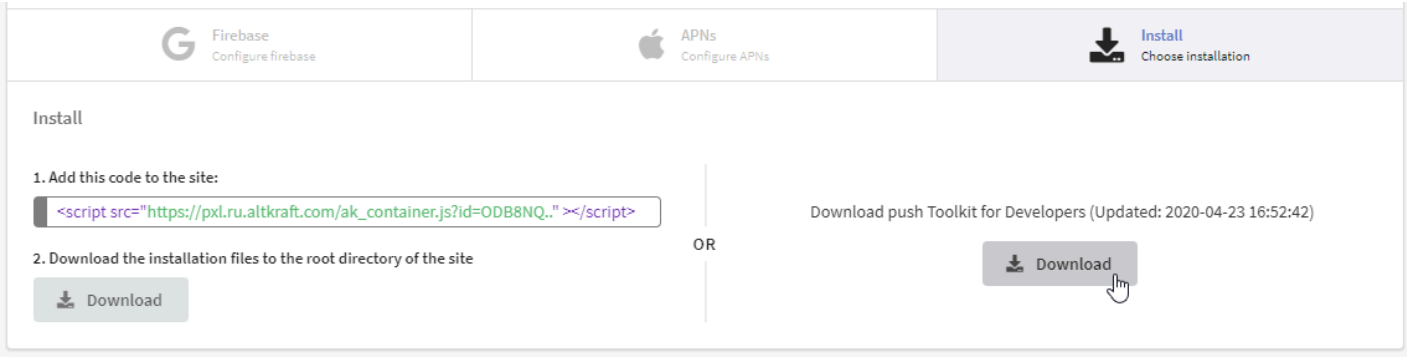
Downloading the Toolkit

Once the necessary resource settings are filled in, you can save it and download the Toolkit, parts of which are then placed on your website. The Toolkit is a set of files designed to enable Web Push notifications on your site. It helps configure user subscriptions and send notifications via browsers.

INFO

The Toolkit itself does not trigger the subscription prompt; it only contains the necessary files and scripts for working with notifications. To show the subscription prompt to users, you need to add code to your website that calls the corresponding function from `ak-push.js`.

The Toolkit will appear in the Install section, replacing the warning message “Save the resource to download the toolkit”:



There are two ways to install the required files on your website:

1. **Adding code to the site and uploading installation files to the root directory of the site;**
2. **Downloading the developer toolkit.**

Method 1: Adding Code to the Site and Uploading Installation Files to the Root Directory

The steps for this method are shown on the left side of the interface. Add the specified code to the website, then save and upload the installation files to the root directory of your site:

- **manifest.json — project data for sending push notifications via Google Chrome;**
- **service-worker.js — a service worker that is registered in users’ browsers and receives push messages.**

The service-worker.js file is dynamic. It automatically updates each time you make changes to the resource settings.

```
importScripts('https://pxl.test.com/service-worker.js?id=MXw3');
```

Method 2: Downloading the Developer Toolkit

The steps for this method are shown on the right side of the interface. Click the “Download” button to download the Toolkit archive. Extract the archive and place the Toolkit files in the root directory of your site.

The Toolkit consists of four files:

- **README.txt — installation instructions for the Toolkit;**
- **ak-push.js — a script that displays the subscription prompt to users;**
- **manifest.json — project data for sending push notifications via Google Chrome;**
- **service-worker.js — a service worker that is registered in users’ browsers and receives push messages.**

Here, the service-worker.js and ak-push.js files are static. When using this method, you must re-download the Toolkit and update the files on the site after making changes to the resource settings.

TIP

When using the developer toolkit, you can change the path to the service-worker.js file. To do this, open the ak-push.js file, find the swPath property, and edit it to specify the correct path to the file.

Adding Scripts to the Website

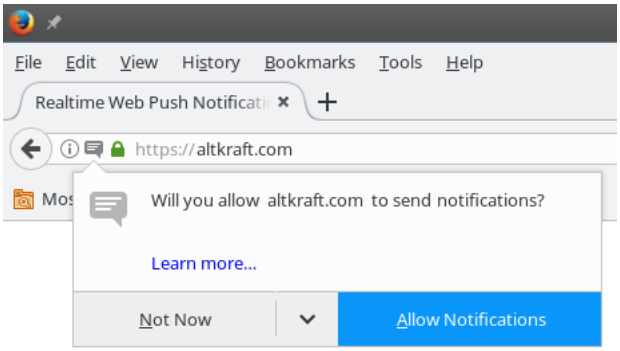
Once the Toolkit is uploaded to the site, add the code that will display the subscription prompt when needed.

Below is an example code snippet for a Subscribe button. This code initializes the library and starts the subscription process:

```
<button id="init_sub">Subscribe to notifications</button> // Subscription button

// ..Main part of the code..

<script>
        document.getElementById('init_sub').addEventListener('click', function() { // On button click..
        try {
                const akPush = new AKPush();
                akPush.initSubscription() // ..show the subscription form
                }
        catch (e) {
                console.log(e); // ..or log the error to the browser console
                }
        });
</script>
```



TIP

Through the Altcraft platform, you can create a custom popup prompting users to subscribe. Learn more in this article.

The code described above works as follows:

- **The user clicks the button;**
- **The `initSubscription` method is called;**
- **A browser system window pops up, asking the user to subscribe to notifications for this site.**

Users can customize the logic and decide how visitors will reach the system subscription window. The styles of the system browser window requesting push notification permissions cannot be changed.

Once everything is set up, users can visit the site and subscribe to push notifications. When confirmed, a Service Worker is installed in the client's browser, allowing messages to be displayed even when the browser window is minimized.

Note that a subscription to the same resource will be created separately for each browser (or account) the user operates. The identifier for the subscription is the browser token or account token. This means that a single profile may have multiple active push subscriptions if the user subscribed from different browsers.

When subscribing to notifications, the script can also transmit additional user data or register event information. Learn more about configuring this in this article.

Transferring Data to the Platform

When a user grants permission to receive notifications, the platform automatically creates a profile for them. In addition to profile creation, other additional data can also be transmitted. In this article, we will explain how to configure a script to collect information about the user and their actions.

TIP

Setting up Web Push on the Altcraft platform requires using Web Push SDK methods. You can learn more about SDK methods in this article. Instructions for installing the SDK on your site can be found here.

Importing Client Profiles When Subscribing to Web Push Notifications

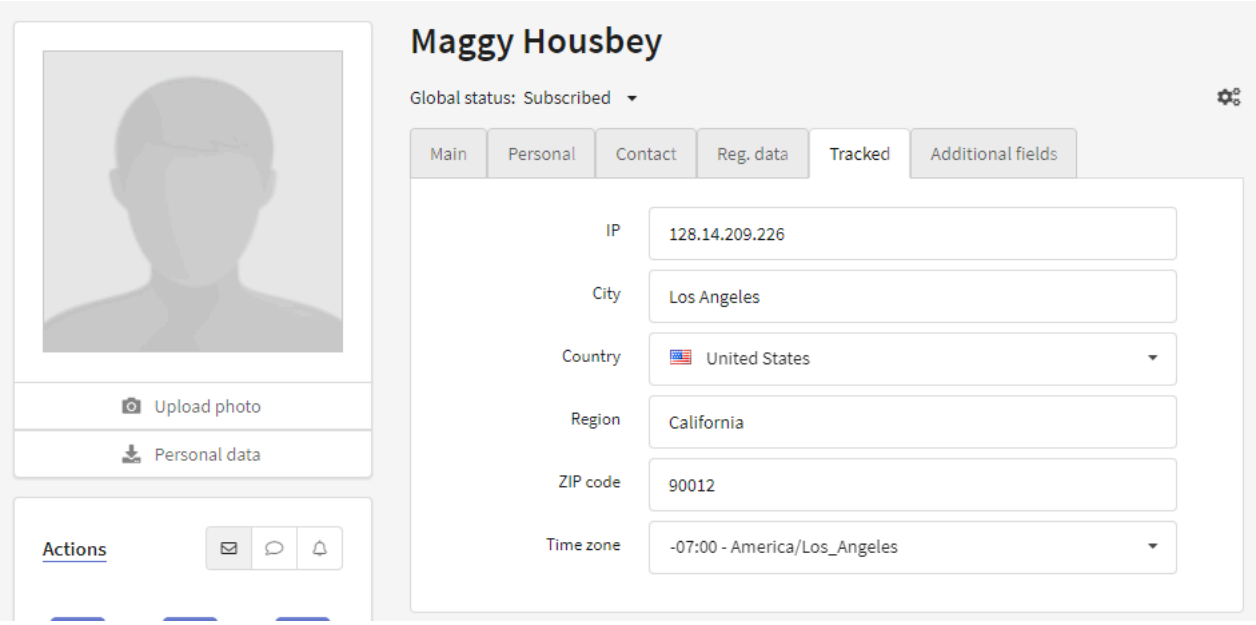
When a user subscribes to Web Push notifications, a new profile will be added to the Databases and the Resource where the Toolkit embedded on the site was generated.

INFO

If a database profile has previously opened your email newsletter from the same browser in which they are now subscribing to push, the subscription will be added to that profile. In this case, a new profile will not be created.

When a subscriber clicks on a push notification (following a link), the Altcraft platform collects profile data:

- **Country, region, time zone, city, postal code, and IP address from which the push subscription occurred;**
- **Browser and operating system of the subscriber.**



If the token becomes outdated or the user resubscribes to push, the platform will automatically update their subscription and issue a new token. This prevents duplicate profiles from being created in the database.

NOTE

Mobile browsers also support Service Worker and Web Push notifications.

Transferring Client Data When Importing a Profile via Push Channel

At the time of push notification subscription registration, any known user data can be recorded into the new user's profile. For example, the client's first and last name, date of birth, pages viewed, and other business-related information. At the same time, you can check if an existing subscriber already has the same data.

Example with email matching in the resource and adding the user's first and last name:

```
const akPush = new AKPush();
akPush.initSubscription(
        {
                email: 'example@email.com'
        }, // Matching of profile by email. If no match is found, it will be recorded in the profile.

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        } // Record first and last name in the subscriber's profile
)
```

If multiple databases are linked to a resource, you can specify the ID of the database where the profile should be imported. Example with email matching in database `id=1`:

```
const akPush = new AKPush();
akPush.initSubscription(
        {
                db: 1
                email: 'example@email.com'
        }, // Matching of profile by email. If no match is found, it will be recorded in the profile.

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        } // Record first and last name in the subscriber's profile
)
```

Example with email matching in databases `id=15` and `id=20`:

```
const akPush = new AKPush();
akPush.initSubscription(
        {
                db: [15,20]
                email: 'example@email.com'
        }, // Matching of profile by email. If no match is found, it will be recorded in the profile.

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        } // Record first and last name in the subscriber's profile
)
```

TIP

The list of profile fields is provided in this article.

Updating Profile Data and Suspending Subscriptions

CAUTION

akPush.updateSubscription will return an error if an up-to-date token has not been obtained via akPush.initSubscription, or if it has been removed from the cache using akPush.removeToken.

A user subscribed to push notifications in a browser can change their status. For example, they may log into their personal account and receive a different matching profile. In this case, the current push subscription must be disabled on the previous profile and enabled on the new one. To achieve this, use the `akPush.updateSubscription` method:

```
const akPush = new AKPush();
akPush.updateSubscription(
        {
                db: [15,20]
                email: 'example@email.com'
        },

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        }, // Record first and last name in the subscriber's profile
        false // Flag for inexclusive, determines whether to pause all subscriptions to the resource with the current browser token for
other profiles (default)
)
```

Web Push Subscription Creation and Update Events

Platform users can subscribe to events for push subscription creation and updates (subscriber token changes). When such an event occurs, subscription data can be sent to an external server using additional code.

To subscribe to events, add the following code to your site’s pages:

```
AKPush.events.on('subscribe', (data) => { // Subscribe to the 'subscribe' event
   console.log(data);
});
AKPush.events.on('update_subscription', (data) => { // Subscribe to the 'update_subscription' event
   console.log(data);
});
```

This code will log the subscriber’s token, as well as browser and push provider information.

Web Push SDK Methods

TIP

For more details on setting up Web Push mailings on the platform and installing the SDK on your website, refer to this section.

This article describes the Web Push SDK methods and their possible parameters. Each method is also accompanied by a usage example.

initSubscription

This method initializes a subscription. It is used to record client data in a profile and perform a search among existing profiles.

Parameters

Parameter	Type	Required	Description
match	**object**	**Yes**	**Defines [profile matching](/en/user-guide/profiles-and-databases/matching). If no profile database is provided for the search, the platform will search within the databases linked to the resource.**
update	**object**	**No**	**Contains data to be added to a new or existing profile.**

Usage Example

```
const akPush = new AKPush();
akPush.initSubscription(
        {
                db: 1
                email: 'example@email.com'
        }, // Searching for a profile by email. If no match is found, it will be recorded in the profile.

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        } // Recording the subscriber's first and last name in the profile
)
```

updateSubscription

This method updates profile data and modifies the status of push subscriptions.

Parameters

Parameter	Type	Required	Description
match	**object**	**Yes**	**Defines [profile matching](/en/user-guide/profiles-and-databases/matching). If no profile database is provided for the search, the platform will search within the databases linked to the resource.**
update	**object**	**No**	**Contains data to be added to a new or existing profile. If you want to pass the **inexclusive** parameter without updating the profile data, you must specify an empty `{}` object in **update**.**
inexclusive	**boolean**	**No**	**If the `inexclusive` flag is not provided or set to `false`: 1. If no database filter is specified, each database of the resource will have only one profile with one subscription for the current token with the status "Subscribed". 2. If a filter is specified, each specified database will have only one profile with one subscription for the current token with the "Subscribed" status. In other databases, all matching subscriptions to the resource for all profiles will be suspended. If the `inexclusive` flag is set to `true`, the call is equivalent to `AKPush.initSubscription`, meaning the profile data will be updated and the subscription status changed to "Subscribed".**

Usage Example

```
const akPush = new AKPush();
akPush.updateSubscription(
        {
                db: [15,20]
                email: 'example@email.com'
        },

        {
                _fname: 'Anatoly',
                _lname: 'Wasserman'
        }, // Recording the subscriber's first and last name in the profile
        false // Whether to suspend all subscriptions to the resource with the current browser token for other profiles (default)
)
```

AKPush.events.on

This method subscribes to events related to push notifications.

Parameters

Parameter	Type	Required	Description
subscribe update_subscription suspend_subscription	**string**	**Yes**	**Name of the triggered event.**

callback	**function**	**Yes**	**Function that will be called when the event occurs. For `subscribe`, the returned parameters will be:**
			{
			browser: "<subscriber's browser>"
			provider: "<push provider>"
			token: "<subscriber's token>"
			}
			For `update_subscription` and `suspend_subscription`, the returned parameters will be:
			{
			browser: "<subscriber's browser>"
			provider: "<push provider>"
			token: "<subscriber's token>"
			tokenCached: "<cached subscriber's token>"
			}

Usage Example

```
AKPush.events.on('subscribe', (data) => { // Subscribe to the subscribe event
   console.log(data);
});
AKPush.events.on('update_subscription', (data) => { // Subscribe to the update_subscription event
   console.log(data);
});
```

suspendSubscription

This method suspends a user's push notification subscription. The subscription can be resumed using the same method.

Parameters

Parameter	Type	Required	Description
match	**object**	**Yes**	**Defines [profile matching](/en/user-guide/profiles-and-databases/matching). If no profile database is provided for the search, the platform will search within the databases linked to the resource.**
update	**object**	**No**	**Contains data to be added to a new or existing profile.**

Usage Example

```
AKPush.suspendSubscription(
        {
                db: 1
                email: 'example@email.com'
        },
        {
                _status: 'inactive'
        } // Updating the subscriber's profile status
)
```

AKPush.removeToken

This method removes a device token from the cache. It is used, for example, when a user changes on the device. No parameters are required for this method.

Usage Example

```
AKPush.removeToken()
```

First Web Push Mailing

	Web push notifications (also known as browser push notifications) are small pop-ups in the browser. These messages are contextual, timely, personalized, and best used to engage, re-engage, and retain website visitors. You can send such notifications to your clients' desktop and mobile browsers. When a user subscribes to Web push notifications, every time you send a push notification it is displayed on the user's device at the top of the screen. Moreover, it is displayed even if the window is inactive. </td> </tr>

This tutorial will help you to create and launch your Push mailing in the Altcraft Platform from scratch.

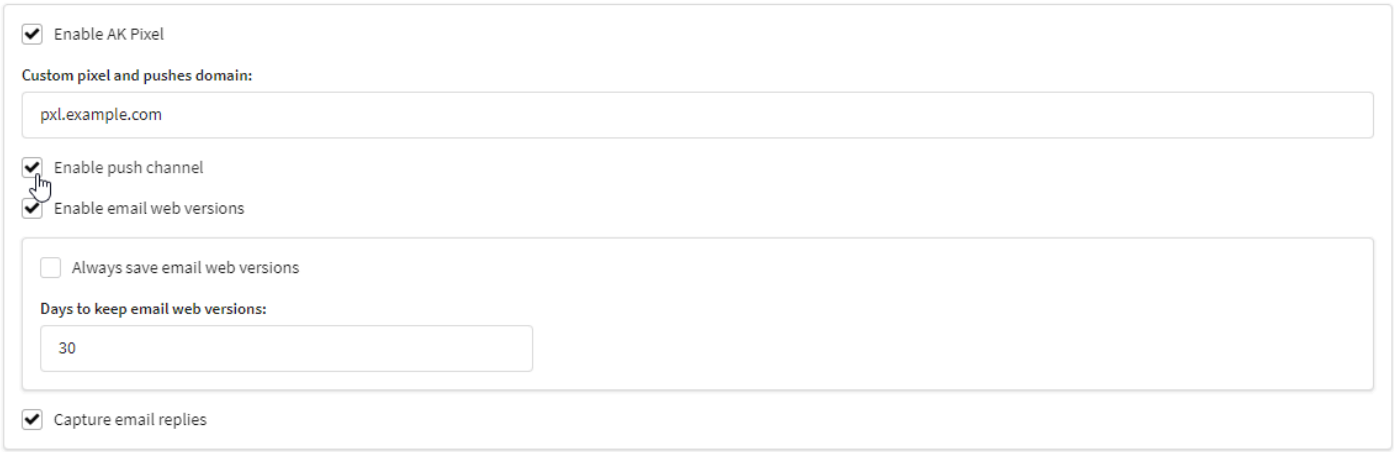
> TIP
>
> **Push notifications are much more profitable than SMS, because sending them practically costs nothing.**

Step 1: Configure a Domain to Work with Push

> CAUTION
>
> **If you use an Altcraft Marketing cloud account, please give the information to the support team at support@altcraft.com. They will help you to connect the integration.**
>
> **If the Altcraft Marketing platform is installed on your equipment (On-Premise), contact your platform administrators. Or just contact our support team!**

Make sure your website is running over HTTPS. In addition, web pixels must be configured in the Altcraft Marketing system.

Go to Administration Panel —> Account Management.



Step 2: Subscribe customers to the Push channel

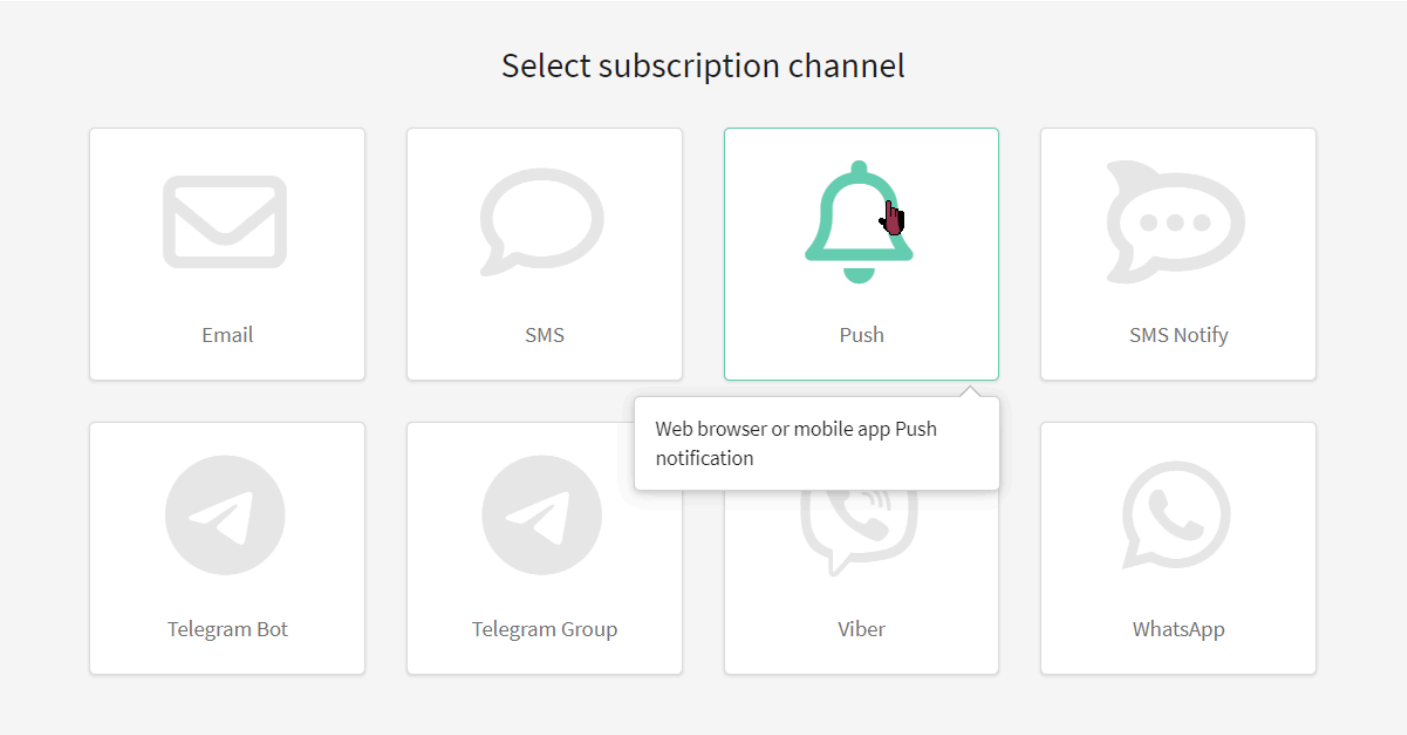
Profile databases and subscription resources

Altcraft Marketing mailings are received by your subscribers. Subscriber profiles are stored in profiles databases.

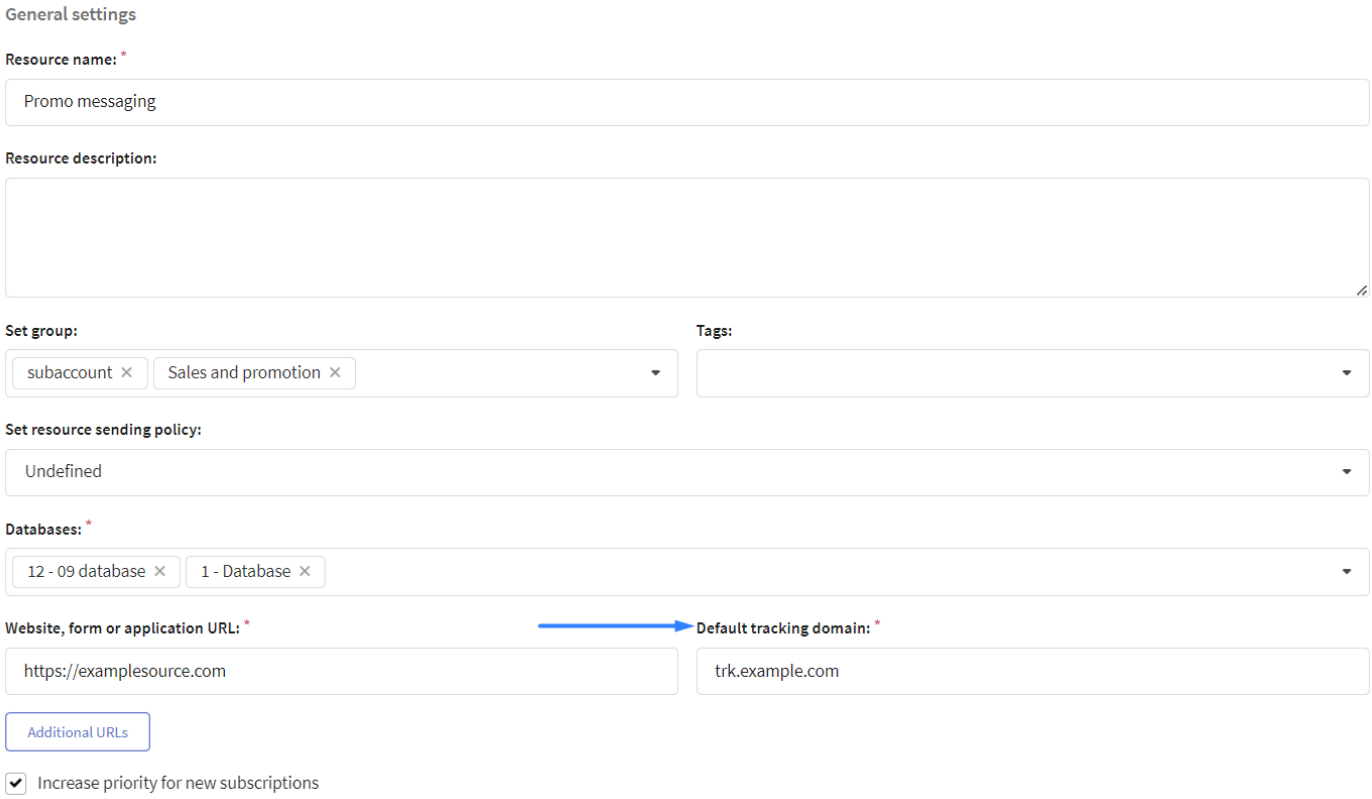
When you start your Push mailing, you use profile subscriptions to resources. This way, you can send notifications to different devices or to different browsers. Each client profile can be subscribed to one or more resources.

What can be a resource? A specific marketing area, a site, an application, or the type of content.

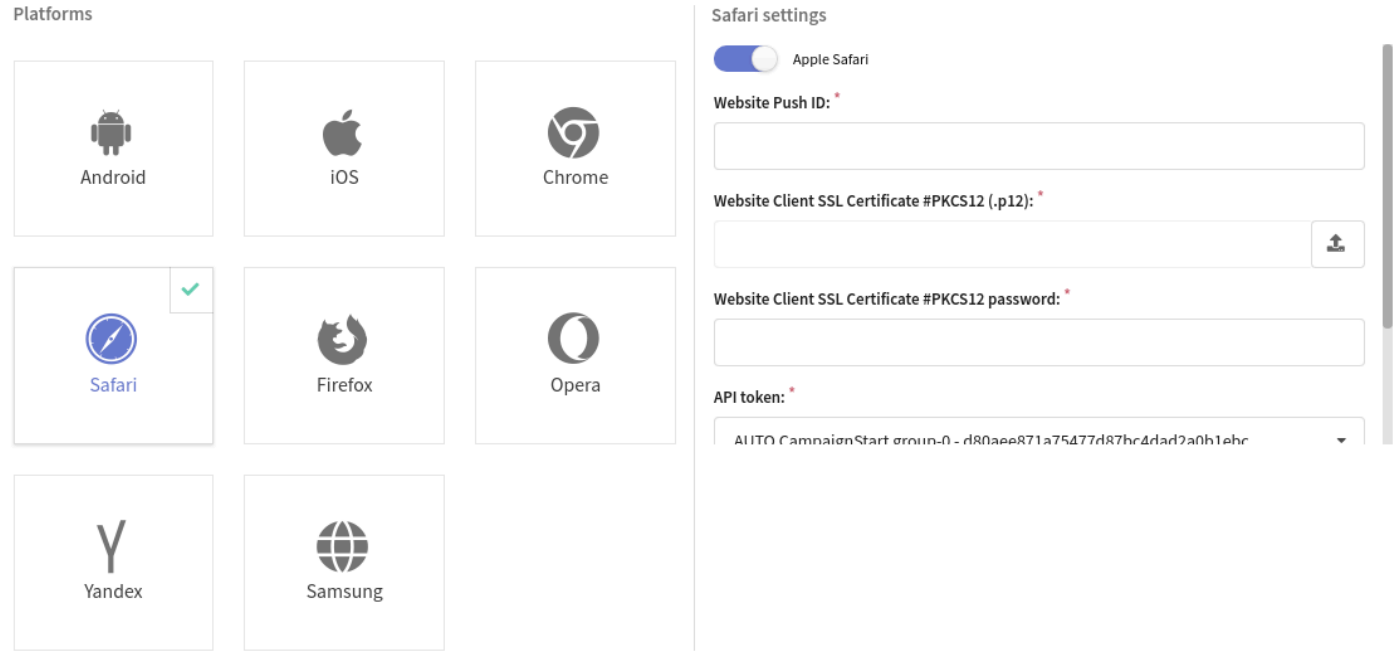
Select the section “Resources” in the main menu, then create a new resource or select an existing one. Select the Push channel for your new resource:



In the general settings, add the necessary databases. If it's necessary, assign a sending policy and specify the Default tracking domain. This domain will process your customers' clicks on links in messages:



In Push settings, select the browsers to which you are going to send notifications to clients and set them up:



INFO

Altcraft Marketing uses different push platforms for the supported browsers:

Service	Which browsers use this service to send Web push notifications?
Google Firebase	**Google Chrome, Mozilla Firefox, Opera, Yandex.Browser, Samsung Internet Browser**
Apple Push Notification Service	**Apple Safari**
Mozilla Services	**Mozilla Firefox**

Please note that sending notifications via Mozilla Firefox is possible both directly - Mozilla Services (SDK is built into the browser), and through the Google Firebase service.

Don't know how to configure each platform? Take a look at this tutorial .

Customer profiles and subscription to the resource

So, we've just configured the profiles database and subscription resource. Now it's time to import customer data into the platform. For each client, Altcraft Marketing creates a single digital profile that includes personal data, history of actions and subscriptions to your resources.

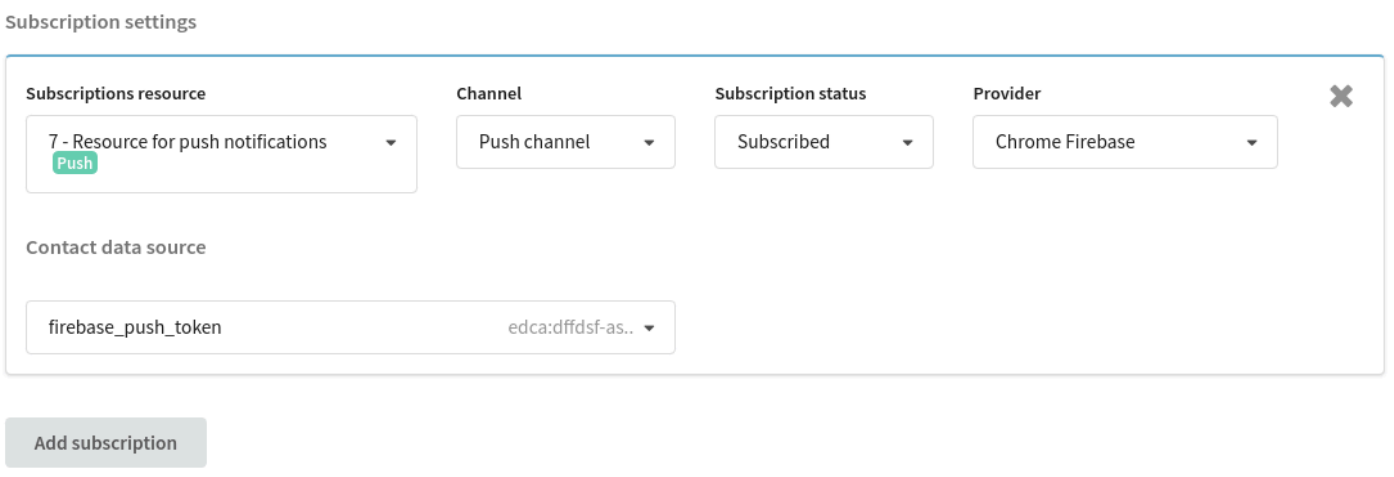
Existing subscriber database

If you already have the database full of tokens for your customers, you can upload customer data to the platform and use it for mailings: one at a time, in a list, or from a table file. Different import methods are described in this article.

You can also set up automatic import of client profiles:

- **on schedule from your SQL database**
- **through the use of the integration with platform API**

When you import data, sign clients to the previously prepared resource. To do this, in the Subscription settings section, select Push as a communication channel, select a notification provider, in the Contact data source field, specify the column with device tokens:



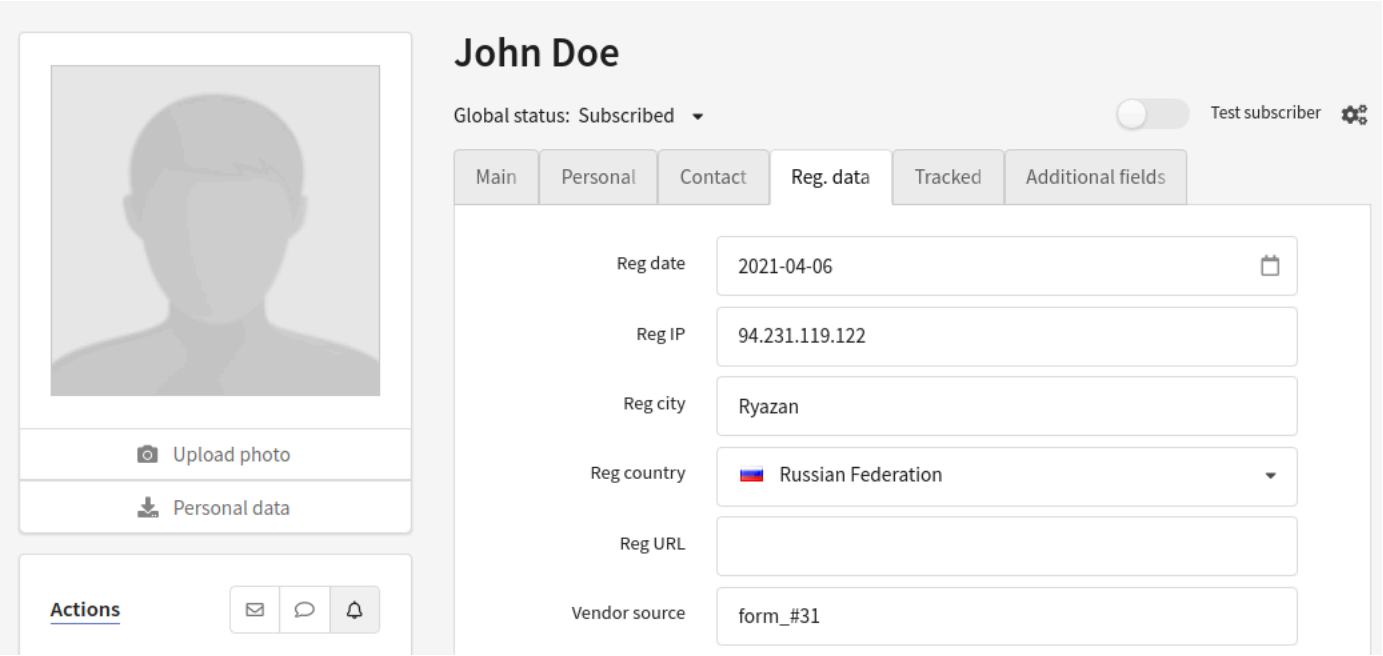
TIP

If you already have a profiles database, add them a subscription to a new resource. There are two options:

- **if customers have subscriptions to another resource with up-to-date email addresses — use bulk profile update**
- **if you need to upload to the platform SMS phones for subscription, use the profile import in the "update only" mode**

New push subscribers of your website

Altcraft Marketing imports the profiles of your website visitors when they allow push notifications. At the same time, information about the subscriber's device, IP address and geolocation determined by it are recorded in the profile:



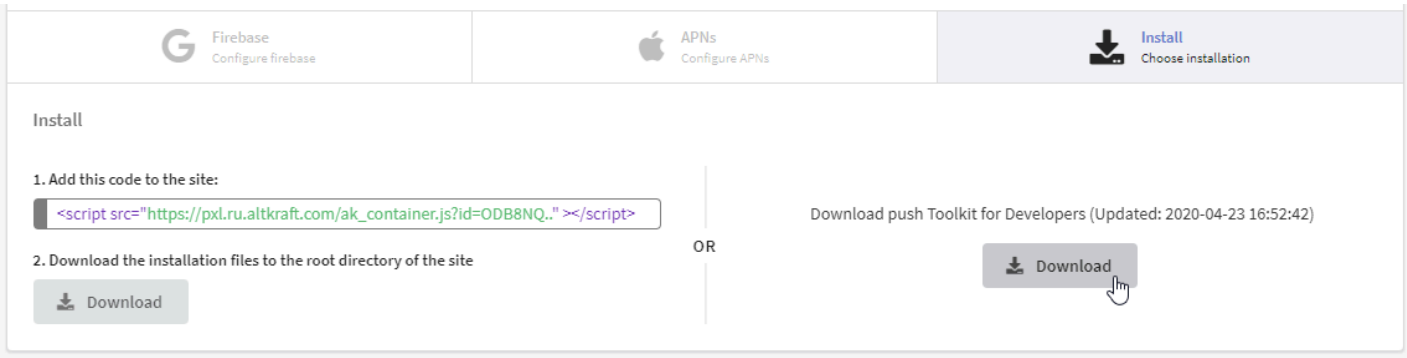
Go to the next step to configure the import of new push subscribers.

Step 3: Add Push Toolkit to your website

After you configure push notification channels, save the resource. This way, you will get access to push implementation toolkit. You can find the toolkit on the tab "Install".

TIP

Download it and hand over the toolkit to the developers of your website. They'll have to add it to certain web-pages. In addition, hand them the link to this article or the article on Web Push configuration.



There are two ways to install the necessary files on your website:

1. Add the code to the site & Upload the installation files to the root directory

If you're going to use this method, read the instructions in the left side. Add the code to the site, save changes and upload the installation files to the root directory of the site:

- ****manifest.json** — project information for Google Chrome;**
- ****service-worker.js** — service worker, service worker, a users' browser extension to receive push notifications.**

Here `service-worker.js` file is dynamic. It is automatically updated every time you make changes to the resource settings.

```
importScripts(
  "https://pxl.polina-zakaryan.dev.altkraft.com/service-worker.js?id=MXw3"
);
```

2. Download push Toolkit for Developers

If you're going to use this method, read the instructions in the right side. Click the "Download" button. The Toolkit will be downloaded as a .zip file. Unpack the archive and put Toolkit files to the root directory of your site.

This toolkit consists of four files:

- **README.txt — installation instructions for the Toolkit**
- **ak-push.js — web push opt in script for your website**
- **manifest.json — project information for Google Chrome**
- **service-worker.js — service worker, a users' browser extension to receive push notifications**

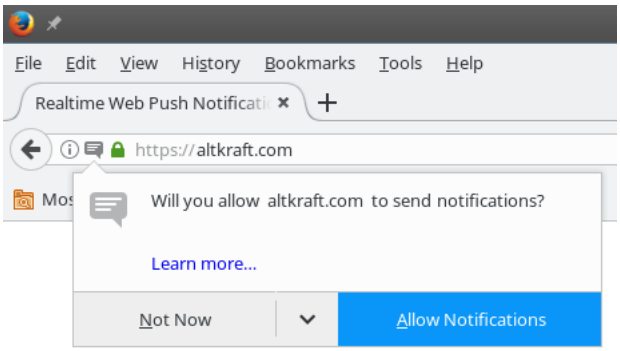
Here `service-worker.js` and `ak-push.js` files are static. It means that every time you change the resource settings you need to re-download the Toolkit and update the files on the site.

When the Toolkit is uploaded to the site, add the code, that will be responsible for showing a window for subscribing to the required action.

Example for the Subscribe button:

```
<button id="init_sub">Subscribe to updates!</button> // The button

<script>
        document.getElementById('init_sub').addEventListener('click', function() { // On button click..
        try {
                const akPush = new AKPush();
                akPush.initSubscription() // ..show the subscription form
                }
        catch (e) {
                console.log(e); // ..or log the error to the browser console
                }
        });
</script>
```



Once everything is set up, you can go to the site and subscribe to push notifications. When the client confirm the subscription, Service worker is installed in his browser. Service worker can display messages even when the browser window is minimized.

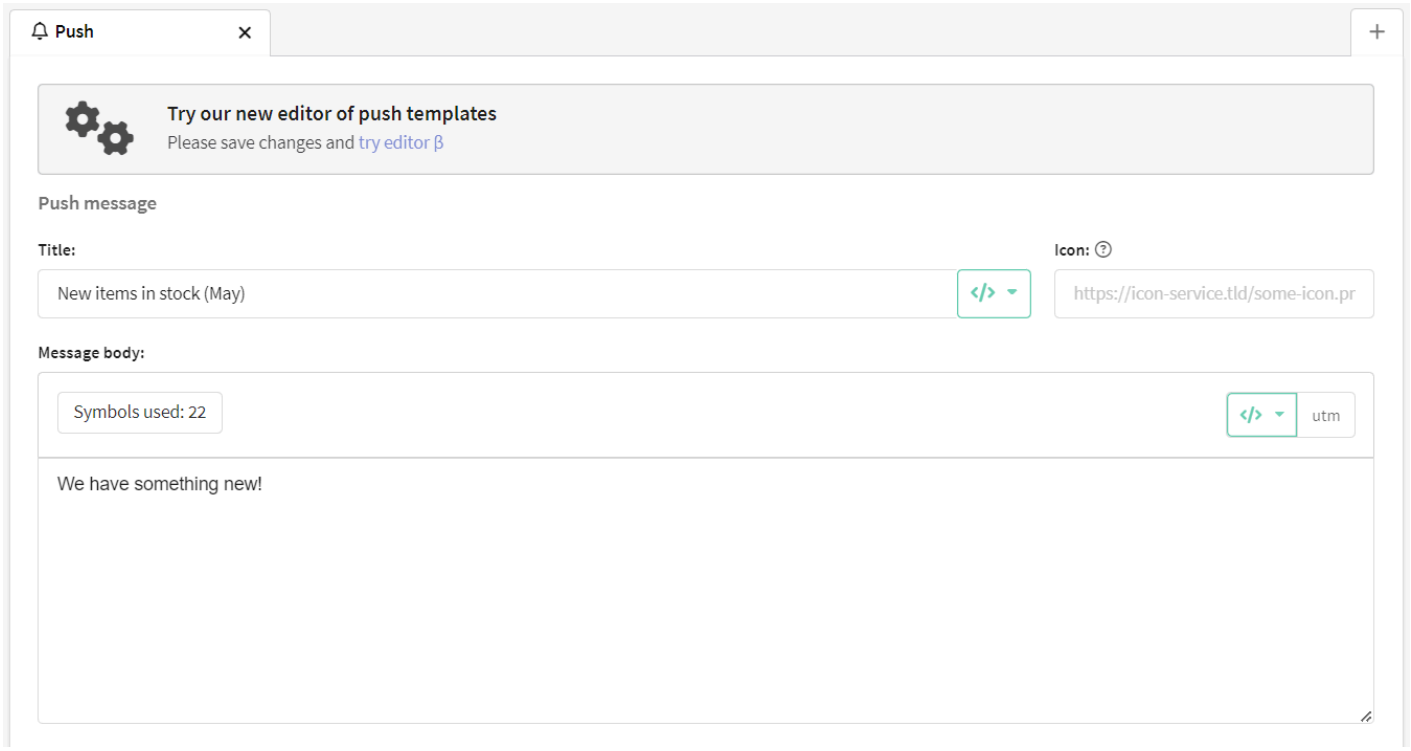
Step 4: Prepare content for sending

The content for your Push mailings is created in Message Templates. The templates contain content for all available channels. When you create a new template, select the Push channel or add it for an existing one.

The template of the push notification can contain:

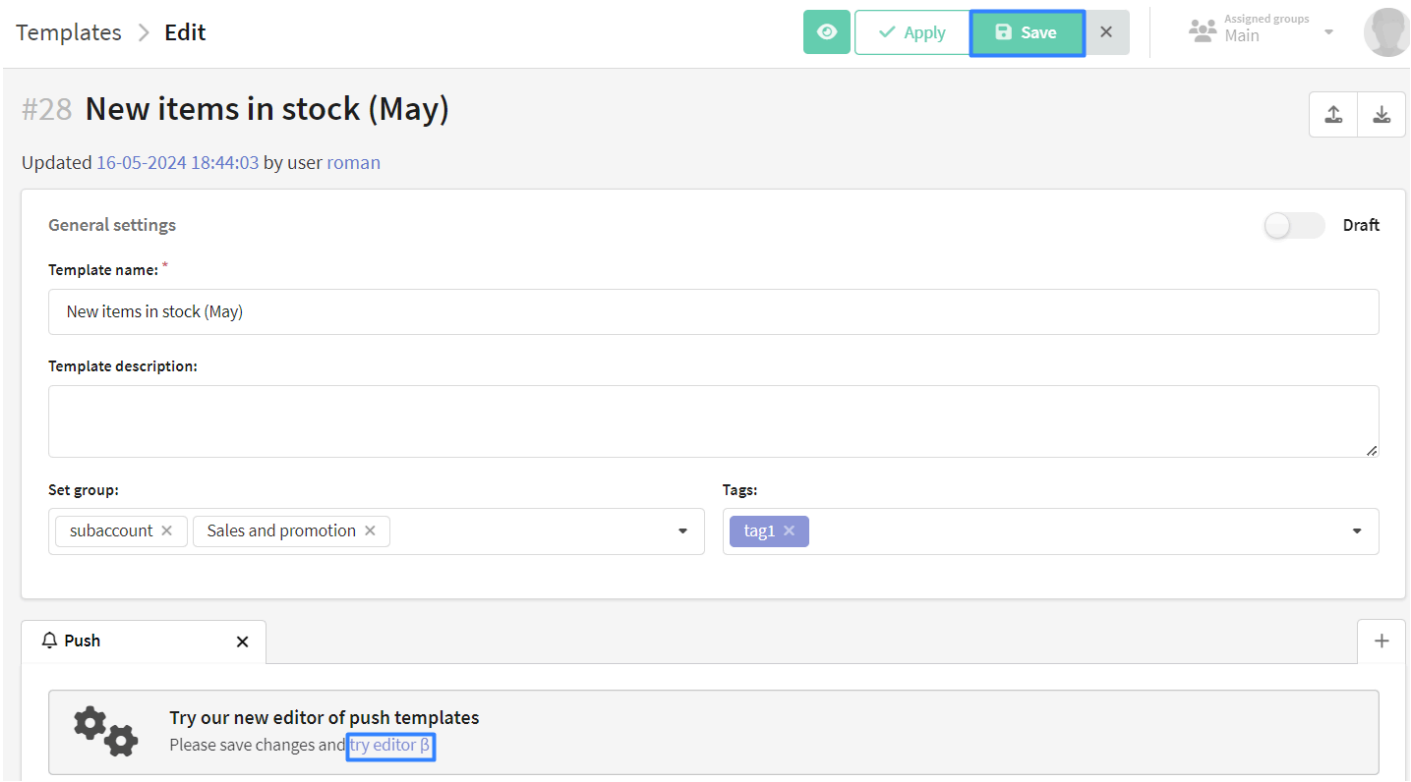
- **A **title** that is similar to email subject**
- **Notification **body** is the main text part of the message**
- **An **image** or an **icon** that will be displayed in the notification**
- **A clickable **link** is the link that the subscriber will follow when he clicks on the push notification.**

To the title, link and body, you can add the data of a specific client, as well as dynamic text content from external sources:



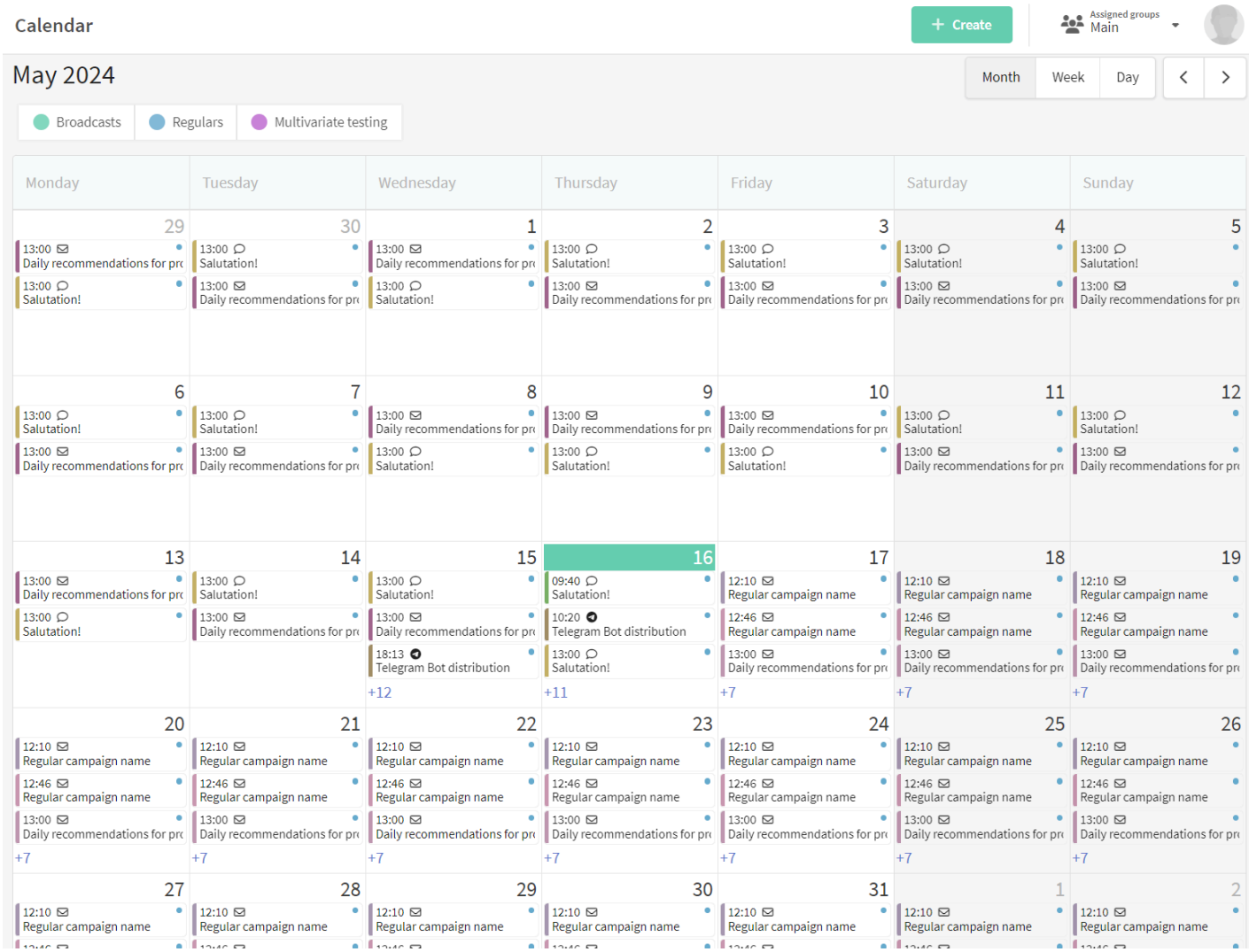
If you send not just Web push, but Rich push notifications, you'd better take a look at our special editor to create a template. In Rich push notifications, you can add not only a text and an icon, but also buttons, images, banners, and even audio and video files. What's more, the new editor will allow you to individually customize the notification for a specific browser, as well as add adaptive links and deeplinks.

To open the Rich push editor, you need to make an initial version of a push message in the usual template editor at first. We've just described how to do this above. Then click Save. Now go back to the created template and click "try editor β" to go to the Rich push editor.

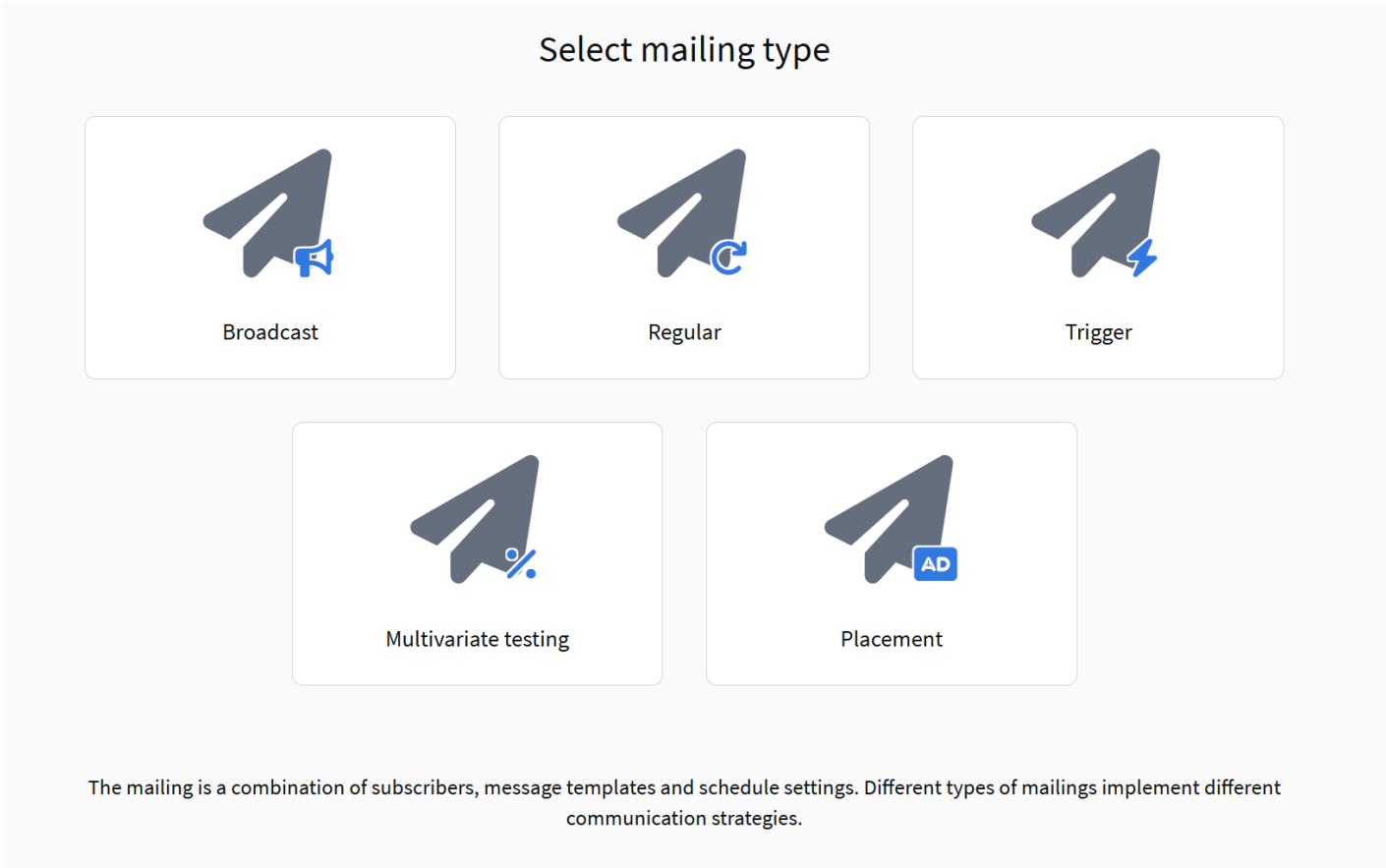


Step 5: Select the mailing type

	If you want to send a Push message, you need to use such a tool as a Mailing. Do you want to send out a **one-time mailing** to all profiles of the database or to the audience of the specific segment? Choose ** Broadcasts** . If the mailing is periodic, your choice is a **Regular** mailing. In the main menu select **Mailing** and open **Calendar.** This is where you'll find all scheduled mailings. </td> </tr>



Decide on a date and create a mailing of the required type:



The mailing is a combination of subscribers, message templates and schedule settings. Different types of mailings implement different communication strategies.

TIP

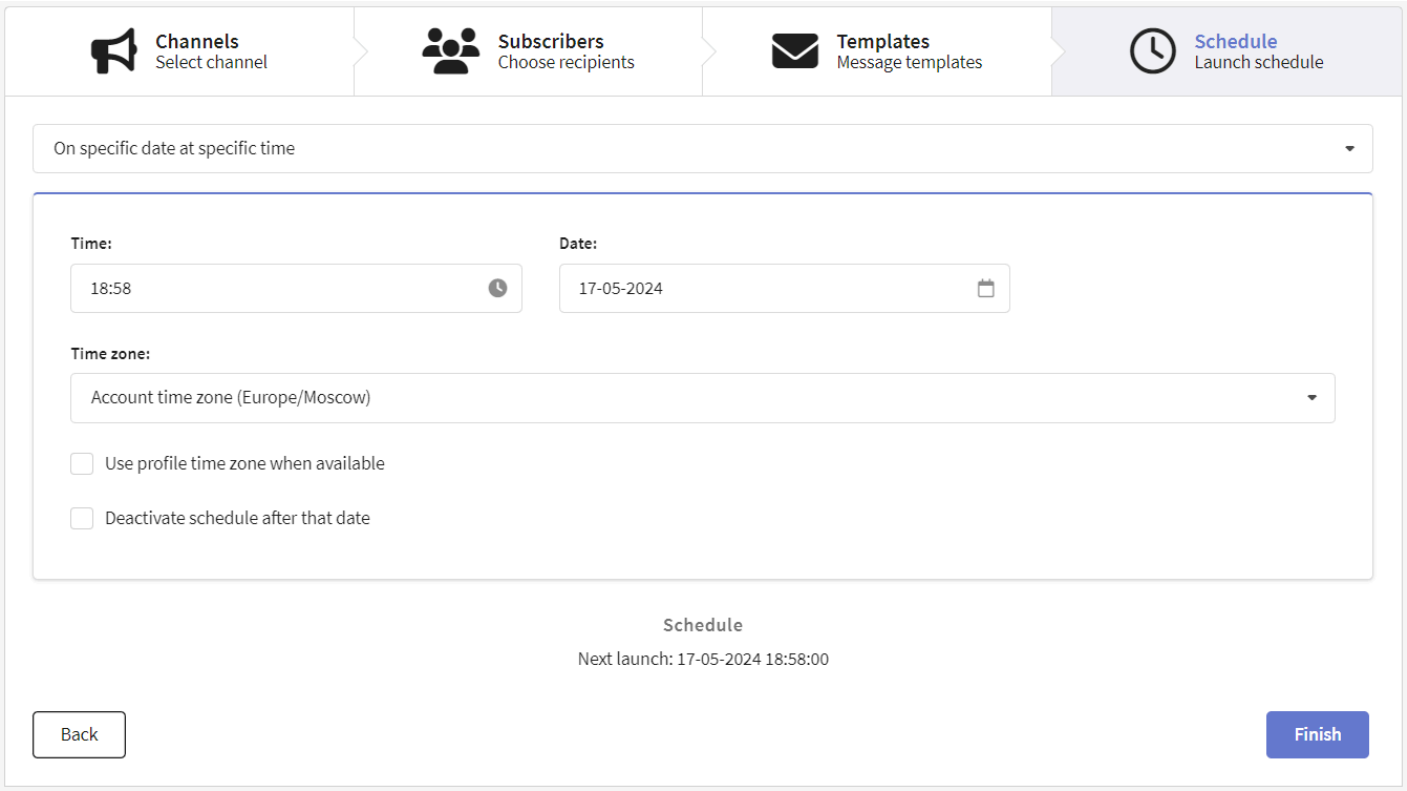
You can find details about how to set up Broadcast on this page.

TIP

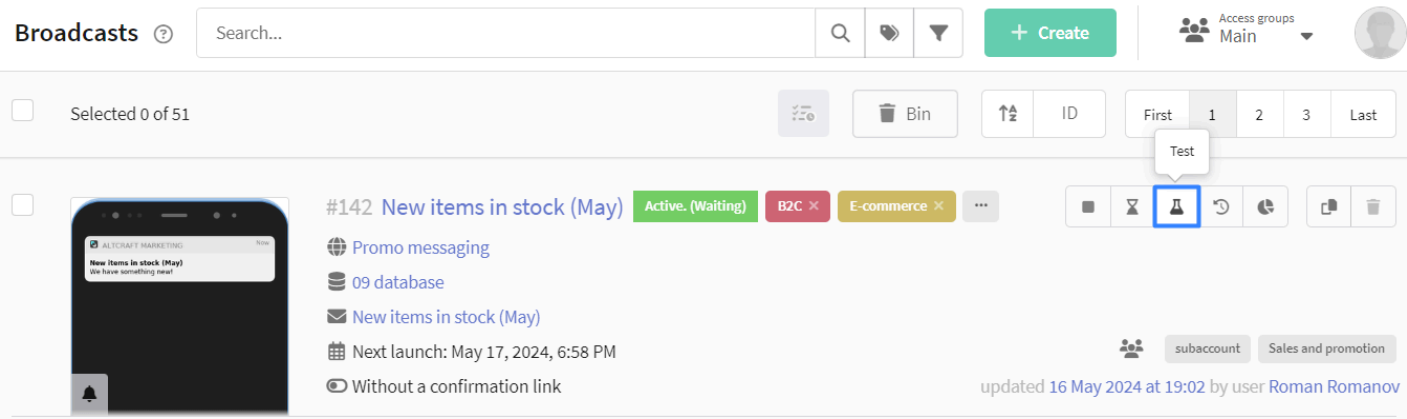
You can find details about how to set up Regular mailing on this page.

Once you're done with the general mailing settings, you need to:

1. **Select **Push** as the communication channel;**
2. **Select providers and sending mode. There are two sending modes - **fanout**, sending to all subscriptions, or **last**, sending to the subscription with the most recent creation date. If there are several of these (created at the same time), then one subscribtion will be selected randomly. If mailings have filters by provider, custom fields or categories, then the filters are first applied, and then the most recent subscription is choosen;**
3. **Select the subscription resource you've just configured and the audience of recipients;**
4. **Add a message template to your message;**
5. **Set a schedule. For example, for a broadcast the option "On specific date at specific time" is available.**



Once the mailing is set up, save it. Later you can find it in the list of mailings of the corresponding type. There is a quick action menu on the right side of each mailing. Use it to test the mailing — and if everything is successful, activate it:



Step 6: Assess the mailing effectiveness

Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

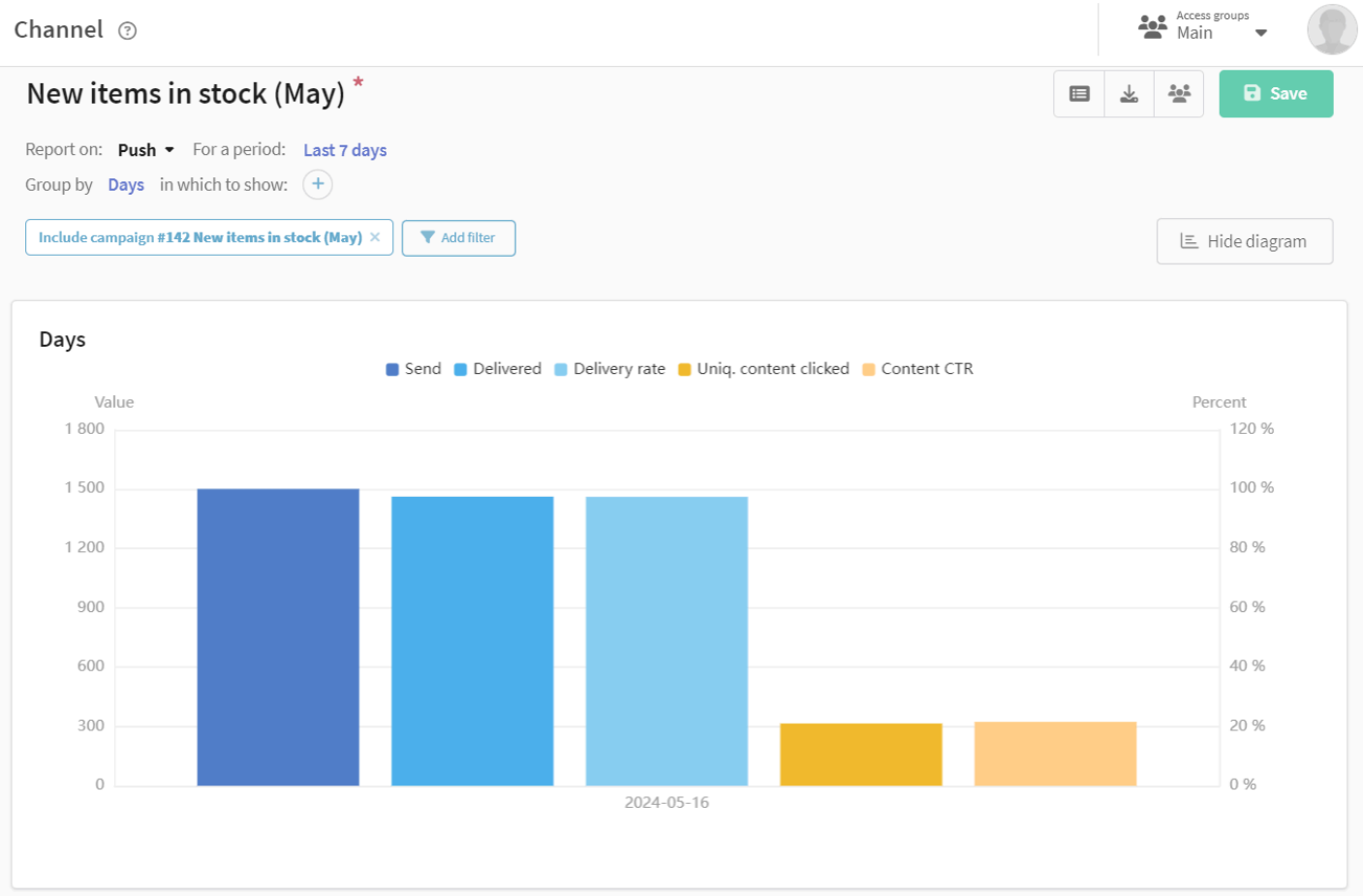
The mailing Log contains the following information:

Type	Description
Mailing started	**Timestamp when the mailing execution began**
Limit	**Indicates whether a sending limit is configured**
Audience source	**Contact origin:** **List — Database** **Segment — Predefined contact group**
Database ID/Segment ID	**Unique identifier of the database being used**
Spent getting segment cursor	**Time required to load contact data from source (milliseconds)**
Number of subscribers received from the source	**Total contacts available in the specified source**
Number of subscribers found	**Contacts with active subscriptions matching mailing criteria**
Number of subscribers without push subscriptions	**Users who haven't opted in to receive push notifications**
Number of Push recipients	**Final count of devices that successfully received notifications**
Number of Push messages sent	**Total push notifications dispatched by the system**
Profiles ignored by resource settings	**Contacts excluded due to mailing configuration rules**
Mailing finished	**Timestamp when mailing processing completed**
Total time spent	**Complete duration from launch to completion (seconds)**

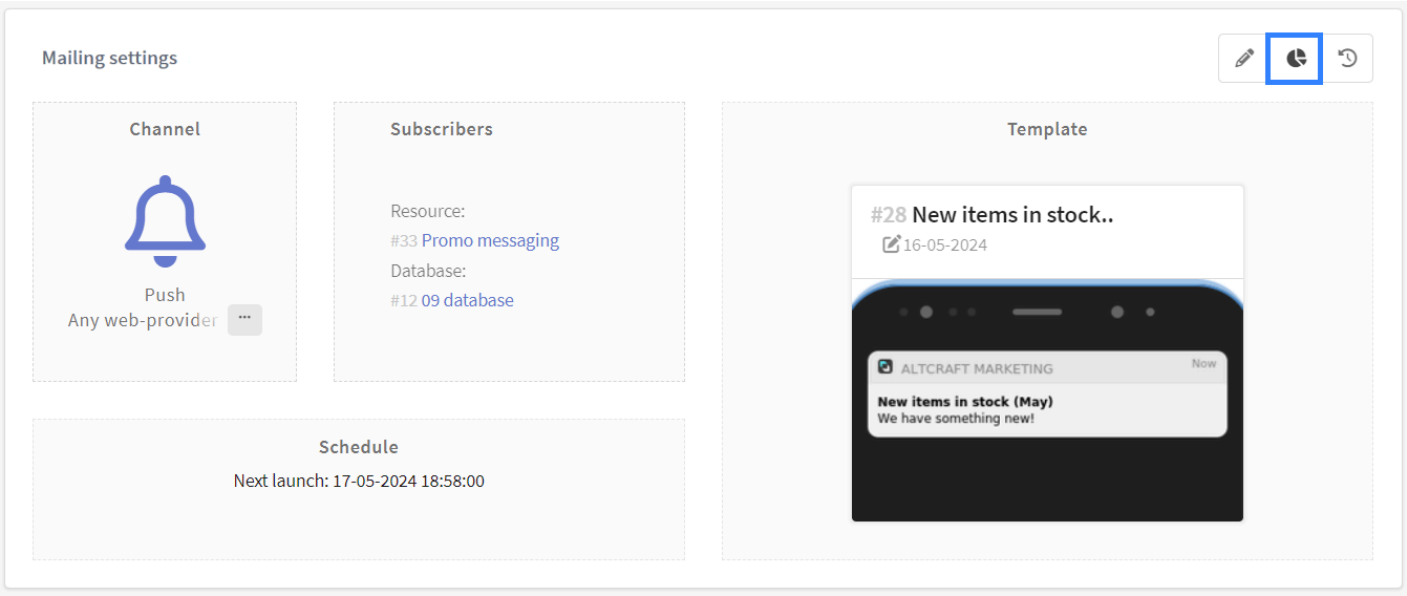
Reports and Analytics

To evaluate the effectiveness of your push mailings go to the section Analytics in the main menu of the platform.

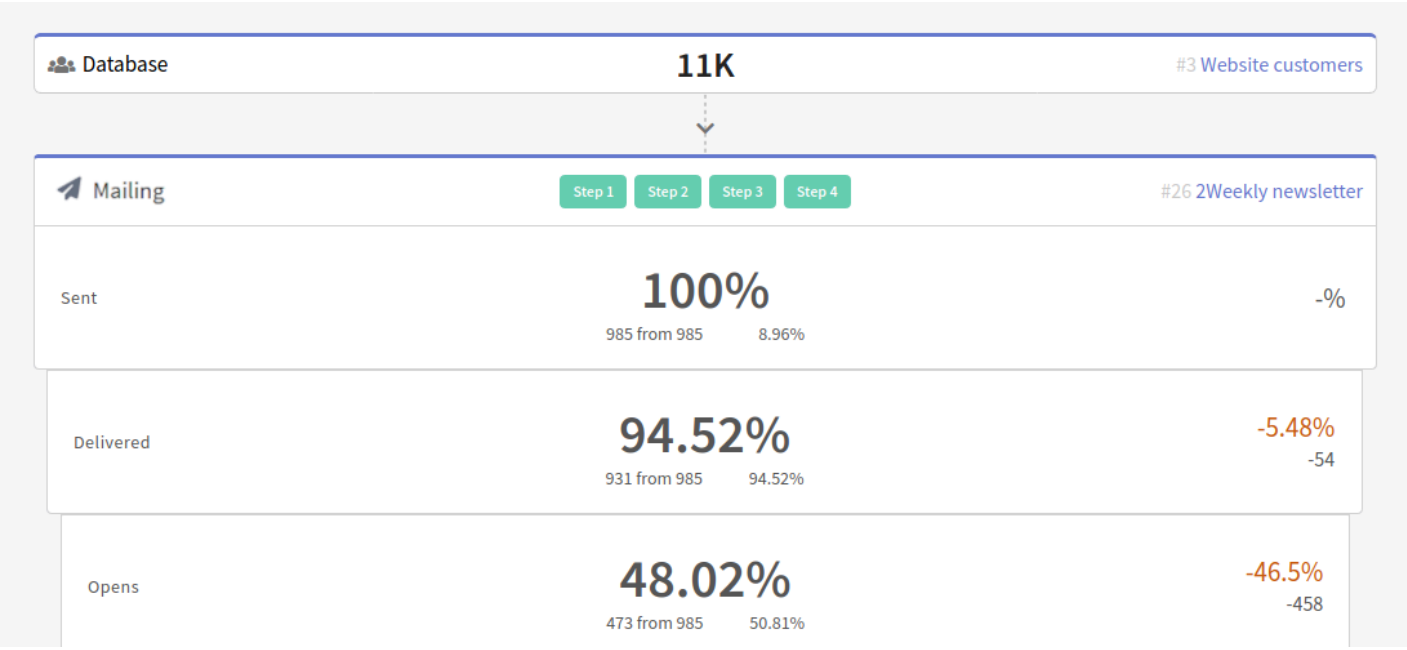
The channel report provides analytics for one or several mailings, for one or several communication channels. For the convenience of grouping, you can select a preset of indicators for Push:



You can also go to the channel report for a specific mailing by following the link in the mailing editor:



Actions in the Push mailing or in the template can turn into steps in the [interactive conversion funnel]:



Web browser push configuration

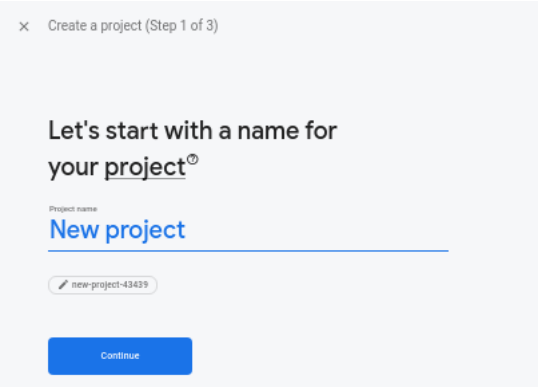
Use these guides to set up Web Push notifications for the browser you need:

Firebase Cloud messaging

Firebase	**If you want to send Web Push to such browsers as **Mozilla Firefox**, **Opera**, **Google Chrome**, **Yandex Browser**, **Samsung Internet Browser**, you need to create a project in Firebase Console.**

Step 1. Creating a Firebase project

Follow the link, create an account in Google Firebase and register your website or application. To do this, click on the Create project button and specify the project name:



TIP

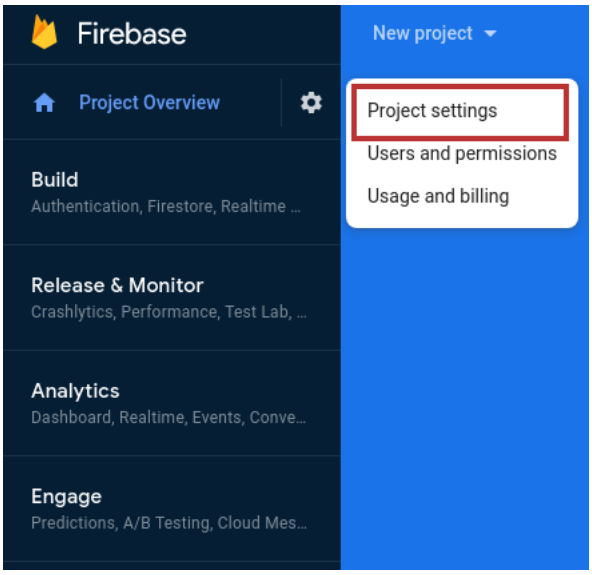
When creating a project, you can set up Google Analytics and also add Firebase to your web application, but these are not required steps.

Step 2: Receiving project data

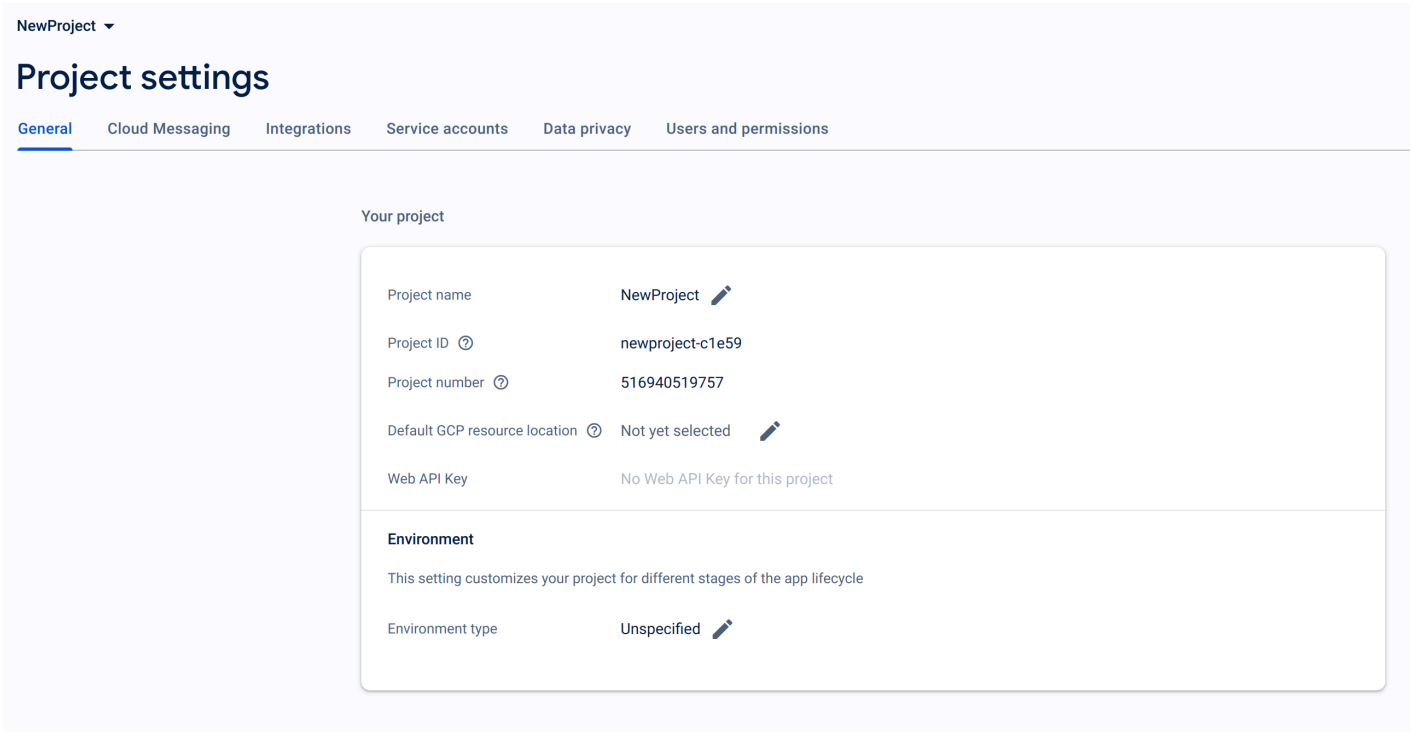
To configure Firebase Cloud Messaging in Altcraft Platform, you need the following information:

- **API key for web application**
- **Firebase project ID**
- **Firebase sender ID**
- **Firebase credentials file (.json)**
- **Application ID**

All this information you can find in the Firebase Console interface. Go to the navigation panel on the left side, click the icon Settings and choose Project Settings:

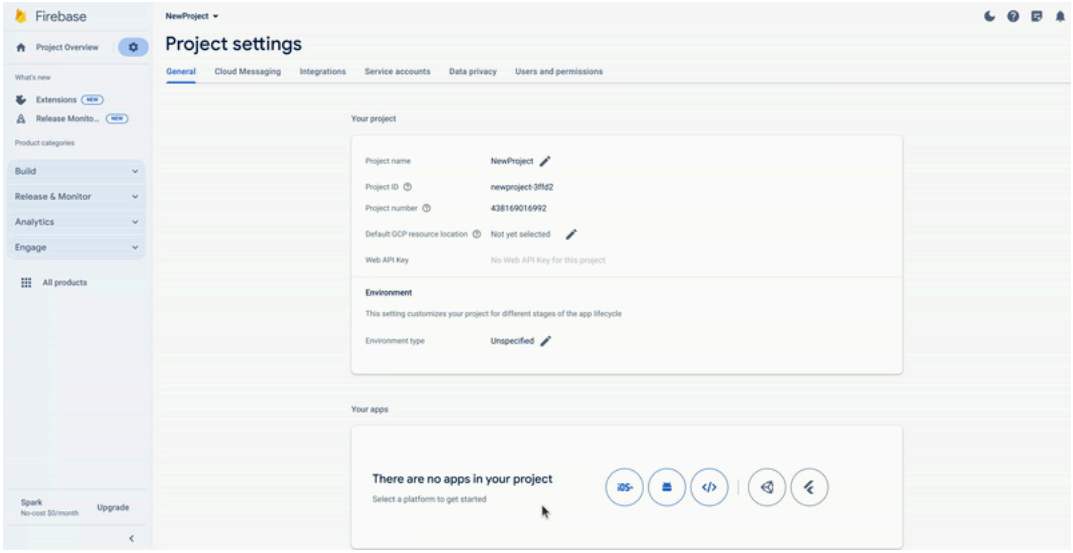


Firebase project ID (Project ID) and API key for web application (Web API key) are in the section “General”:

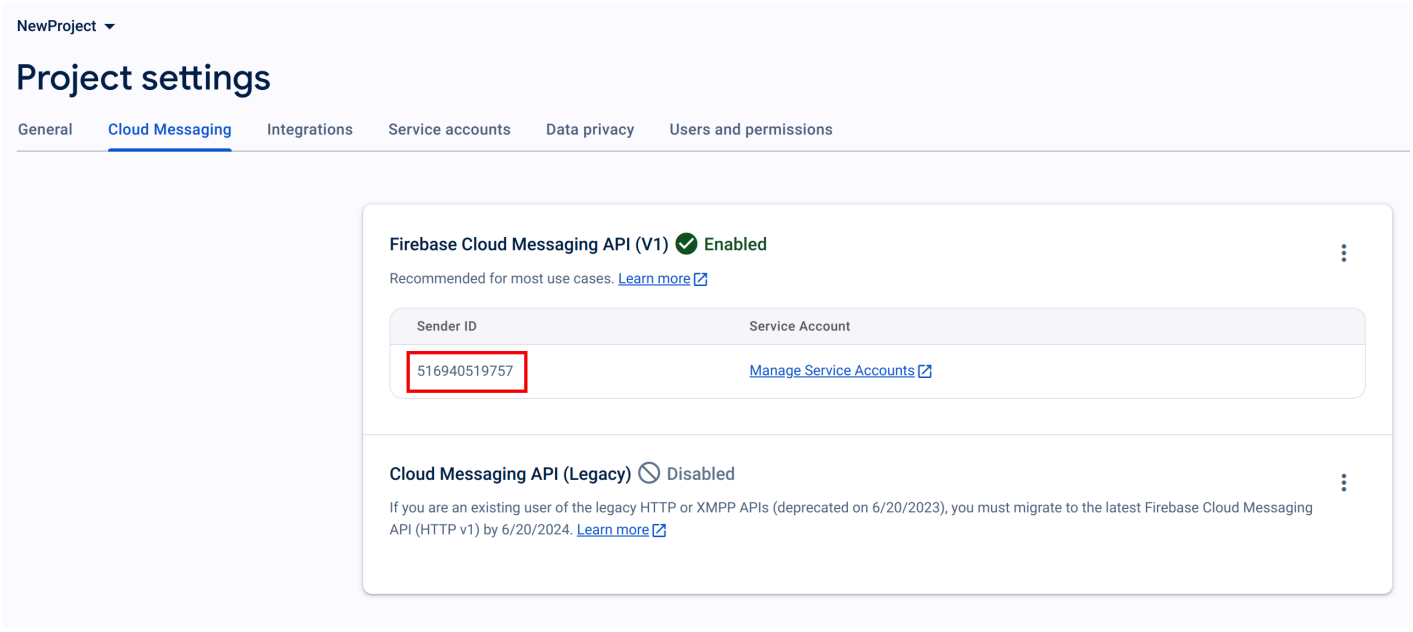


TIP

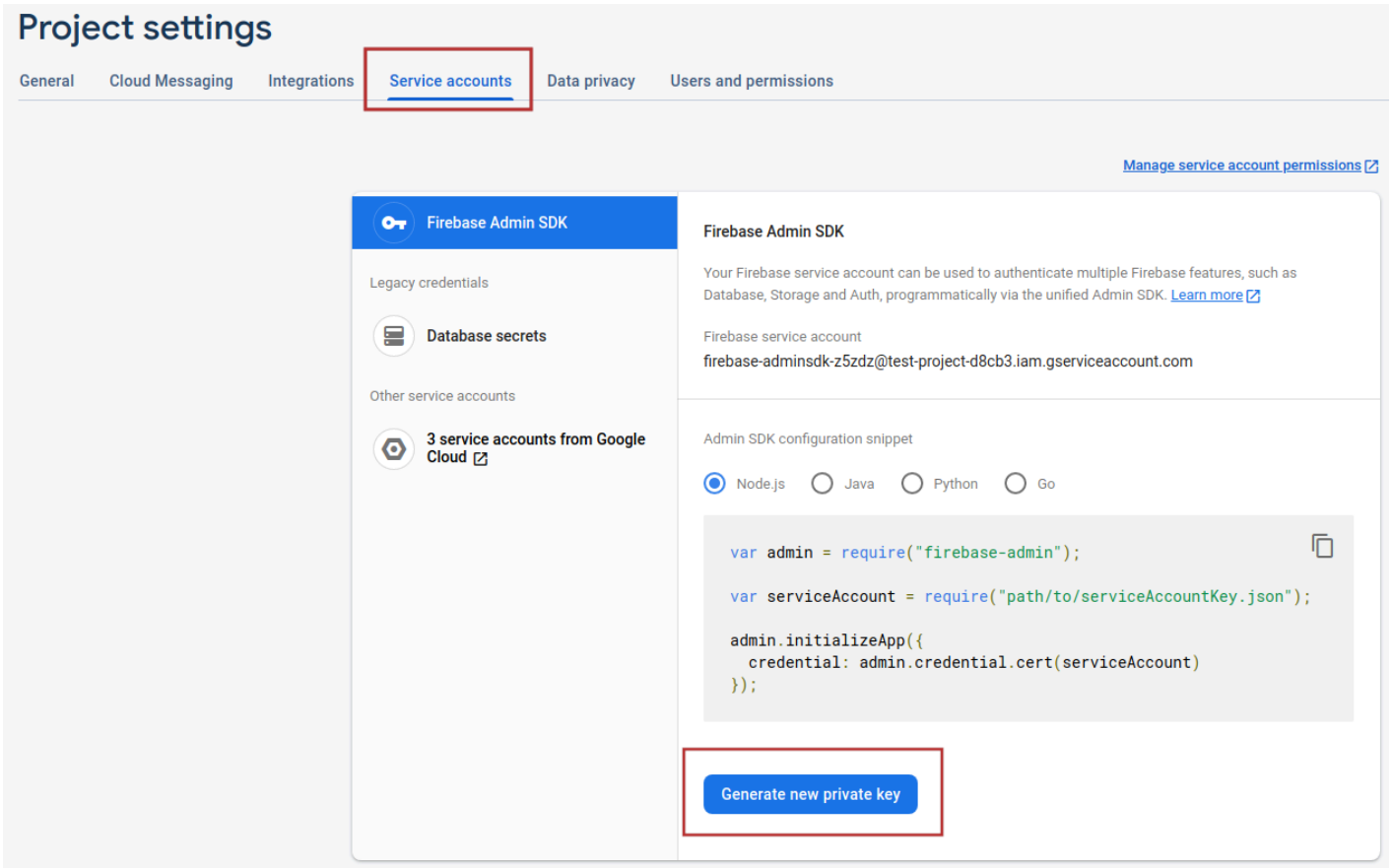
If you do not see the Web API Key, go to the “Build” panel and open the “Authentication” panel in the drop-down menu. Click on the “Get started” button on the screen. Now you can go back to the settings menu at the left panel:



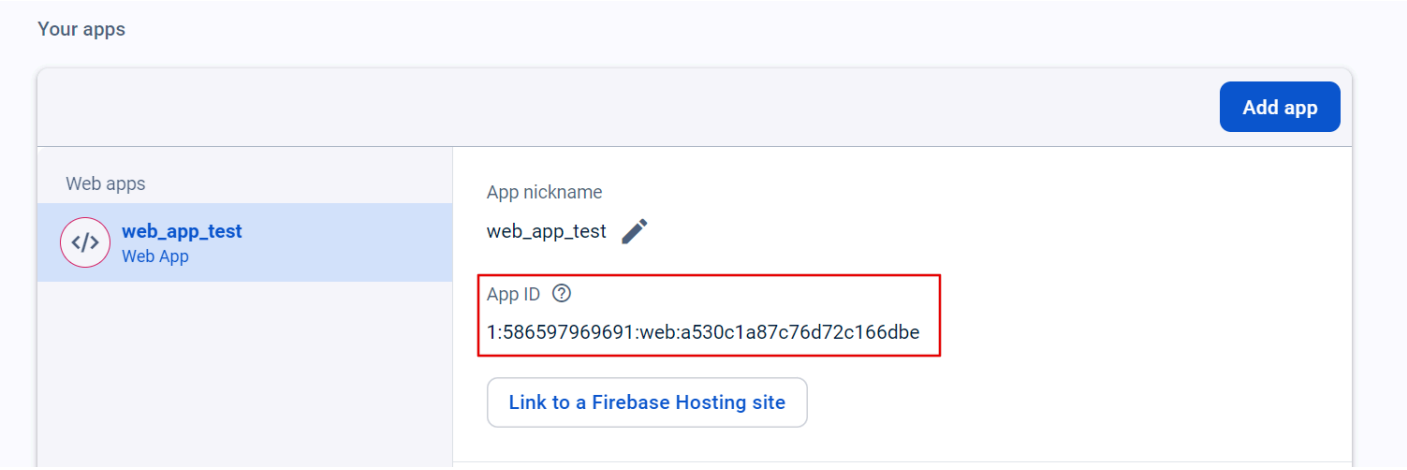
Firebase sender ID (Sender ID) is in the section “Cloud Messaging”:



To configure Firebase, you also need Firebase credentials file. To download it, go to the "Service accounts" tab. Click "Generate new private key". The file will begin downloading:

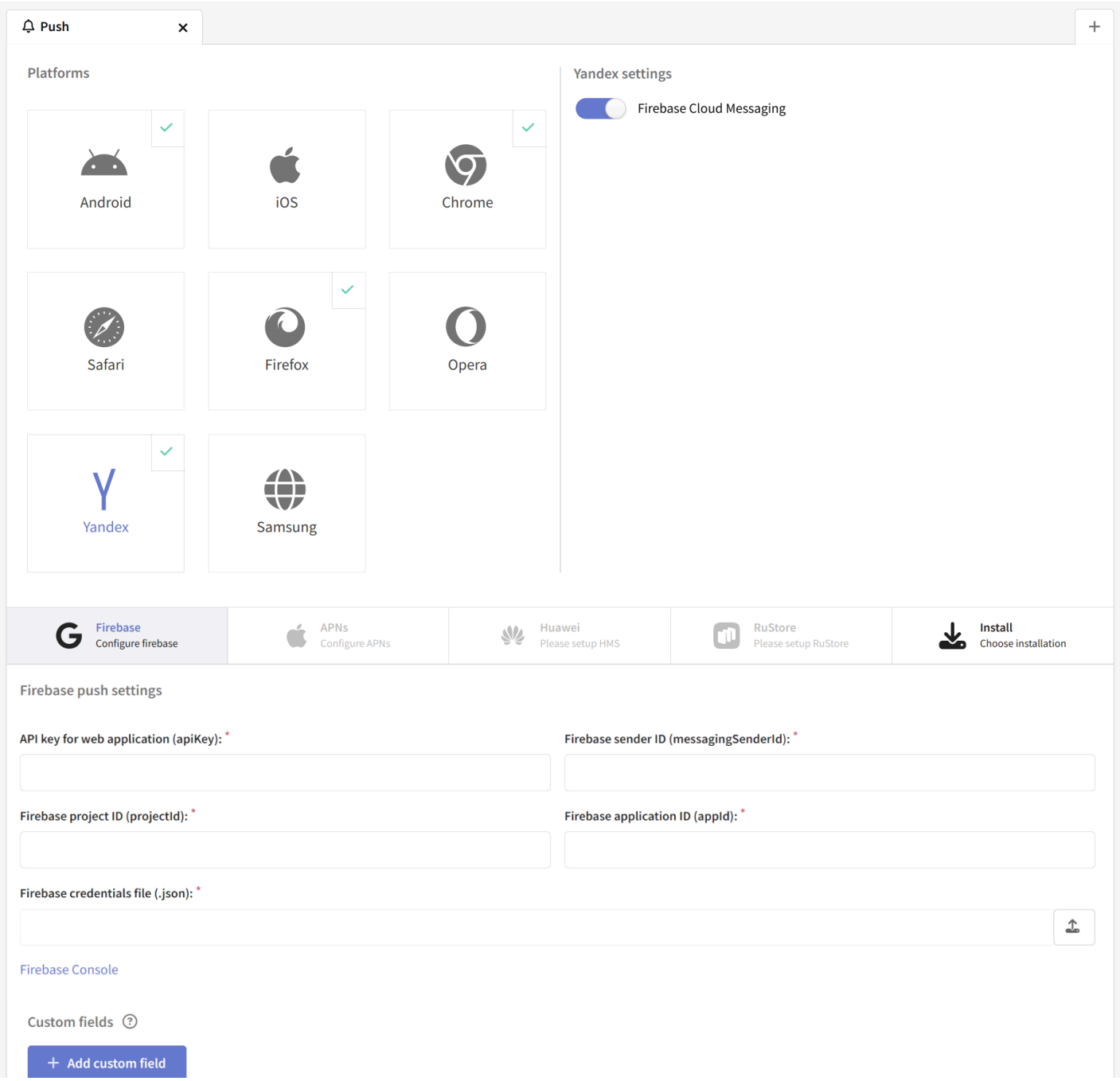


App ID can be found in Your Apps section on Firebase Console:



Step 3. Setting up a push channel in a resource

Go back to Altcraft Platform and go to the “Resource Settings”. Configure Firebase Cloud Messaging for supported browsers using data from the Firebase Console. Upload your .json file to the Firebase credentials file (.json) window:



Save the resource or Apply changes by using the buttons in the upper right corner of the screen. Only after that go to the tab Install.

TIP

The next step is Implementing Web push.

Apple Safari

	If you want to configure Web Push to Safari browser, you must have the Developer extension in your Apple account.

CAUTION

All actions are performed on a macOS X device:

INFO

Sending push notifications to PWA

In iOS 16.4+ versions, Apple has introduced support for push notifications from PWA.

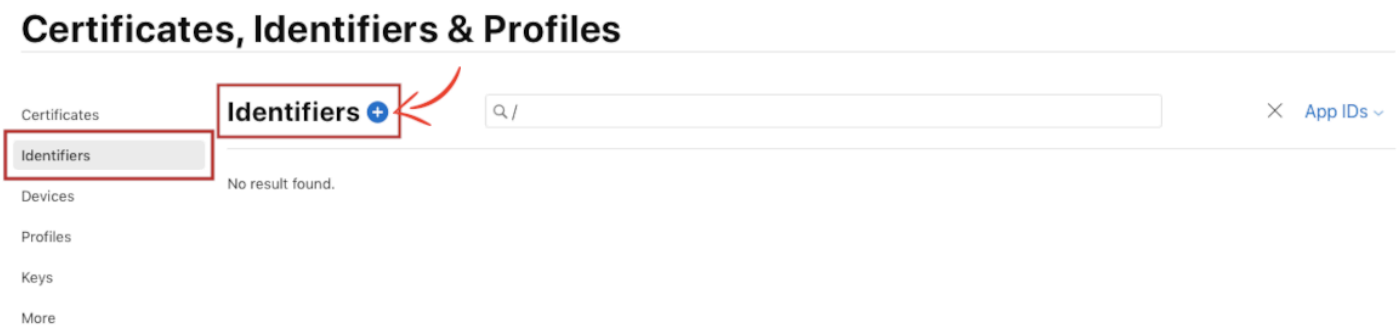
PWA (Progressive Web App) is a technology that allows you to install a website on smartphones or computers as an application. Unlike traditional app downloads from app stores, users can simply visit the website and add an icon to their device's home screen. PWA applications offer native-like functionality, including offline mode, data caching, and the ability to receive push notifications.
To send push notifications to PWA's users, you need to configure the APNS provider for Safari within the resource settings. Once the settings are saved, the PWA provider will be automatically enabled.

Please note that the subscription ID for PWA push notifications is attached to a specific resource. If you transfer subscriptions from one resource to another, they will stop working.

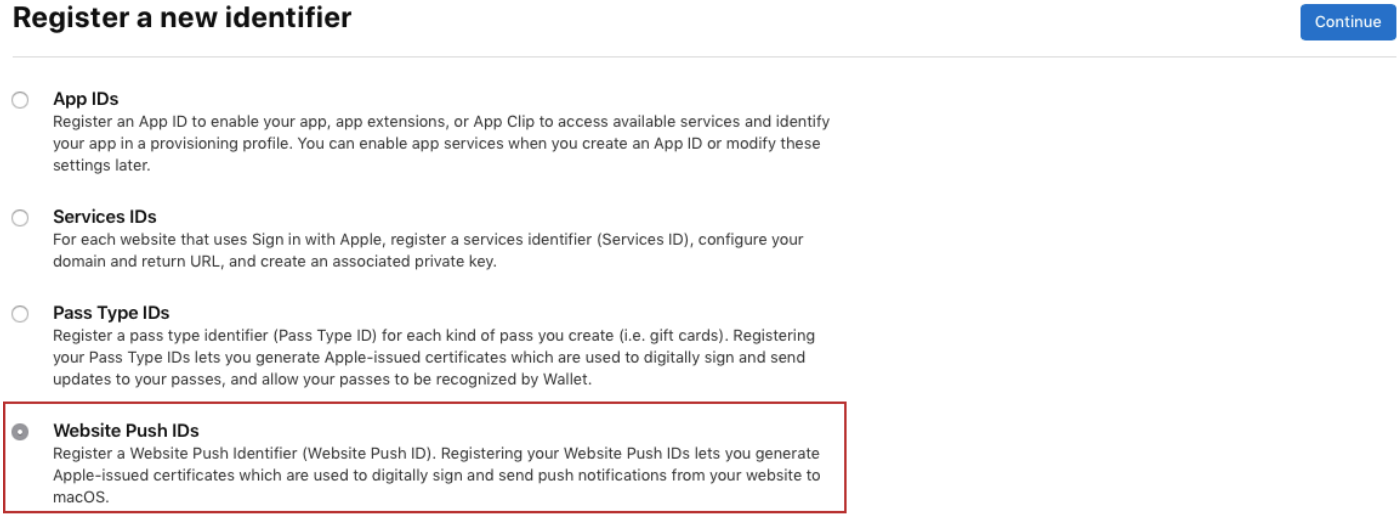
Step 1. Register a new identifier

1.Open [Certificates, Identifiers & Profiles]() and log into your Apple developer account.

2. On the left navigation bar, select the Identifiers tab and click + next to the "Identifiers" heading.



3. In the new window choose **Website Push IDs.It lets you send push notifications from your website to macOS.**

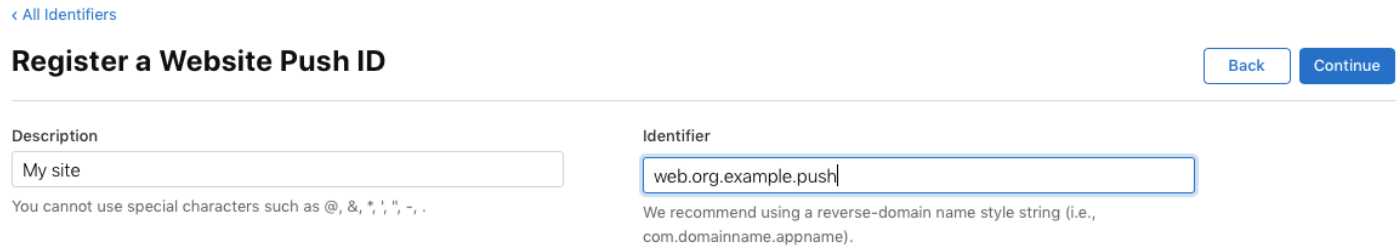


4. Enter Description and Identifier.

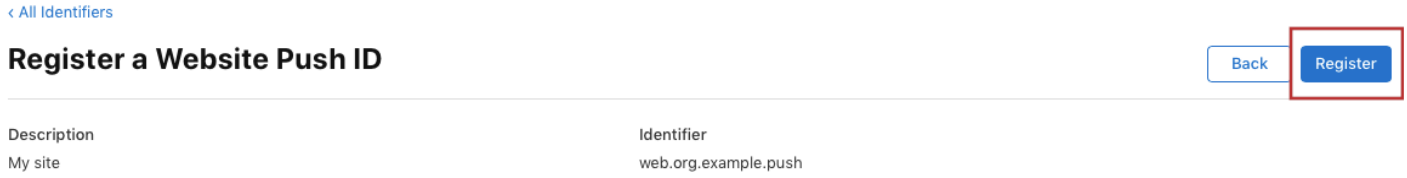
INFO

`ID` **should be entered in such a format:** `web.com.domainname.appname`.

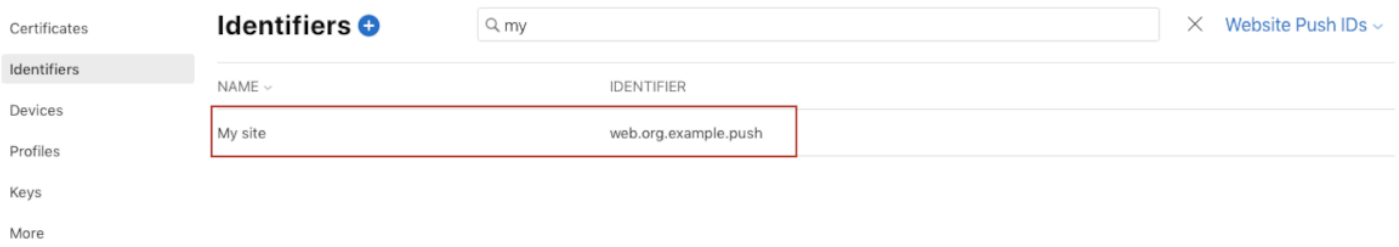
Important! The first part is always `web`**, there can be as many parts as you like. It is welcomed to use the reverse domain style:** `"web.org.example.push"`.



5. Click Continue, then Register.

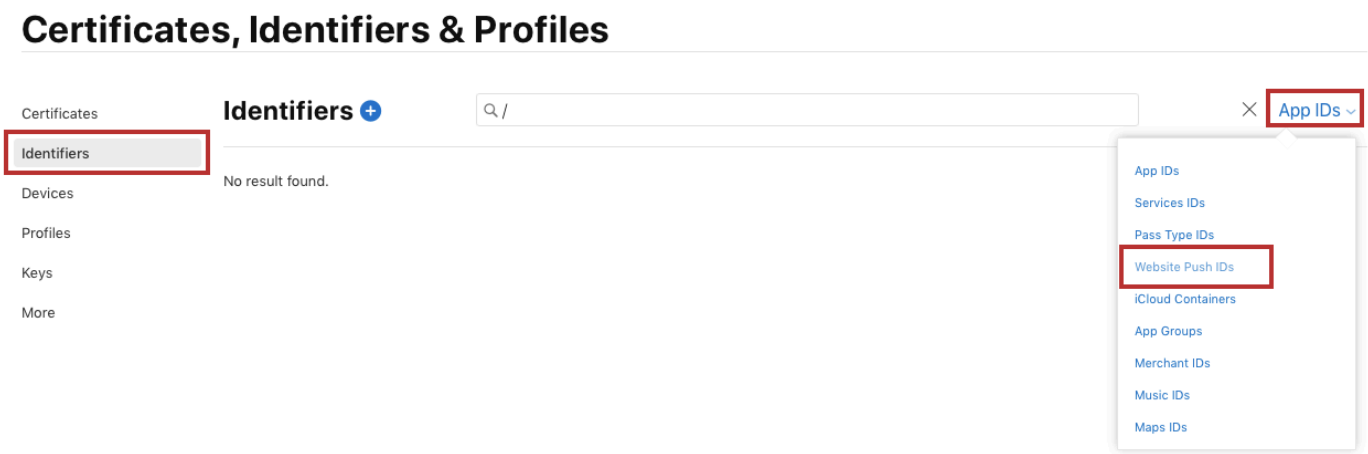


So, you have created and registered a Website Push ID. Now you can check information about this ID on the Identifiers tab at any time.



INFO

To see identifiers for sending push notifications from websites, do not forget to open the drop-down list in the right corner by clicking on App IDs and select Website Push IDs.



Step 2. Create Certificate

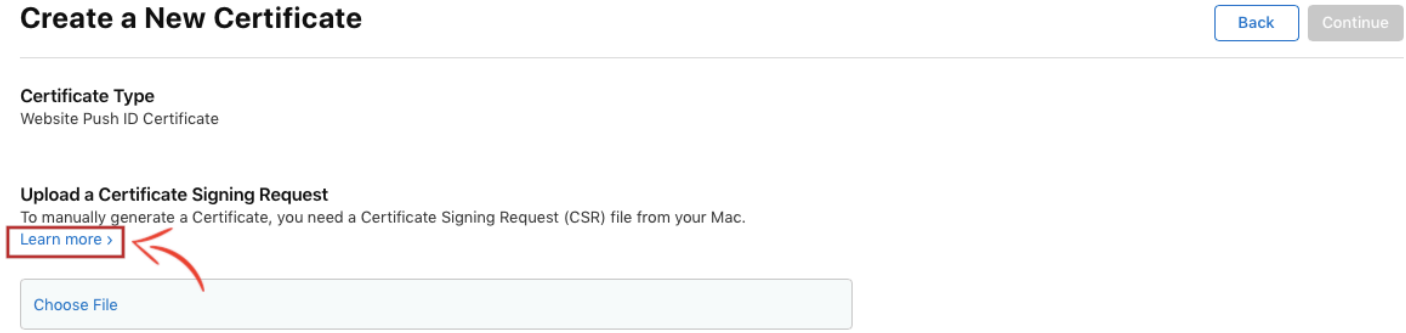
1. Each identifier need a certificate. To create it, in the list find Web Push ID you've just created and click it.

2. Pay attention to the Identifier field. The information in it will be needed to configure the resource for sending Web push notifications on the Altcraft platform — when you'll be filling in the Website Push ID field.

3. Click Create Certificate .

4. Now it's time to generate the certificate. You need Certificate Signing Request (CSR) (the request format is ".certSigningRequest").

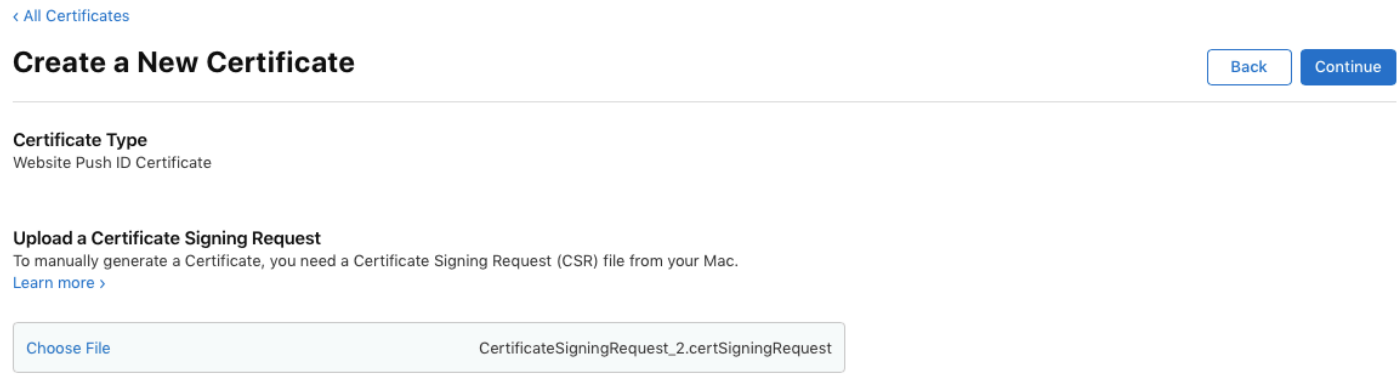
To make the request, click Learn more.



5. You'll be taken to a new page. Follow the instructions on it.

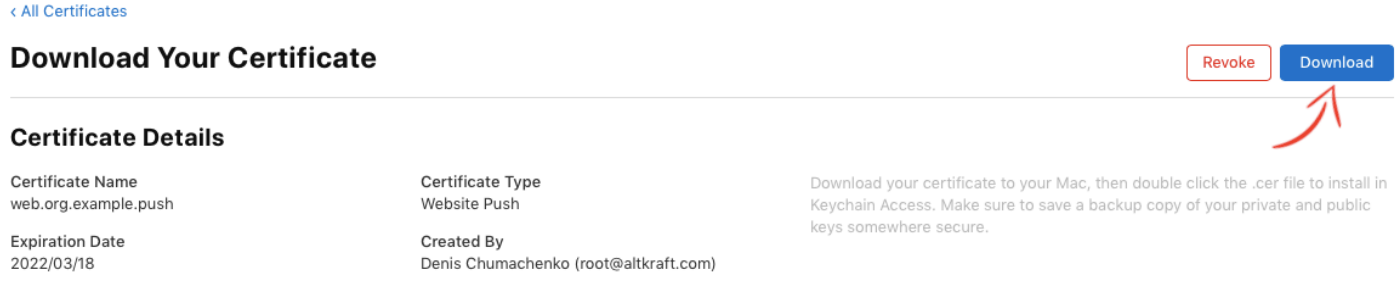
6. When you create the Certificate Signing Request, go back to the previous page. Click Choose File and in the new window select the CSR you created.

Then click Continue.

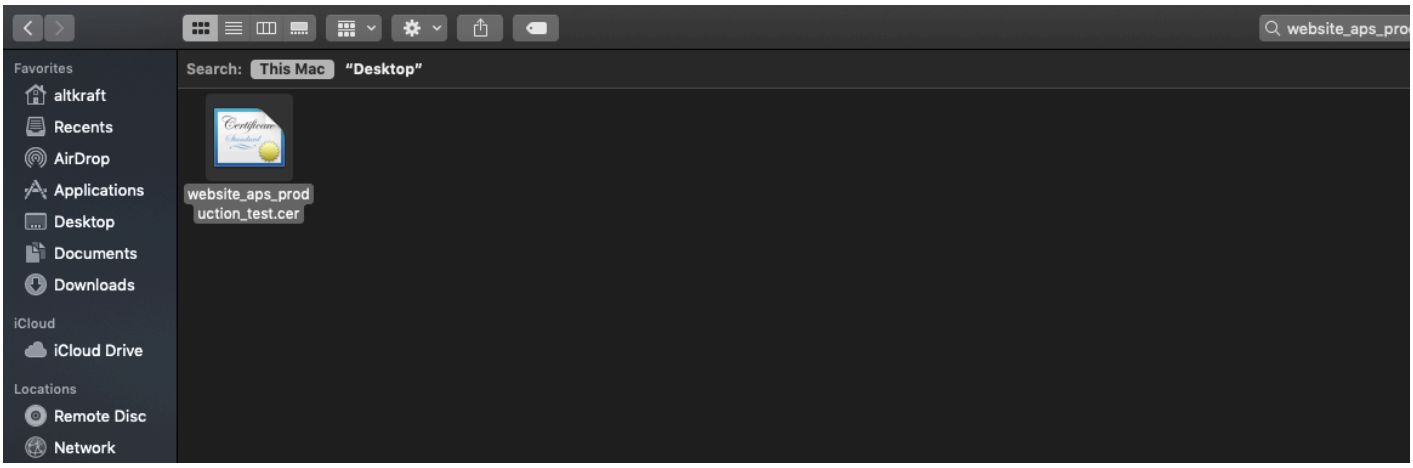


Step 3. Save the certificate

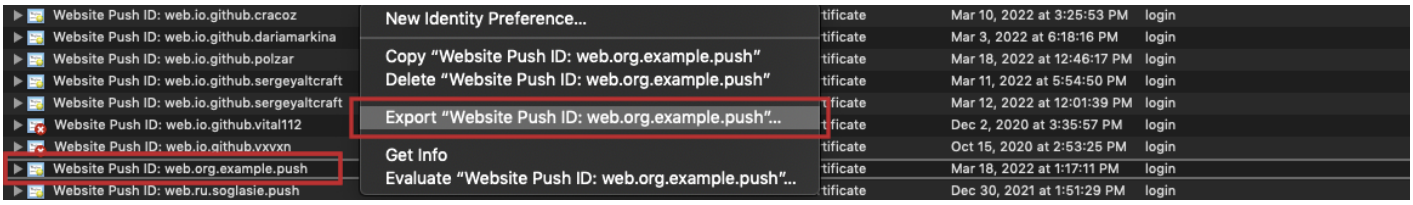
1. On the next tab, download the Certificate to your Mac by clicking the Download button. The file will be saved in .cer format.

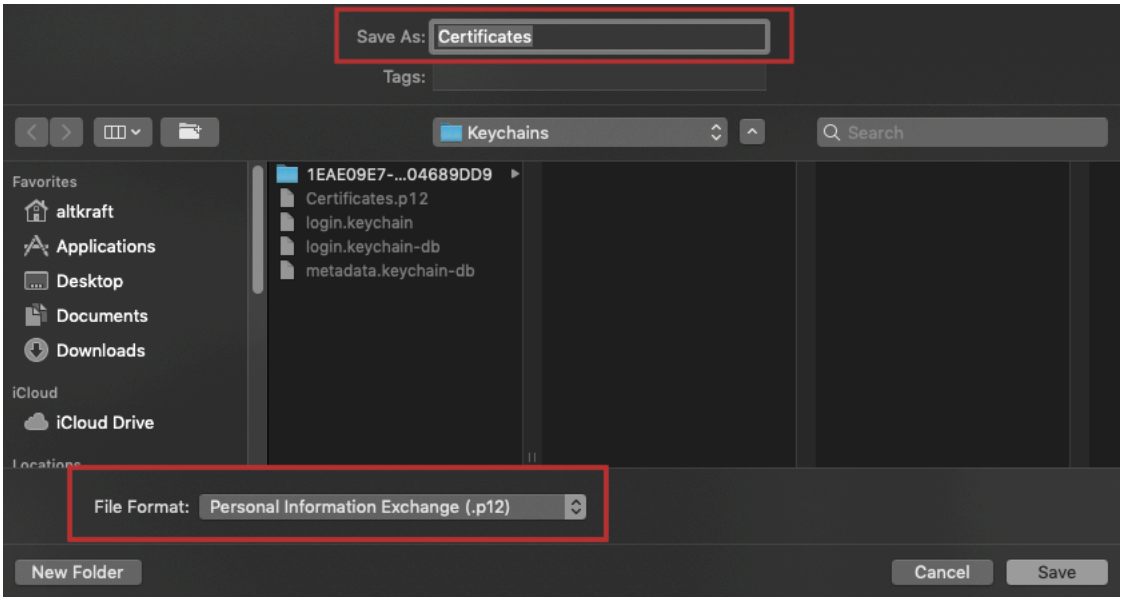


2. Find it on your device and double-click it. This will install the.cer file** into KeyChains.**

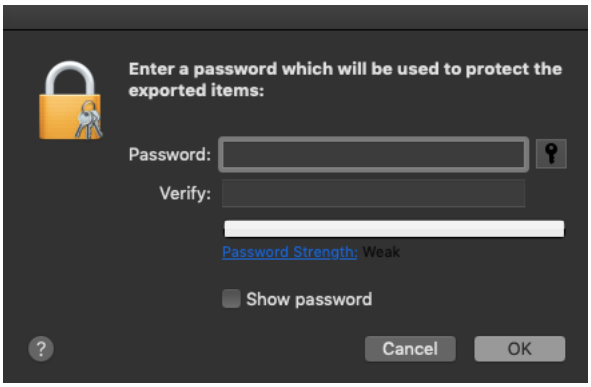


3. Now find the installed certificate in KeyChains, right-click on it and export it to disk in .p12 format.





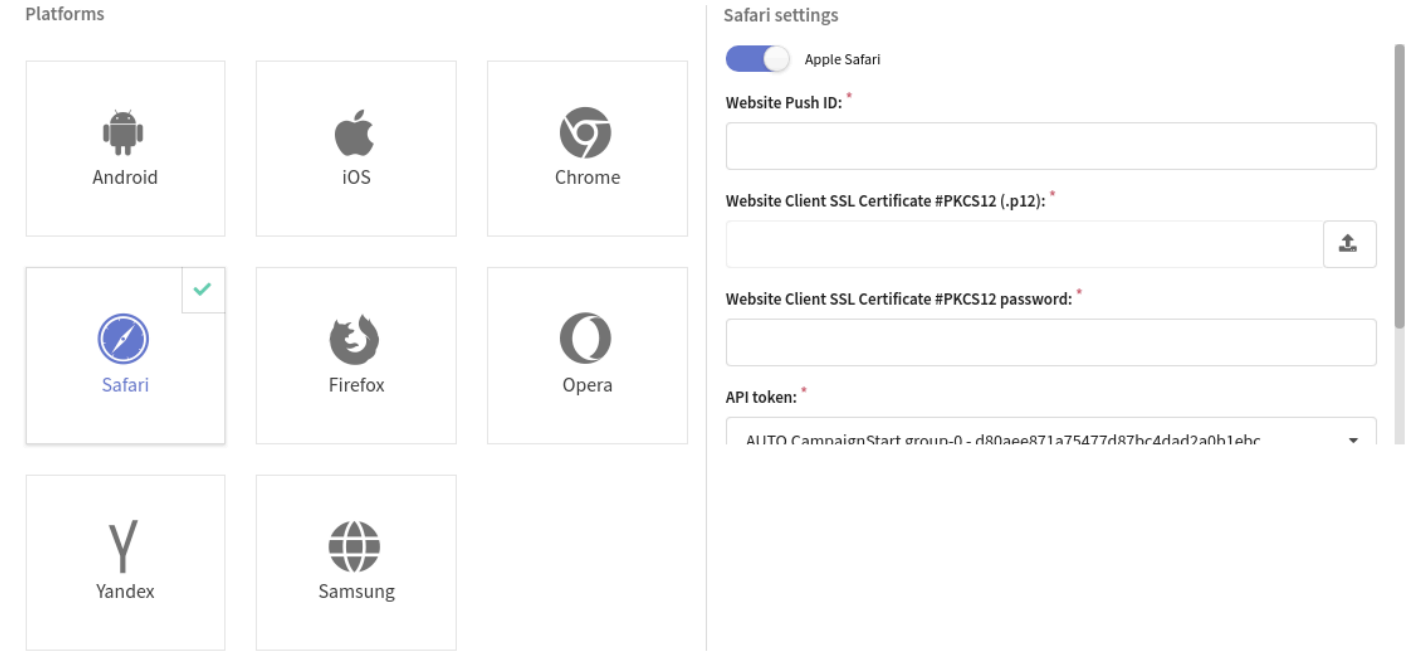
4. During the export process, you will be asked to enter a Password. The same password will later be required to set up push notifications on the Altcraft platform. Remember it!



Step 4. Configure our resource in the Altcraft platform

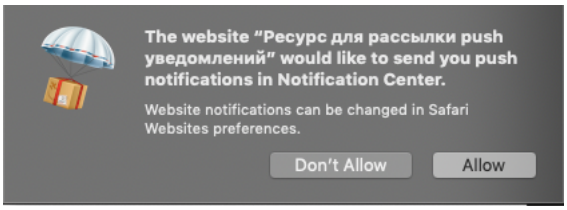
Let's go back to the Altcraft platform — to the settings of the resource for sending Web push notifications.

1. Choose Safari platform. Activate Apple Safari service.



2. Fill in fields:

- **Website Push ID — identifier in your Apple Developer account. You can find this value in the “Identifier” field of your Apple Developer account.**
- **Website Client SSL Certificate — the .p12 certificate file. This certificate is exported from Keychain Access on your Mac device.**
- **Website Client SSL Certificate пароль — the password used when exporting the .p12 certificate.**
- **API token — a platform token that enables APNS to send subscription requests to the platform. You can generate this token in the Settings —> Tokens section. If the token is deleted, the platform will no longer accept subscription requests from Safari for this resource.**
- **Icon — an image that will be displayed in the push notification. It helps users identify the source of the notification. The format is png, The recommended size is 256x256 or larger. Note that Safari supports only one icon per resource (website).**

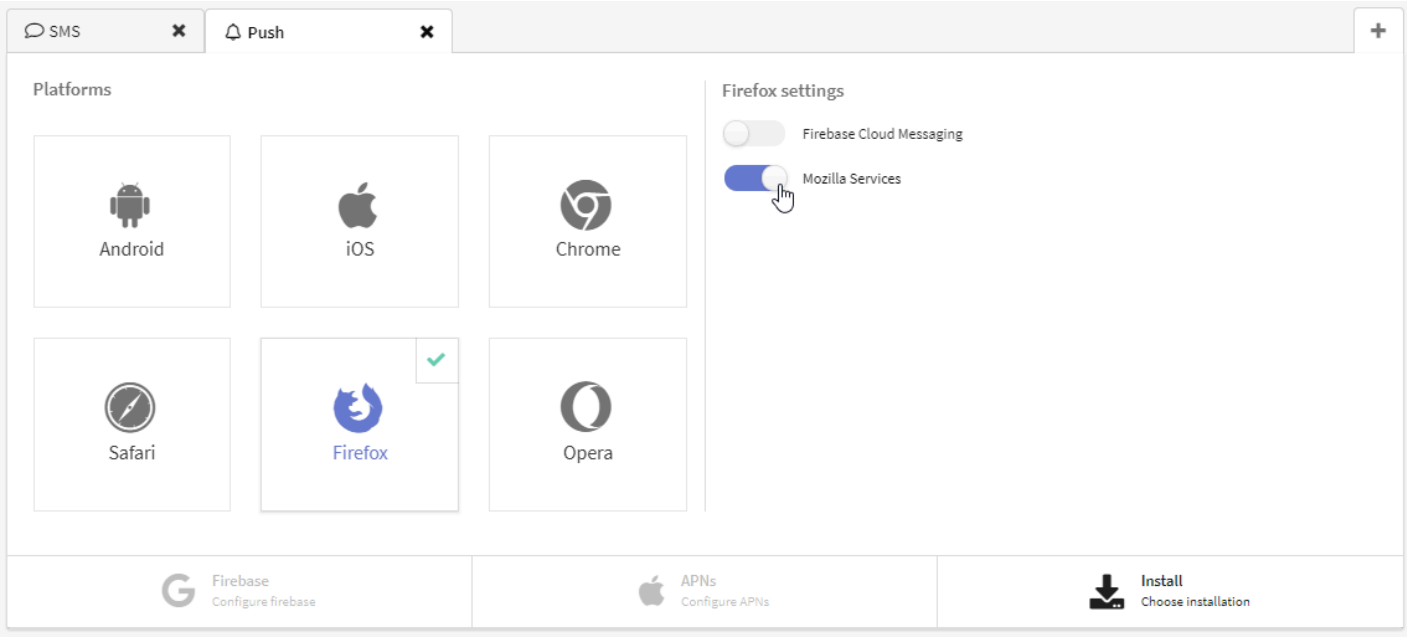


TIP

The next step is Implementing Web push.

Mozilla Services

	Do you want to begin sending push notifications to Mozilla Firefox browser? All you need is to enable "Mozilla Services" in the resource push options. Firefox has a built-in push SDK.



TIP

The next step is Implementing Web push.

Migrating push subscriptions from third-party services

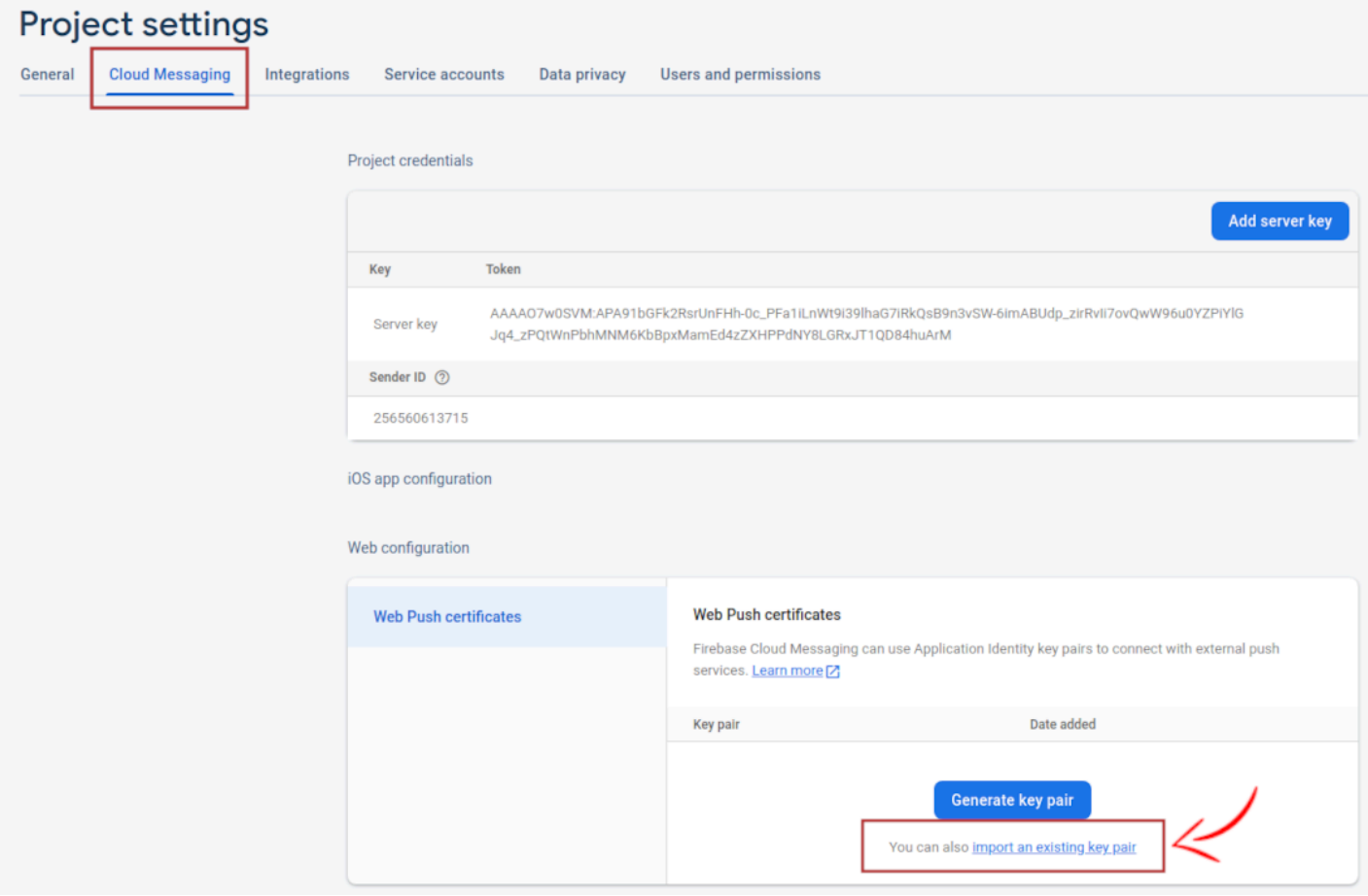
Step 1. Get VAPID keys from the service from which you're going to migrate

Request the keys from the technical support of your service. Both private and public keys are required: private key, public key.

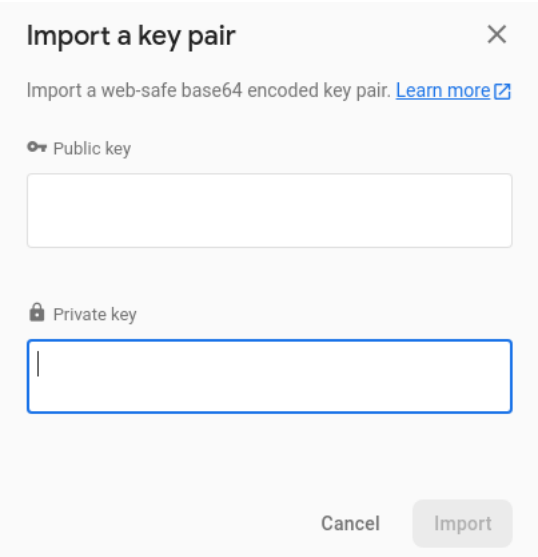
> TIP
>
> **VAPID keys and Application Server keys are the same thing.**

Step 2. Import the VAPID keys into your Firebase project

Go to project settings in Firebase and go to the Cloud Messaging tab. At the bottom of the page, you will see the Web Configuration section. Here you can generate keys yourself, but we do not need this, since you have already received VAPID keys from your service. Click import an existing key pair at the very bottom of the section.



In the new window enter Public key and Private key that you've got from the technical support of your service.



Step 3. Configure the resource

For information on configuring a resource for Web push notifications, see Web push: setting and implementation.

> **TIP**
>
> **Please note that resource settings differ from browser to browser.**

Step 4. Create a csv file with subscriptions

When a new user subscribes to push notifications, a PushSubscription object is created. It contains all the information that is necessary to send push notifications to this user. The object is stored in JSON format:

```
{
"endpoint": "https://some.pushservice.com/something-unique",
"keys": {
        "p256dh": "BIPUL12DLfytvTajnryr2PRdAgXS3HGKiLqndGcJGabyhHheJYlNGCeXl1dn18gSJ1WAkAPIxr4gK0_dQds4yiI=",
        "auth": "FPssNDTKnInHVndSTdbKFw=="
        }
}
```

Now we need to unload the subscription fields (i.e. the key values of the PushSubscription object) from a third-party service. In other words, you need to get a csv file in any way available for the service. In this file each subscription should have the values of the following fields: endpoint, p256dh, auth.

Step 5. Preparing the CSV File for import

If the Altcraft platform is deployed inside your business (On-Premise solution), transfer the generated csv file with subscriptions to the platform administrator.

If the platform is installed on the cloud (SaaS solution), transfer the csv file to Altcraft support.

he administrator (or support team) should use a specialized utility to convert the data into a format compatible with the platform import. After processing, you will receive a CSV file ready for import.

Step 6. Import profiles

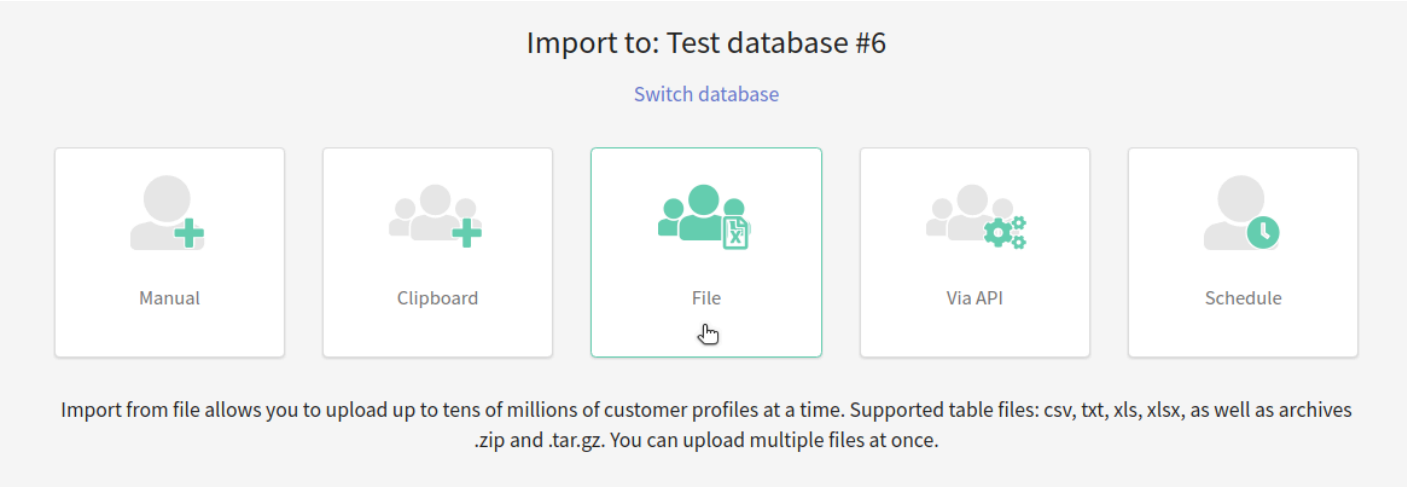
Now you can import profiles into Profile Database from a csv file. During the import a subscription to the required provider will be created.

In addition, you have the opportunity to update the database by creating a subscription to the same provider. Please note that in this case it must be one of the Firebase providers.

> **TIP**
>
> **The provider identifier allows you to find out which browser the subscription belongs to and through which service it was created. Examples of providers: Chrome Firebase, Apple Safari, Mozilla Firefox.**

So, create or select a database and click Import customer profiles in the drop-down list.

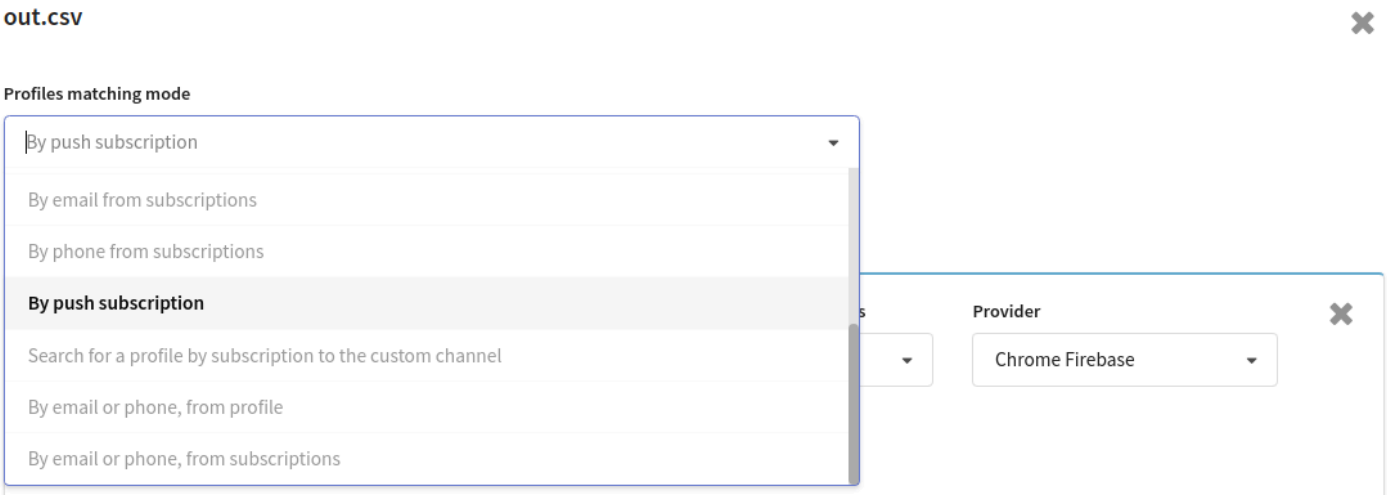
Select the import method - File. Upload the required csv file with subscriptions.



Click Profiles matching mode and select By push subscription in the drop-down list.

CAUTION

If your csv file contains any data that can be used to configure correspondences between profiles (“match users”), then before the import process, select the appropriate profile search mode.

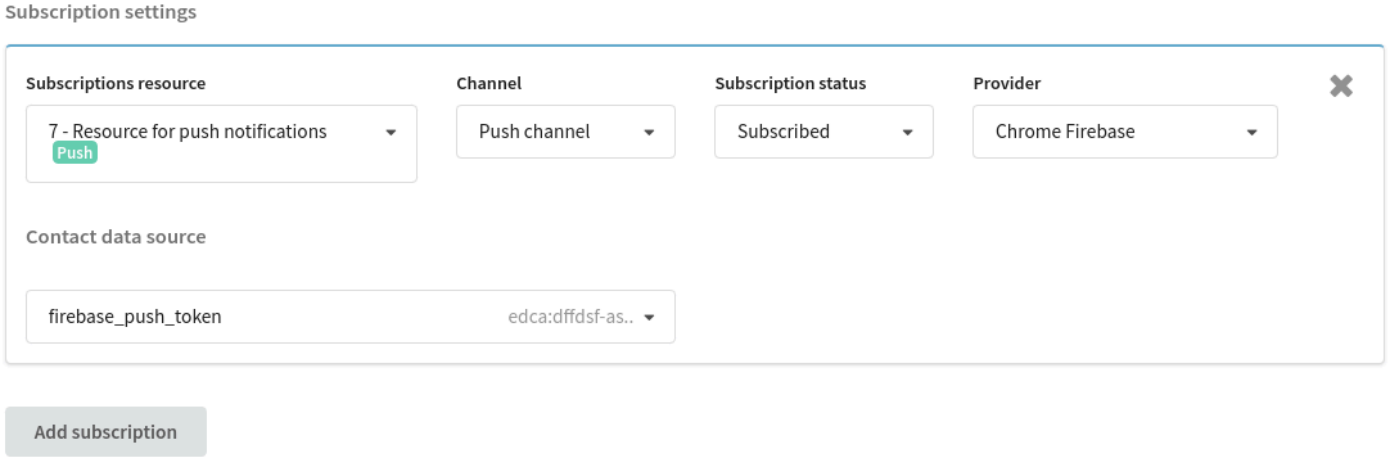


Click the “Add subscription” button and then configure the subscription:

1. **Channel is Push channel;**
2. **Provider — if the subscription browser is unknown, you can import subscriptions with the Chrome Firebase provider;**
3. **Contact data source — firebase_push_token.**

CAUTION

To create a subscription, you need the name of the field where the id of the subscription you are importing is stored. In our csv file, this field will be called firebase_push_token.



Now it's time to configure column mapping:

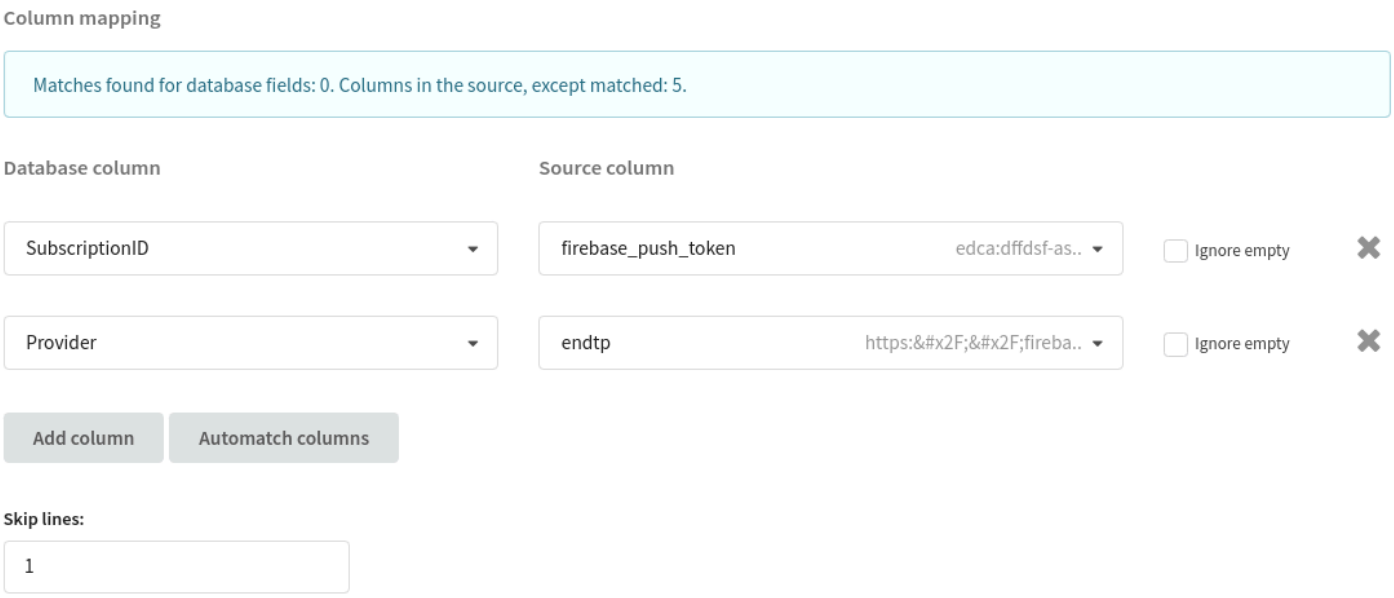
- **SubscriptionID — firebase_push_token**
- **Provider — Altcraft provider id field**

TIP

If the Altcraft provider identifier field is missing in the csv file, then specify the same field as in the SubscriptionID. This way you'll create a new subscriber for each subscription.

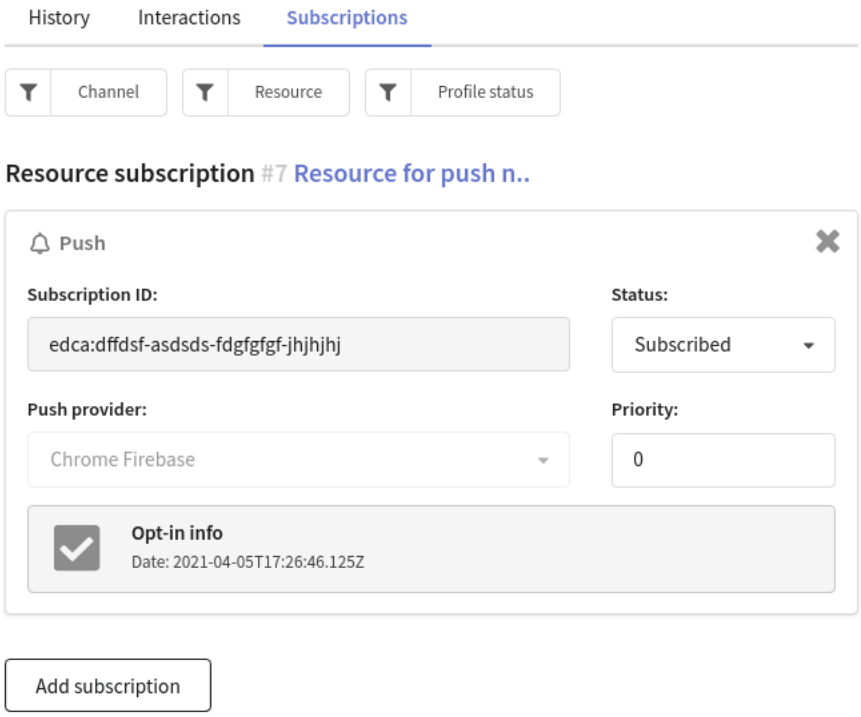
CAUTION

In the Skip Lines field, enter "1" if the file contains headers. Otherwise, the headers will be captured on import.



After you click the Import button in the upper right corner of the screen, the process of importing new profiles into the Database will begin.

Once the profiles have been successfully imported, you can check them in the selected Profiles Database. Open the Subscriptions tab to see the new profile subscription for push notifications and information about it.



TIP

If you would like to migrate from OneSignal, see this article: [Migration from OneSignal](#).

How to transfer push subscriptions configured for Safari

CAUTION

If you want to transfer push subscriptions that are configured for the Safari browser, you need to have the Developer extension in your Apple account. In addition, the transfer is possible only if they were created with a certificate and a key from this account.

Step 1. Configure the resource

For information on configuring a resource for Web push notifications, see Web push: setting and implementation.

Step 2. Create a csv file with subscriptions

Unload identifiers of Safari subscriptions from external service in csv file format.

Step 3. Import profiles

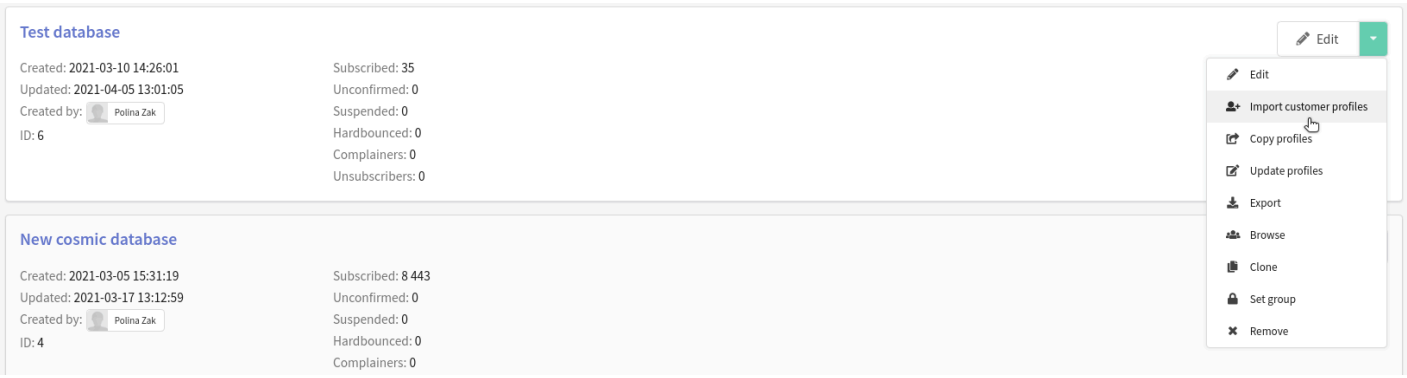
Now you can import profiles into Profile Database from a csv file. During the import a subscription to the required provider will be created.

In addition, you have the opportunity to update the database by creating a subscription to the same provider. Please note that in this case it must be one of the Firebase providers.

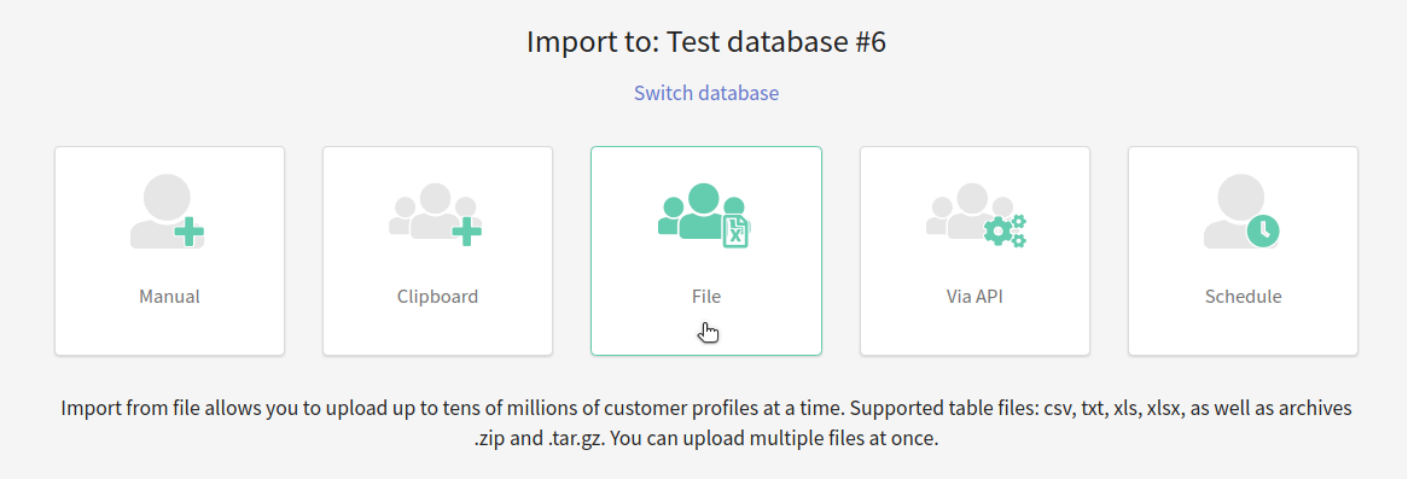
TIP

The provider identifier allows you to find out which browser the subscription belongs to and through which service it was created. Examples of providers: Chrome Firebase, Apple Safari, Mozilla Firefox.

So, create or select a database and click Import customer profiles in the drop-down list.



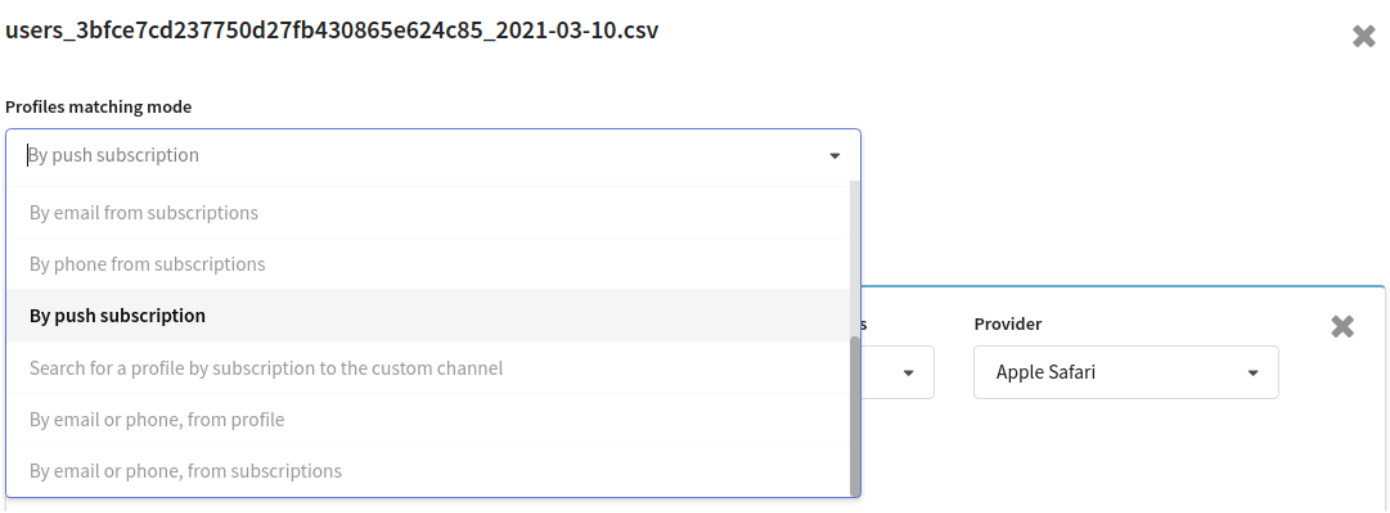
Select the import method - File. Upload the required csv file with subscriptions.



Click Profiles matching mode and select By push subscription in the drop-down list.

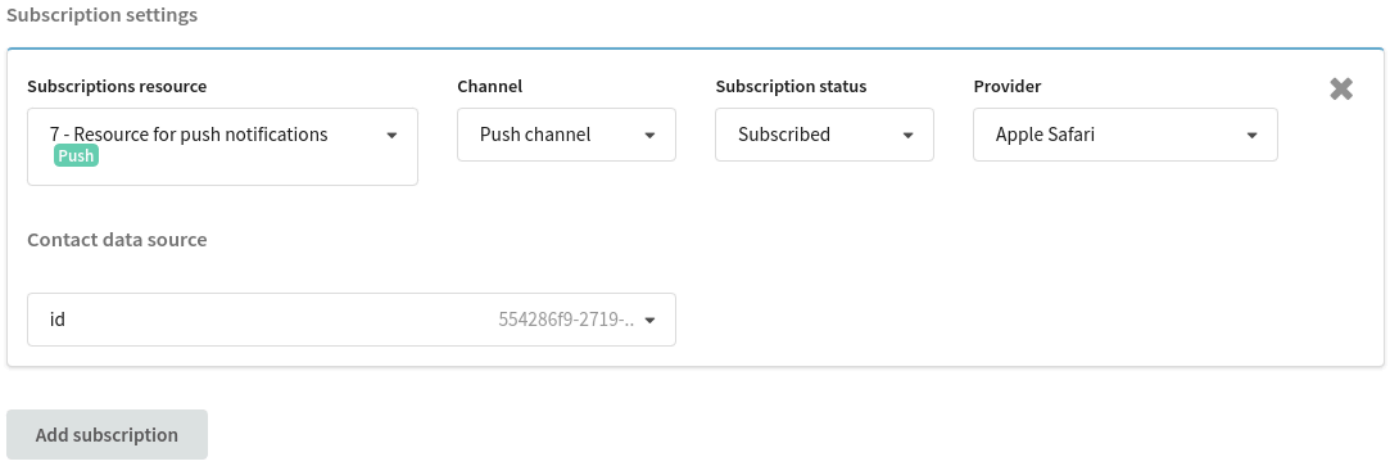
CAUTION

If your csv file contains any data that can be used to configure correspondences between profiles (“match users”), then before the import process, select the appropriate profile search mode.



Click the Add subscription button and then configure the subscription:

1. **Channel is Push channel;**
2. **Provider is Apple Safari;**
3. **Contact data source: you need to specify the subscription identifier field from the file.**



Now it's time to configure column mapping:

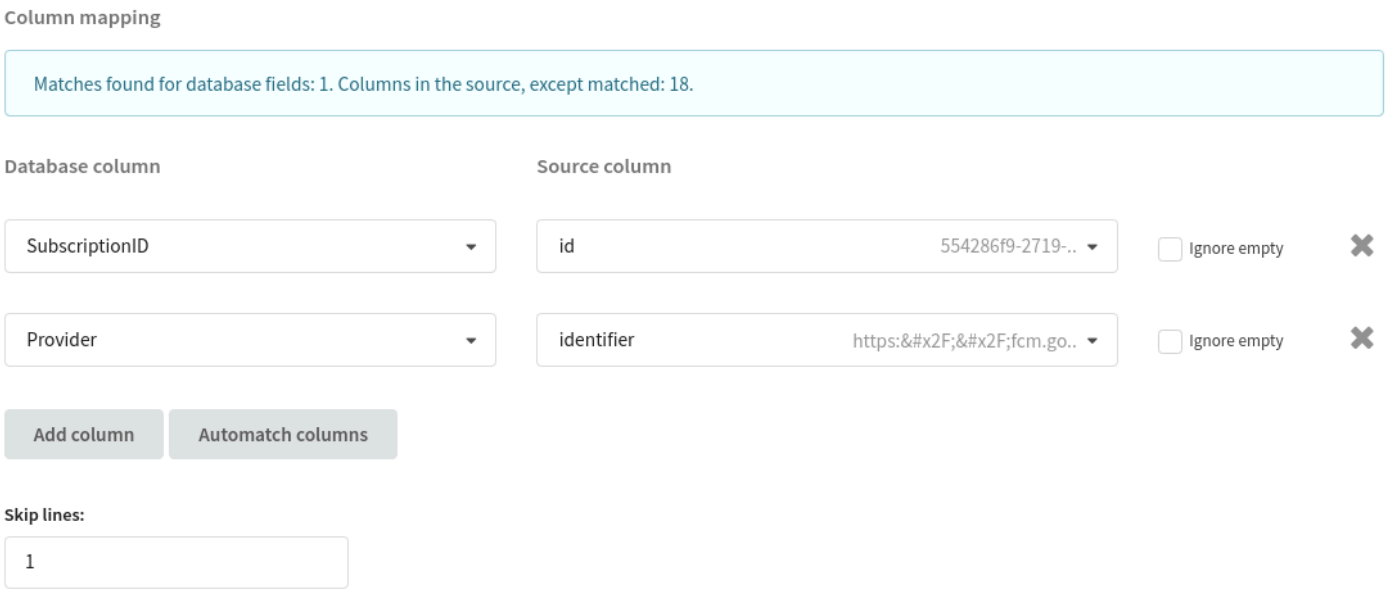
- **SubscriptionID: subscription identifier field from the file;**
- **Provider: Altcraft provider identifier field;**

TIP

If the Altcraft provider identifier field is missing in the csv file, then specify the same field as in the SubscriptionID. This way you'll create a new subscriber for each subscription.

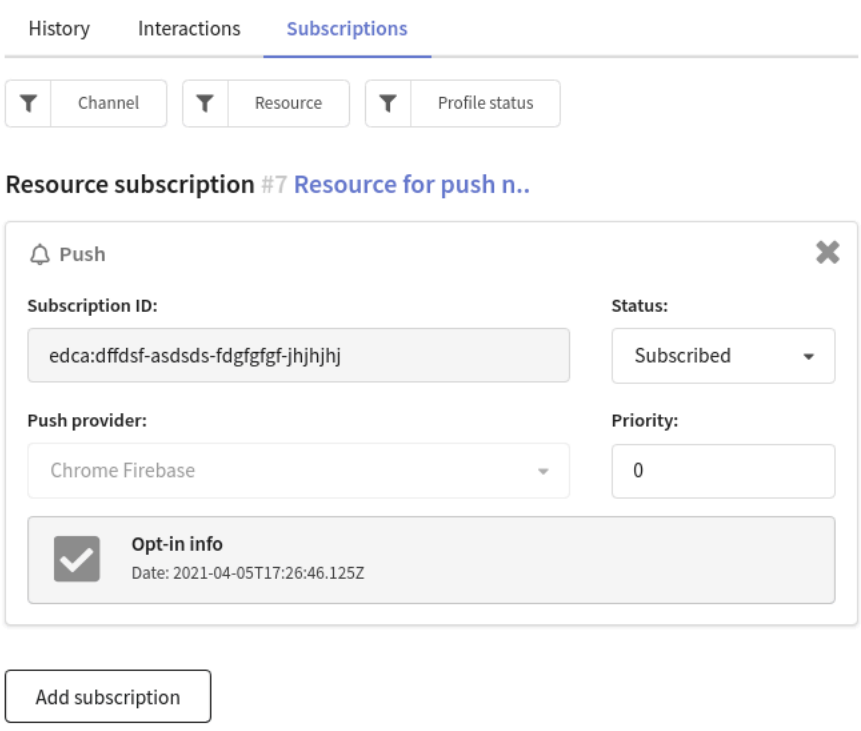
CAUTION

In the Skip Lines field, enter "1" if the file contains headers. Otherwise, the headers will be captured on import.



After you click the Import button in the upper right corner of the screen, the process of importing new profiles into the Database will begin.

Once the profiles have been successfully imported, you can check them in the selected Profiles Database. Open the Subscriptions tab to see the new profilesubscription for push notifications and information about it.



Migration from OneSignal

Step 1. Get VAPID keys from OneSignal service

> CAUTION
>
> **Skip the step if you are migrating subscriptions configured for the Safari browser.**

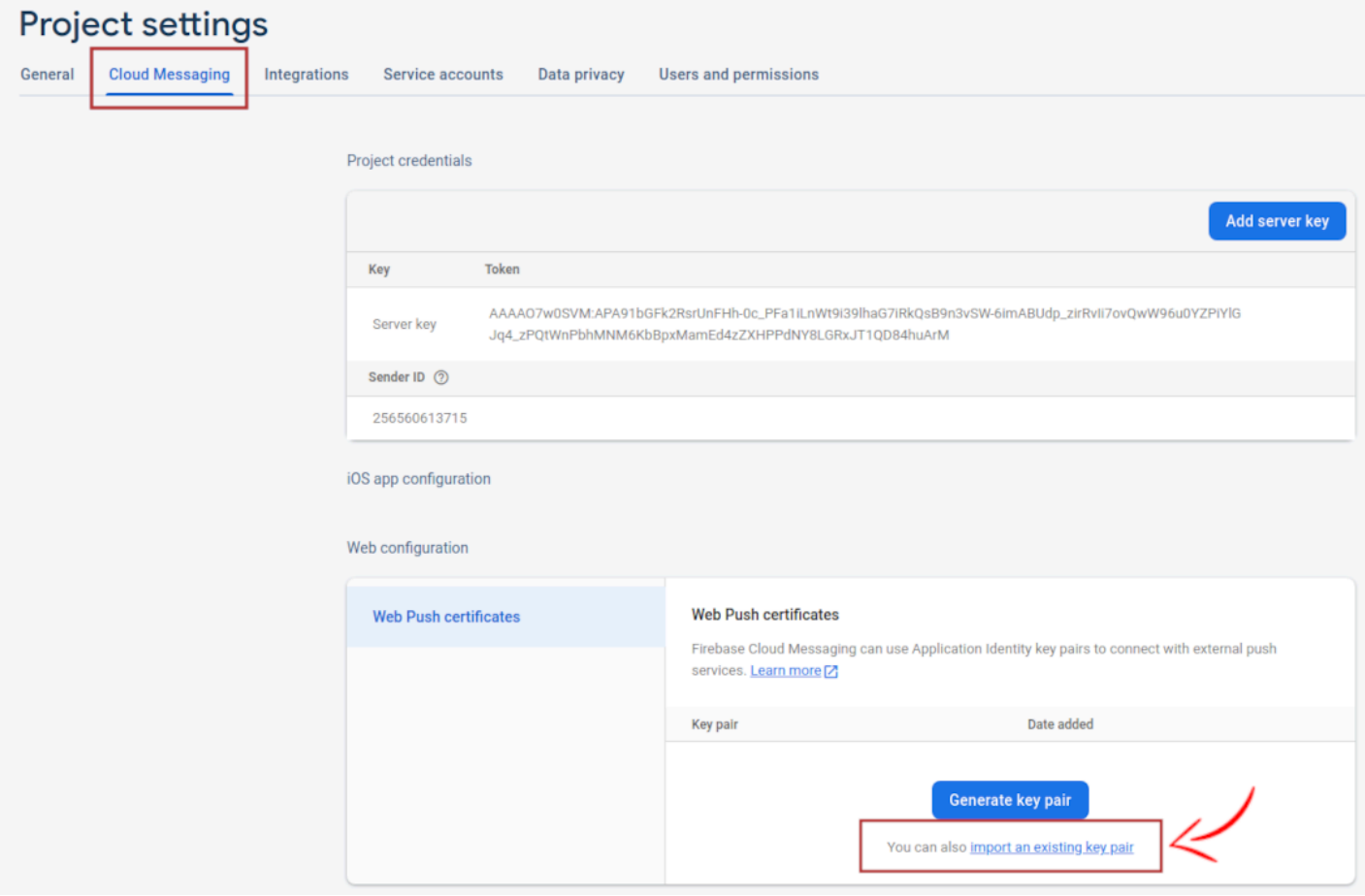
Request the keys from OneSignal technical support. Both private and public keys are required .

> TIP
>
> **VAPID keys and Application Server keys are the same thing.**

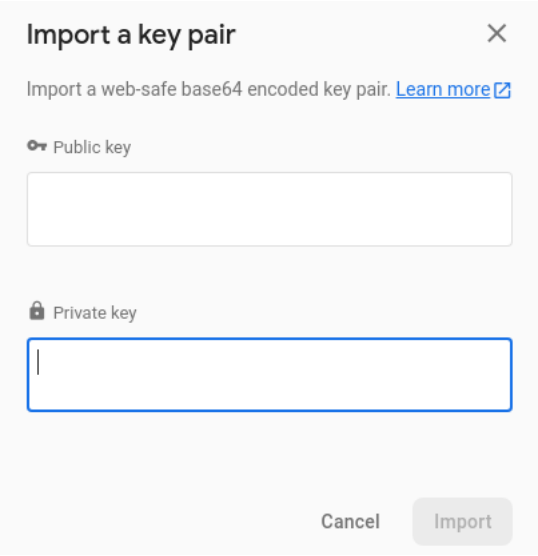
Step 2. Import the VAPID keys into your Firebase project

> CAUTION
>
> **Skip the step if you are migrating subscriptions configured for the Safari browser.**

Go to project settings in Firebase and go to the Cloud Messaging tab. At the bottom of the page, you will see the Web Configuration section. Here you can generate keys yourself, but we do not need this, since you have already received VAPID keys from your service. Click import an existing key pair at the very bottom of the section.



In the new window enter Public key and Private key that you've got from OneSignal technical support.



Step 3. Configure the resource

For information on configuring a resource for Web push notifications, see Web push: settings and implementation.

> **TIP**
>
> **Please note that resource settings differ from browser to browser.**

Step 4. Export subscriptions

Exporting subscriptions involves creating a CSV file that contains all of your current user data.

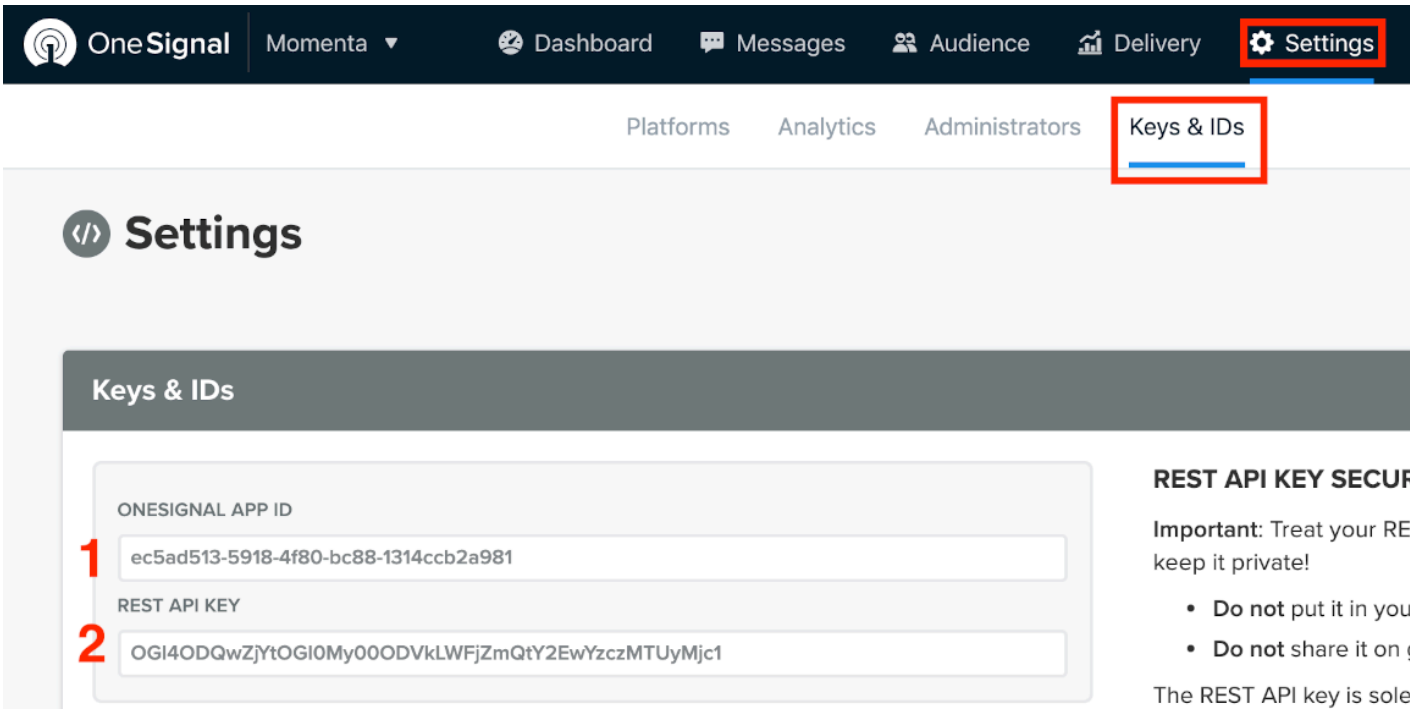
Use the Curl utility to make a POST request. The request will generate a URL that to can use to download a compressed CSV file (.csv .gz).

> **TIP**
>
> **The file may take several minutes to generate depending on the number of users in your app.**
>
> **You can test if it is complete by making a GET request to the `csv_file_url` value.**
>
> **If the file is not ready, a 403 error will be returned. Otherwise the file itself will be returned.**

Below is an example of the request that can be run in the terminal.

```
curl -X POST -H "Authorization: Basic YOUR_REST_API_KEY" -H "Content-Type: application/json" -d '{
    "extra_fields": ["web_auth", "web_p256"],
    "last_active_since": "1469392779", "segment_name": "Active Users"
}' "https://onesignal.com/api/v1/players/csv_export?app_id=YOUR_APP_ID"
```

The request requires your OneSignal App's REST_API_Key and APP_ID. You can find this data on the OneSignal service. Go to Settings and then - to the Keys & IDs tab.



The following parameters must be passed in the request body:

Field	Description
“extra_fields”	**Additional fields that you wish to include. Currently supports `location`, `country`, `rooted`, `notification_types`, `ip`, `external_user_id`, `web_auth`, and `web_p256`.** **In our case, you must specify `web_auth` and `web_p256`.**
“last_active_since”	**Export all devices with a `last_active` timestamp greater than this time.**
“segment_name”	**Export all devices belong to the segment** **You can use this field to export only the segments you are interested in.** **More details about segments in OneSignal: https://documentation.onesignal.com/docs/segmentation.**

TIP

You can get more information on exporting subscriptions from OneSignal here: https://documentation.onesignal.com/reference/csv-export.

Step 5. Provide the csv file with these fields to the administrator or Altcraft support service

CAUTION

Skip the step if you are migrating subscriptions configured for the Safari browser.

If the Altcraft platform is deployed inside your business (On-Premise solution), transfer the generated csv file with subscriptions to the platform administrator.

If the platform is installed on the cloud (SaaS solution), transfer the csv file to Altcraft support.

Step 6. Take the processed csv file from the administrator

CAUTION

Skip the step if you are migrating subscriptions configured for the Safari browser.

Step 7. Import profiles

Now you can import profiles into Profile Database from a csv file. During the import a subscription to the required provider will be created.

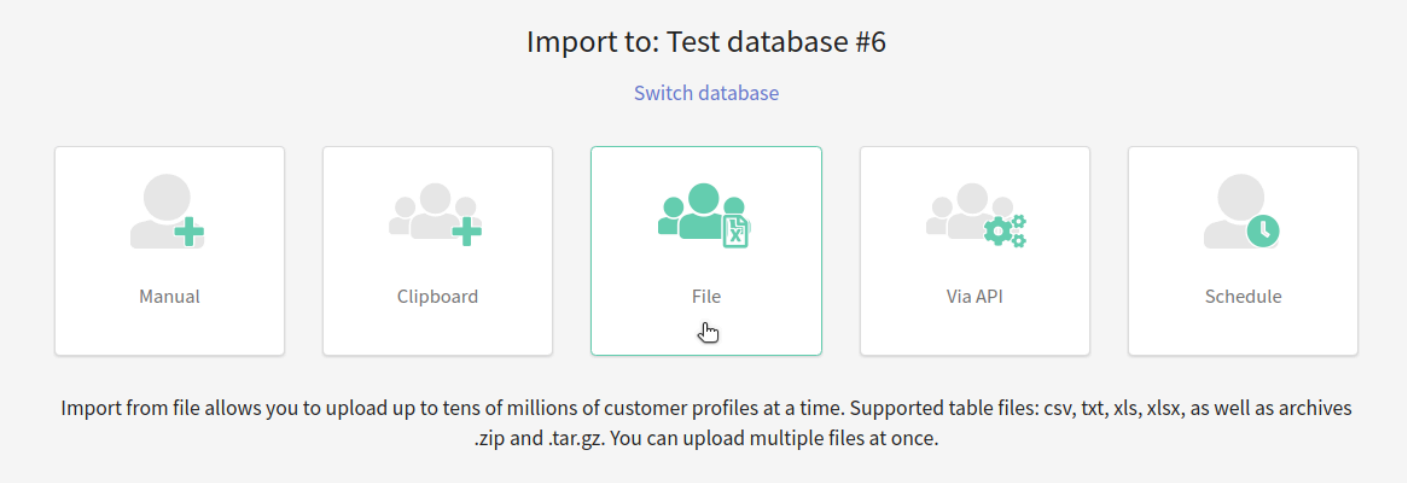
In addition, you have the opportunity to update the database by creating a subscription to the same provider. Please note that in this case it must be one of the Firebase providers.

TIP

The provider identifier allows you to find out which browser the subscription belongs to and through which service it was created. Examples of providers: Chrome Firebase, Apple Safari, Mozilla Firefox.

So, create or select a database and click Import customer profiles in the drop-down list.

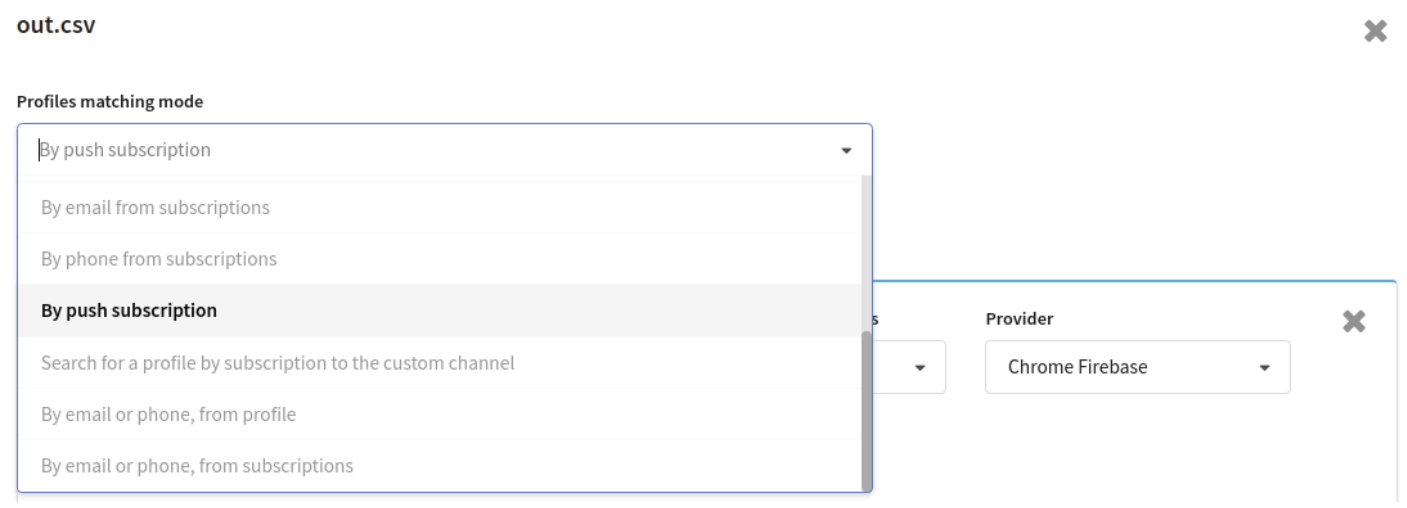
Select the import method - File. Upload the required csv file with subscriptions.



Click Profiles matching mode and select By push subscription in the drop-down list.

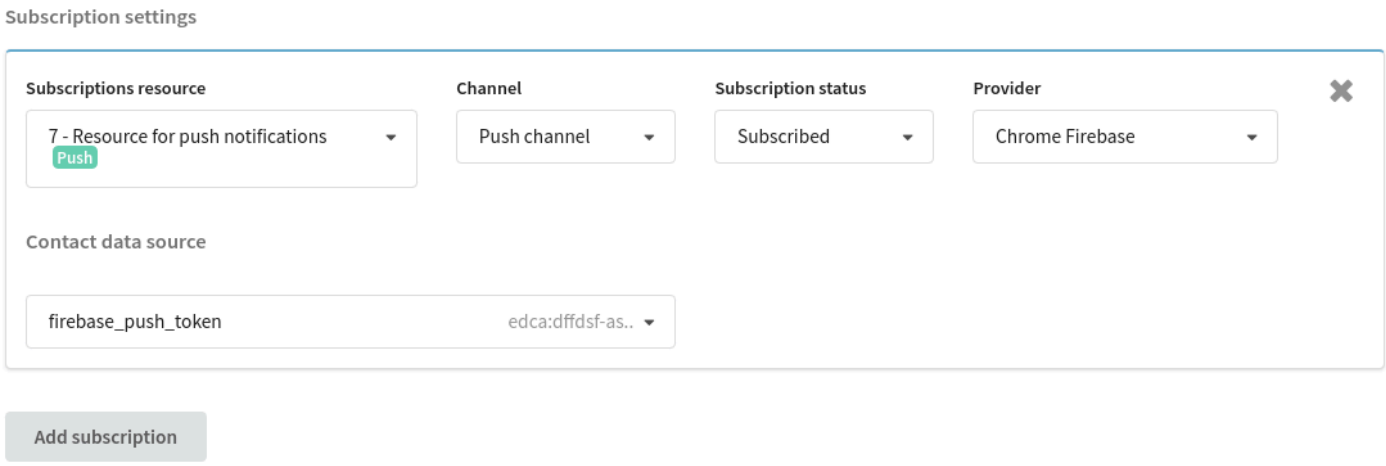
CAUTION

If your csv file contains any data that can be used to configure correspondences between profiles (“match users”), then before the import process, select the appropriate profile search mode.



Click the “Add subscription” button and then configure the subscription:

1. **Channel is Push channel;**
2. **Provider is the browser through which the subscription was made. If the user signed up using the Safari browser, then specify Apple Safari. If you don't know the subscription browser, you can import the subscriptions with the Chrome Firebase provider.**
3. **Contact data source is the subscription id field from csv file. If the user signed up through the Safari browser, then you need to define this field yourself. For other browsers, you need to specify firebase_push_token here.**

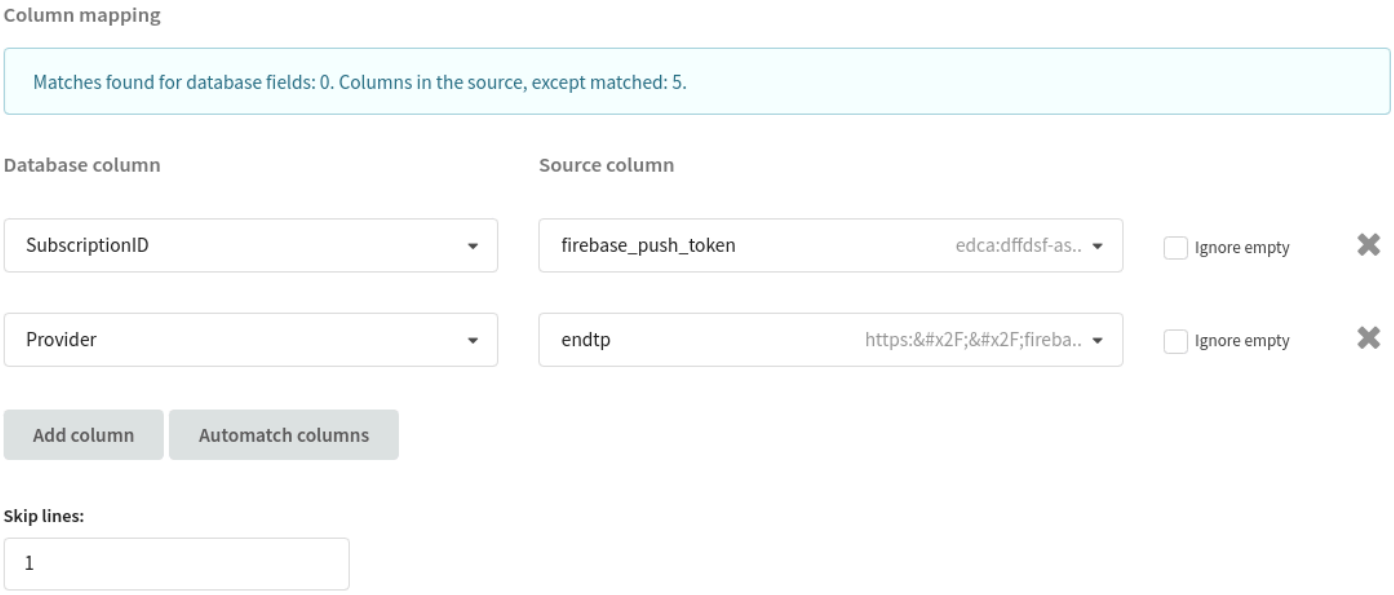


Now it's time to configure column mapping:

Field	For Safari	For other browsers
SubscriptionID	Subscription ID field from csv file	firebase_push_token
Provider	If the Altcraft provider identifier field is missing in the csv file, then specify the same field as in the SubscriptionID. This will create a new subscriber for each subscription.	

CAUTION

In the Skip Lines field, enter "1" if the file contains headers. Otherwise, the headers will be captured on import.



After you click the Import button in the upper right corner of the screen, the process of importing new profiles into the Database will begin.

Once the profiles have been successfully imported, you can check them in the selected Profiles Database. Open the Subscriptions tab to see the new profile subscription for push notifications and information about it.

History Interactions Subscriptions

Channel Resource Profile status

Resource subscription #7 Resource for push n..

Push

Subscription ID:

edca:dffdsf-asdsds-fdgfgfgf-jhjhjhj

Status:

Subscribed

Push provider:

Chrome Firebase

Priority:

0

Opt-in info

Date: 2021-04-05T17:26:46.125Z

Add subscription

SMS: ready-made integrations

Description

Altcraft Platform uses integrations with gateways of mobile operators to send SMS to your customers. If you can't see operator you need in the list below - we can develop a custom integration during a pilot project (on-premise product trial). Contact support@altcraft.com for details.

Integrations

The following integrations are available out of the box:

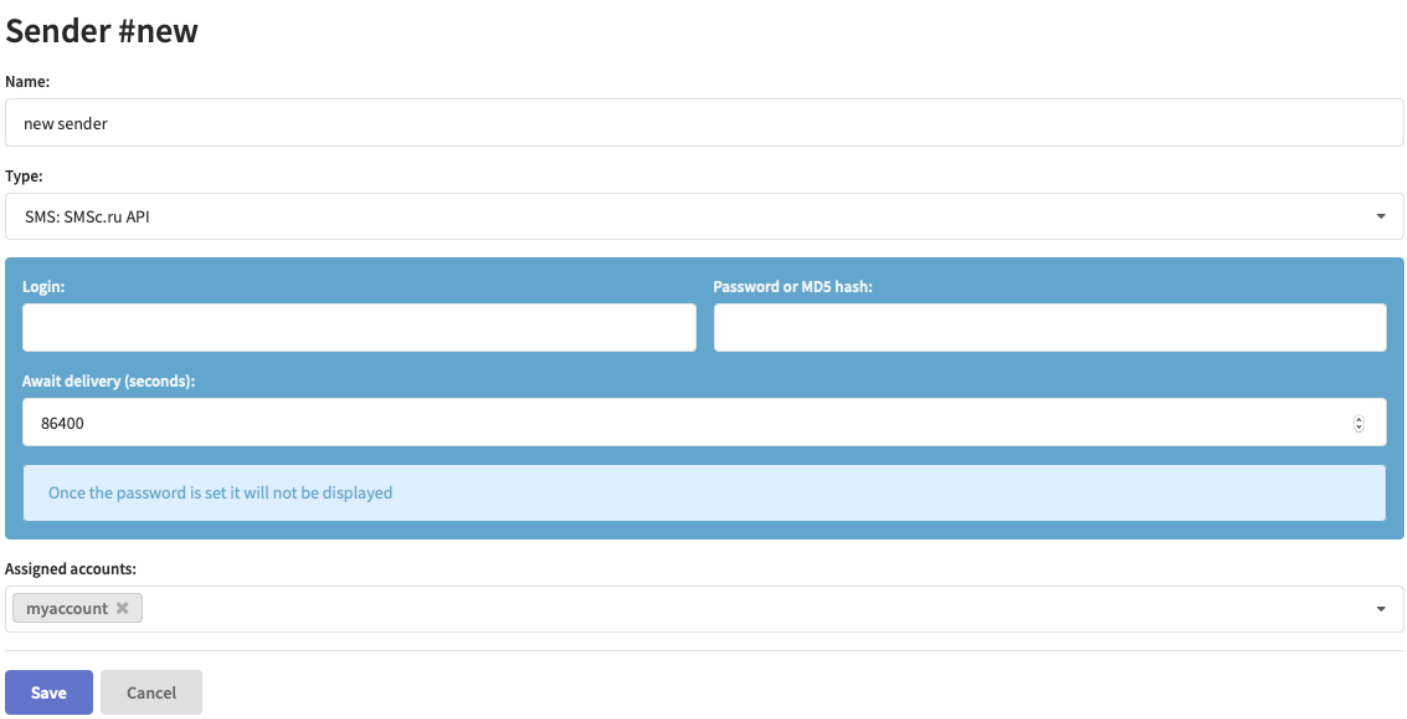
- **SMSc.ru — SMS gateway by "СМС-Центр". Link to documentation: https://smsc.ru/api/**
- **SMSimple.ru — SMS gateway by "Юлабс". Link to documentation: http://www.smsimple.ru/api/**
- **BulkSMS.com — SMS gateway by "Celerity Systems". Link to documentation: : https://www.bulksms.com/developer/json/v1/#**
- **Devino.Online SMS — integration with API of Devino Telecom gateway. Link to documentation: https://docs.devino.online/0.2.0**
- **SMSTraffic — SMS gateway by "СМС-Трафик". Link to documentation: https://lk.smstraffic.ru/techdocs**
- **Infobip — SMS gateway by "Infobip".Link to documentation: https://www.infobip.com/docs/api#channels/sms/send-sms-message**
- **MTS — SMS outlet by "МТС". Link to documentation: https://support.mts.ru/mts_marketolog/Rassilki-po-svoei-baze-PRO-i-API-k-nim/dokumentatsiya-rest-api**
- **MFMS EDNA — SMS gateway by "EDNA". Link to documentation: SMS-MFMS-PROTOCOL-HPG.pdf**
- **Sky Media — SMS gateway by Sky Media.Link to documentation: https://sky-ms.ru/api.html**
- **Play Mobile — SMS gateway by Play Mobile (documentation).**
- **Rapporto — multipurpose platform Rapporto (ex. Zagruzka). Link to documentation: https://doc.rapporto.ru/api/http/http_main.html**
- **VK Notify — SMS gateway by Notify.**
- **SberMobile - SMS gateway by SberMobile. Link to documentation: SMS-SBER-PROTOCOL.pdf**
- **Stream telecom — SMS gateway by "Stream telecom". Link to documentation: https://stream-telecom.ru/solutions/integrations/**
- **Intellin — SMS gateway by Intellin.**
- **Seven Tech — integration with the SEVENTECH SMS gateway. Documentation: https://docs.seven.tech/**

TIP

For proper operation of certain integrations, you need to provide specific information to the provider (for example, to receive statuses of sent SMS messages, you must provide a Rapporto URL to accept delivery status reports), or specify a provider-specific parameter. To do this, contact the platform administrators.

Sender configuration

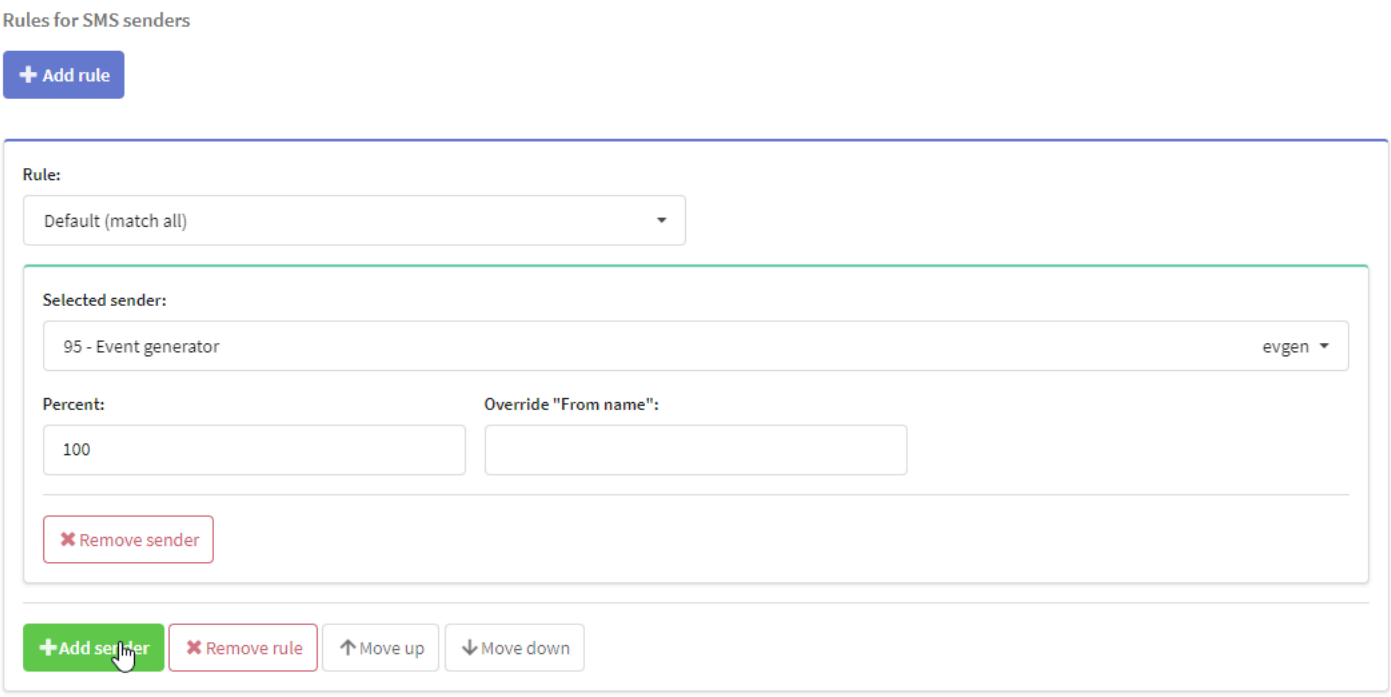
To start messaging using a ready-made integration, get the service account and credentials and add a new sender in Altcraft Marketing Administrative panel:



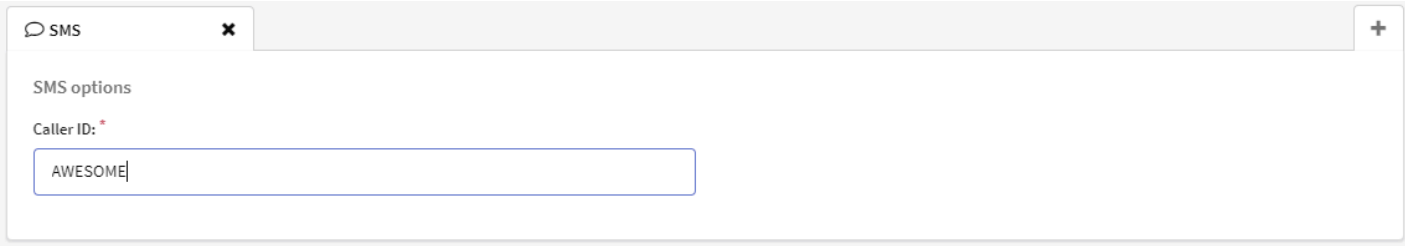
NOTE

Altcraft Marketing also supports Faktura SMS outlet. For details please contact support@altcraft.com.

Then add your sender to one of virtual sender rules for SMS:



After you get a CALLER_ID from your telecom service, enter it in your Resource SMS tab:



First SMS Mailing

	This tutorial will help you to create and launch your **SMS mailing in the Altcraft Platform from scratch.** **SMS** as a channel of communication with clients is impossible to replace. It does not matter if a customer has Internet access, because today everyone has a mobile phone. The subscription process is also simpler than in other channels: the only thing you need to get started is a customer's phone number. Yes, there is no way to add media content to SMS, but you can add links to resources on the Internet. Altcraft Marketing automatically **shortens** links in a message and also **tracks** recipients' clicks. </td> </tr>

Step 1: Connect SMS gateway

Altcraft Marketing uses integrations with gateways of mobile operators to send SMS to your customers.

> TIP
>
> **You can read about our ready integrations in this article.**
>
> **In case you use different SMS gateway, write to us at support@altcraft.com and we will agree with you the terms of connecting the new gateway.**

At first you'll have to connect the SMS sender. You need the authorization data of the telecommunications service provider, as well as the CALLER ID agreed with it.

> CAUTION
>
> **If you use an Altcraft Marketing cloud account, please give the information to the support team at support@altcraft.com. They will help you to connect the integration.**

If the Altcraft Marketing platform is installed on your equipment (On-Premise), you need to configure SMS sender in the platform administrative panel. Contact your platform administrators to set up senders and virtual senders using the instructions below:

- **Senders are configuration records that combine parameters for sender identification, sending speed, and settings for the sender's behavior in case of sending errors.**
- **Virtual senders are the rules for sending from multiple senders. Can be configured both in the Administration Panel and in the User Interface.**

> TIP
>
> **Or just contact our support team — and we will help you set up SMS sending.**

Step 2: Subscribe customers to the SMS channel

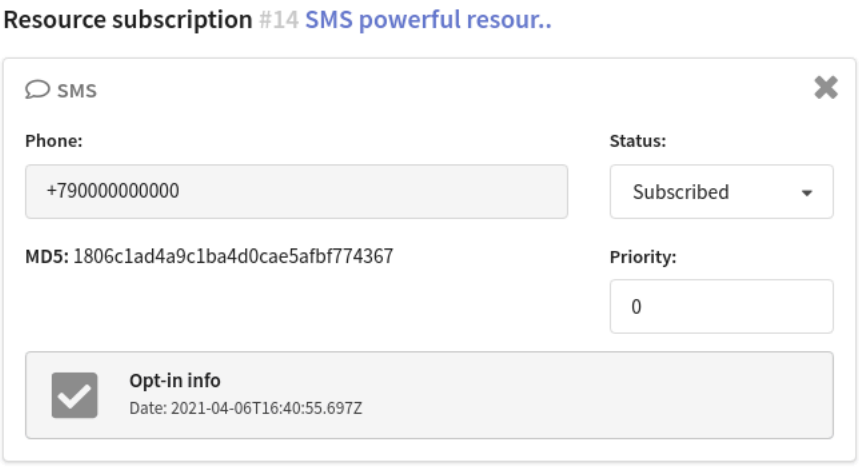
Profiles Databases

Altcraft Marketing mailings are received by your subscribers. Subscriber profiles are stored in profiles databases.

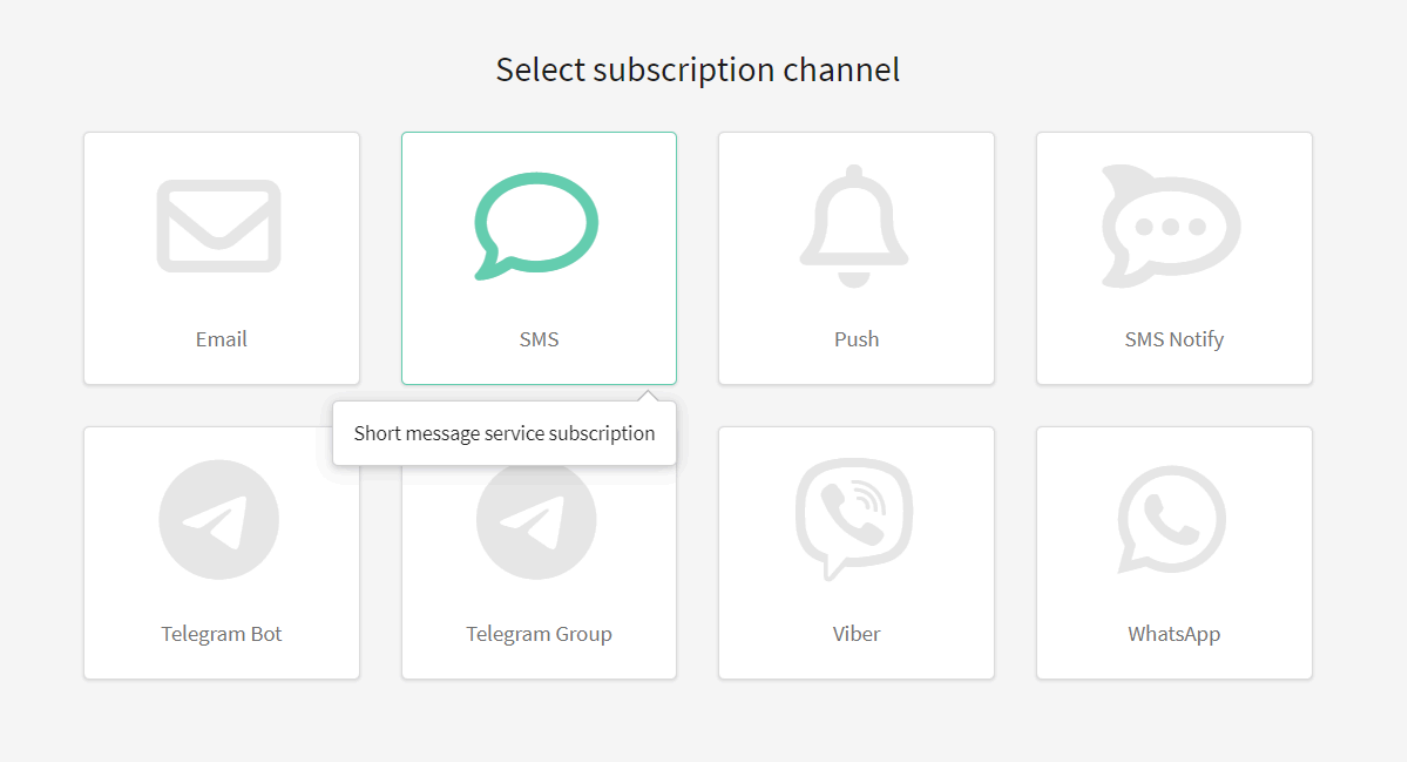
Each customer profile has a system field "Phone". You can store your customers' phones there. However, it is used not to send, but to identify a customer in the platform.

Subscription resources

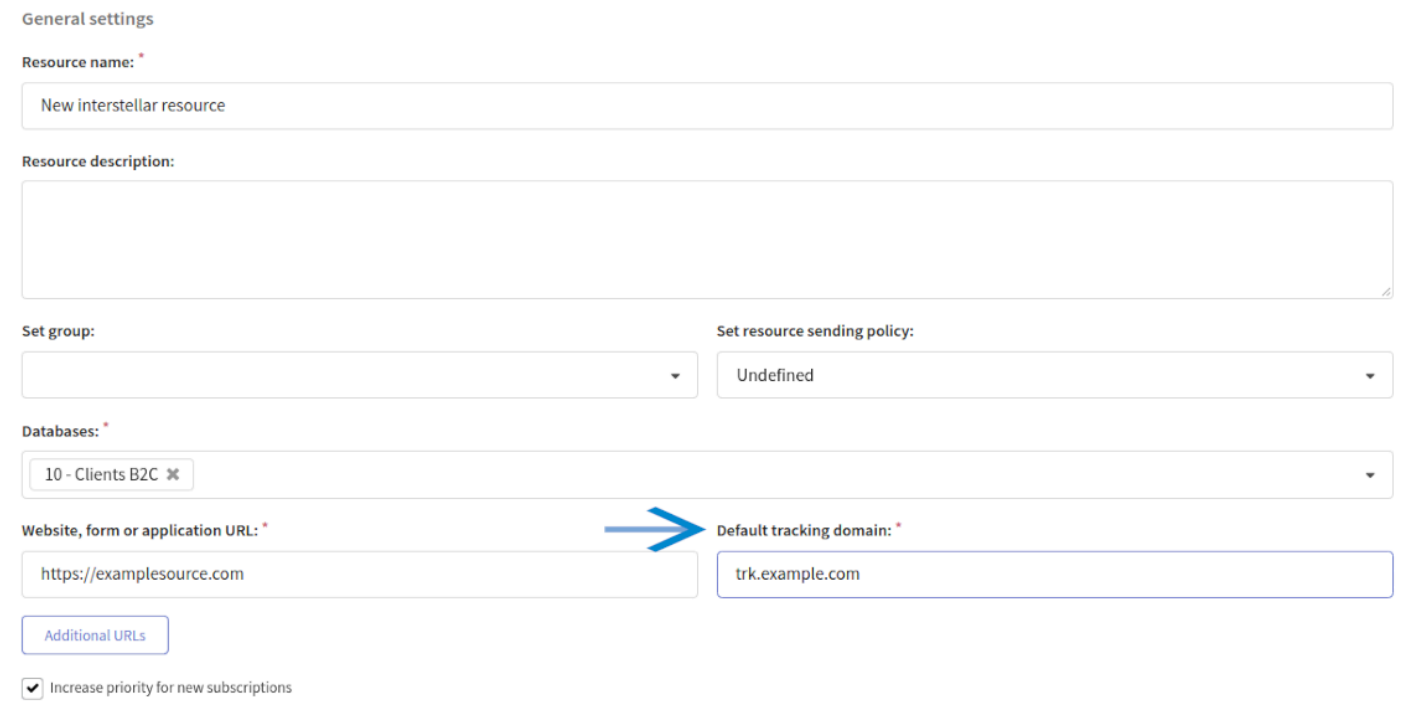
When you start your SMS mailing, you use profile subscriptions to resources. This way, you can notify the client by several phone numbers. Each client profile can be subscribed to one or more resources . What can be a resource? A specific marketing area, a site, an application, or the type of content.



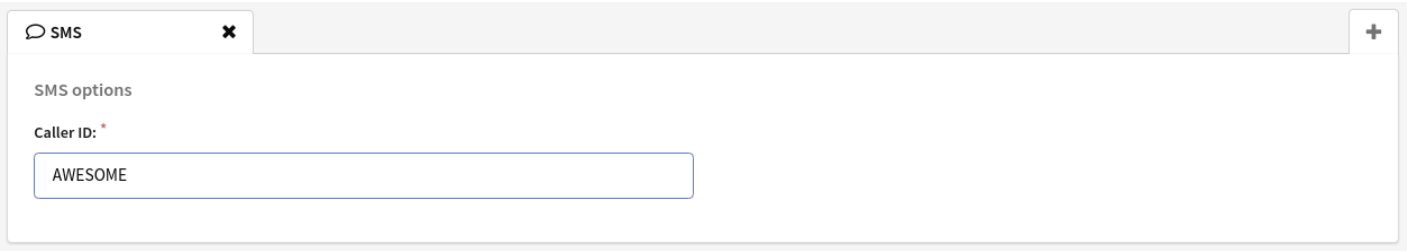
Select the corresponding item in the main menu, then create a new resource or select an existing one. Select an SMS channel for a new resource:



In the general settings, add the necessary databases. If it's necessary, assign a sending policy and specify the Default tracking domain. This domain will process your customers' clicks on links in SMS messages:



In the SMS options of the resource channel Caller ID agreed with your telecommunications service provider:



Customer profiles and subscription to the resource

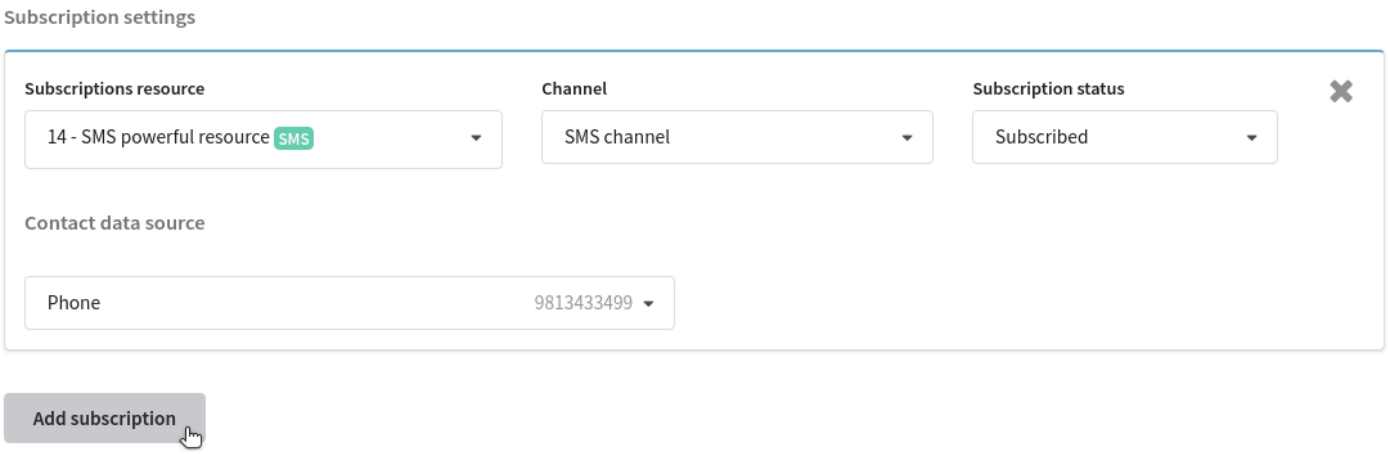
So, we've just configured the profiles database and subscription resource. Now it's time to import customer data into the platform. For each client, Altcraft Marketing creates a single digital profile that includes personal data, history of actions and subscriptions to your resources.

You can upload customer data manually: one at a time, in a list, or from a table file. Different import methods are described in this article.

You can also set up automatic import of client profiles:

- **on schedule from your SQL database**
- **through the use of the integration with platform API**

When you import data, sign clients to the previously prepared resource. To do this, select SMS as the communication channel and specify the column with subscribers' phones:



TIP

If you already have a profiles database, add them a subscription to a new resource. There are two options:

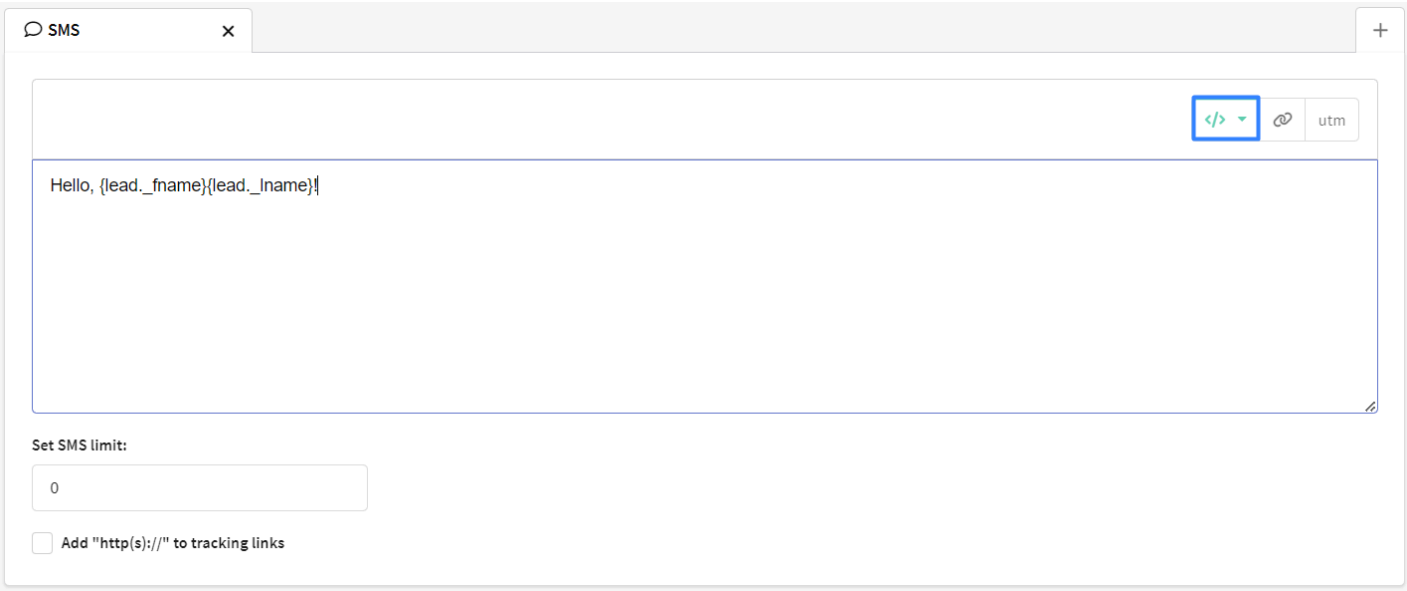
- **if customers have subscriptions to another resource with up-to-date SMS phone numbers — use bulk profile update**

- **if you need to upload to the platform SMS phone numbers for subscription, use the profile import in the “update only” mode**

Step 3: Prepare content for submission

The content for your SMS mailings is created in Message Templates. The templates contain content for all available channels. When you create a new template, select the SMS channel - or add it for an existing one.

The SMS version of the template is a simple text editor where you can add data of a specific client, as well as dynamic text content from external sources:



INFO: Notice

When a template includes logical expressions or loops, the platform counts every formatting character — spaces and line breaks — as part of the final message. Each space or line break adds one character to the SMS and may affect the cost. To save characters and control message length, keep logical constructions on one line.

For example, instead of:

```
{if lead._city equal "Moscow"}
We invite you to a conference in Moscow on June 30!
{else}
We invite you to a conference in your city!
{end}
```

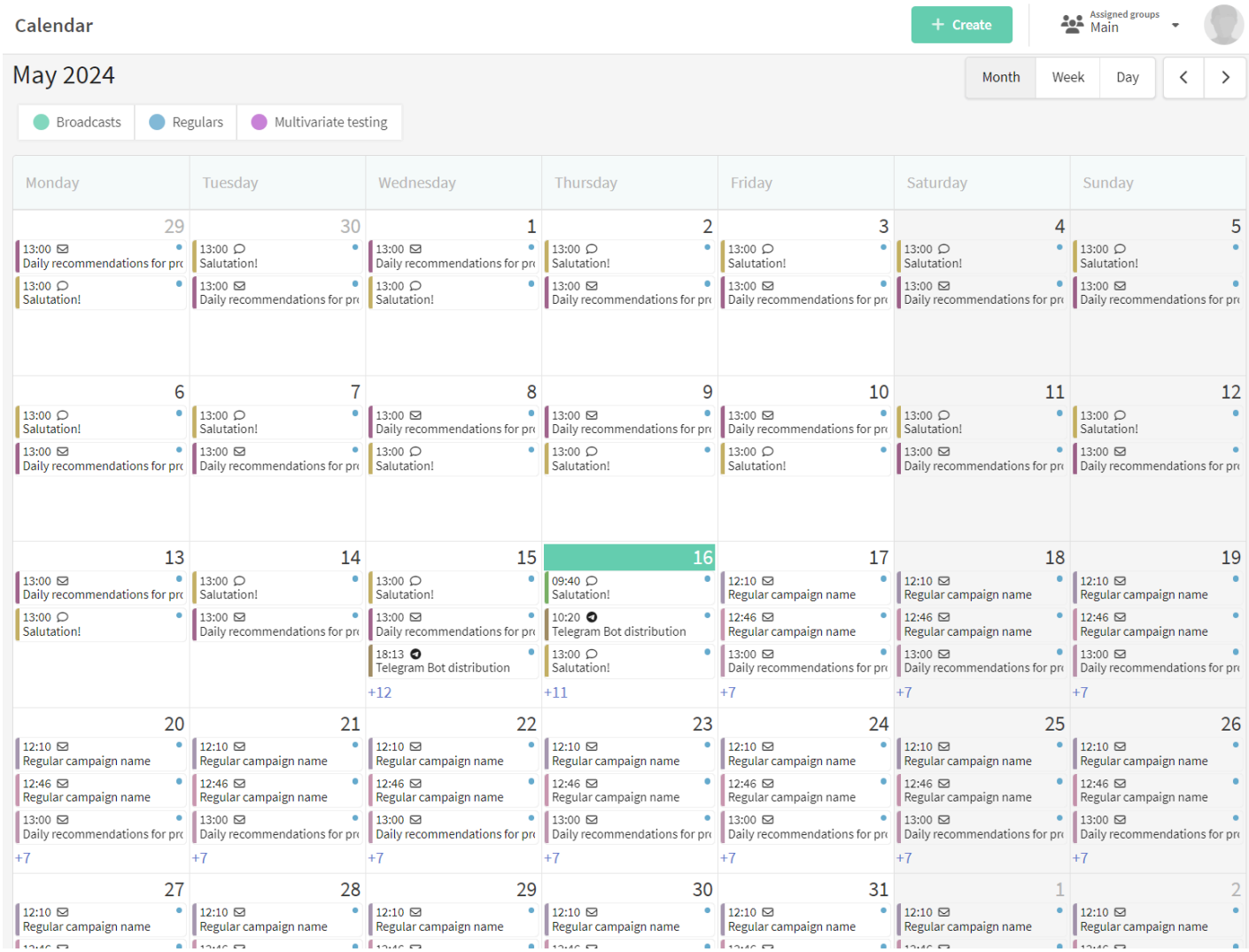
Use:

```
{if lead._city equal "Moscow"} We invite you in Moscow on June 30!{else}We invite you in your city!{end}
```

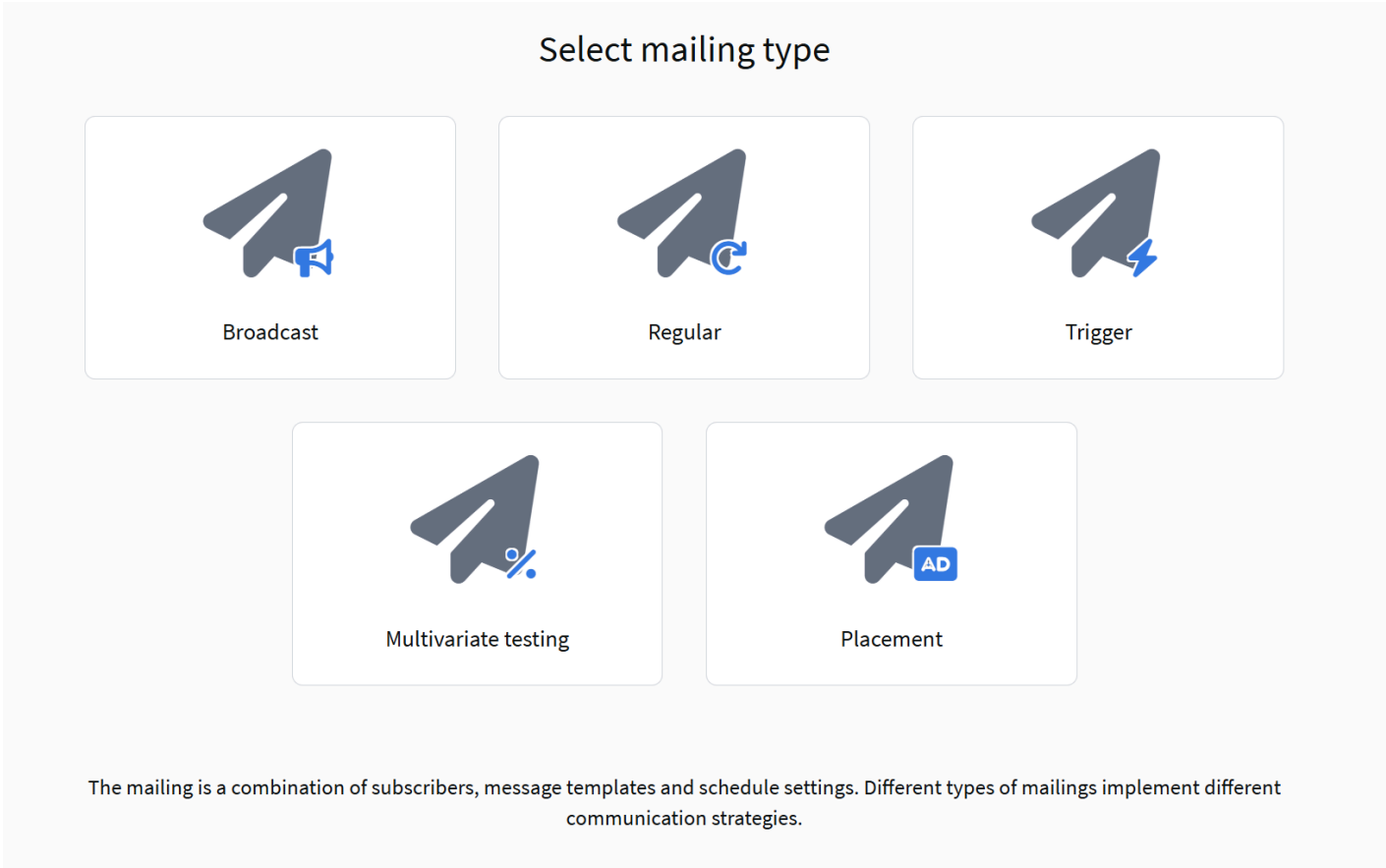
This approach keeps the logic intact and avoids unexpected SMS length growth.

Step 4: Select the type of email mailing

	If you want to send an SMS message, you need to use such a tool as a **Mailing** Do you want to send out a **one-time mailing** to all profiles of the database or to the audience of the specific segment? Choose **Broadcasts** If the mailing is periodic, your choice is a **Regular** mailing. In the main menu select **Mailing** and open **Calendar.** This is where you'll find all scheduled mailings. </td> </tr>



Decide on a date and create a mailing of the required type:



The mailing is a combination of subscribers, message templates and schedule settings. Different types of mailings implement different communication strategies.

TIP

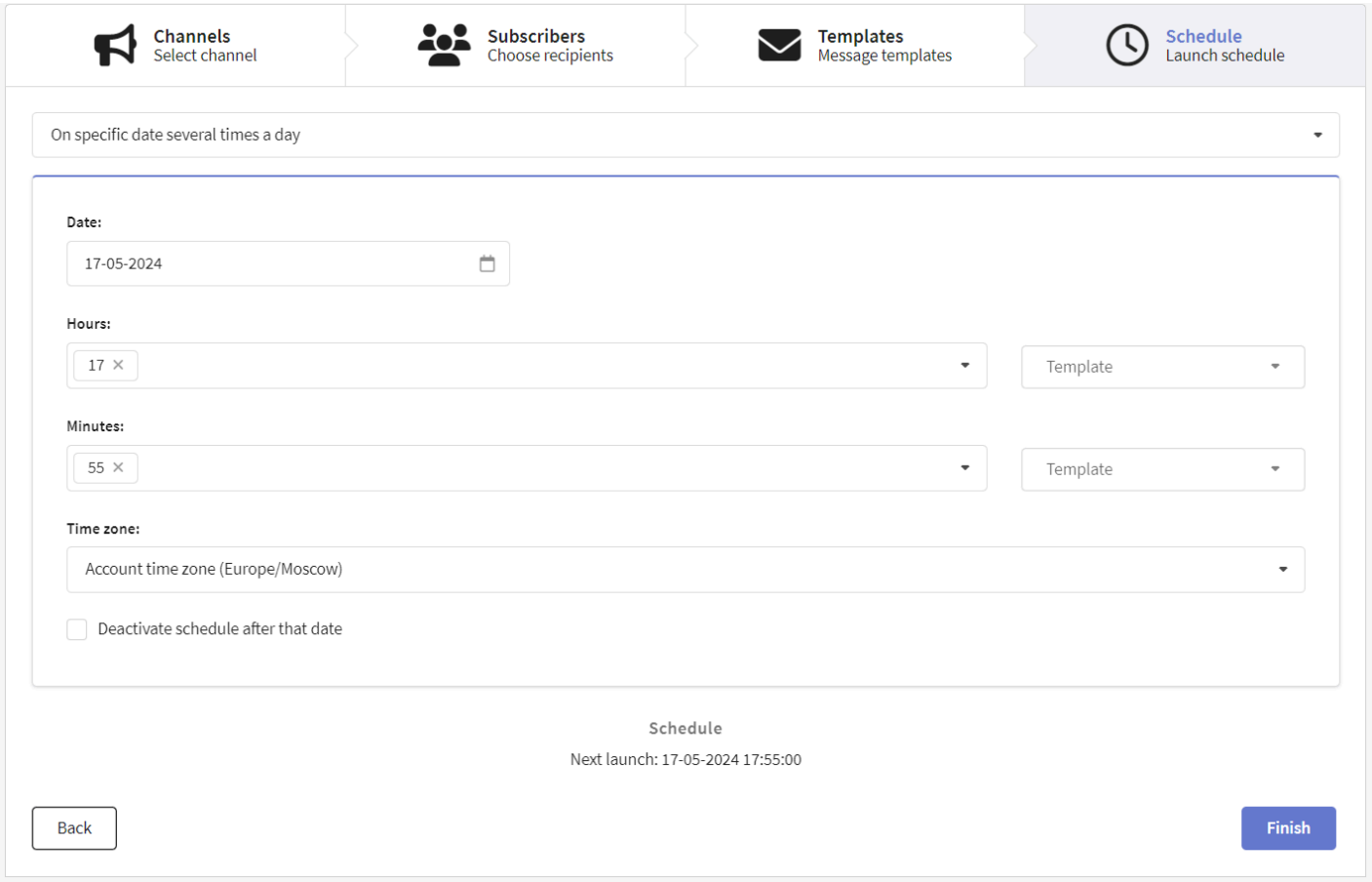
You can find details about how to set up Broadcast on this page.

TIP

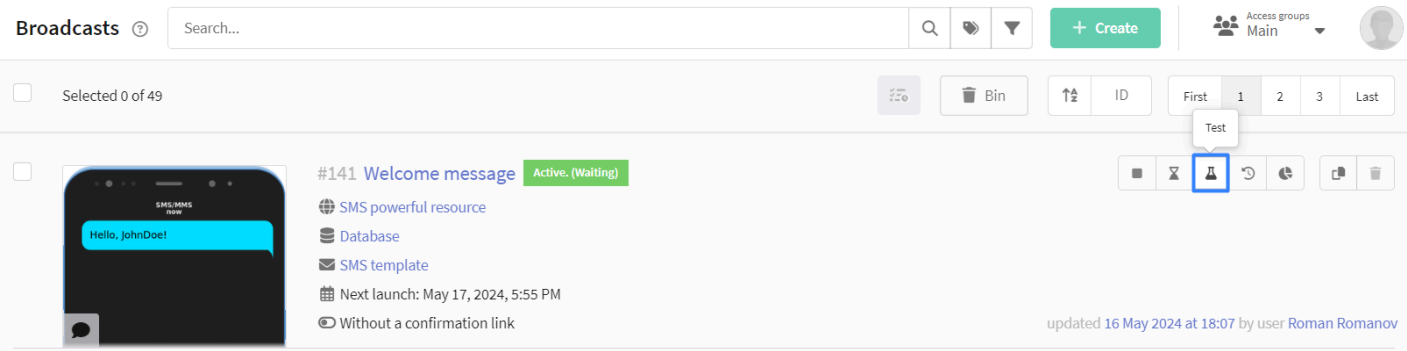
You can find details about how to set up Regular mailing on this page.

Once you're done with the general mailing settings, you need to:

1. **Select **SMS** as the communication channel;**
2. **Select the subscription resource you've just configured and the audience of recipients;**
3. **Add a message template to your message;**
4. **Set a schedule. For example, for a broadcast the option "On specific date at specific time" is available.**



Once the mailing is set up, save it. Later you can find it in the list of mailings of the corresponding type. There is a quick action menu on the right side of each mailing. Use it to test the mailing — and if everything is successful, activate it:



Step 5: Assess the effectiveness of the email mailing

Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

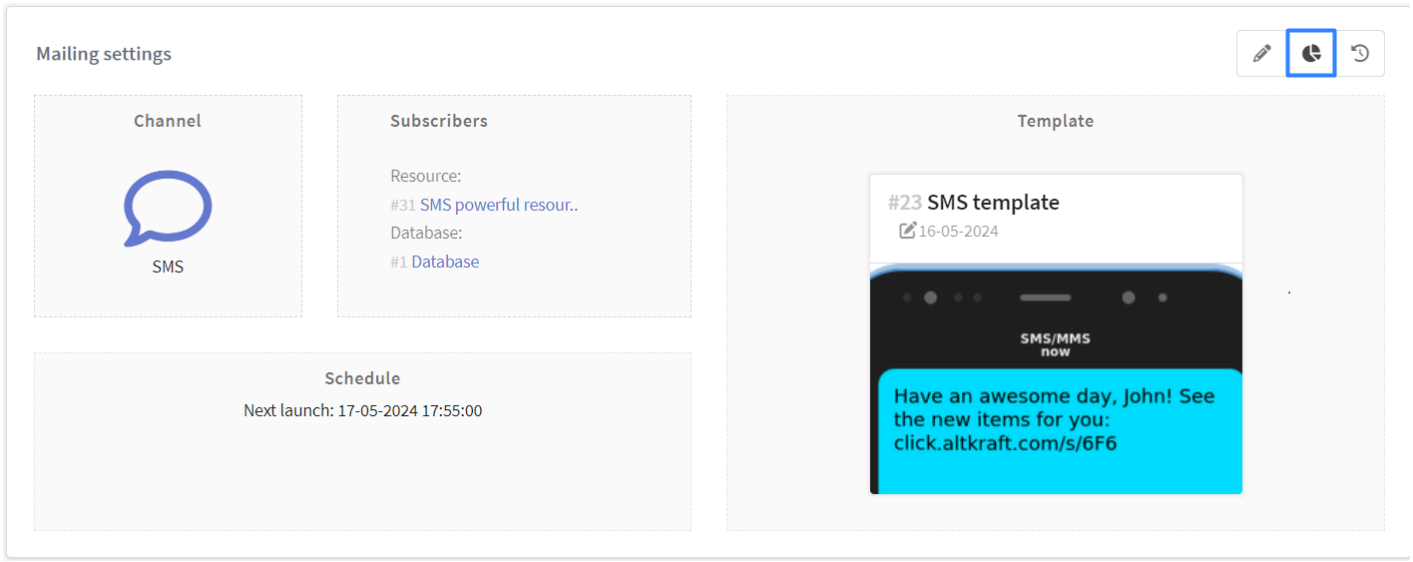
Type	Description
Mailing started	**The timestamp when the mailing execution began**
Limit	**Indicates whether a sending limit is set**
Audience source	**Contact source:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database being used**
Spent getting segment cursor	**Time spent loading contact data from the source (in milliseconds)**
Number of subscribers received from the source	**Total number of contacts in the specified source**
Number of subscribers found	**Number of contacts with active subscription to the selected resource**
Duplicate phones	**Count of duplicate phone numbers (excluded from sending)**
Number of SMS recipients	**Final count of successful SMS deliveries**
Mailing finished	**Completion timestamp of the mailing execution**
Total time spent	**Total duration from mailing start to completion (in seconds)**

Reports and Analytics

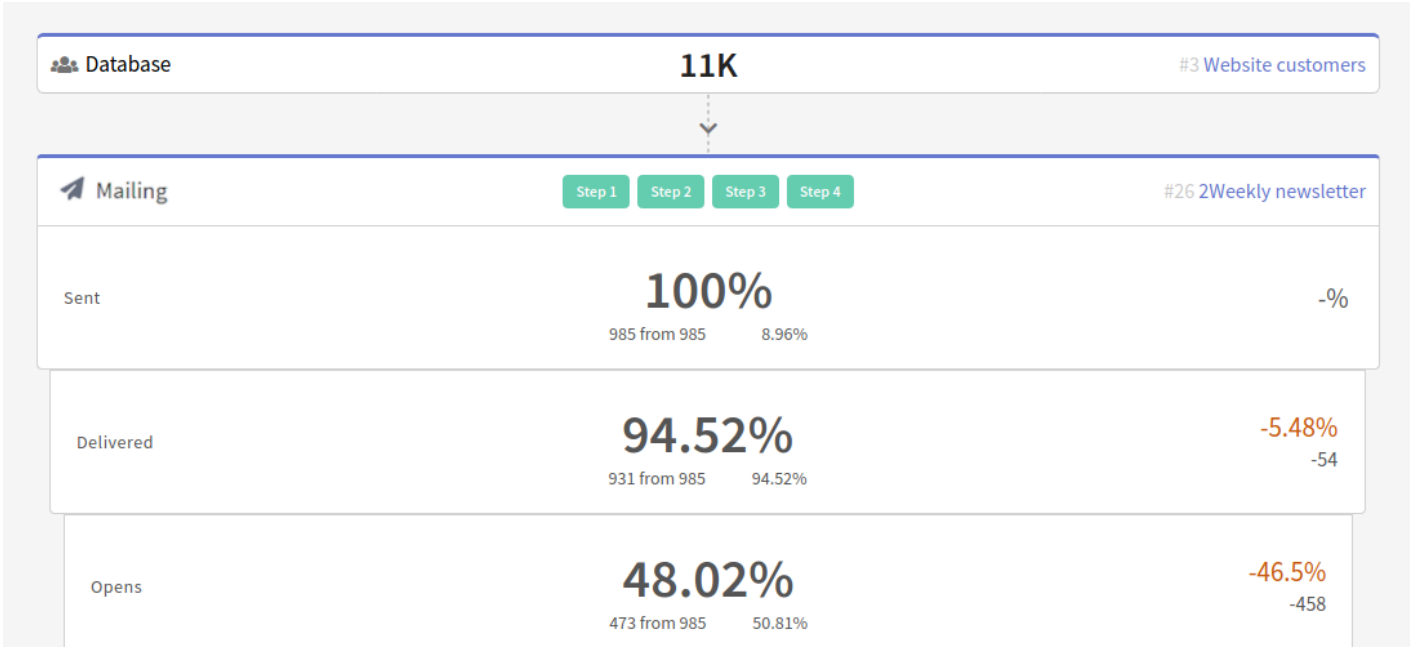
To evaluate the effectiveness of your email mailings go to the section Analytics in the main menu of the platform.

The channel report provides analytics for one or several mailings, for one or several communication channels. For the convenience of grouping, you can select a preset of indicators for SMS:

You can also go to the channel report for a specific mailing by following the link in the mailing editor:



**Actions in the SMS mailing or in the template can turn into steps in the [interactive conversion funnel](): **



Telegram

This section contains guides for working with Telegram in Altcraft Platform.

Telegram Bot — a channel for personalized messages. Subscribers interact with the bot individually, receive personal notifications and mailings. Suitable for customer communication, sending receipts, codes, order status notifications. Requires passing each subscriber's chat_id to the platform.

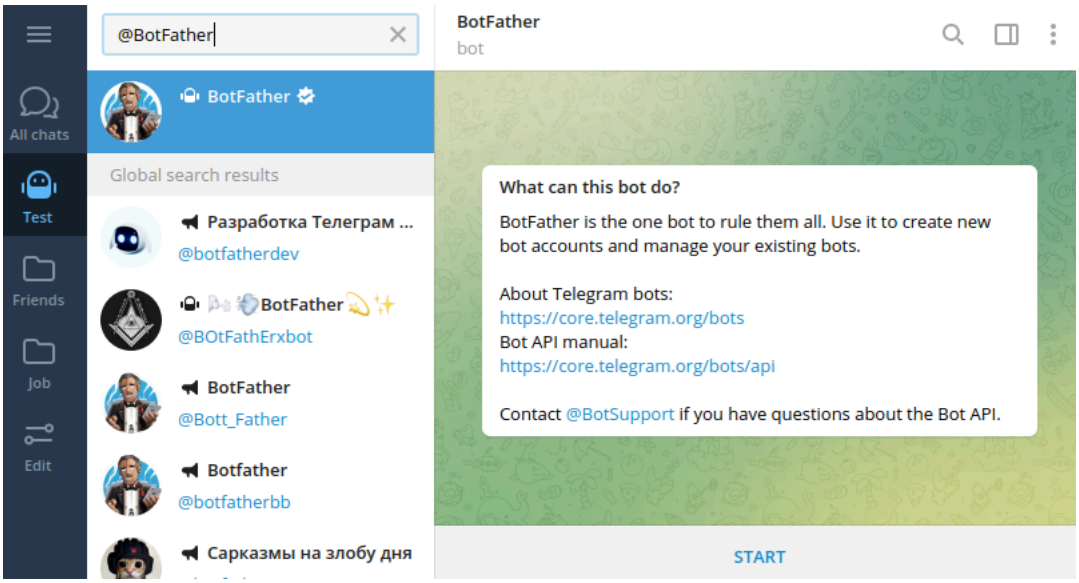
Telegram Group — a channel for publishing posts in groups and channels via placements. Suitable for mass content and announcements in public chats. Does not require individual profile subscriptions — just add the bot to the group with administrator rights and get the chat_id of the channel or group.

Telegram Bot: a Guide for the First Mailing

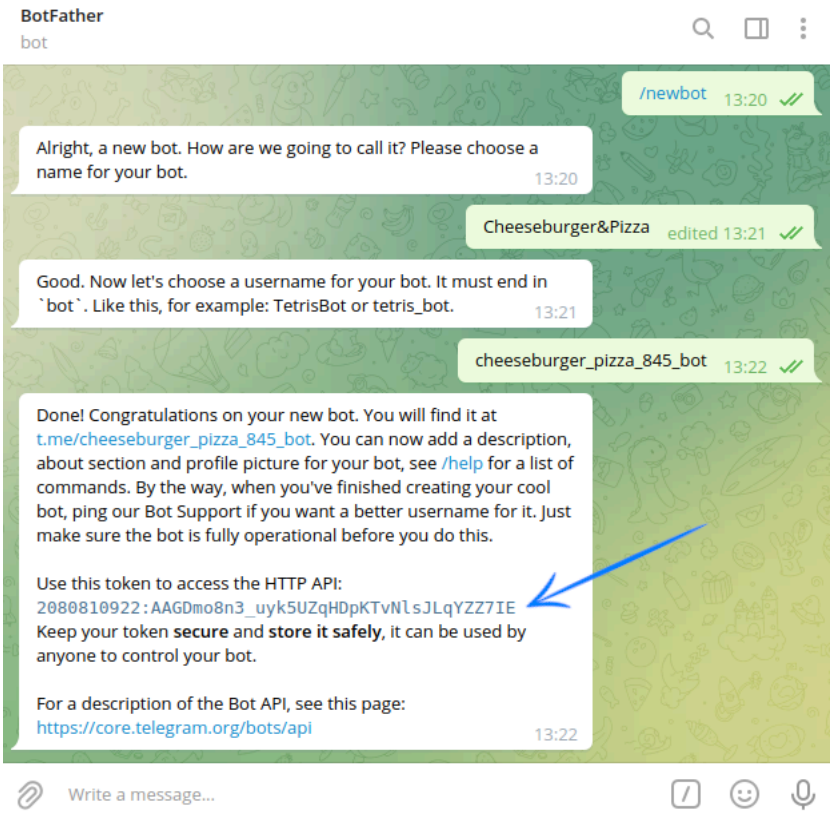
	A bot is a direct channel that allows you to personally send messages to subscribers who have entered into a chat with the bot. Read this guide to set your first mailing using the Telegram bot.

Step 1. Create a bot and get a token

1. Log in to Telegram, enter `@BotFather` in the search box and select the desired bot.



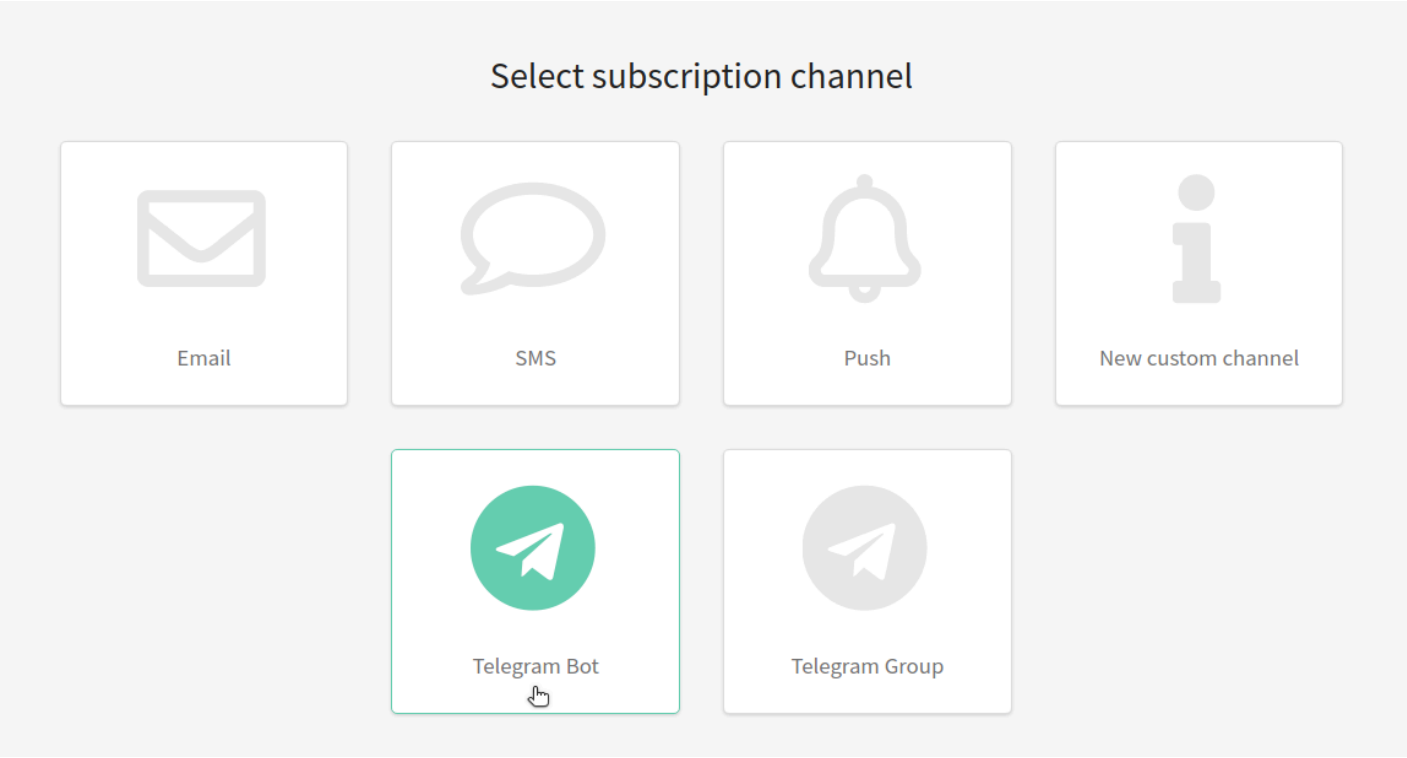
2. Start a dialogue with the bot by clicking the START button in the chat. Then send the `/newbot` command to it. Enter the name and then the username for your new bot. Please note that the username must end with `"bot"` or `"Bot"`. Copy the received bot token:



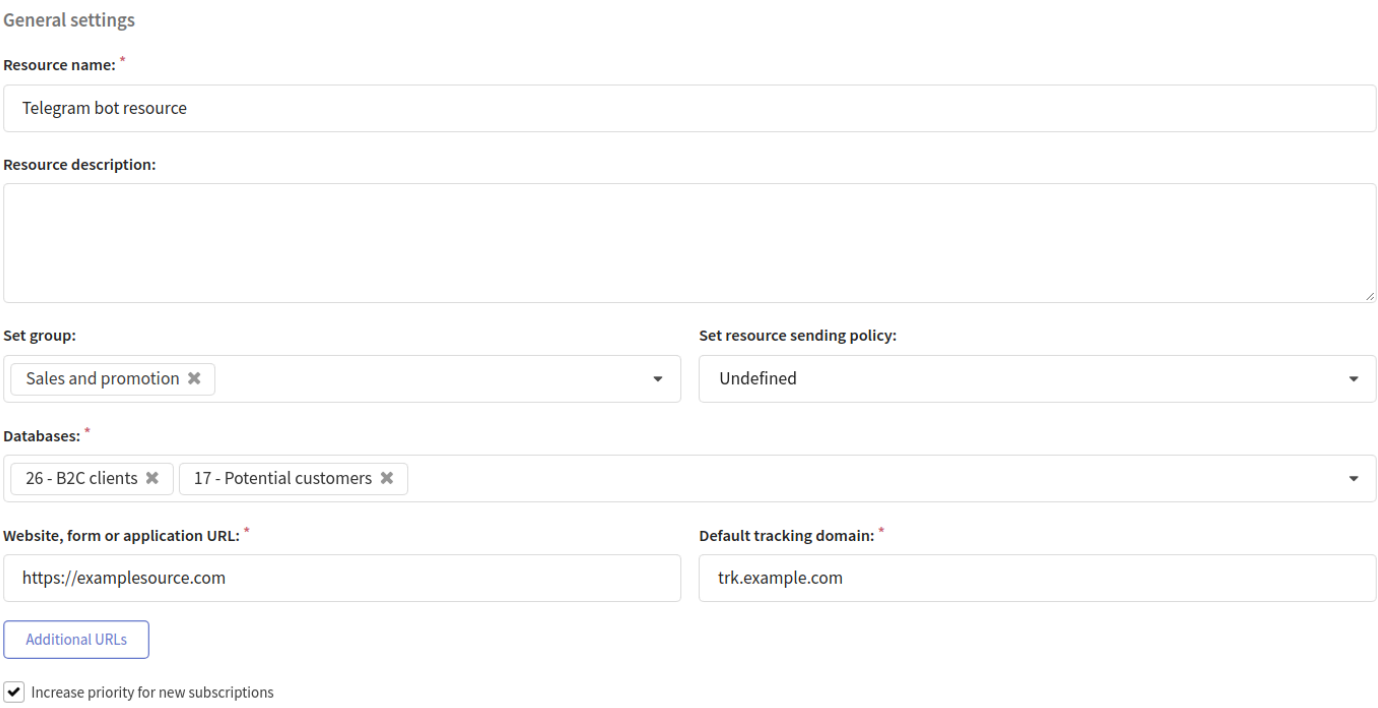
Step 2. Subscribe your customers to this Telegram Bot

Go to the “Data” → “Profile database” and create a new database. The data of users subscribed to the bot will be stored here, as well as their Telegram IDs.

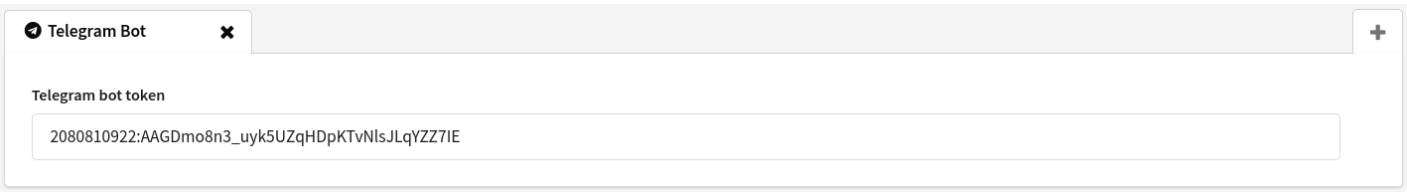
Then in the “Resources” section create a new resource or select a pre-existing one. Use Telegram Bot as a channel:



Add the necessary databases in the basic settings. Specify the default tracking domain that will process subscribers' clicks on the links in your Telegram Bot messages.



Add the bot token copied from Telegram and save the resource:



Now you need to import user data into the database by subscribing them to the resource.

INFO

Please note that the platform only sends Telegram mailings. Subscriber data is collected and transmitted to the platform on the client side. Read more about how to do this in the Telegram documentation.

Example of getting a user's chat_id using the getUpdates method

Method: POST

URL:

`https://api.telegram.org/bot<bot_token>/getUpdates`

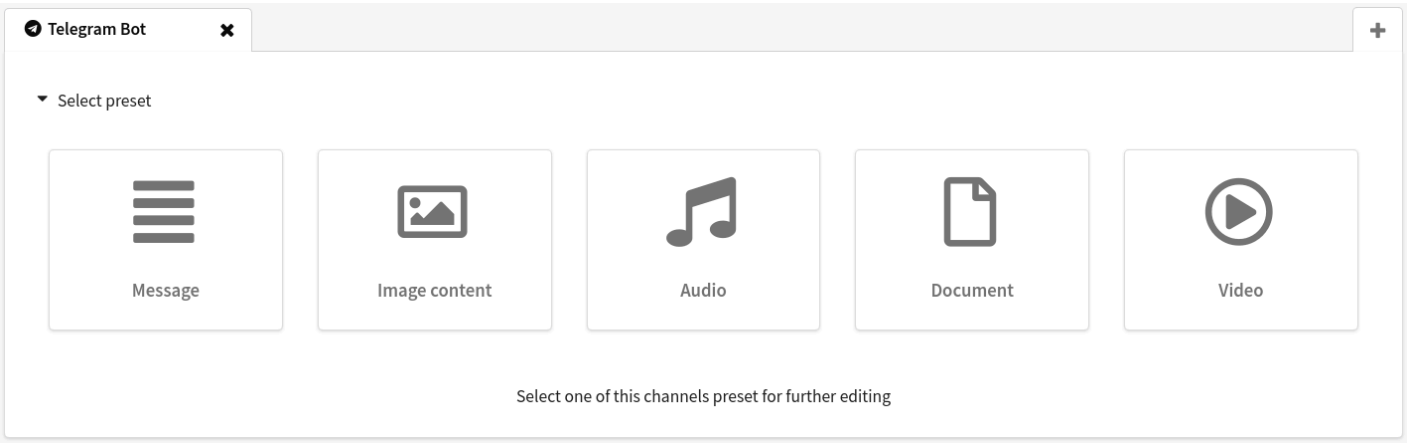
Response:

```
{
    "ok": true,
    "result": [
        {
            "update_id": 262756455,
            "message": {
                "message_id": 47,
                "from": {
                    "id": <chat_id>,
                    "is_bot": false,
                    "first_name": "John",
                    "last_name": "Doe",
                    "username": "john_doe",
                    "language_code": "en"
                },
                "chat": {
                    "id": <chat_id>,
                    "first_name": "John",
                    "last_name": "Doe",
                    "username": "john_doe",
                    "type": "private"
                },
                "date": 1652284613,
                "text": "Hello, world"
            }
        }
    ]
}
```

`<chat_id>` **is a unique identifier for a chat with a specific user. You need to pass it to the platform.**

Step 3. Create a template

Go to the "Templates" section. When creating a new template, select the Telegram Bot channel and add this channel to a pre-existing template. Select the content type in the Telegram version of the template:



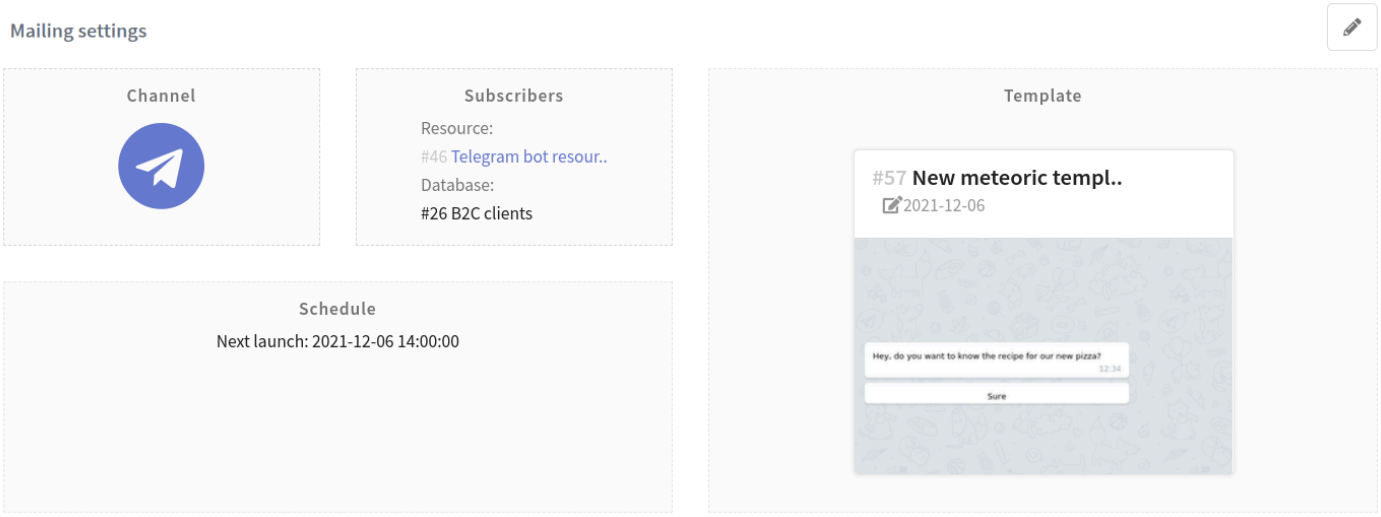
You can add data of a particular customer, as well as dynamic text content from external sources.

Step 4. Configure and launch a mailing

Open the section "Mailings" and select the distribution method: broadcast, regular or trigger mailing.

Create a new mailing. After filling in the basic settings, you need to:

1. **Choose Telegram Bot as a communication channel**
2. **Select a previously configured subscription resource and target audience**
3. **Add a message template to your mailing**
4. **Set a schedule**



When the mailing is fully configured, save it. It will be displayed in the list of mailings of the corresponding type. There is a quick action menu on the right side of each entry. By its means, you can send test message

and, if everything is good, activate your actual mailing

:

Step 5. Evaluate your mailing effectiveness

Mailing Log

The Mailing Log is a tool that provides a summary of the progress of your email mailing. It logs key events, mailing parameters, and statistics, helping you identify potential issues.

To open the mailing journal, click the Journal button in the quick actions menu:

The mailing Log contains the following information:

Type	Description
Mailing started	**Start time of mailing execution**
Limit	**Indicates whether a sending limit is set**

Type	Description
Audience source	**Source of contacts:** **List — Database** **Segment — Segment**
Database ID/Segment ID	**Identifier of the database/segment**
Spent getting segment cursor	**Time spent loading data from source (ms)**
Number of subscribers received from source	**Total number of profiles in the specified source**
Number of subscribers found	**Number of users in source with active subscription to selected resource**
Number of subscribers without subscriptions to channel	**Users without valid channel subscriptions**
Number of subscribers with incorrect subscription fields	**Users with invalid subscription data fields**
Number of duplicate messages for channel	**Count of duplicate contacts detected**
Number of messages sent to channel	**Final count of successfully delivered messages**
Mailing finished	**Mailing completion timestamp**
Total time spent	**Total execution duration from start to finish (sec)**

Reports and Analytics

To evaluate the effectiveness of your push campaign use the Analytics section in the main menu of the platform. A [channel report] provides the analysis of one or more mailings.

Name	Send	Delivered	Delivery rate	Opened	Read	Undelivered	Undelivery rate
Total	3 533	3 257	92,19 %	1 603	1 295	276	7,81 %
2024-05-15	470	428	91,06 %	201	163	42	8,94 %
2024-05-14	483	442	91,51 %	219	176	41	8,49 %
2024-05-13	499	454	90,98 %	243	195	45	9,02 %
2024-05-12	506	469	92,69 %	222	180	37	7,31 %
2024-05-11	513	476	92,79 %	232	189	37	7,21 %
2024-05-10	524	486	92,75 %	249	200	38	7,25 %
2024-05-09	538	502	93,31 %	237	192	36	6,69 %

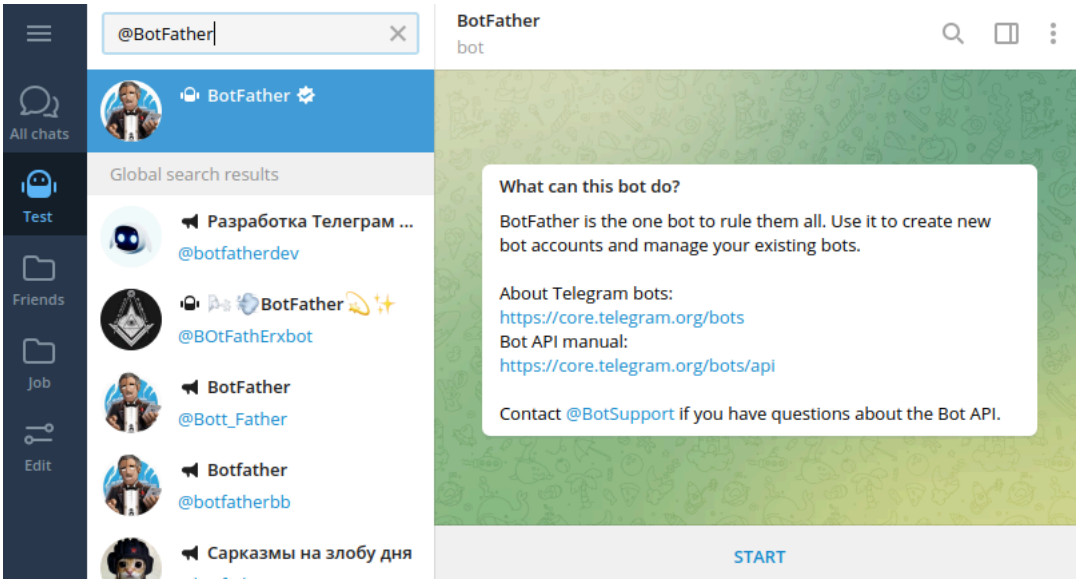
To make grouping easier, you can select a preset of indicators for Telegram Bot. You will see the data on the number of sent, delivered or undelivered messages in the table, as well as information on unique clicks and openings.

Telegram Group: a Guide for the First Mailing

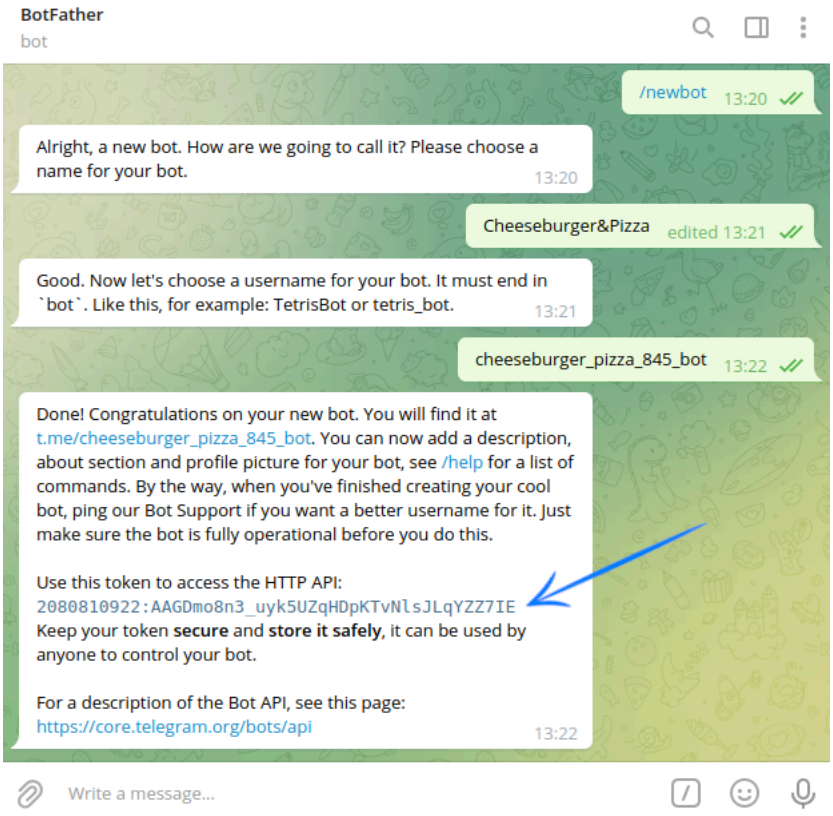
	Telegram channel is a feed of text messages, which are complemented by media files and emoji. Channel members can comment on posts and participate in polls. In groups, unlike channels, two-way communication is available. This means that each group user can send a message to the chat. The Telegram Group channel will allow you to set up automatic publication of posts in channels and groups of the messaging app using placement mailings. </td> </tr>

Step 1. Create a bot and get a token

1. Log in to Telegram, enter `@BotFather` in the search box and select the desired bot.



2. Start a dialogue with the bot by clicking the START button in the chat. Then send the `/newbot` command to it. Enter the name and then the username for your new bot. Please note that the username must end with `"bot"` or `"Bot"`. Copy the received bot token:



Step 2. Add the bot to the group or channel and get a chat_id.

1. 1. Create a group or channel. You will find more detailed information on how to do this in the Telegram documentation.

2. Add the bot to a group or channel. Assign admin rights to the bot.

3. Send a test message to chat using an API request. This can be done through [Postman] or any other API testing tool.

Method: POST

URL: `https://api.telegram.org/bot[BOT_API_KEY]/sendMessage?chat_id=[CHANNEL_NAME]&text=[MESSAGE_TEXT]`

- `[BOT_API_KEY]` — **the bot token received in the conversation with** `@BotFather`.
- `[CHANNEL_NAME]` — **channel or group name using** `@`.
- `[MESSAGE_TEXT]` — **message text.**

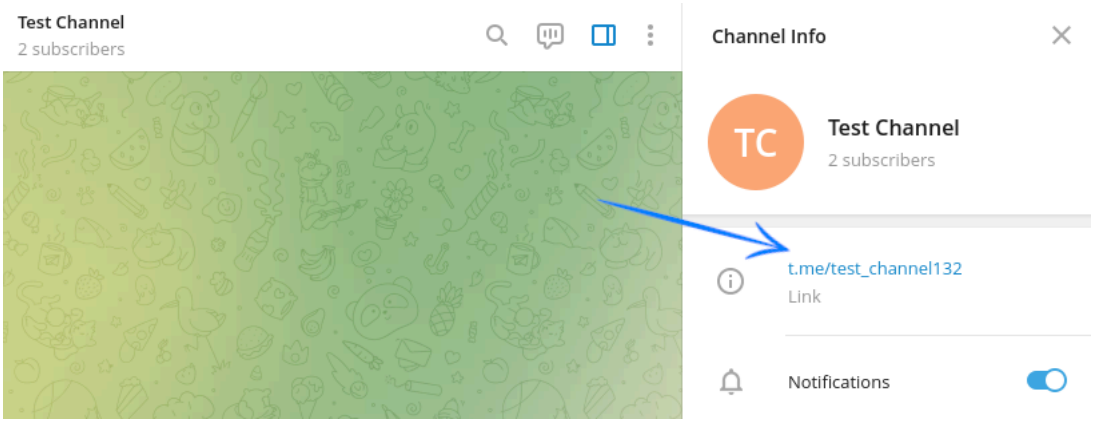
Example:

`https://api.telegram.org/bot2080910120:AAGfho8n3_uyk5trh4DpKTvNlsJLkadsZ7IE/sendMessage?chat_id=@new_test_group132&text=Hey!`

INFO

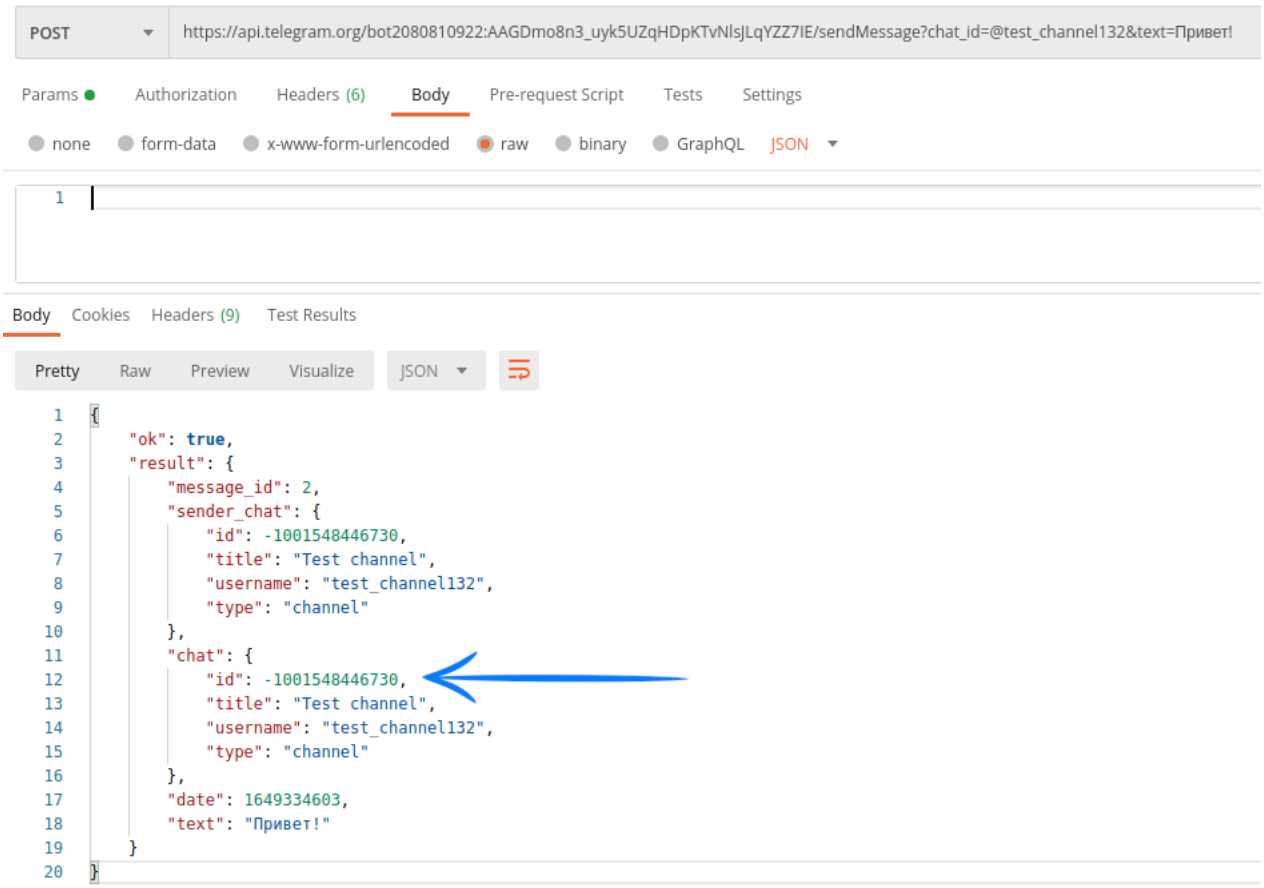
Where can I find the name of the channel or group?

The name is embedded in the link to the channel or group. For example, if the link is `t.me/test_channel132`**, then the part after the slash will become the name of the channel —** `test_channel132`.



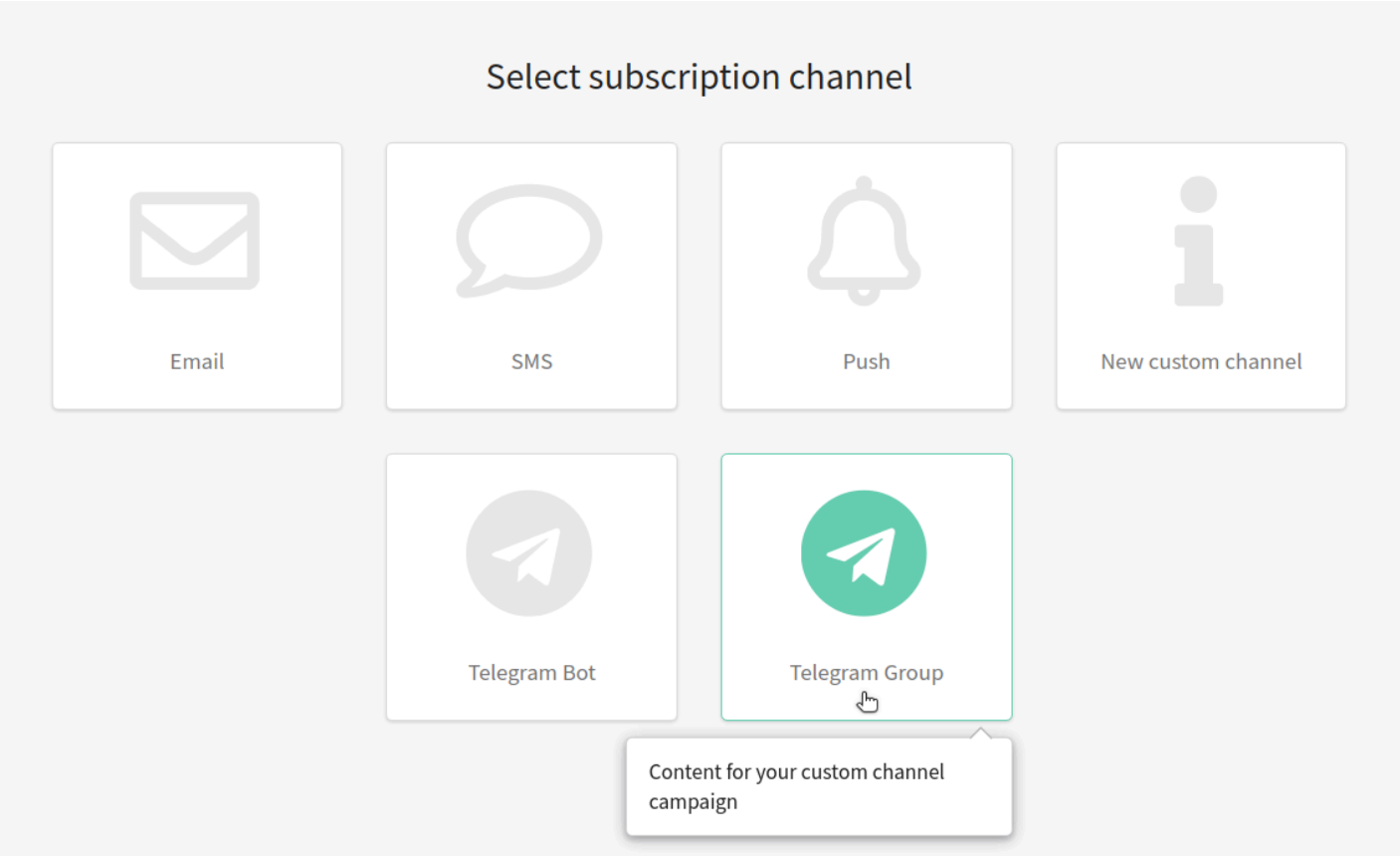
Response:

In the body of the response, you will receive the `chat_id.` **It must start with** `-100`.



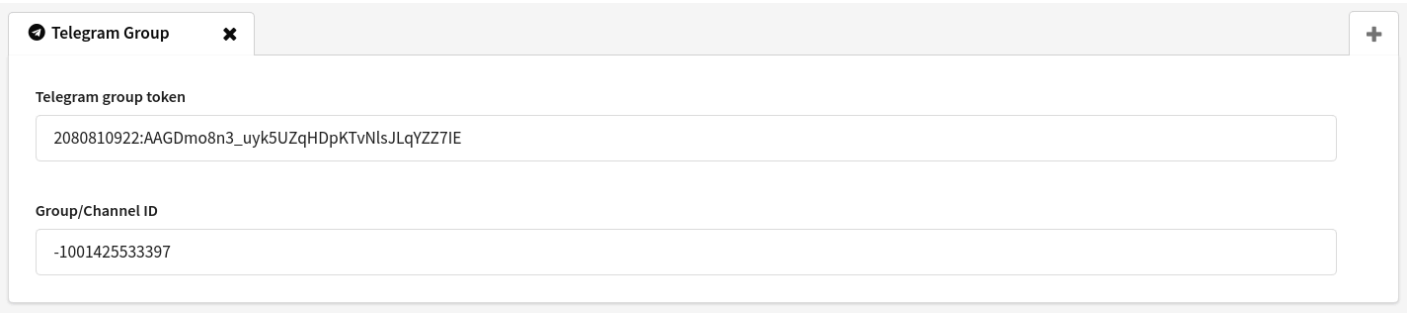
Step 3. Create a resource

Go to the “Resources” section create a new resource or select a pre-existing one. Use Telegram Group as a channel:



Add the necessary databases in the basic settings. Specify the default tracking domain that will process subscribers’ clicks on the links in your Telegram Bot messages.

Add the bot token and group/channel ID copied from Telegram and save the resource:

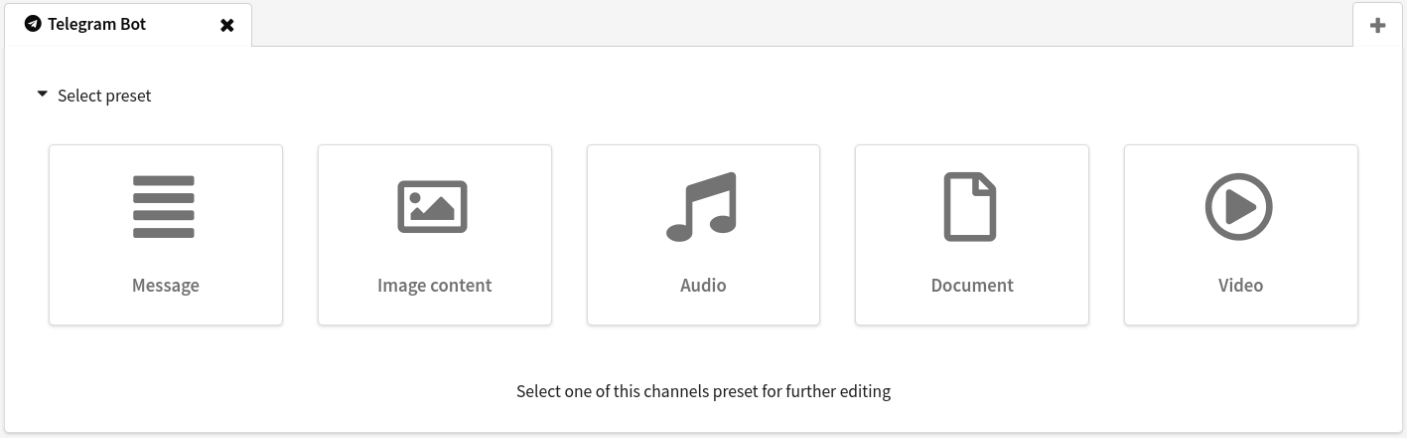


Now you need to import user data into the database by subscribing them to the resource.

Please note that the platform only sends messages. Subscriber data is collected and transmitted to the platform on the client side.

Step 4. Create a template

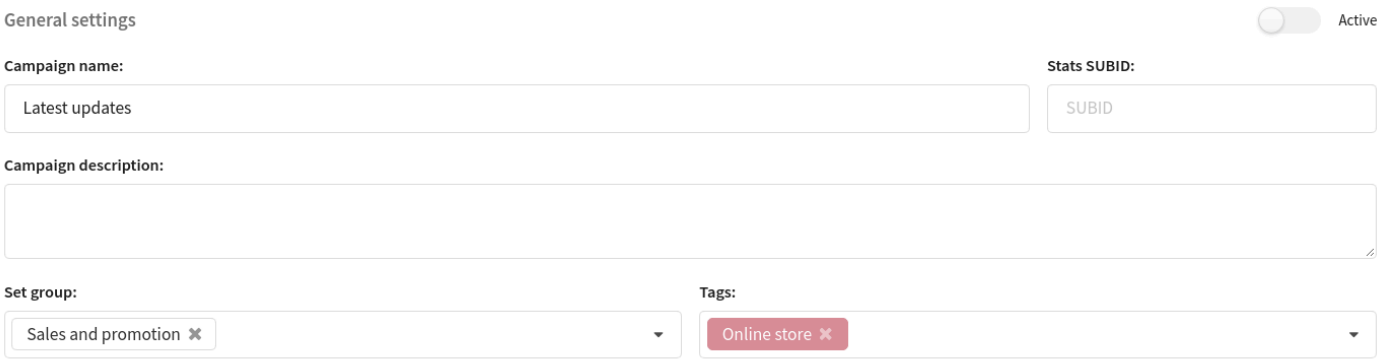
Go to the “Templates” section. When creating a new template, select the Telegram Bot channel and add this channel to a pre-existing template. Select the content type in the Telegram version of the template:



You can add data of a particular customer, as well as dynamic text content from external sources.

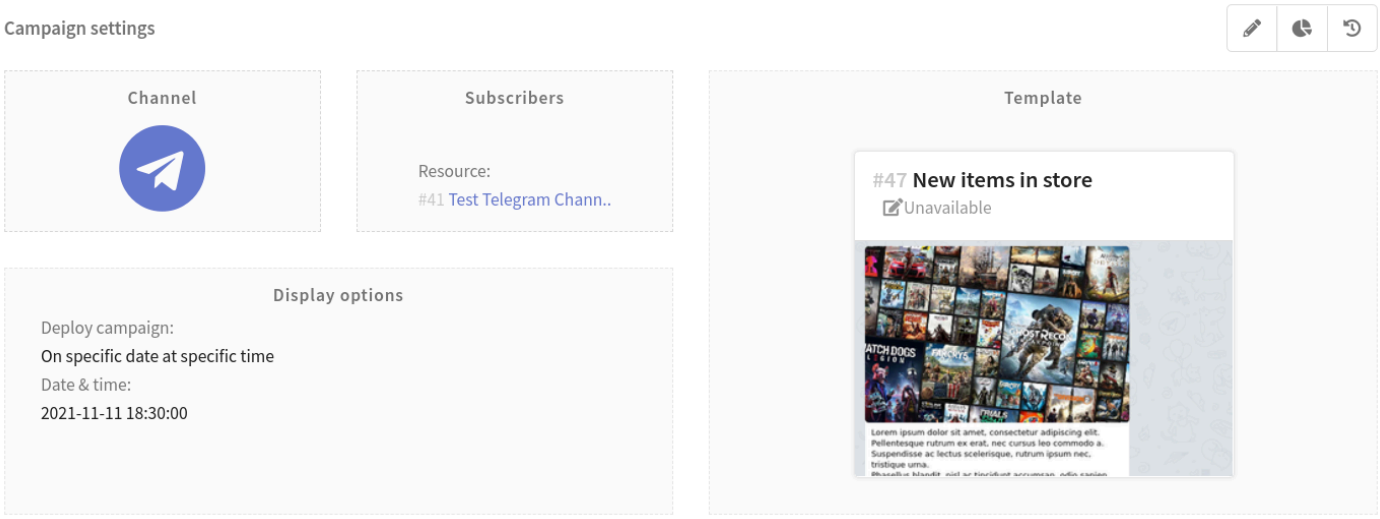
Step 4. Configure and launch a placement mailing

Open the section “Mailings and select Placements”. Create a new mailing.



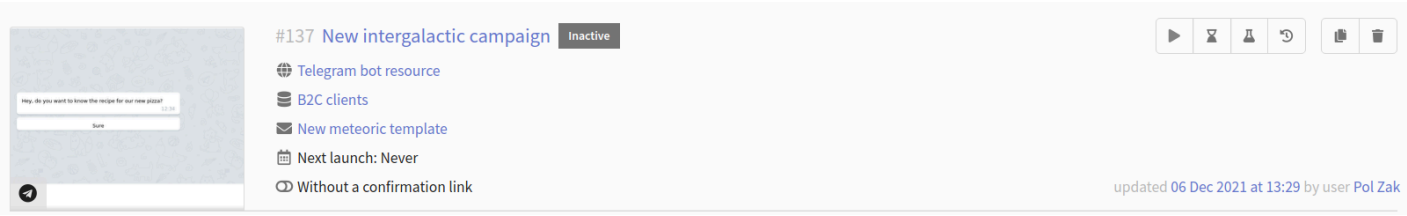
After filling in the basic settings, you need to:

1. **Choose Telegram Group as a communication channel.**
2. **Select a previously configured subscription resource.**
3. **Add a message template.**
4. **Add the date and time the post will be posted to the group.**



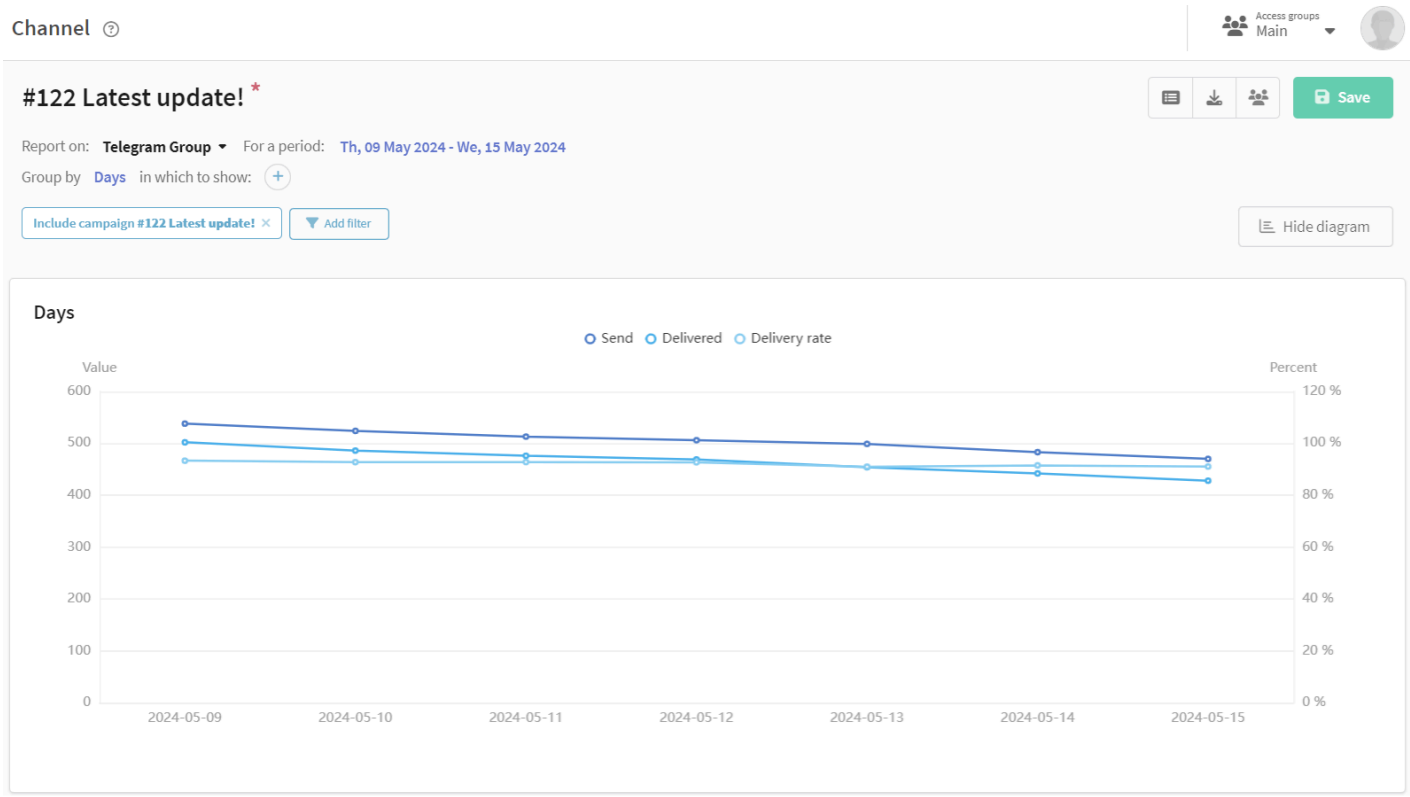
When the mailing is fully configured, save it. It will be displayed in the list of mailings of the corresponding type. There is a quick action menu on the right side of each entry. By its means, you can activate your actual newsletter

:



Step 5. Evaluate the effectiveness of your newsletter

To evaluate the result of your newsletters, use the “Analytics” section in the main menu of the platform. A channel report provides the analysis of one or more mailings.



To make grouping easier, you can select a preset of indicators for Telegram Group. You will see the data on the number of sent, delivered or undelivered messages in the table, as well as information on clicks and openings.

Max

This section contains guides for working with Max in Altcraft Platform.

Max Bot — a channel for personalized messages via bot. Suitable for customer communication, sending notifications, receipts, and personalized mailings. Requires passing the subscriber identifier to the platform.

Max Group — a channel for publishing messages in groups and chats. Suitable for mass content and announcements. Does not require individual profile subscriptions.

For more information about integration with the Max messenger, see this article.

First Campaign via MAX Bot

A MAX bot is an automated account in the MAX messenger that interacts with users via chat according to predefined logic.

To send Mailings via MAX Bot, users must be subscribed to the MAX Bot resource in the Altcraft platform.

The subscription process works as follows:

1. **The user launches the bot.**
2. **The bot receives the user's `user_id`.**
3. **This identifier is sent to the Altcraft platform via API.**
4. **The profile is subscribed to the MAX Bot resource.**

After that, the user can receive Mailings. Below is an example of how to configure such a Mailing via a MAX bot in Altcraft.

Step 1. Create an organization and complete verification

Creating bots and channels in MAX is available only for verified corporate accounts and individual entrepreneurs. To confirm your status, create an organization in MAX and complete verification. More details about creating and verifying an organization are available in the MAX documentation.

Step 2. Create a bot

On the MAX for Partners platform, go to your profile and open the Chatbots section:

Click Create and fill in the required fields:

Click Create. The chatbot will be sent for moderation.

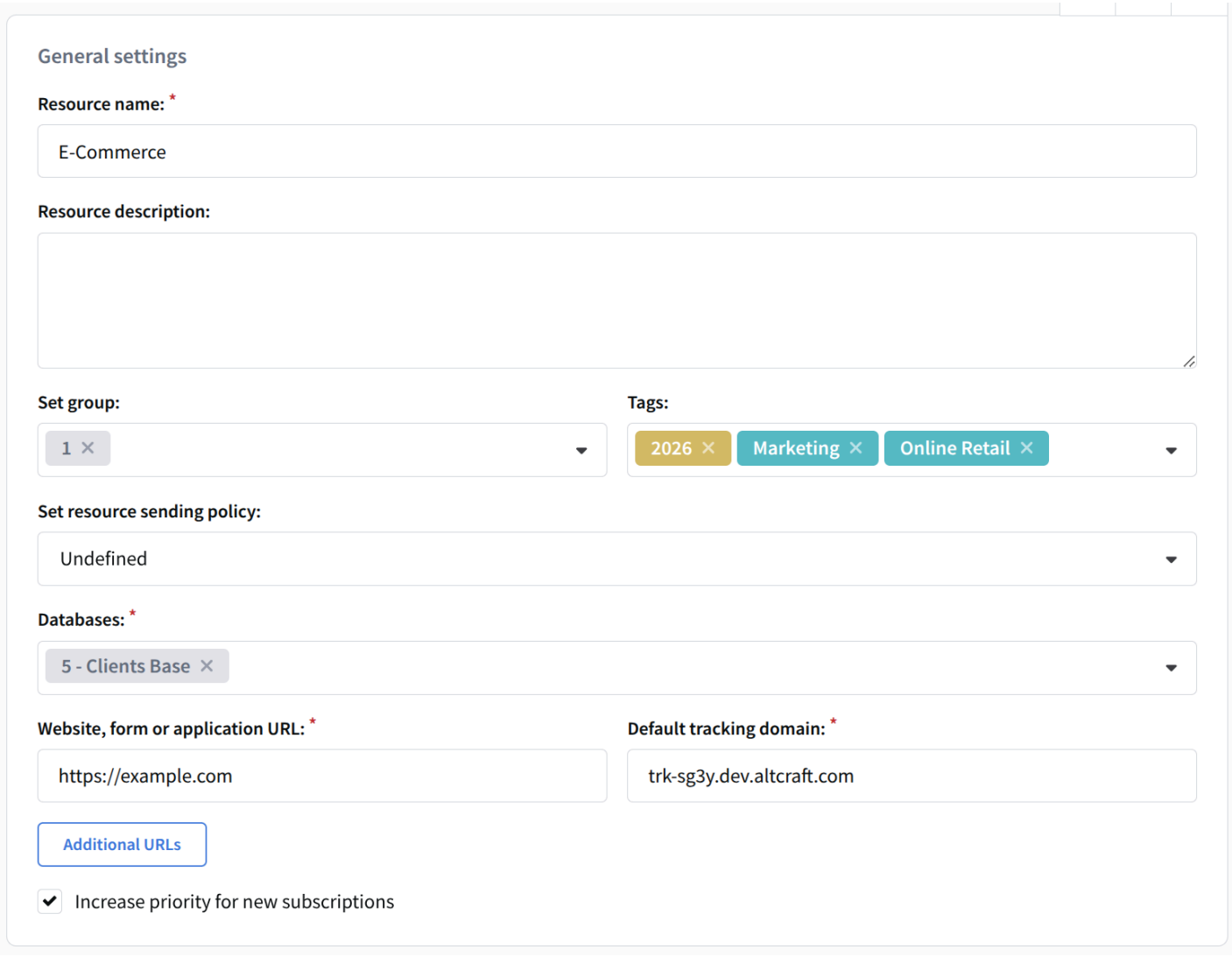
> TIP: MAX chatbots
>
> **More information about creating a bot is available in the MAX documentation.**

Step 3. Subscribe clients to MAX Bot in the platform

Go to Data — Profile database and select a database where the data of users subscribed to the MAX bot will be stored.

Then open Resources and create a new resource or select an existing one. Use MAX Bot as the channel:

In the main settings, add the required databases. Specify the default tracking domain that will process subscriber clicks on links in MAX bot messages:



Save the resource. Next, import user data into the database and subscribe them to the corresponding resource. The subscription must include the user's `user_id`.

NOTE

Note: the platform only performs message sending. Collecting and transferring user data to the platform, including sending the `user_id` of users who subscribed to the MAX bot, must be implemented on the client side. More details are available in the MAX documentation.

Example of retrieving a client `user_id` using the GET method `updates`

Method: GET

URL:

`https://platform-api.max.ru/updates`

Response:

```
{
  "updates": [
    {
      "update_id": "9f3c2a71-8c4b-4c92-9e57-1d7f2c0b8a11",
      "message": {
        "message_id": "msg_847362915",
        "chat": {
          "chat_id": "7349281745621",
          "type": "private"
        },
        "from": {
          "user_id": "48291057",
          "first_name": "Ivan"
        },
        "date": 1719832145,
        "text": "/start"
      }
    },
    {
      "update_id": "b1a0c5e2-3c91-4a6e-8fbb-5e92c0a47d19",
      "callback": {
        "callback_id": "cb_12984756123",
        "data": "order_create"
      }
    }
  ],
  "marker": "1719832145:7349281745621:msg_847362915"
}
```


`<user_id>` **is the identifier of a specific user that must be sent to the platform.**

Step 4. Create a template

Go to the Templates section. When creating a new template, select the MAX Bot channel or add this channel to an existing template.

You can include specific client data as well as dynamic text content from external sources.

More details about creating a template for MAX are available in this article.

Step 5. Configure and launch the Mailing

Open the Mailings section and choose the appropriate type of Mailing: broadcast, regular, or triggered.

Create a new Mailing. After filling in the main settings, you need to:

1. **Select MAX Bot as the communication channel**
2. **Select the previously configured subscription resource and the recipient audience**
3. **Add a message template to the Mailing**
4. **Set the schedule**

After completing the Mailing configuration, save it. It will appear in the list of Mailings of the corresponding type. On the right side of each record there is a quick actions menu. From there you can perform a **test send**

and, if everything is successful, **activate the Mailing**

:

Step 6. Evaluate mailing performance

Mailing journal

The Mailing journal is a tool that provides a brief overview of Mailing execution. It records key events, Mailing parameters, and statistics, which helps identify potential issues.

To open the Mailing journal, click the Journal button in the quick actions menu:

The Mailing journal contains the following information:

Type	Description
Mailing start	**Time when the Mailing execution started**
Limit	**Indicates whether a message sending limit is set**

Type	Description
Subscriber source	**list — Database** **segment — Segment (if applicable)**
Database ID / Segment ID	**Identifier of the database or segment used**
Data retrieval time	**Time spent loading contacts from the source (ms)**
Number of profiles in source	**Total number of contacts in the specified source**
Number of valid subscribers	**Number of users with an active subscription**
Subscribers without subscriptions	**Users without a subscription to the target channel (0 in the example)**
Records with incorrect fields	**Errors in subscription data (0 in the example)**
Number of duplicates	**Duplicate contacts (40 in the example)**
Number of messages sent	**Actual number of successful sends**
Campaign completed	**Time when the Campaign finished**
Total time spent	**Total Campaign duration**

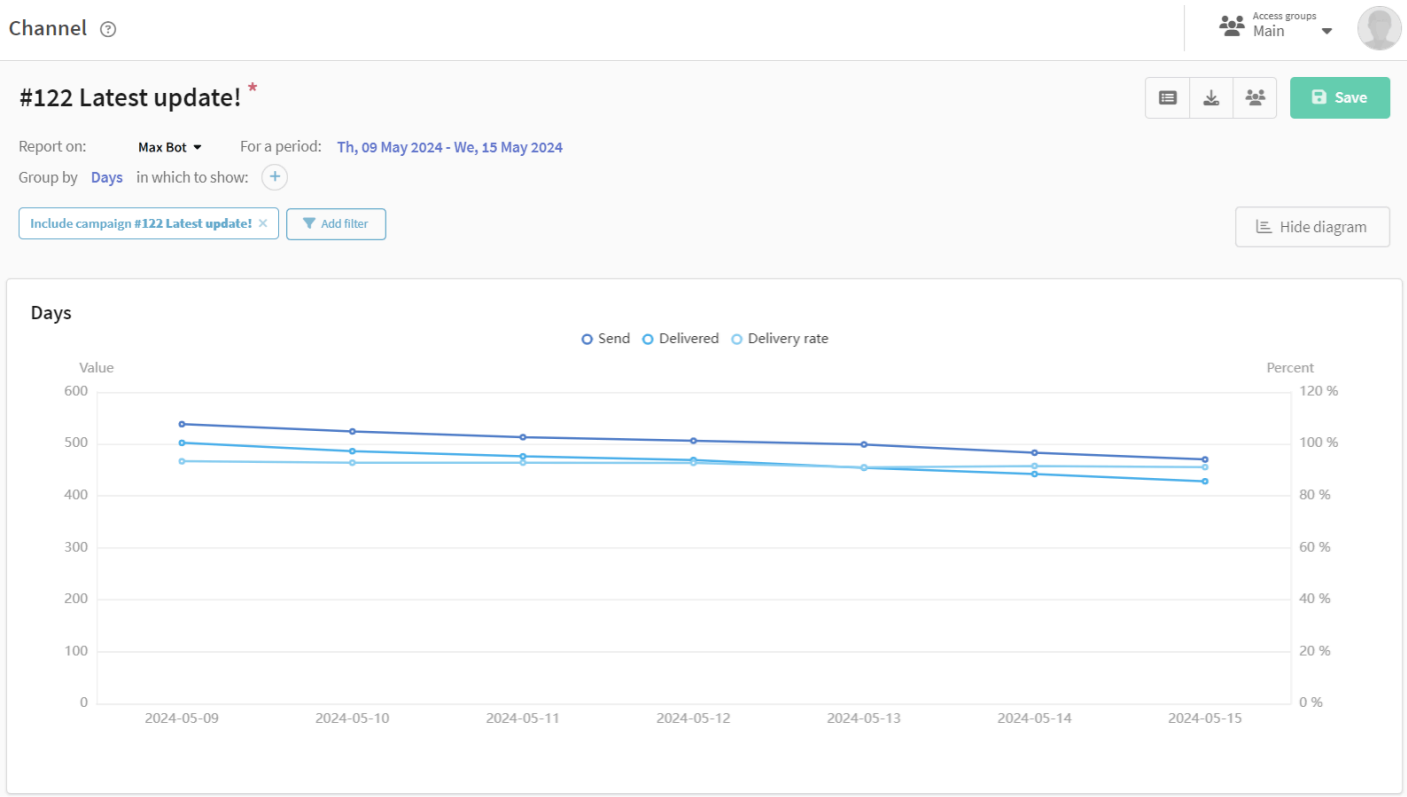
Reports and analytics

To evaluate the results of the current Mailing, you can view the channel report. The Channel report provides analytics for one or several Mailings.

To access statistics, open the Analytics section in the main platform menu. By default, the report for All channels for the last 7 days is displayed.

To group data by channel, select Report by: MAX Bot

Next, add a filter for specific Mailings. Select the required Mailing from the list. As a result, you will receive statistics for the selected Mailing.



First Mailing in MAX Group

A channel in MAX is a feed of text messages that can include media files and emojis. Channels also support user comments and polls.

The MAX Group channel enables automatic publishing of posts in messenger channels and groups using placements.

> TIP: Indirect channel
>
> **MAX Group is an indirect communication channel, which means Mailings in this channel cannot be personalized using profile data variables.**

Step 1. Create an organization and complete verification

Creating bots and channels in MAX is available only for verified corporate accounts and individual entrepreneurs. To confirm your status, create an organization in MAX and complete verification. More details about creating and verifying an organization are available in the MAX documentation.

Step 2. Create a bot

On the MAX for Partners platform, go to your profile and open the Chatbots section:

Click Create and fill in the required fields:

Click Create. The chatbot will be sent for moderation.

> TIP: MAX chatbots
>
> **More details about creating a bot are available in the MAX documentation.**

Step 3. Create a channel or group in MAX

On the MAX for Partners platform, open Channels — Create — Create a new business channel. Scan the QR code or open the bot using the button:

In the channel creation bot, confirm your phone number and wait for the request to be processed. After that, you will be able to manage the created channel.

Creating a group does not require special permission.

> TIP: Private channels
>
> **MAX supports private channels with access available only by invitation. More information about creating and managing channels is available in the MAX documentation.**

Step 4. Add the bot to the channel or group

Add the created bot to the group or channel. Then grant the bot administrator rights. Messages will be published on behalf of this bot.

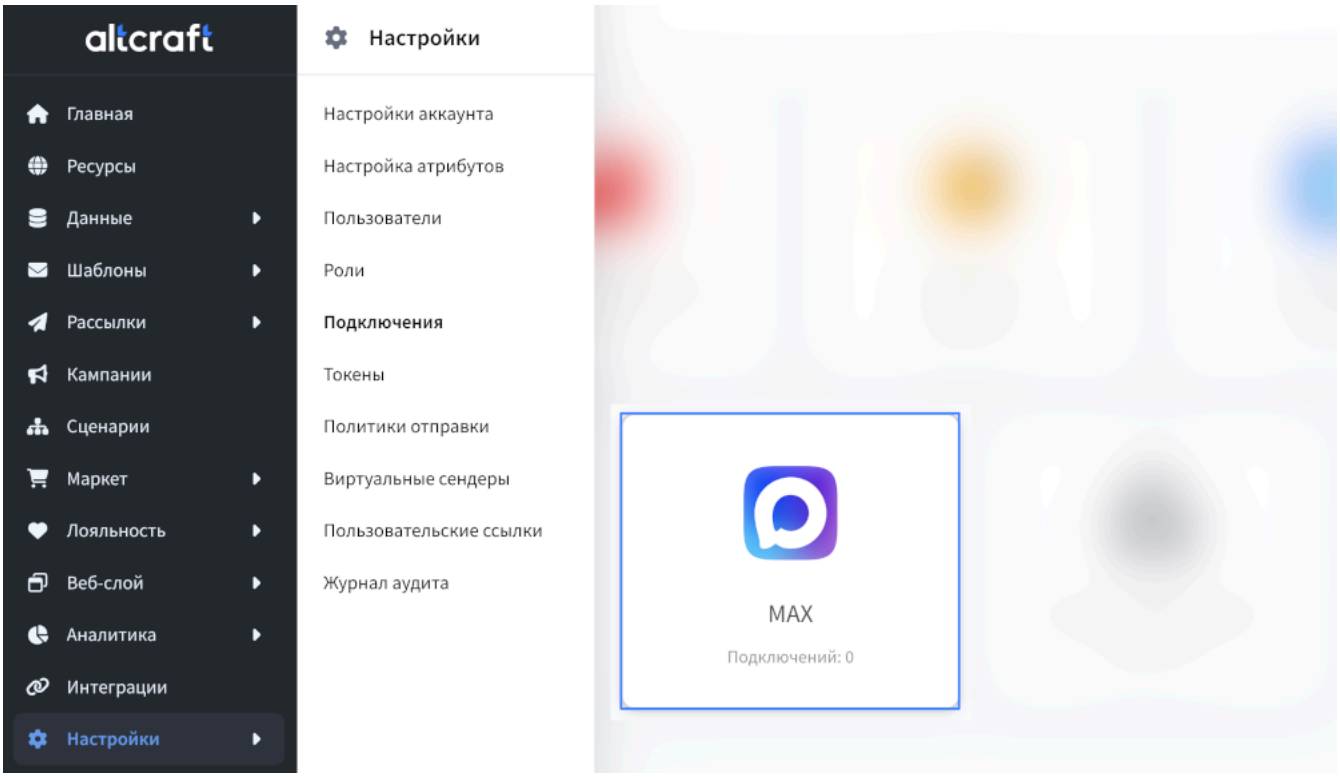
Step 5. Create a connection and integration

To enable interaction between the platform and MAX, you first need to create a bot connection and then configure the integration used by communication channels.

To create the connection, you need the MAX bot token. It can be obtained on the MAX for Partners platform.

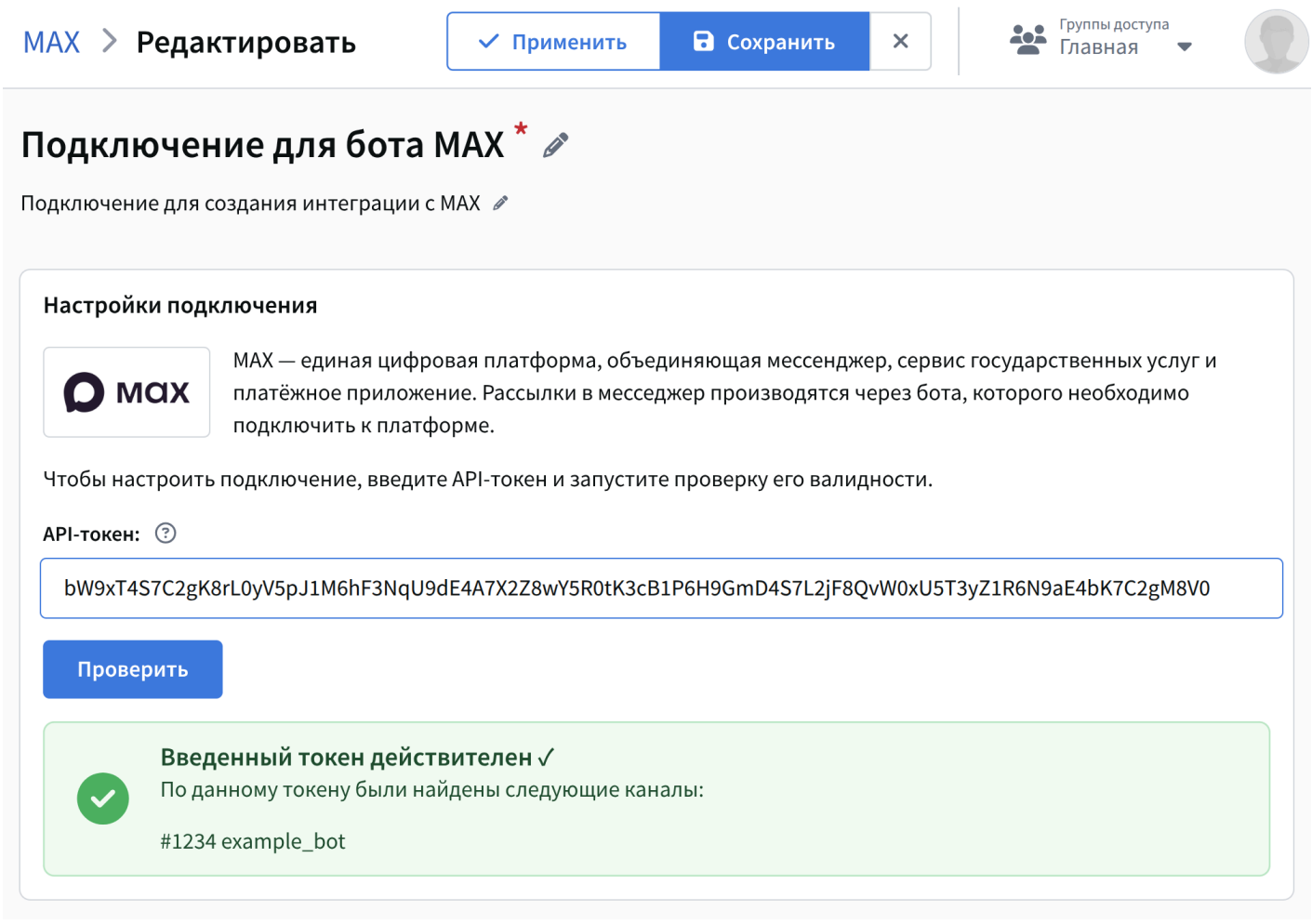
Open the Chatbots section on the MAX for Partners platform, then go to Integration — Get token. The API token will appear in the field:

Then go to the platform and open Settings — Connections. In the list of connection types, select MAX:



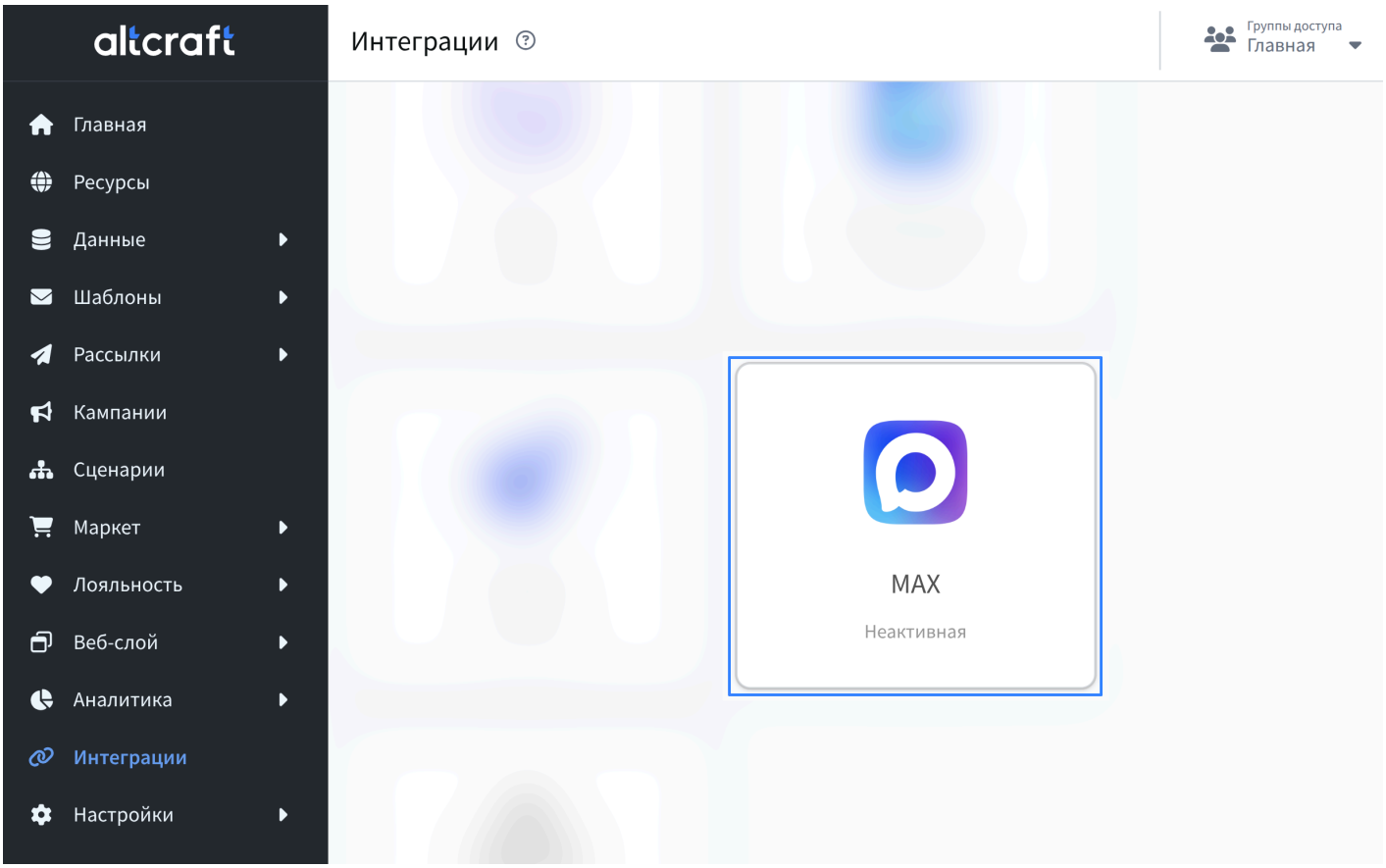
Click Create on the top panel and enter the connection name and description.

In the API token field, paste the token obtained from the MAX platform and click Check. If the data is correct, the platform will display a successful verification message.

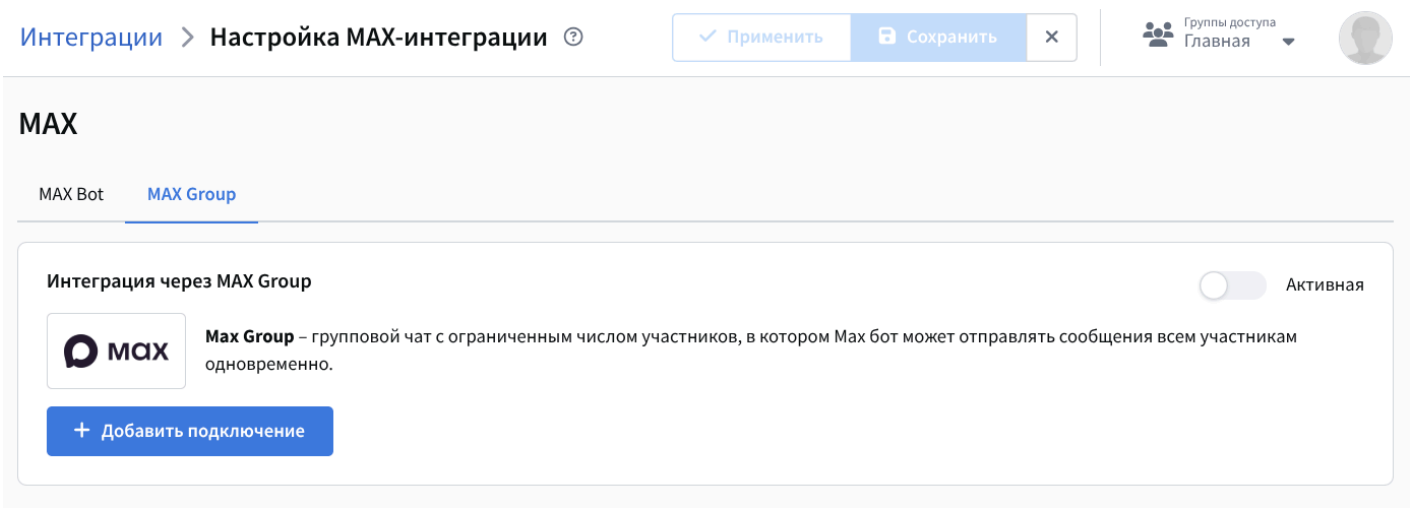


Click Save to finish creating the connection.

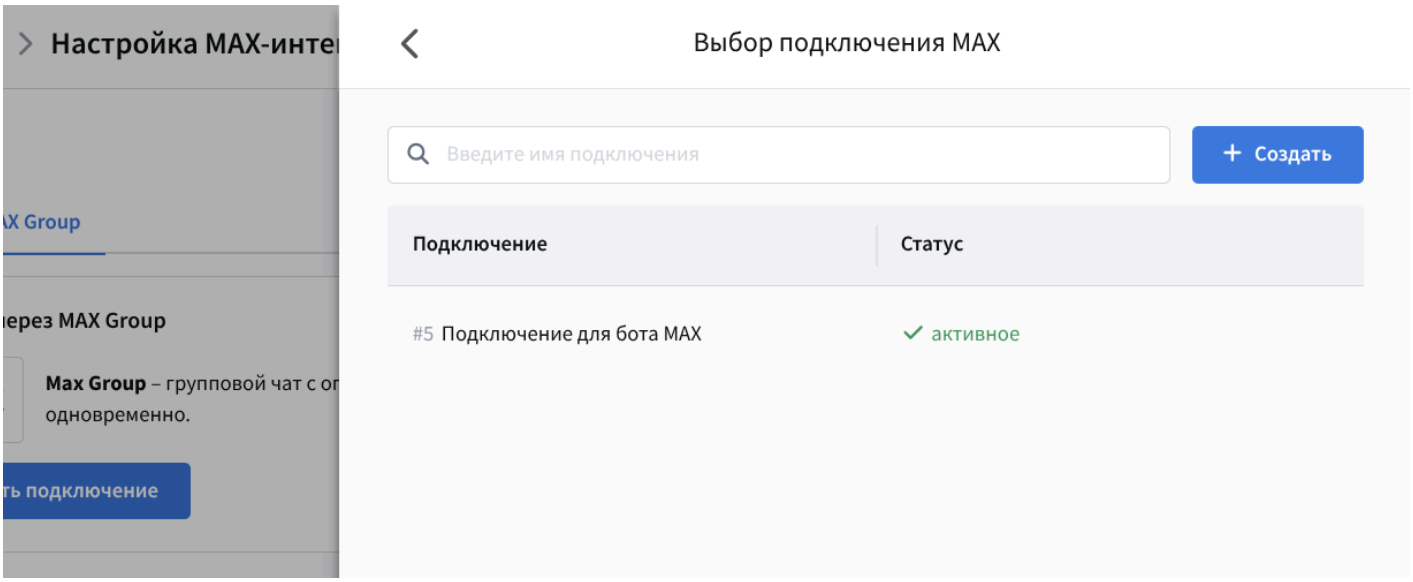
After creating the connection, configure the integration. Open the Integrations section and select MAX:



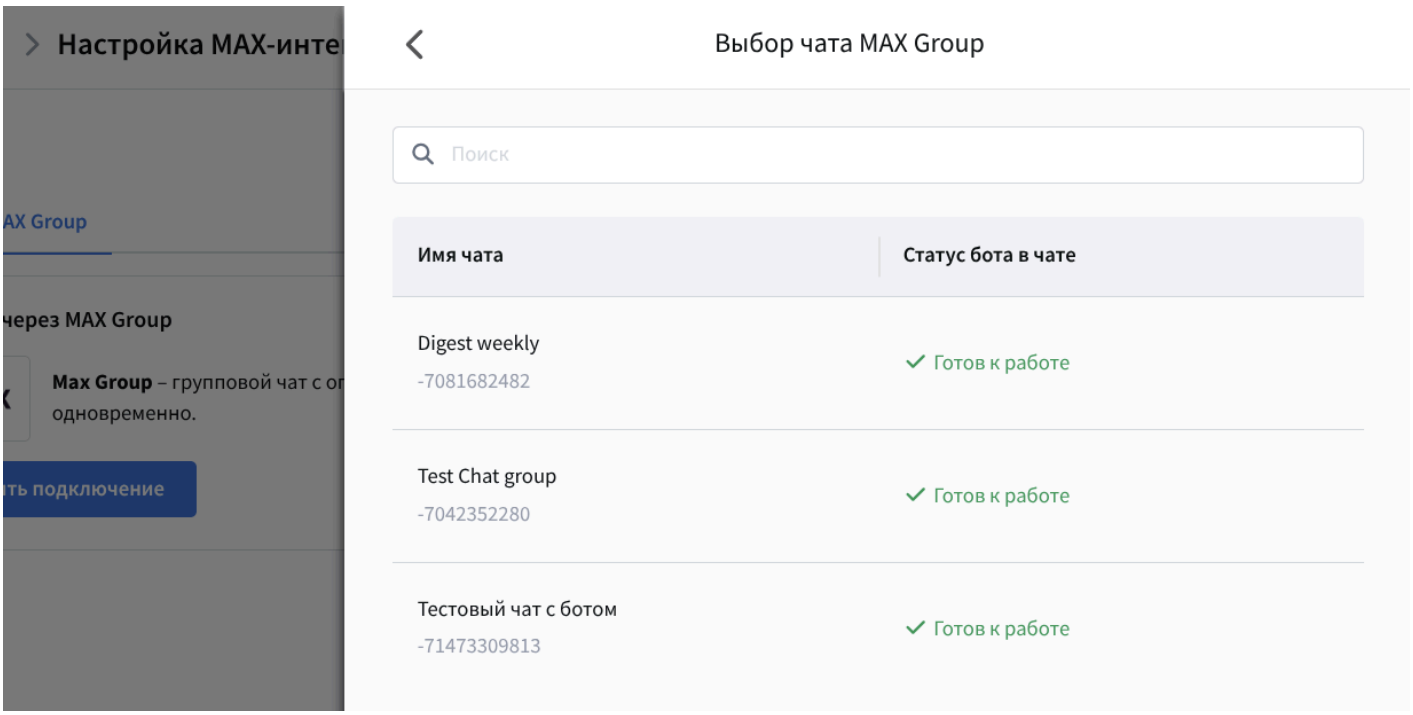
Open the MAX Group tab and click Add connection:



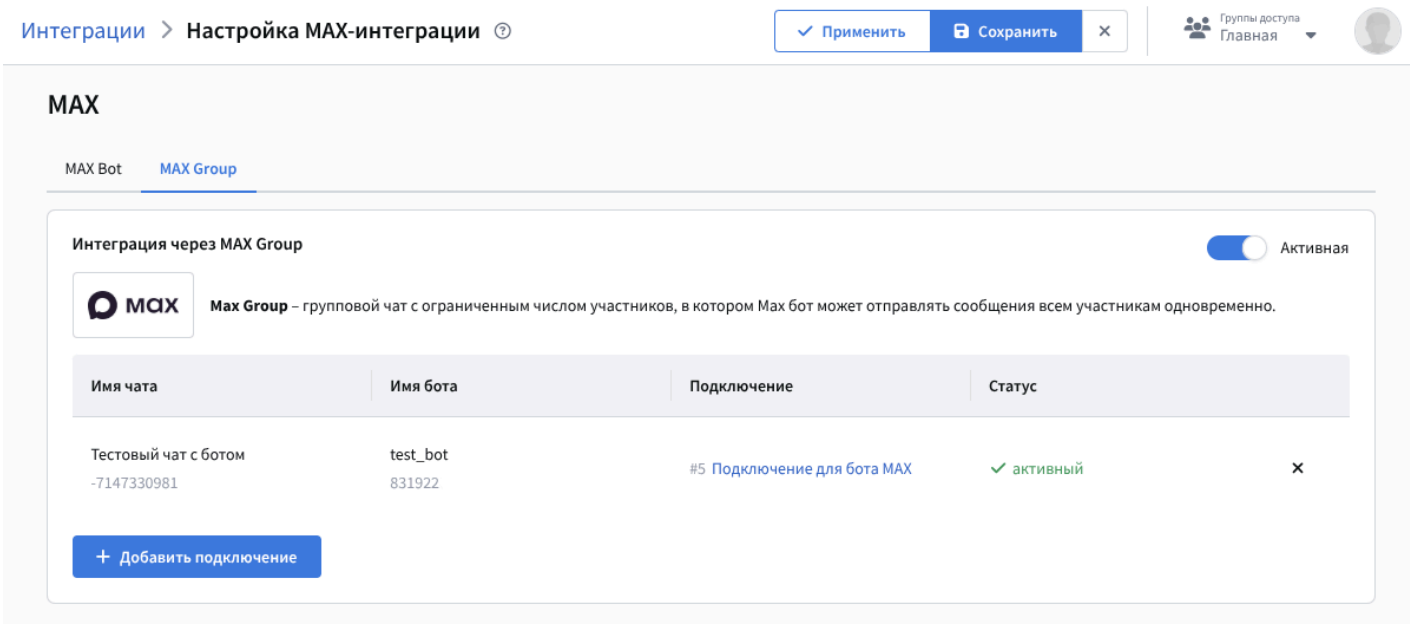
Select the created bot connection:



Then select the chats where the bot will send messages:



Enable the integration using the switch at the top of the page and click Save:



The integration is ready to use. Next, configure a resource that will be used to send messages to MAX messenger chats.

Step 6. Create a resource

Go to the Resources section and create a new resource or select an existing one. Use MAX Group as the channel:

In the main settings, add the required databases. Specify the default tracking domain that will process subscriber clicks on links in MAX group posts.

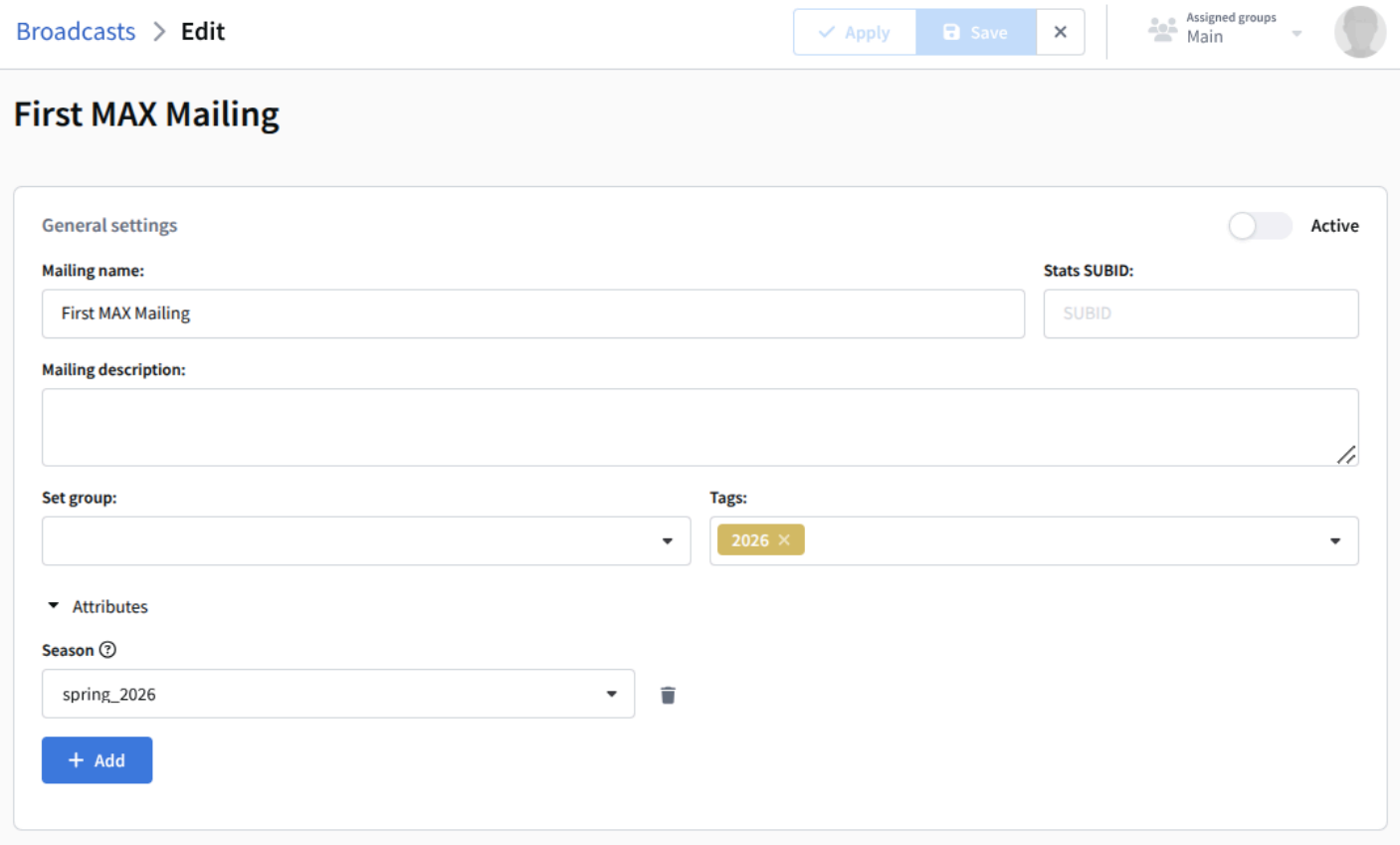
Step 7. Create a template

Go to the Templates section. When creating a new template, select the MAX Group channel or add this channel to an existing template.

You can add dynamic text content from external sources.

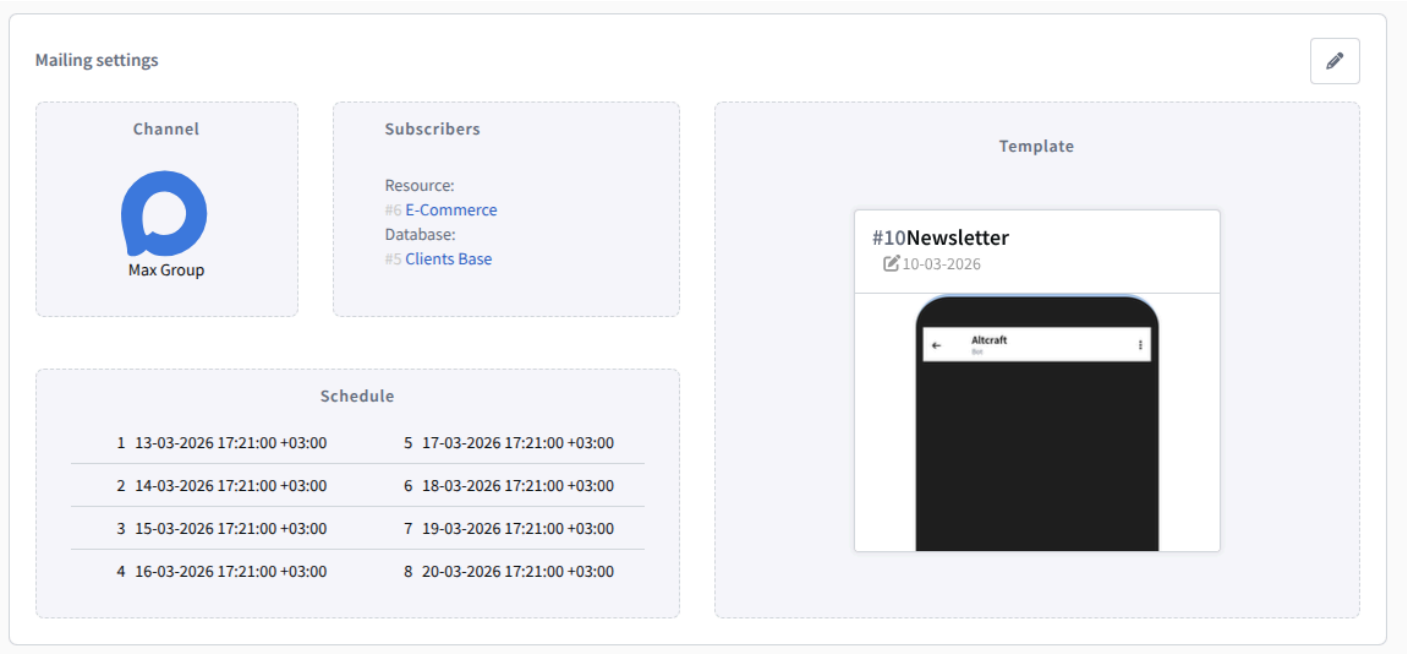
Step 8. Configure and launch a placement

Open the Mailings section and select Placements. Create a new Mailing.



After filling in the main settings, select:

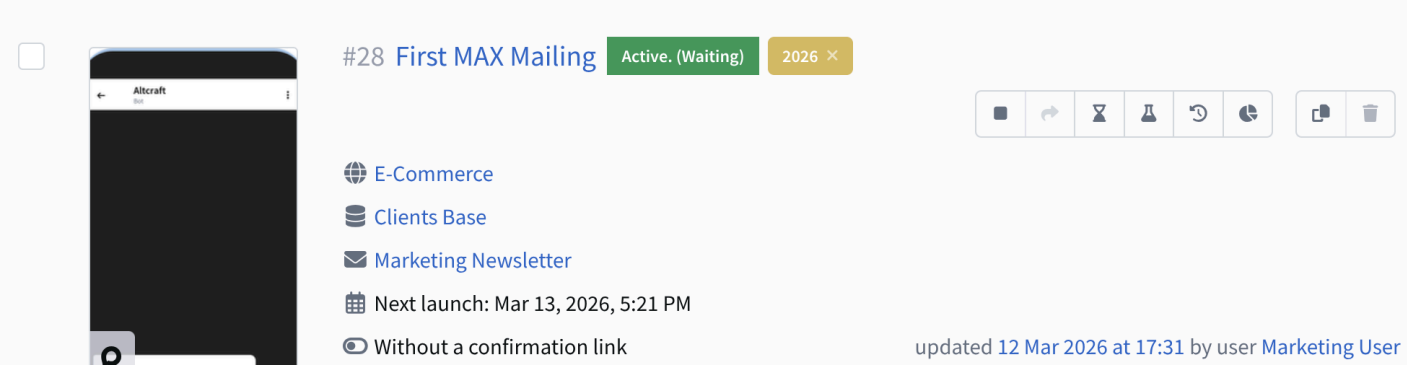
1. **MAX Group as the communication channel;**
2. **The previously configured subscription resource;**
3. **The message template;**
4. **The date and time when the post will be published in the group.**



After completing the configuration, save the Mailing. It will appear in the list of Mailings. On the right side of each entry there is a quick actions menu. From there you can activate the placement

▶

:

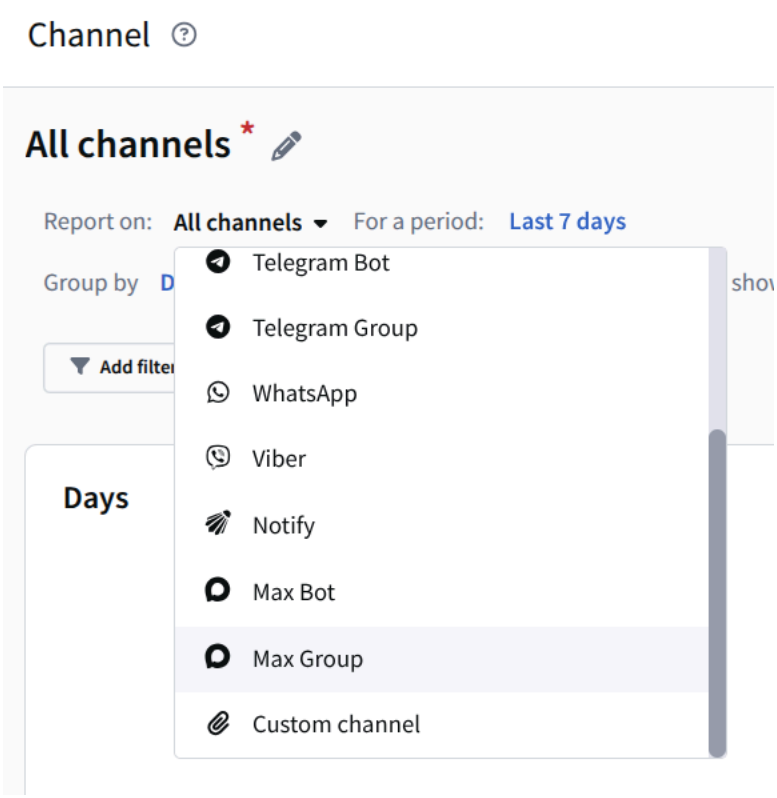


Step 9. Evaluate Mailing performance

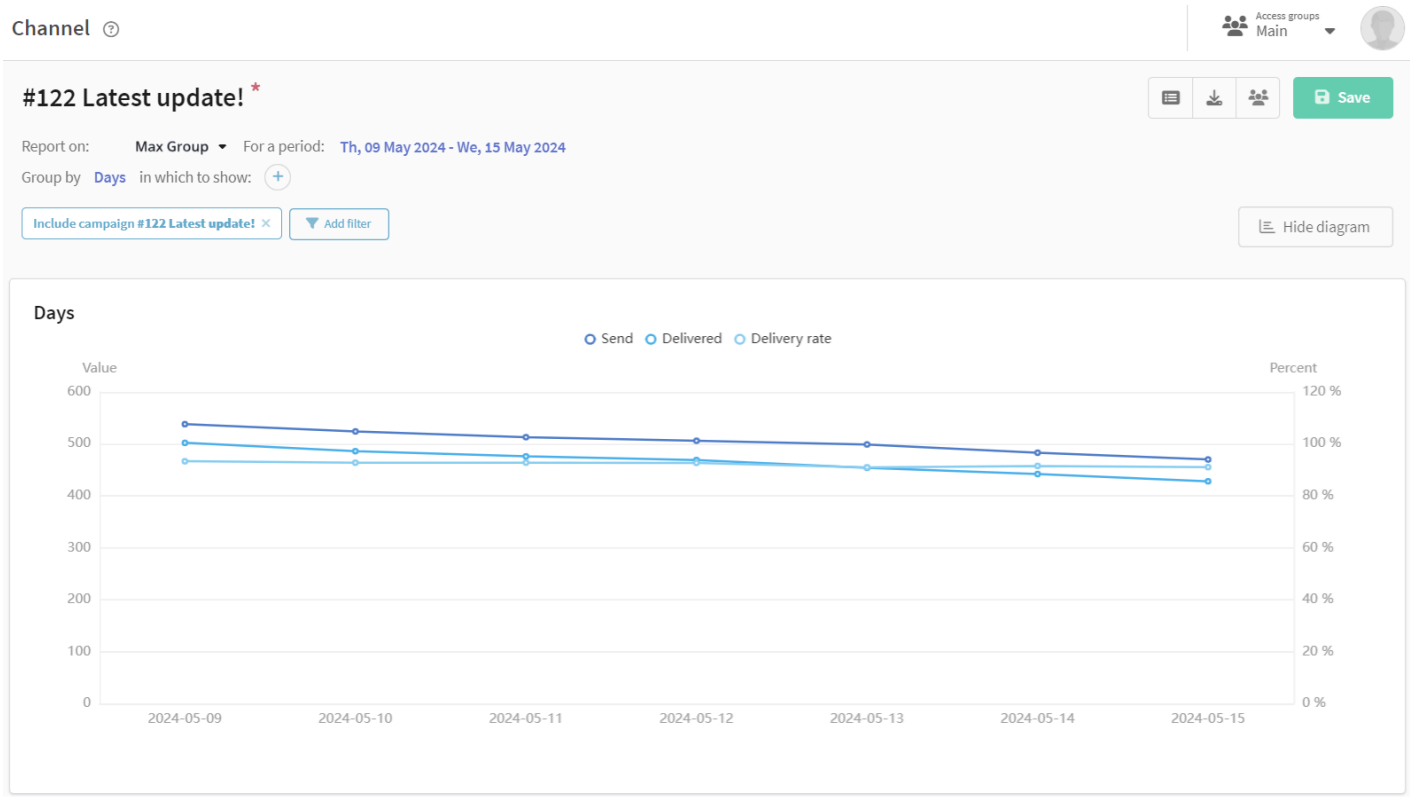
To evaluate the results of the current Mailing, you can view the channel report. The Channel report provides analytics for one or several Mailings.

To access statistics, open the Analytics section in the main platform menu. By default, the report for All channels for the last 7 days is displayed.

To group data by channel, select Report by: MAX Group



Next, add a filter for specific Mailings. Select the required Mailing from the list. As a result, you will receive statistics for the selected Mailing.

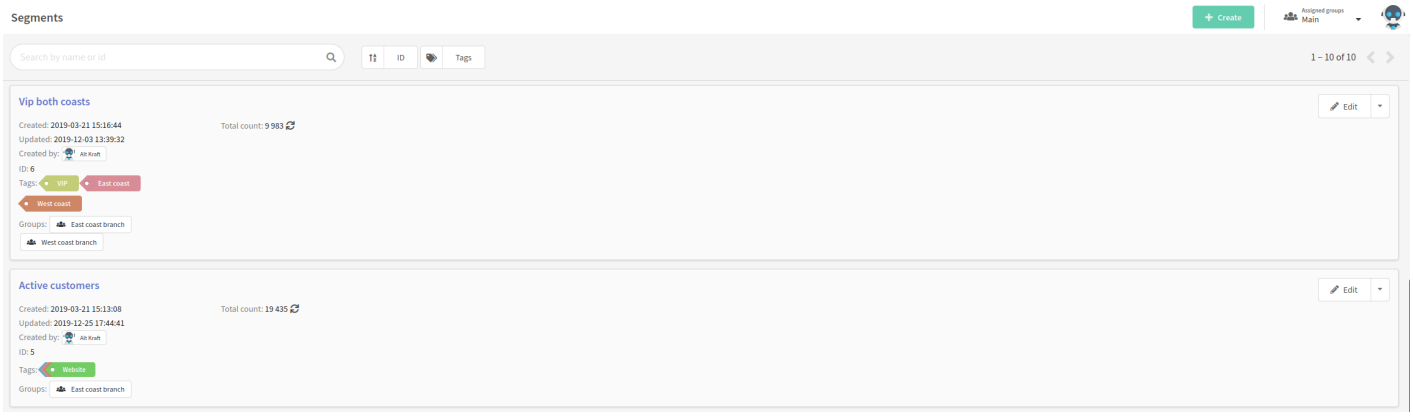


Segmentation

	Segment is a part of the database that includes profiles that meet certain conditions defined in the segment. After the segment is created, it can be included in a campaign or mailing as a source of subscribers.

Profiles segmentation allows you to target your marketing mailings and campaigns. Selection is based on customer profile data, action history in messages, on the website or in the application, as well as based on queries to external sources such as SQL databases or web service APIs.

To segment the audience of subscribers, open Data → Segments in the main menu. In the list, for each segment, the type and brief information will be displayed. Using the context menu, you can perform additional actions, for example, mass update profiles of the segment.



Open an existing segment or add a new query by clicking + Create at the top of the page.

Segment types

Static segments

Static segment can be counted by selection from the database, but it will not be recounted automatically after saving. You can add profiles into the static segment manually, and remove profiles in the same way. You can use these segments when you deal with fixed customers groups.

Dynamic segments

Dynamic segments are mainly used for targeting your marketing campaigns. Such segment does not save customers list, which fit segmentation conditions. Instead, the main thing which is saved is selection query structure. Counting is performed exactly in the moment of campaign launch - and actual information is used.

Updatable segments

The updatable segment is used for dynamic scheduled updates. It includes profiles that meet the specified conditions. The audience is updated until this scheduled check. If necessary, the calculation can be performed manually.

Static Segments

A static segment is a fixed list of profiles from a selected database. After filling, its composition does not change automatically when the data in the database is updated.

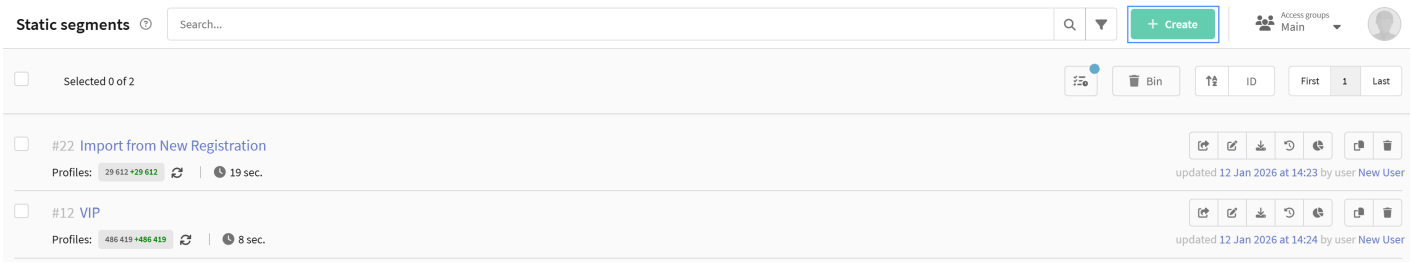
This type of segment is suitable for tasks where a fixed audience is needed, for example: mailings to a pre-approved list of clients or for one-time selections for reports or integrations. Also, the audience of a mailing is formed based on a static segment.

	To get started, go to **Data → Segments → Static Segments.**

Creating and General Segment Management

Creating a Segment

In the segments list, click + Create:



In the form that opens, specify the Name and Description of the segment, and select the Database from which profiles will be selected. Then click Create:

> INFO: Important
>
> **The database is set only during creation and cannot be changed later.**

Segment Configuration and Management

After creating or selecting a segment, its editor will open. In the upper right corner of the segment editor, an action panel is available for working with the entire current list of profiles:

- **Set Tags: Assign or change tags for categorizing the segment.**
- **Specify Access Groups: Configure which user groups can see and edit this segment.**
- **Copy Profiles to Profile Database: Copy all profiles from this segment to another selected profile database.**
- **Copy Profiles to Static Segment: Copy all profiles from this segment to another selected static segment.**
- **Edit Profiles: Open the interface to modify data (for example, fields or subscriptions) for all profiles in the segment simultaneously.**
- **Add to Scenario: Send segment participants to a selected scenario. By default, only profiles that are not already participating in this scenario are sent. To send all segment participants, including those already participating, check the corresponding checkbox.**

> INFO: Note
>
> **You can set tags and access groups right after creating the segment. Other actions will become available after the first data calculation.**

Audience Management

The segment composition can be changed in different ways: manually, by configuring conditions, using the API, or via import. Please note that the conditions work only once when they are applied and are not saved for constant segment updates. To add or remove profiles, click:

- **Add: Clicking opens a side panel for configuring conditions by which profiles will be added to the segment.**
- **Remove (becomes available after adding profiles): Clicking opens a similar side panel for configuring conditions by which profiles will be removed from the segment.**

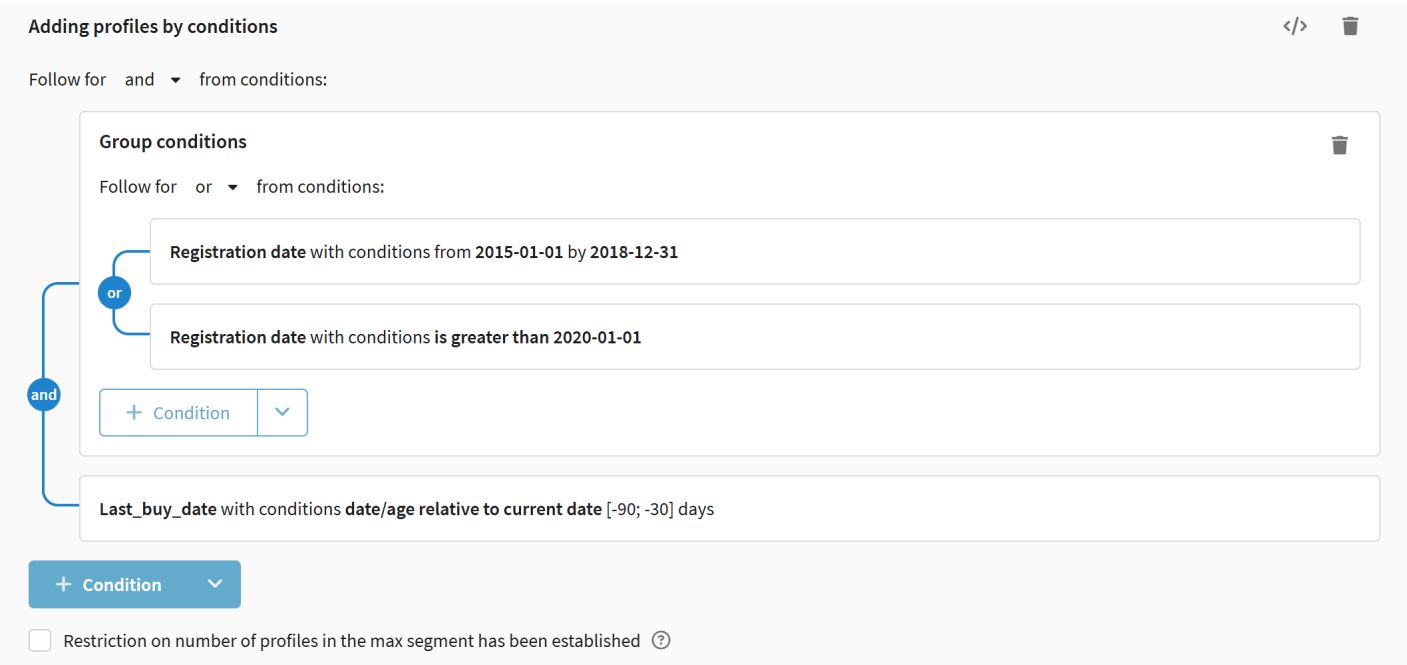
Adding Profiles by Conditions

When you click the Add button, the conditions editor opens in the side panel. Here, categories are available for configuring selection rules from the selected profile database:

Rules are built based on filters, which can be combined into groups. The following data categories are available for filtering:

- **Profile data, including raw queries, profile structure, browser and operating system data, statuses, system fields (UTM, subscriptions, identifier), main fields (email, phones, first name, last name, date of birth, gender, registration and location data), as well as additional custom fields.**
- **Interactions with entities: participation in scenarios and segments (dynamic or static), behavior in popups (opening, filling, clicks), working with promo codes, connections, forms, orders, goal achievements, participation in mailings (including belonging to a control or test group).**
- **Based on Behavior in channels: interactions with Push notifications, SMS, Email, Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, Viber, and VK Notify.**

Within a group, conditions can work using "AND" logic (must comply with each condition) or "OR" logic (must comply with one of the conditions):



You can quickly load pre-configured conditions by clicking Import Conditions and selecting an existing segment — only the internal filtering rules will be copied.

> INFO: Note
>
> **A static segment does not save segmentation conditions, so importing conditions is only available from dynamic or updatable segments.**

After configuring the rules, click Add. The system will start calculating the number of matching profiles. Once the calculation is complete, the profile list in the segment will be updated.

Script

The segmentation query can also be created manually using the built-in JSON editor. To do this, click the Script button in the query:

Removing Profiles by Conditions

To remove profiles from the segment, click Remove. A similar conditions editor will open on the right. It allows you to configure filters for excluding profiles from the current segment. It also includes rules and rule groups, condition import, and audience limitation settings.

> NOTE
>
> **The Remove profiles that are in another segment function allows you to quickly exclude from the current segment all profiles that intersect with another segment.**

Audience Limitation

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can be included. If more people match the segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 profiles will be included in the segment. This restriction applies to the sample result of the segment. If a range is also used, the range restriction is applied first, and then the profile count limit is applied to the selected portion of the database.

Range

The range determines which portion of the entire database will participate in segmentation. By default, it is set to 100%, meaning the conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in the database is assigned a numeric value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls within the selected range. This allows you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% and a range from 80 to 100% will never intersect.

When new users are added to the database, they are distributed according to the same rules: if a 20% range is selected, a new profile has the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the database, and then the limit takes effect within that portion.

Importing Profiles from Another Segment

To import profiles from another segment, click the down arrow ▾ located next to the main Add button, and in the context menu, select Import from Segment:

This option allows you to quickly add to the current segment all profiles from another existing segment.

INFO: Important

You can only select a segment created based on the same profile database. After selecting a segment, profiles will be added, and the list, statistics, and history will be updated.

Manual Addition and Removal of Profiles

You can manually add or remove a profile from a segment. To do this, in the profile card, go to the Interactions tab in the Segments block, select the desired segment, and edit:

Viewing and Analyzing Segment Composition

Profiles Table

After calculation, a table with all profiles included in the segment is displayed in the static segment window. In addition to profile data, the table displays system fields, such as “Source Type” (for example, import, conditions) and “Addition Source” (segment or mailing identifier). These fields help track the history of how each profile entered the segment.

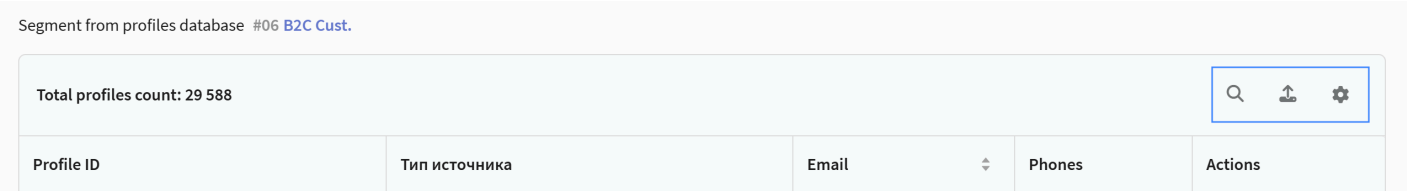
The following actions are available for working with this table:

- **Search**

  : find a specific profile by name, email, or other data.
- **Export Profiles**

  : export the displayed list to an external file.
- **Configure Displayed Fields**

  : to add or remove columns in the table, drag the desired Field from the side column Add Field to the Displayed Fields area or back.



Update History

Below the profiles table is the Update History block. It visually presents information about changes in the total number of profiles in the segment over the selected period. You can specify the desired period (start and end dates) and group data by days or weeks:

When hovering the cursor over a chart column, a tooltip appears. It shows the total change in the number of profiles and details: by which specific actions and in what quantity profiles were added to or removed from the segment:

The tooltip groups changes by the following action types:

Added

- **Import from another segment: adding all profiles from another segment.**
- **Import from file to database: automatic addition of profiles after importing a file into the Profile Database, if the corresponding option was enabled.**
- **By segment conditions: addition based on filters (the “Add” operation).**
- **From profile card: manual addition via the profile card (the Interactions tab).**
- **From scenario node: addition of profiles by the “Update Static Segment” node in a scenario.**
- **Via API: addition via API.**

Removed

- **Import from another segment: removal of all profiles that are in another segment (the “Remove profiles that are in another segment” operation).**
- **By segment conditions: removal based on filters (the “Remove” operation).**
- **From profile card: manual removal via the profile card.**
- **From scenario node: removal of profiles by the “Update Static Segment” node in a scenario.**
- **From profile database: removal of a profile from the Profile Database itself (leads to its automatic removal from all segments).**
- **Via API: removal via API.**

Next to each action type, the exact number of affected profiles is indicated.

Manual Changes Log

Below the update history is the Manual Changes Log. This is a detailed log of all operations that affected the segment’s composition. For each operation, the following is displayed:

- **Start Date: the time when the segmentation process started (addition/removal by conditions).**
- **End Date: the time when the process ended.**
- **Initiator: the name of the user who initiated the operation.**
- **Added: the number of profiles added as a result of the operation.**
- **Removed: the number of profiles removed as a result of the operation.**
- **Status: the current state of the operation (for example, “Completed”, “In Progress”, “Error”).**

Manual updates journal

Start date	End date	Initiator	Added	Deleted	Status
29-12-2025 14:11	29-12-2025 14:11	New User	17265	0	Completed
29-12-2025 14:07	29-12-2025 14:08	New User	5	0	Completed
16-12-2025 16:41	16-12-2025 16:41	New User	12317	0	Completed

NOTE

The log does not display entries about adding profiles via API or from a scenario.

Dependent Objects

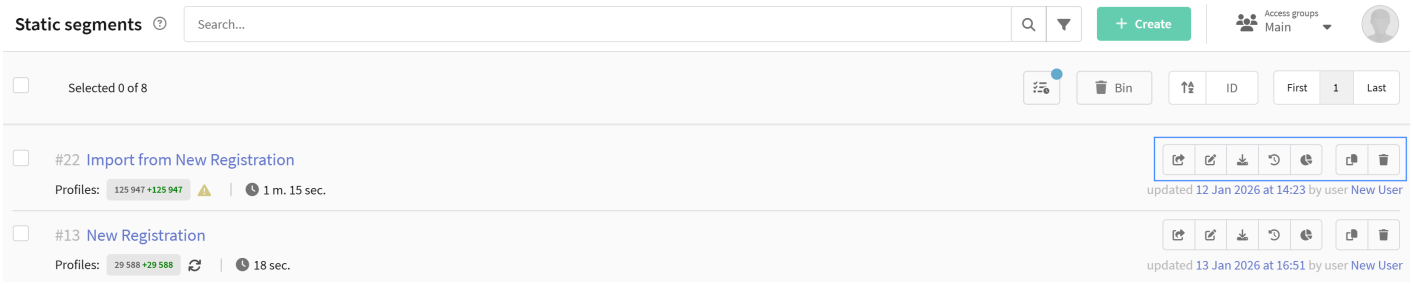
Below the manual changes log, the Dependent Objects block may appear. It appears automatically if the current static segment is used in other platform entities.

This block shows exactly where your segment is used (for example, in mailings or scenarios).

Segment Management

In the row of the desired static segment in the general list, the following actions are available for working with static segments:

- **Copy Profiles from Segment: Copy the list of profiles from the current segment to another selected segment.**
- **Update Profiles: Mass-modify profile data in the segment (for example, update fields, add a subscription).**
- **Export Profiles: Export data of profiles included in the segment to an external file.**
- **Export Profile History: Export interaction and change history for profiles from the segment.**
- **Analytics: Go to analytical reports for this segment.**
- **Clone Segment: Create an exact copy of the segment (with profiles, or without them).**
- **Delete Segment: Move the segment to the bin.**



Dynamic Segments

A dynamic segment is a segment that selects users directly at the moment of each use.

You can work with such a segment without saving it if you need a one-time quick selection. If you decide to save the segment, it can be used in mailings, scenarios, etc. Each time it runs, the segment will recalculate the profile selection based on the specified conditions.

	To get started, go to **Data → Segments → Dynamic Segments.**

Creating a Dynamic Segment

In the dynamic segments list, click + Create. In the form that opens, specify the Name of the segment and click on the Select source block. In the side panel, select one Profile Database — this is a mandatory step; segment creation is impossible without selecting a database:

INFO

Please note that after selecting a source, it cannot be changed.

Configuring Range and Conditions

Here you can also set the Range, which defines the portion of the database to which the segmentation conditions will be applied:

By default, the range is set from 0% to 100% of the entire selected profile database. Profiles are selected proportionally — the same range always selects the same users.

The main block is Condition Settings. To add a rule, click the + Condition button. In the dropdown menu, the options Add Condition Group or Import Conditions from another segment are available. After selection, the condition settings window will open:

The mechanics are completely identical to static segments: all the same filter categories by profile data, interactions, and behavior in channels are available. Conditions within a group can be combined using must comply with each of the conditions (AND) or must comply with one of the conditions (OR) logic.

In the “Condition Settings” block, the Delete button (clears the set rules) and the Script button (opens an editor for manually writing conditions in JSON format) are also available.

After configuring the conditions, in the same window, you can activate the checkbox Set a limit on the maximum in the segment and specify the desired number. This limitation is applied to the selection result after selection by range and conditions:

More about how audience limitations work

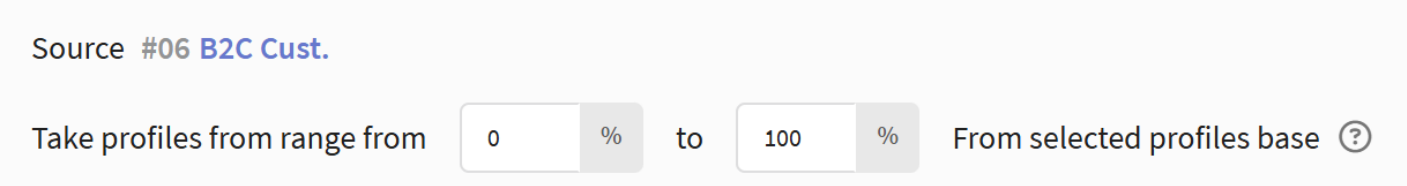
Limit on the maximum

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can be included. If more people match the segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 profiles will be included in the segment. This restriction applies to the sample result of the segment. If a range is also used, the range restriction is applied first, and then the profile count limit is applied to the selected portion of the database.

Range



The range determines which portion of the entire database will participate in segmentation. By default, it is set to 100%, meaning the conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in the database is assigned a numeric value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls within the selected range. This allows you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% and a range from 80 to 100% will never intersect.

When new users are added to the database, they are distributed according to the same rules: if a 20% range is selected, a new profile has the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the database, and then the limit takes effect within that portion.

Working with the Segment

Quick Segment

Before clicking the “Save” button, a dynamic segment can be used as a quick segment — for example, to export profiles or copy them to a static segment without the need to save the segment itself.

After adding at least one condition, additional actions for working with the current profile selection become available in the upper right corner, next to the fields for tags and access groups:

- **Copy Profiles to Profile Database;**
- **Copy Profiles to Static Segment;**
- **Export Profiles;**
- **Bulk Edit Profiles;**
- **Add Profiles to a Scenario.**

Preview and Save

Also, after configuring the conditions, you can immediately estimate the size of the future audience. Click the Calculate button in the profile calculation block — the system will process the database and show how many profiles match the conditions:

To see the profiles themselves, click Show — a table with the first 100 profiles from the calculated selection will open:

After calculation, the profile count can be manually updated using the corresponding button next to the counter.

Fill in the remaining fields in the form: if necessary, specify Description, Tags, and Access Groups. Then click Save. After saving, the segment will automatically calculate its audience for the first time.

INFO

If no conditions are set when saving the segment, all profiles from the selected database within the set range will automatically be included in the segment.

Dependent Objects

If a dynamic segment is used in other platform entities (for example, in mailings or scenarios), the Dependent Objects block will automatically appear below the update log. This block shows exactly where your segment is used:

Managing Segments from the General List

In the row of the desired dynamic segment in the general list, the following actions are available for working with dynamic segments:

- **Copy Profiles from Segment: Copy the list of profiles from the current segment to another selected segment.**
- **Update Profiles: Mass-modify profile data in the segment (e.g., update fields, add a subscription).**
- **Export Profiles: Export data of profiles included in the segment to an external file.**
- **Export Profile History: Export interaction and change history for profiles from the segment.**
- **Analytics: Go to analytical reports for this segment.**
- **Clone Segment: Clone the segment conditions, tags, access groups, and Profile Database.**
- **Delete Segment: Move the segment to the bin.**

Updatable Segments

A scheduled segment is a segment that automatically recalculates its composition based on specified conditions according to a set schedule (for example, once a day). Its list of profiles is current at the moment of the last recalculation.

INFO

- **Filling: The segment cannot be filled manually or via API. It is filled only based on specified conditions according to the schedule or during manual recalculation.**
- **Usage: The segment can be used in the same places as a static segment (for example, in mailings, scenarios, triggers).**

Creating a Segment

	To get started, go to **Data → Segments → Scheduled Segments.**

In the scheduled segments list, click + Create. In the form that opens, specify the Name of the segment — this is a required field. Then select the segment source (Database). Segment creation is impossible without selecting a database:

INFO

After selecting a source, it cannot be changed.

Configuring Conditions

After selecting the database, the Conditions and Schedule block will open. The mechanics of configuring conditions are completely identical to dynamic and static segments. You can set the Range, which defines the portion of the database for applying conditions.

To add a rule, click the + Condition button. In the dropdown menu, the options Add Condition Group and Import Conditions are available. After configuring conditions within a group, they can be combined using must comply with each of the conditions (AND) or must comply with one of the conditions (OR) logic:

In the same block, the Delete button, which completely clears all set rules, and the Script button, which opens an editor for manually writing conditions, are available.

After configuring the conditions, you can set limitations by activating the checkbox Set a limit on the maximum number of profiles in the segment to specify a limit. This limitation is applied to the selection result after selection by range and conditions.

More about how audience limitations work

Limit on the maximum

Maximum profile count limit — the limit on the number of profiles that can be included in a segment.

This method limits the size of the final segment. You specify the maximum number of profiles that can be included. If more people match the segmentation conditions, the platform takes the first profiles from the resulting sample.

For example, if 1,000 people match the conditions and the maximum count is set to 400, the first 400 profiles will be included in the segment. This restriction applies to the sample result of the segment. If a range is also used, the range restriction is applied first, and then the profile count limit is applied to the selected portion of the database.

Range

The range determines which portion of the entire database will participate in segmentation. By default, it is set to 100%, meaning the conditions apply to all profiles. The key feature is that the range is tied to specific users: each profile in the database is assigned a numeric value, and the same range always selects the same people.

This means the selection is not random. A profile is included in the segment if its assigned value falls within the selected range. This allows you to create non-overlapping groups by setting different ranges — for example, a range from 0 to 20% and a range from 80 to 100% will never

intersect.

When new users are added to the database, they are distributed according to the same rules: if a 20% range is selected, a new profile has the same probability of falling into it.

If a maximum profile count limit is also used, the range is applied first to determine the portion of the database, and then the limit takes effect within that portion.

Configuring the Schedule

In the Conditions and Schedule block, configure the parameters for automatic updates. Specify the exact time for launch (HH:MM) and select the timezone. Then set the frequency of recalculation from the available options: daily, on a specific date, by time (days of the week, days of the month, etc.):

Optionally, you can activate the function Deactivate schedule after the specified date so that automatic updates stop on a given day.

After configuring all parameters, click Save. The first calculation will start at the nearest scheduled time.

Working with a Created Segment

After creating and saving the segment, its page will open. Now, in the upper right corner, next to the fields for tags and access groups, additional actions for working with the current profile selection become available:

- **Copy Profiles to Profile Database;**
- **Copy Profiles to Static Segment;**
- **Export Profiles;**
- **Bulk Edit Profiles;**
- **Add Profiles to Scenario.**

> INFO
>
> **Actions with profiles will become available after the first segment update.**

Managing Automatic Updates

The Automatic Segment Update block displays the current schedule. From here, you can Stop"/"Start (depending on the current state), Reconfigure Schedule, or Launch Manual Recalculation outside the scheduled time.

> CAUTION
>
> **Manual recalculation and editing of the update schedule are unavailable during segment update execution (automatic or manual).**

Below is the Show Update Conditions spoiler. When expanded, it displays all set conditions, range, and limit. They can be edited, the script can be viewed, or they can be deleted.

Analyzing Composition and History

After the first update, the Profiles Table with the current composition will become available on the segment page:

Update History

The Update History block is a graph visualizing changes in the number of profiles in the segment over the selected period. The period can be set using the calendar or ready-made options (today, last 7 days, this month, etc.), and data can be grouped by days, weeks, months, quarters, or years. When hovering over a point on the graph, a breakdown by profiles is displayed: added or removed automatically and manually:

Log

The Log block is a table with detailed information for each recalculation run. For each operation, the start and end date and time, type (Manual / Automatic), initiator (user or “System”), number of added and removed profiles, and status (`In Progress`, `Completed`, `Error`) are indicated.

Dependent Objects

If the segment is used in other platform entities (mailings, scenarios, loyalty programs), the Dependent Objects block will appear below the update log:

Managing Segments from the General List

In the row of the desired scheduled segment in the general list, the following actions are available:

- **Copy Profiles from Segment: Copy the list of profiles from the current segment to another selected segment.**
- **Update Profiles: Mass-modify profile data in the segment (e.g., update fields, add a subscription).**
- **Export Profiles: Export data of profiles included in the segment to an external file.**
- **Export Profile History: Export interaction and change history for profiles from the segment.**
- **Analytics: Go to analytical reports for this segment.**
- **Clone Segment: Clone the segment conditions, tags, access groups, profile database, and schedule.**

- **Delete Segment: Completely delete the segment.**

Best Send Time (BST)

Best Send Time (BST) is a mechanism that determines the optimal time to send messages for each profile based on its behavior. It is powered by a machine learning algorithm that identifies the time of highest profile activity based on data collected by the platform.

Sending messages at the right time directly impacts communication performance:

- **Open Rate increases as messages arrive when users are most active;**
- **engagement grows — users interact with content more frequently;**
- **the risk of messages being marked as spam decreases due to higher open rates;**
- **customer loyalty improves thanks to more relevant communication;**
- **resources are optimized through more efficient campaigns.**

How Best Send Time is calculated

The BST model uses machine learning algorithms to analyze user behavior. The ML service analyzes profile activity and determines the hour when the probability of message opening is highest. For each profile, one or several preferred sending hours are calculated.

During analysis:

- **data from the last 6 months is used;**
- **message opens and other behavioral events are processed;**
- **the best hour or several hours for sending are determined.**

BST data is regularly recalculated (by default, once a day at 00:00 in the account time zone).

> CAUTION
>
> **By default, the ML model uses profile activity data from the last 6 months to determine the optimal sending time. If a profile has no history of sends and opens within this period, the system cannot calculate BST for it.**

Using Best Send Time

BST-based segmentation allows selecting profiles for which the current or specified time is optimal for sending. Then you can configure a Mailing by selecting the segment audience as recipients — each user will receive the message at the most appropriate time.

> INFO
>
> **To use the ML service, it must be activated at the platform level. If you are using the On-premises version, contact the Altcraft support team.**

To calculate the best send time, you need to enable ML for the profile database. Go to the database settings, open Machine Learning Settings, and activate the module.

After activation, the model will start collecting and processing data. You can check the next update time in the database settings. If ML was enabled recently, wait for the first recalculation.

<Slider images={[{src:“/img/attachments/89325640/en/enable.png”}, {src:“/img/attachments/89325640/en/recount.png”}]}/>

BST can be used as a condition for building a dynamic segment or in an automation Scenario node.

BST Segmentation

Scenario Node

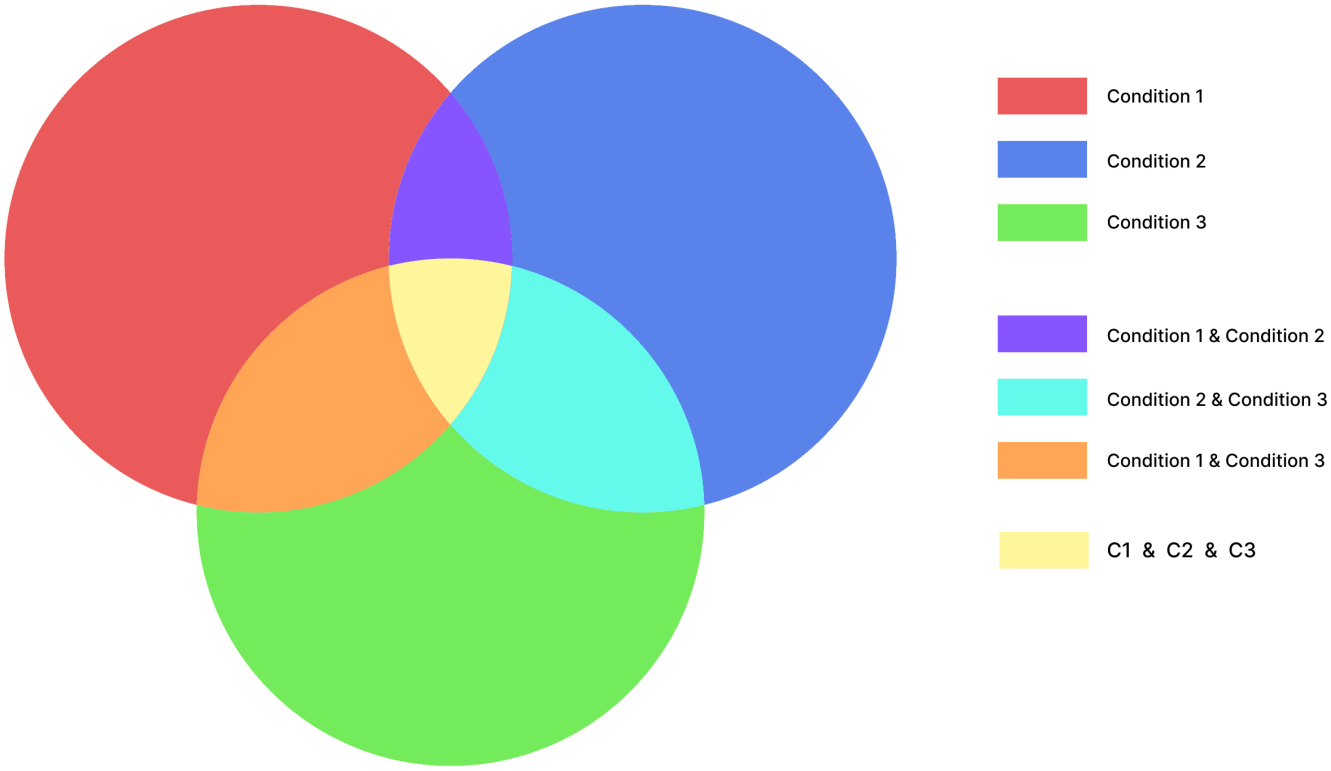
This approach in Scenarios is suitable for complex communications with trigger chains and advanced logic.

Logical operators “AND” and “OR”

Segmentation on the platform is based on conditions. Often, a single condition is not enough to calculate the audience required for a mailing. In such cases, it is necessary to specify two or more selection conditions or group of rules. The conditions are linked together using the logical operators of formal logic - AND and OR. In this article, we will use examples to explain how conditions work with these operators.

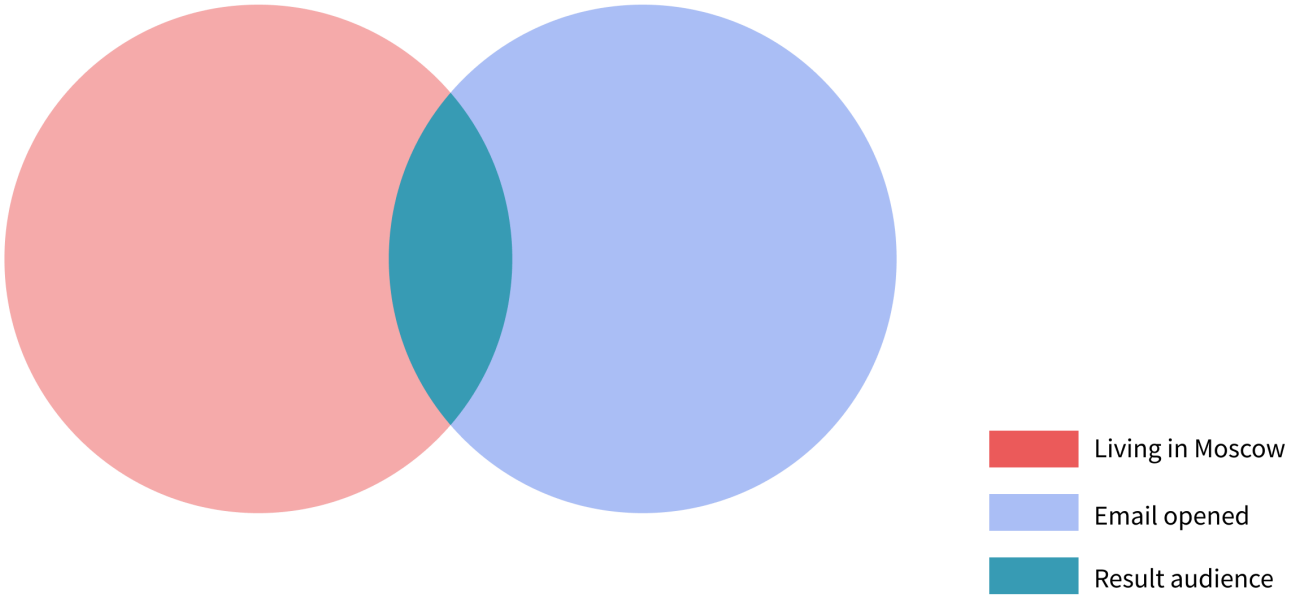
Operator "AND"

The “AND” operator will select those subscriberss for which all rules and rule groups in the query are valid. This operator implies that all elements of the elements being joined must strictly match the condition. It can be represented schematically:

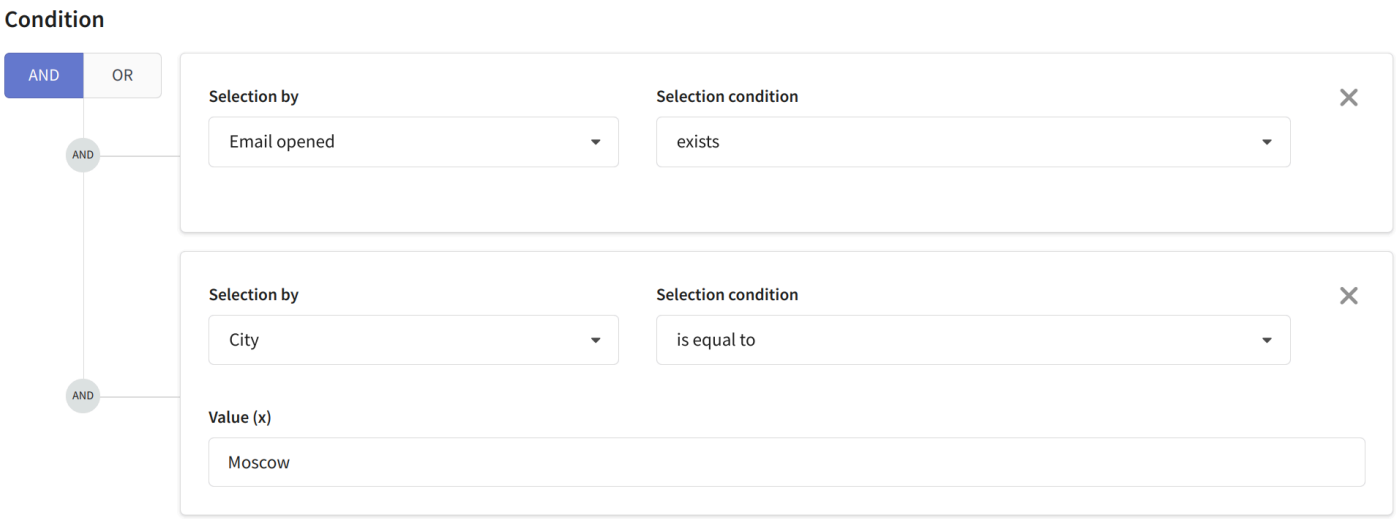


Example

Let’s model the situation. To send out an invitation to a promotional event, we need to select subscribers living in Moscow who have opened your email at least once:

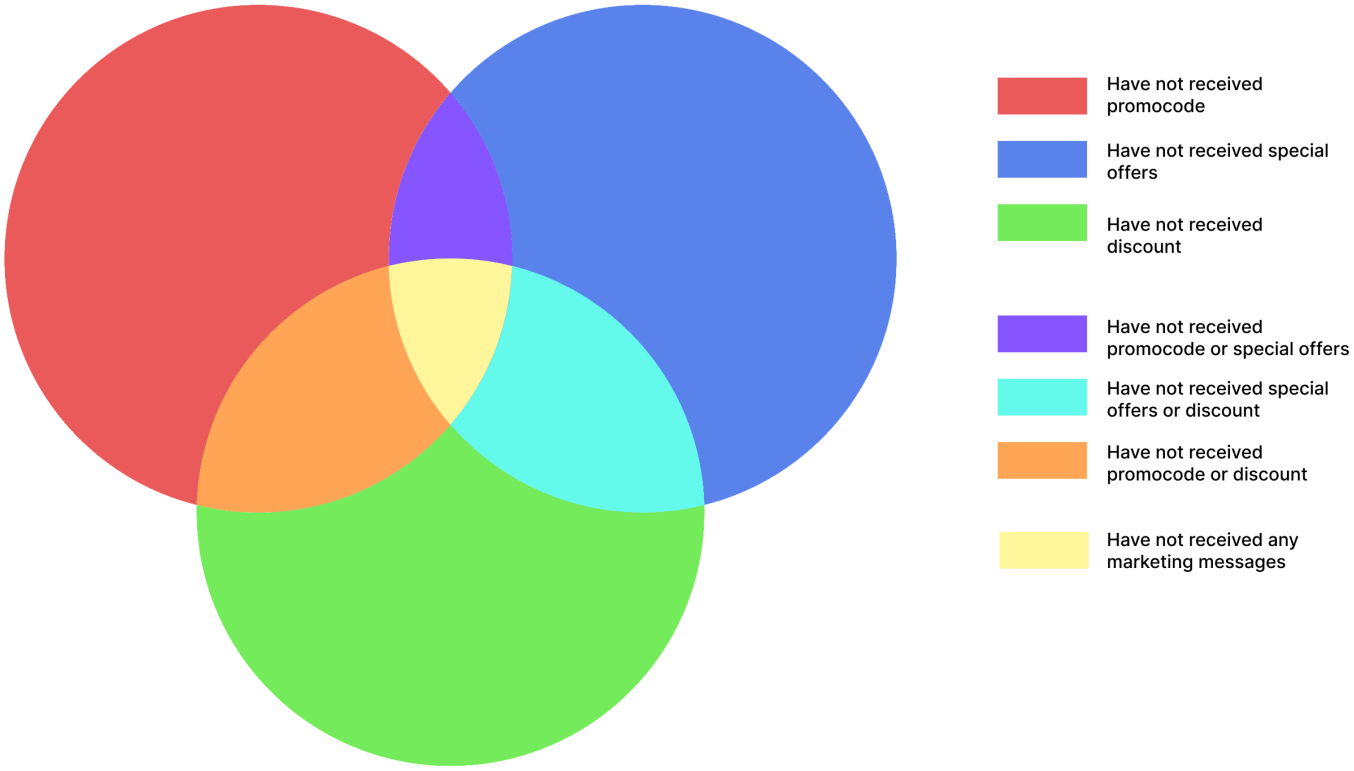


The conditions will look like this:



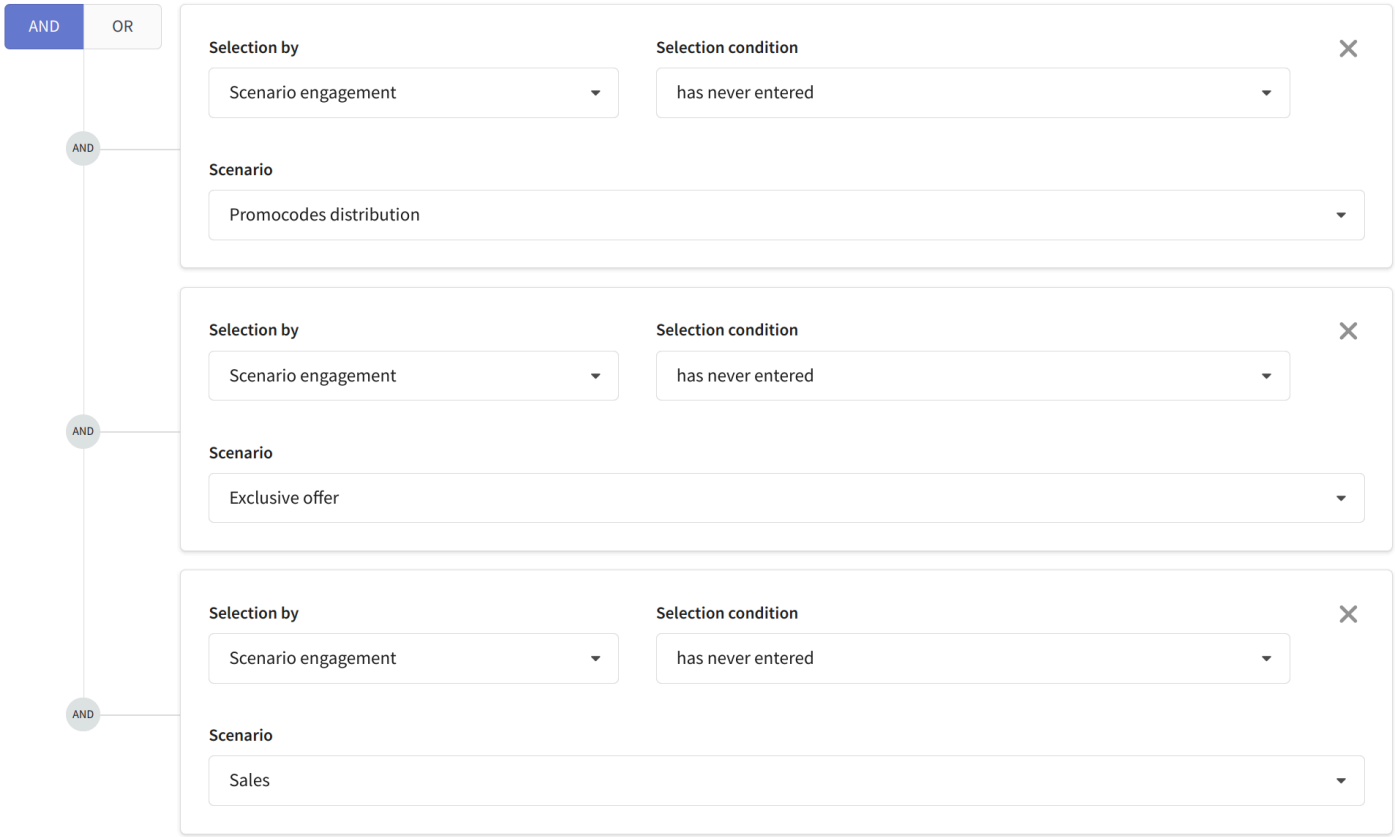
This way, we will select those customers who meet both conditions.

Let's say you've previously created several scenarios with different offers. Now you need to select those who have not yet received messages from you:



In this case, the conditions will also be connected by the "AND" operator:

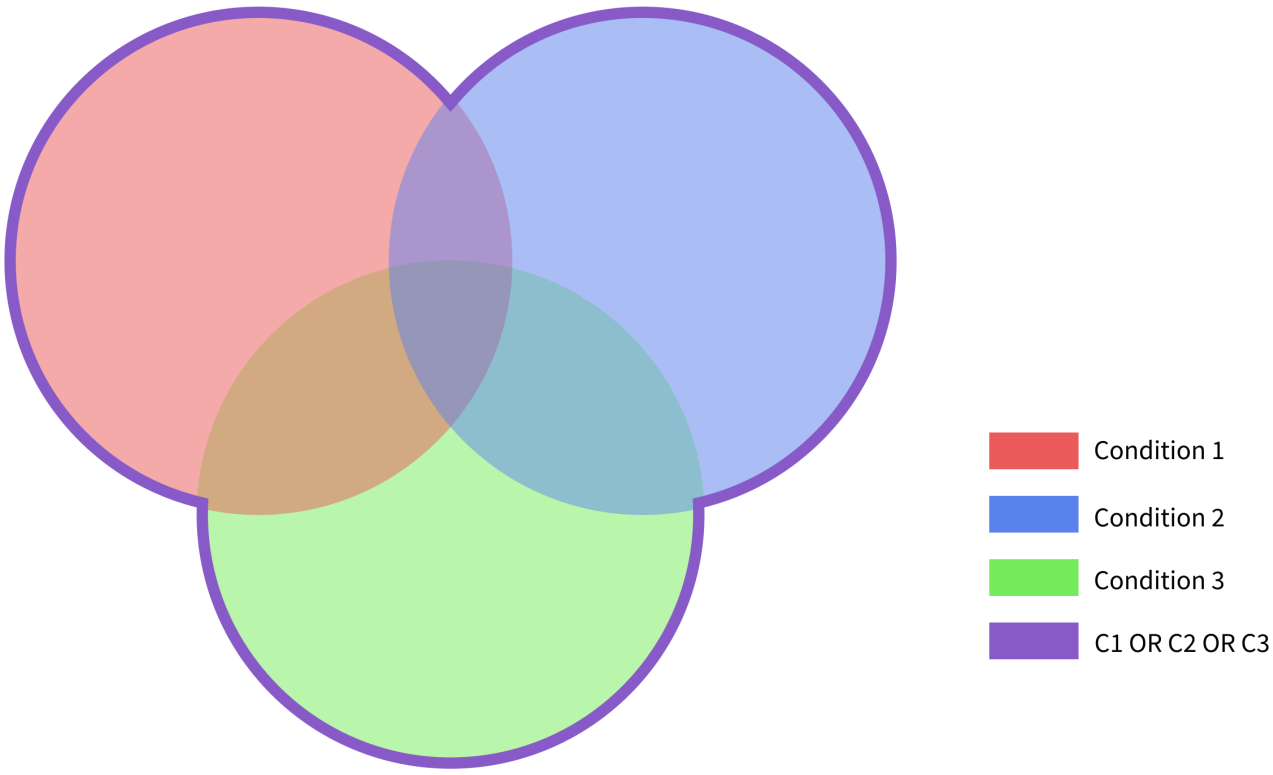
Condition



This segment will include those profiles that have not received any discount messages, emails about exclusive offers, or distributions with promo codes.

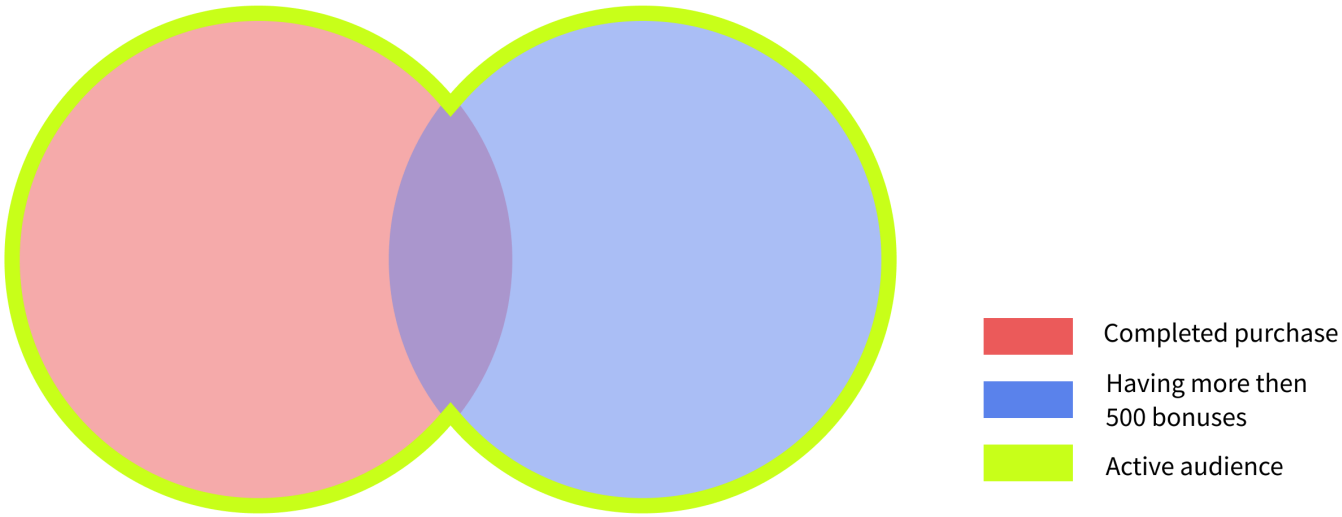
Operator "OR"

The “OR” operator selects those profiles for which at least one rule or group of rules is valid. If a profile meets a single condition, it will be included in such a segment. The scheme of its work is as follows:

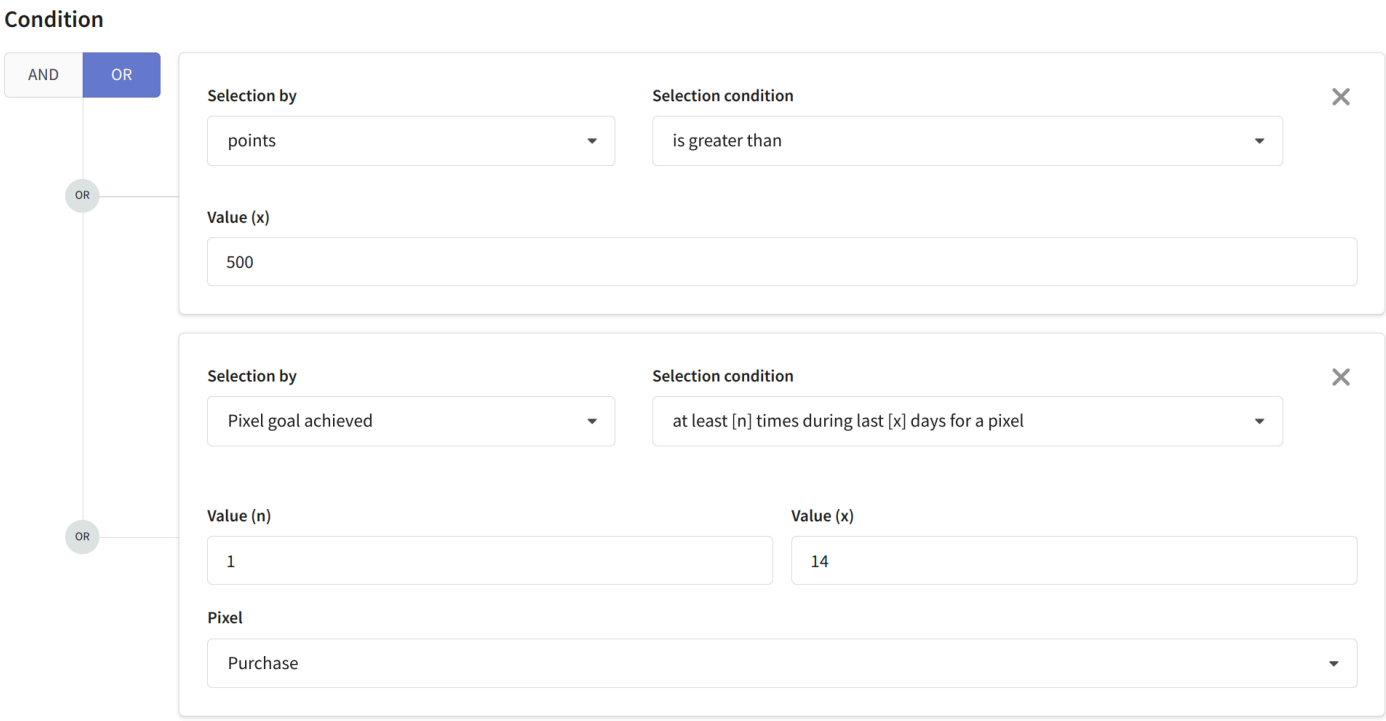


Example

Let's assume that you came up with an idea to reward active customers with a special promo code. We will consider as active those who either made a purchase in the last 2 weeks or have more than 500 bonus points on their account:

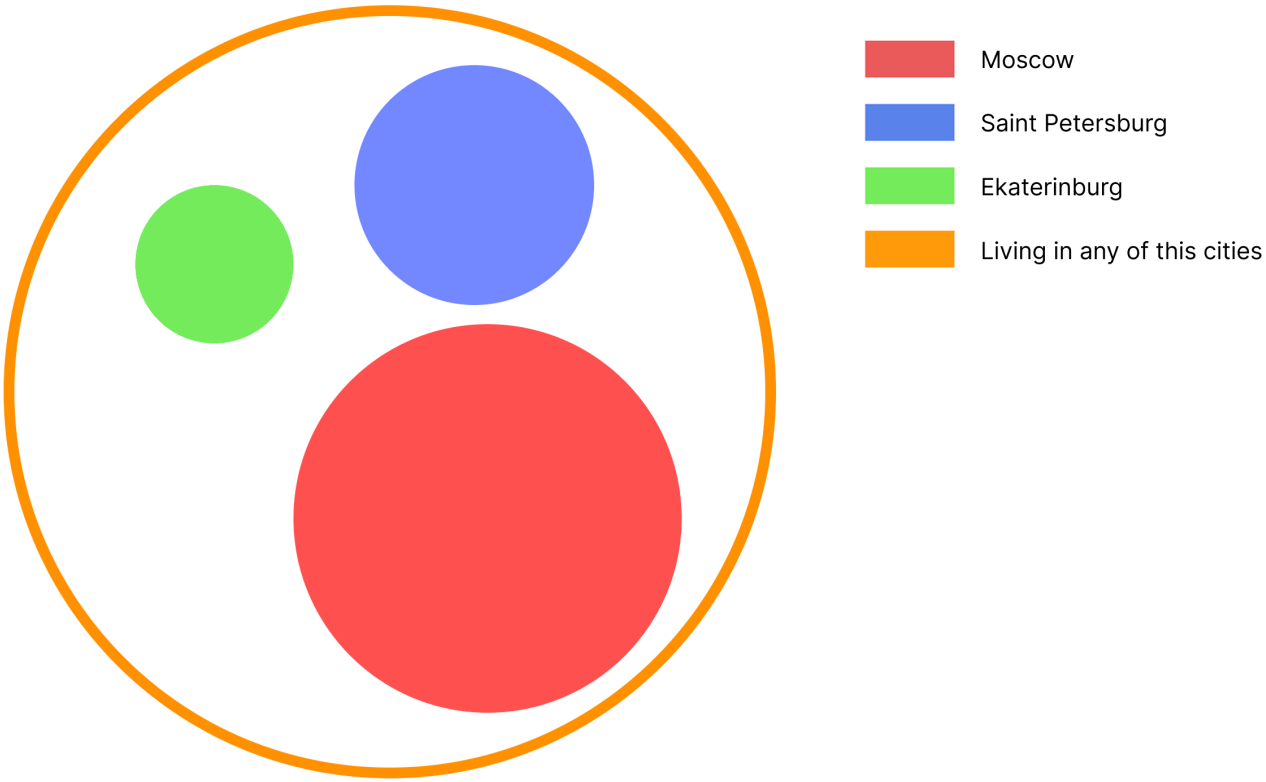


And a condition:



If any of these conditions are met, the customer will receive a promo code.

If you need to select people from few different cities as your audience, you would also use the OR operator when building your segment:



The conditions will look like this:

Condition

AND OR

OR

Selection by: City — Selection condition: is equal to — Value (x): Moscow

OR

Selection by: City — Selection condition: is equal to — Value (x): Saint Petersburg

OR

Selection by: City — Selection condition: is equal to — Value (x): Ekaterinburg

INFO

Using a condition with the “OR” operator and negations can lead to errors. Such conditions often contain a built-in logical flaw.

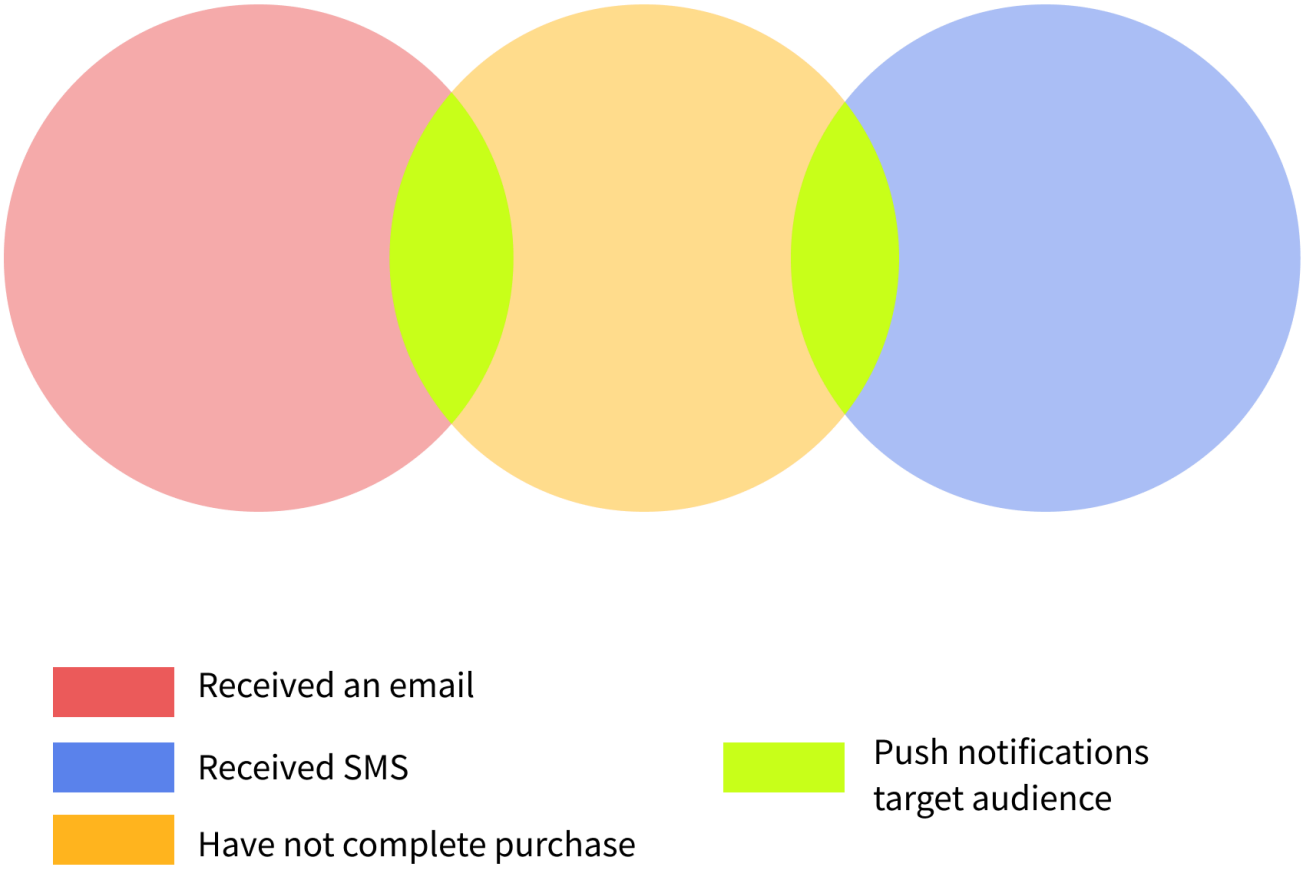
Avoid using these constructions — it’s better to replace them with equivalent conditions using “AND”. If you do use them, be sure to double-check with a visual logic diagram.

Group of rules

You can link multiple rules of the same level using either OR or AND, but you cannot use them together. To build complex schemes, you must use a group of rules. Within a rule group, you can use an operator different from the one used to link conditions of another level. This allows you to build more complex audiences for your marketing mailings.

Example

Let's explain it with an example. Let's imagine a situation in which you need to create a Viber mailing for inactive customers - for those who have received both email and SMS messages, but never made a purchase. Schematically, the audience will look like this:



For such a condition, the rule set will be composed as follows:

Condition

AND | OR

Query group

AND | OR

AND

Selection by: SMS sent

Selection condition: exists

OR

AND

Selection by: Pixel goal achieved

Selection condition: not happened to a pixel

Pixel: Purchase

+ Add rule | + Import conditions

Query group

AND | OR

AND

Selection by: Email sent

Selection condition: exists

OR

AND

Selection by: Pixel goal achieved

Selection condition: not happened to a pixel

Pixel: Purchase

+ Add rule | + Import conditions

Recommendations for working with segments

Segments are a key tool for working with data in a CDP platform. The quality of selections, analytics, and marketing mailing launches depends on how they are configured. However, in practice, there are situations when a segment does not work as expected: it is not recalculated, not updated, or returns unexpected results.

To simplify troubleshooting, this article provides recommendations for working with segments.

Too many “NOT” conditions

Excessive use of negations, especially the NOT IN SEGMENT operator, complicates logic and reduces performance.

Special attention – the “NOT IN SEGMENT” operator.
It is the most performance-heavy operator. When used, the system dumps the entire profile database to intersect with the excluded segment. This is very slow.

Example of a problematic construct:

```
NOT IN (segment A)
AND NOT IN (segment B)
AND NOT IN (segment C)
...
```

Recommendations:

- **Avoid using “NOT IN SEGMENT” whenever possible. Instead, duplicate the negative conditions from the selected segment. For example, if the segment has a single condition “not in Global Control Group”, use a direct condition “is in GCG”.**
- **Do not use more than 2–3 “NOT” conditions (of any kind) in a row without a good reason.**
- **Replace a set of “NOT” conditions with a single positive rule. Instead of listing where an object should not be, describe where it should be.**
- **Document segmentation rules: state why each negation is added. This helps avoid situations where one condition becomes irrelevant but the logic remains unnecessarily complex.**
- **Use “NOT IN SEGMENT” only when it is really unavoidable (e.g., the excluded segment is dynamically updated by another team).**

Using the “NOT” condition with “OR”

When building a segment, a condition like the following is often set:

```
NOT (has active contract) OR NOT (verified)
```

At first glance, it may seem that this condition excludes all customers without a contract and without verification. In practice, it works differently: the segment will include all customers except those who simultaneously have both an active contract and verification completed.

This means that the condition excludes only those for whom both conditions are true at the same time, and not everyone who has at least one of them.

Recommendations:

- **Avoid long expressions that alternate NOT and OR.**
- **If you need to select only customers who have neither a contract nor verification, use:**

```
NOT (has active contract) AND NOT (verified)
```

Indexes and segment performance

Indexes are internal database structures that speed up searching for profiles by field values. Without an index, the system has to iterate through all customers — the segment builds slowly, especially on large databases.

Some indexes are created automatically when the platform starts. These are system indexes necessary for platform operation. You cannot manage them, and their set may differ between installations. Queries on such fields are usually fast, but you shouldn’t rely on this for any field.

Recommendations

Analyze which segments you build most often and which fields you use in them. Pass this list to your administrator. The administrator will check which indexes already exist and, if necessary, create additional ones within the limits.

CAUTION

No more than 64 indexes can be created per account. For more details on database indexes, see the administrator documentation.

Storing lists of values

If you need to select profiles by a specific value from a list (e.g., interests, categories, tags), it is better to store such lists in a “tags” field type rather than in a string.

For example, the `interests` field contains `tennis, football, chess`. If you need to select everyone interested in tennis, the segmentation process will take longer when stored in a string than when stored in a “tags” field.

Date range

If a segment condition uses a date (registration date, order date, goal reached, email opened, etc.), the recalculation will be faster when the date range is limited. The fewer dates need checking, the less time segmentation takes.

Recommendations:

- **Specify a concrete period whenever possible. Instead of “goal reached”, use “goal reached in the last 7 days”.**
- **Instead of checking events for all time, use profile attributes that don’t require scanning history (e.g., “last order date” instead of “had at least one order”).**
- **For recurring segments, use relative dates (“last N days”), but don’t make N too large — usually 30–90 days are enough.**

Segmentation Conditions

This section contains conditions for segmenting profiles:

Segmentation by Profile data

Profile data is the main information about a subscriber that can be used for precise audience segmentation. This section combines everything: from name and email to system labels and subscription statuses.

Main Fields

Main fields include standard profile data such as Email, Phone, First Name, Last Name, Date of Birth, and others. These fields can be of different types, and each type has its own comparison operators.

Operators

For most fields, the following operators are available:

- **Is equal to / Is not equal to — exact match of the subscriber's field to the segmentation rule. The value in the rule is set manually or selected from a list.**
- **Exists / Does not exist — checks whether the field is active for a specific profile. An existing field can be empty or contain a value.**
- **Empty / Not empty — checks whether an existing field contains any value.**
- **Is in file in column / Is not in file in column — the rule looks for a match of the field in a column of a tabular file (CSV, xls) that can be attached to the rule.**
- **Is in data table / Is not in data table — the rule looks for a match of the field in a query to an external SQL database.**
- **Is in a field of post url response / Is not in a field of post url response — the rule looks for a match of the field in the response of a post method request to an external API.**
- **Is in list / Is not in list — the rule looks for a match of the field in a list that can be copied into the data input area. Values must be separated by a line break (pipe).**

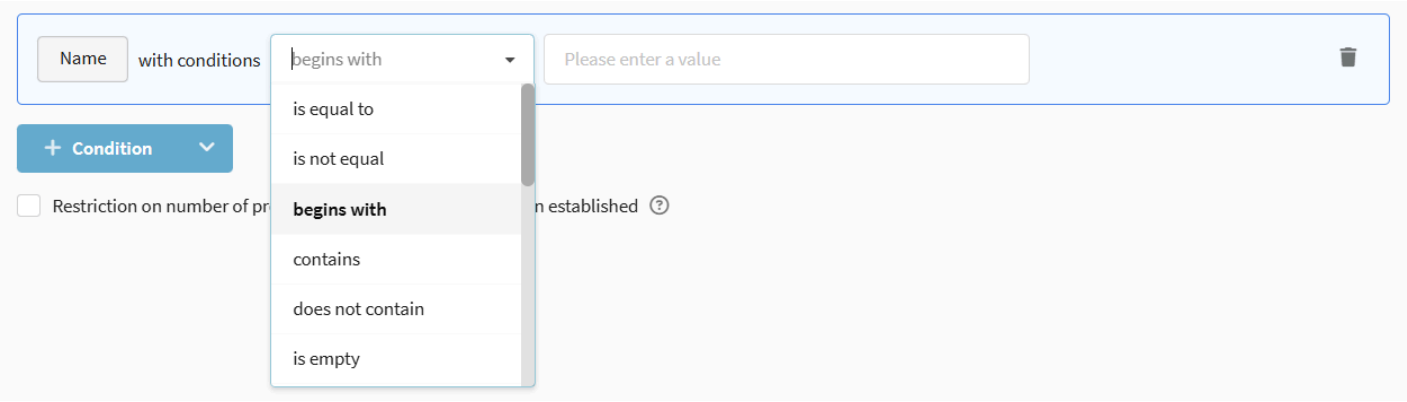
Field Types and Their Operators

String Fields

String fields store any values: letters, numbers, special characters. These include: `First Name`, `Last Name`, `Registration Link`, `City`, `Region`, `Zip Code`, **and** `Provider`.

For string fields, an additional set of operators is available:

- **Starts with — search based on the beginning of the field value.**
- **Contains / Does not contain — search based on a part of the field value (case-insensitive). The "Does not contain" operator consumes a lot of system resources.**

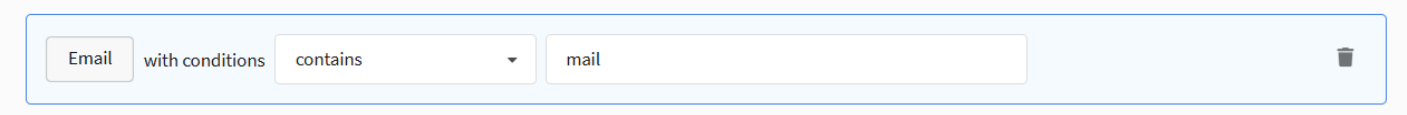


> **NOTE**
>
> **The standard Phones field has the same set of segmentation rules as string fields. However, this field can only store phone numbers in the format +71234567890.**

Email Type Fields

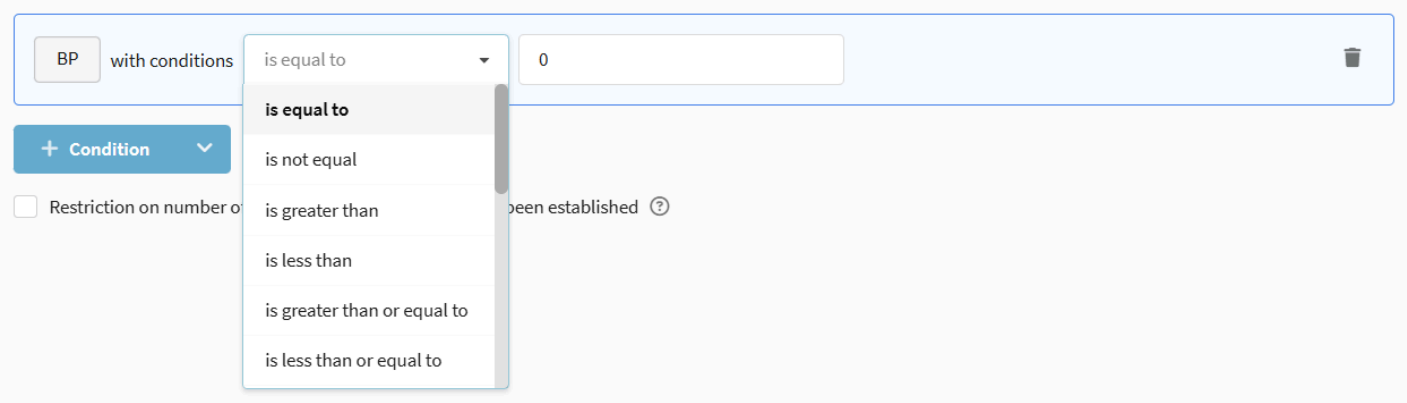
The Email field is used to identify a profile and has a special set of operators related to domains and domain groups.

- **Domain group is / Domain group is not — search for email addresses of a selected domain group (mail.ru, yandex.ru).**
- **Domain is / Domain is not / Domain contains — search by the domain of subscribers' email addresses or part of it (bk.ru, list.ru, ya.ru, organizational domains).**



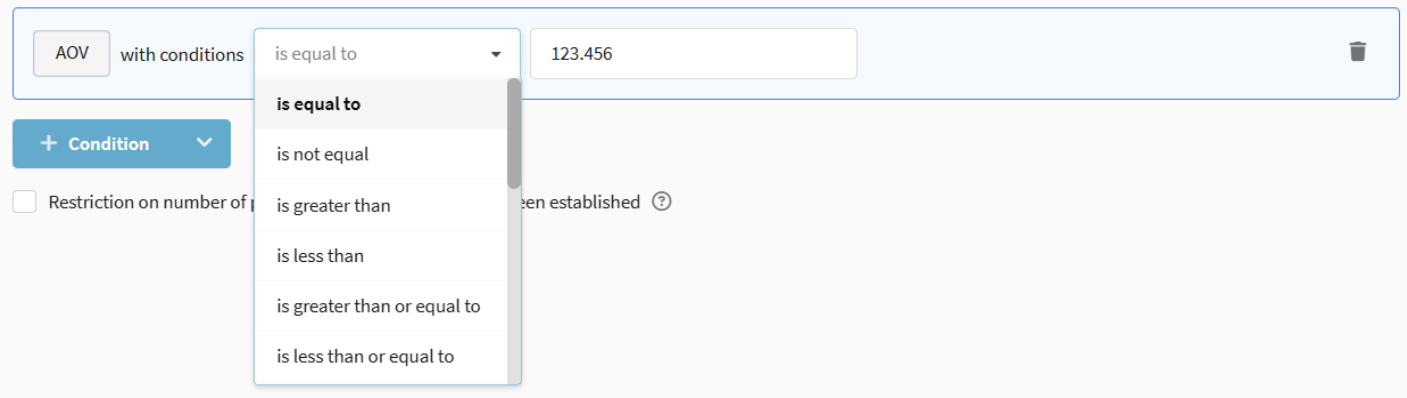
Integer Fields

Integer type fields can contain positive and negative whole numbers. Available operators: Is equal to / Is not equal to / Greater than / Less than.



Floating Point Fields

Floating-point fields can contain positive and negative decimal values. Available operators: Is equal to / Is not equal to / Greater than / Less than.

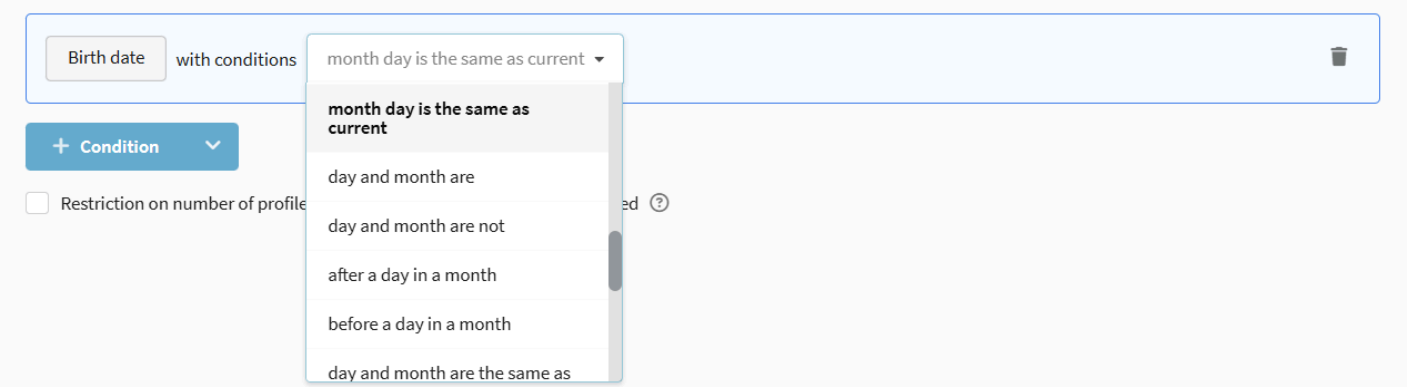


Date

Date type fields store dates in RFC 3339 format (e.g., 2018-02-22T21:00:00Z). These include: `Registration Date` **and** `Date of Birth`**.**

Date fields have an extended set of segmentation operators:

- **Greater than / Less than — selection by date after or before the specified one.**
- **Is in a date range — selection by date falling within the specified period.**
- **Date/age relative to current date — selection by a moment in the past or future.**
- **Month is / Month is not / Month is the same as current — selection by month.**
- **Month day is / is not / after month day / before month day / month day is the same as current — selection by day of the month.**
- **Day and month are / are not / after / before a day in the month / day and month are the same as current — selection by day and month (without year).**

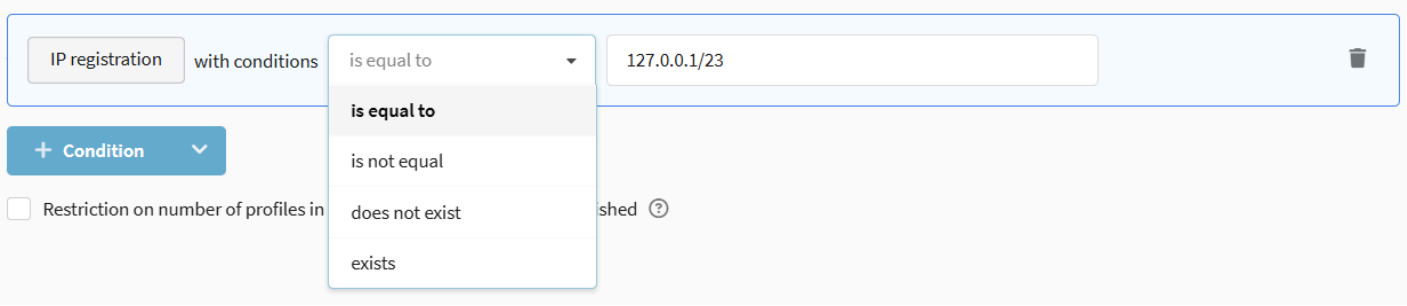


TIP

Dynamic segmentation by a date field is excellent for trigger reminders (e.g., about the end of a service term). To do this, set up a daily regular mailing based on such a segment.

IP Address

IP Address type fields store addresses in IPv4 format. These include `IP Address` and `Registration IP Address`. For IP addresses, the following operators are available: Is equal to / Is not equal to, Exists / Does not exist.



NOTE

For IP, network mask search is implemented:

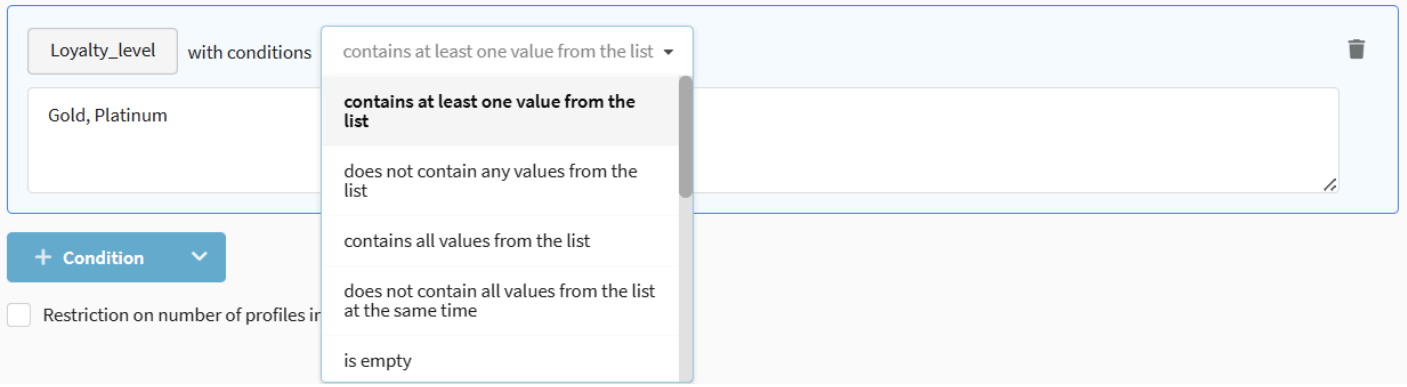
- **Is equal to — find subscribers whose IP addresses are within the specified subnet (e.g., `127.0.0.1/25`).**
- **Is not equal to — find subscribers whose IP addresses are not within the specified subnet. To search by range, combine two rules with the AND condition.**

Tags

The Tags field is a list separated by commas. These can be, for example, purchased products.

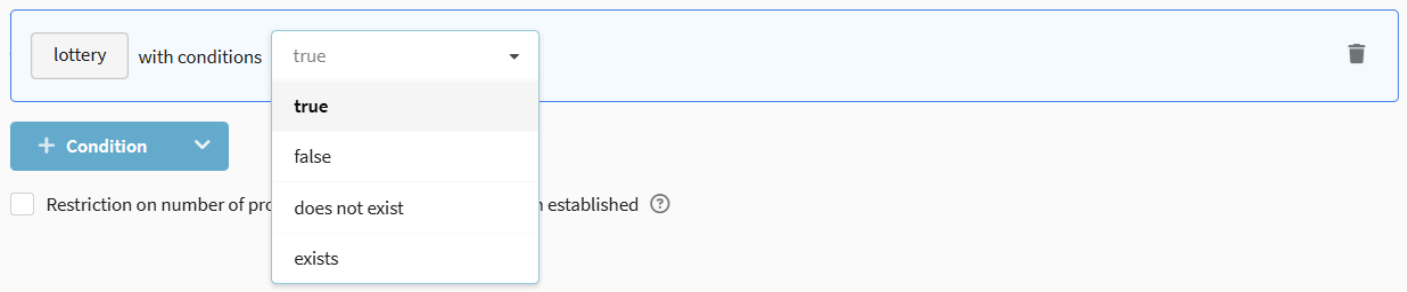
Tag segmentation operators:

- **Contains one of — the field contains at least one of the values from the rule.**
- **Contains none of — the field does not contain any of the values from the rule.**
- **Contains all of — the field contains all values from the rule.**
- **Contains not all of — the field does not contain some of the values from the rule.**



Boolean Field

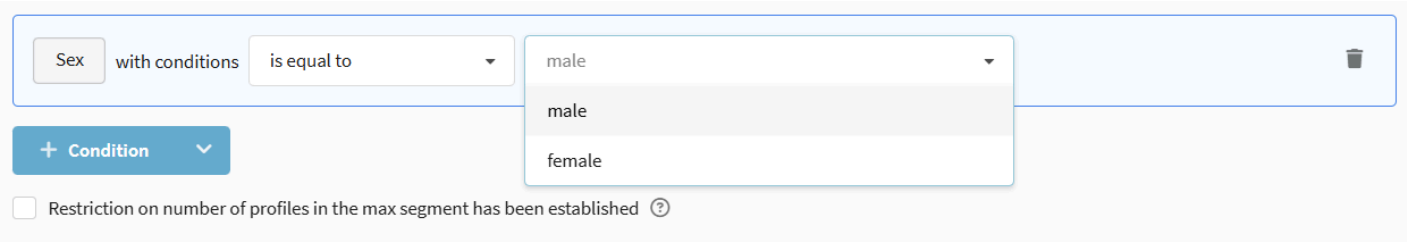
A field containing logical YES/NO (true/false). Convenient for statuses or presence of characteristics. Operators: Is equal to true / Is equal to false.



List (Enumeration)

A field containing one of predefined values. Among standard fields, the Sex field has this type.

Operators can only compare the value with one selected from the list or check for the existence of the field.



Custom (Additional) Fields

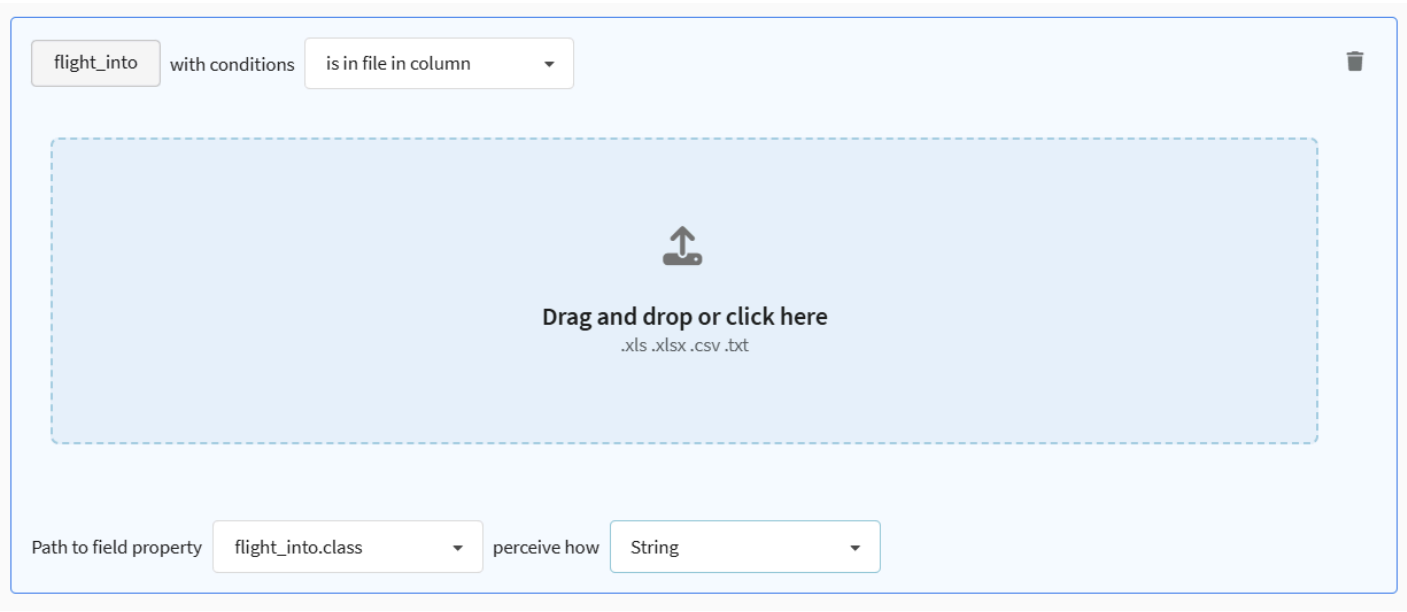
You can create your own fields in the database structure to store specific information. The same operators are available for segmentation as for main fields, depending on the field type.

Object

The field contains an arbitrary object for storing complex data structures.

Available conditions: Exists / Does not exist, as well as checks in lists and external sources.

If necessary, specify the path to a specific property of the field. The type of the property will be determined automatically, but you can override it by changing the value in the “Perceive type as” field.

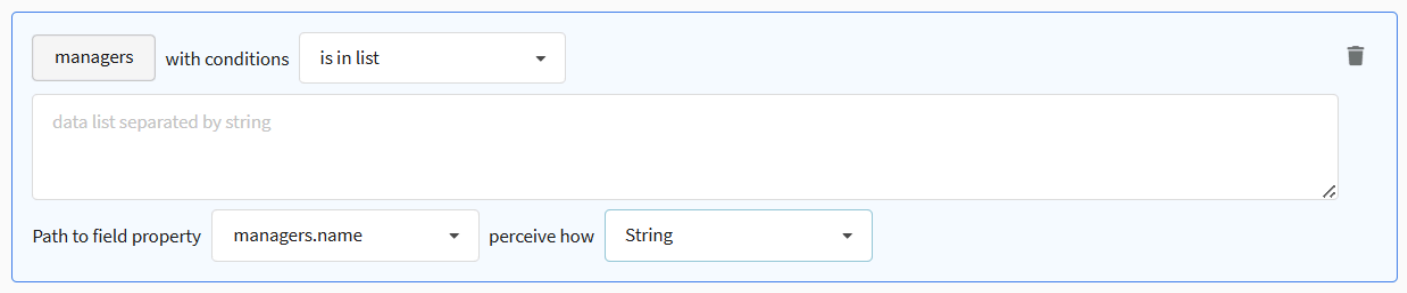


Array of Objects

The field contains an array of arbitrary objects for storing a sub-table (e.g., a list of contracts).

Available conditions: Exists / Does not exist, as well as checks in lists and external sources with the specification “all are in”, “at least one is in”, or “is not in”.

You can also specify the path to the property and its type.



System Fields

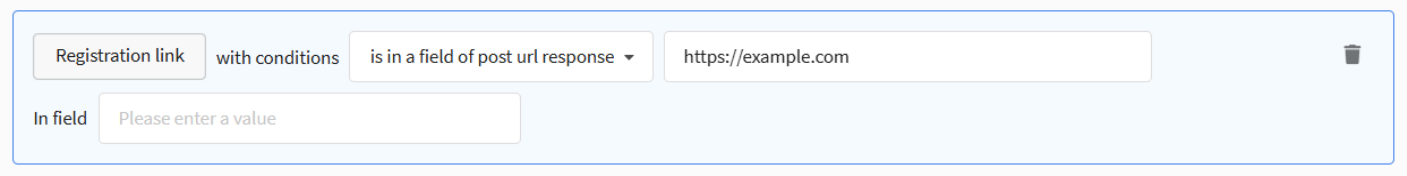
System data available for segmentation includes fields automatically filled by the platform.

Profile Status

Contains only the global profile status, which determines the possibility of communication with the client. Select the required status from the dropdown menu.

Profile Identifier

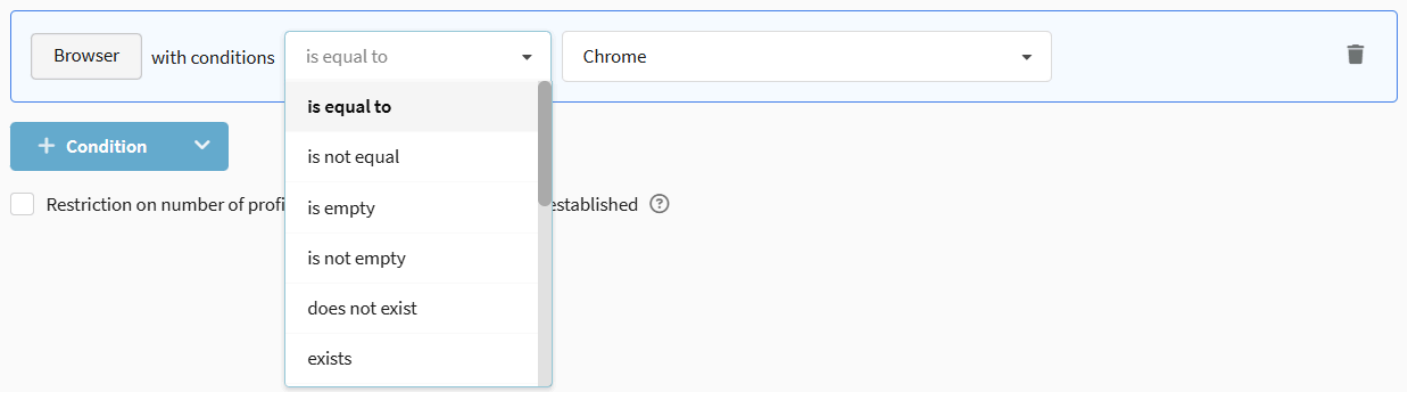
A unique identifier for each customer profile. Options available: Is equal to / Is not equal to, as well as checks for presence/absence of the identifier in lists and external sources (file, SQL, API).



Browser and Operating System

The Browser and Operating System fields are automatically filled when a client opens an email or performs an action on the site (using the User agent header).

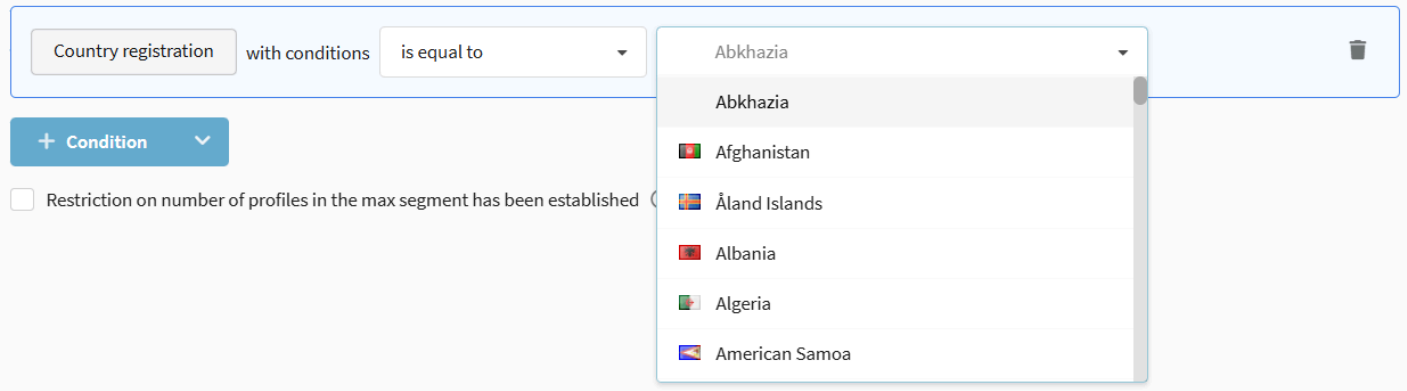
Options available: Is equal to / Is not equal to, Exists / Does not exist, Empty / Not empty, as well as checks in lists and external sources.



Geolocation Data

Location data (Country, Region, City, Zip Code, Time Zone) can be imported manually or determined automatically by IP address at the moment of opening an email or clicking a link.

Use standard string field operators for segmentation by geolocation.



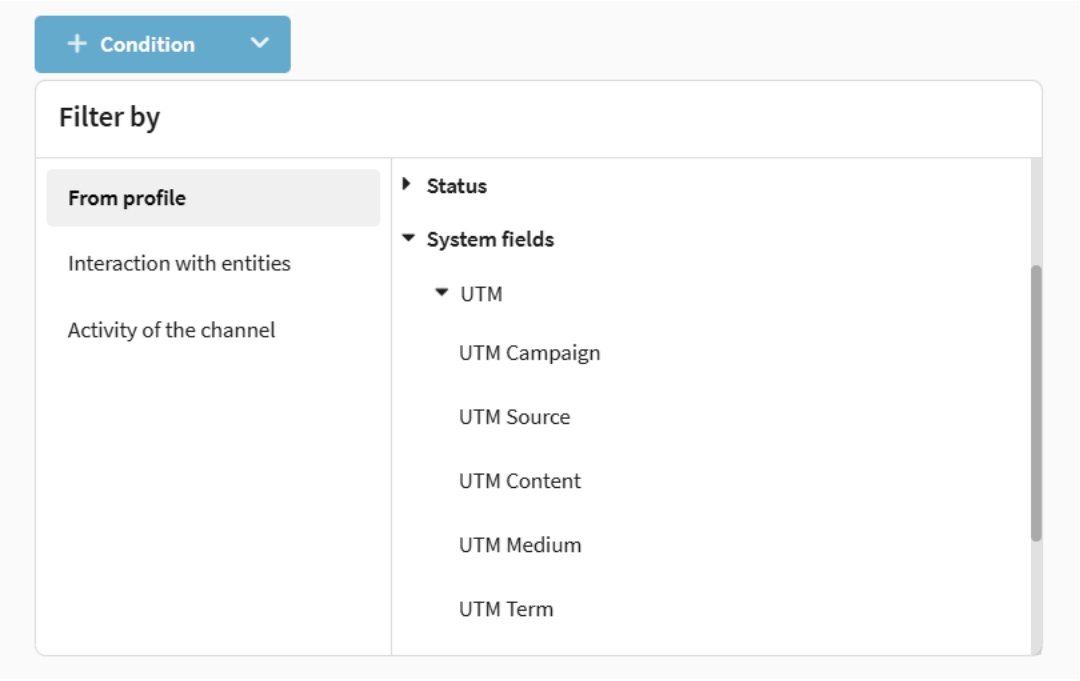
UTM Labels

UTM labels are used to track traffic sources. The following fields are available:

- **UTM mailing — the name of the mailing.**
- **UTM Source — the traffic source (advertising system).**
- **UTM Medium — the type of mailing (contextual, banner, etc.).**
- **UTM Content — additional information for differentiating ads.**
- **UTM Term — the keyword.**

For UTM fields, the following operators are available:

- **Is equal to / Is not equal to**
- **Starts with**
- **Contains / Does not contain**
- **Is empty / Is not empty**
- **Exists / Does not exist**
- **Is in file in column / Is not in file in column**
- **Is in data table / Is not in data table**
- **Is in a field of post url response / Is not in a field of post url response**
- **Is in list / Is not in list**



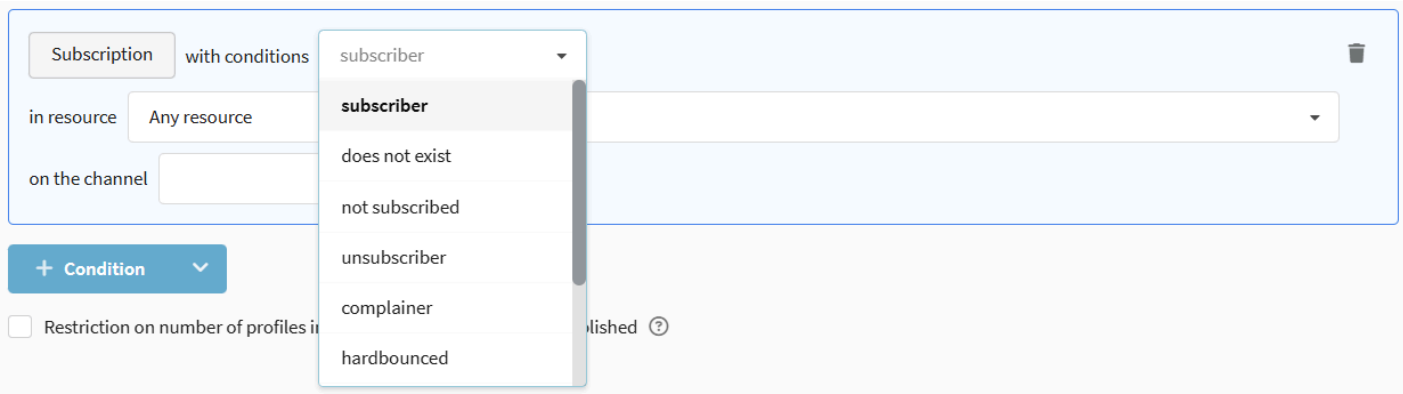
Subscriptions

Segmentation by subscription data allows you to determine the possibility of communication with clients through various communication channels and set up omnichannel mailings.

Subscription

Allows selecting profiles by the presence and status of a subscription to a specific resource and communication channel.

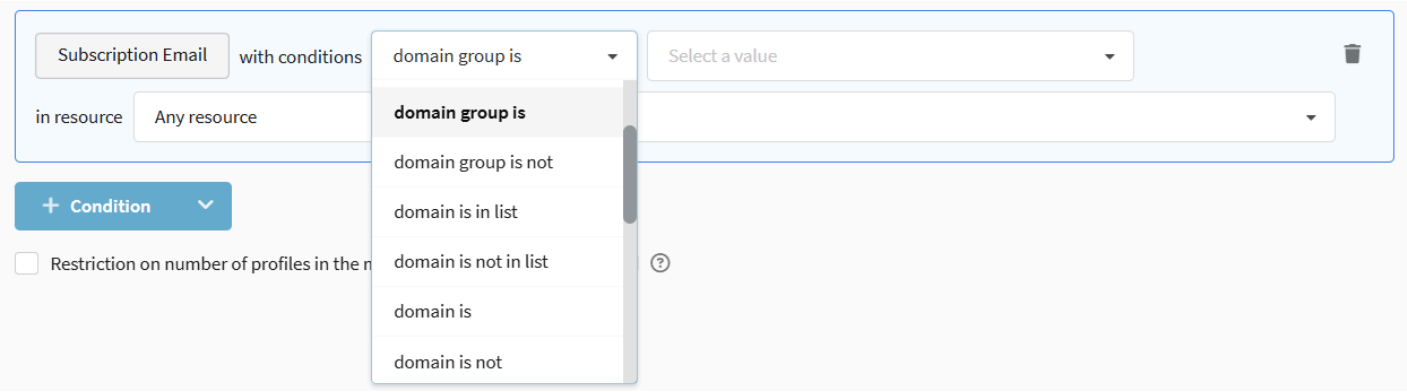
- **Does not exist — the profile does not have a subscription to the selected resource and channel.**
- **Subscribed — the profile has active subscriptions (“Subscribed” status).**
- **Not subscribed — the subscription exists, but its status does not allow sending messages.**
- **Unsubscribed, complainer, hard bounced, unconfirmed, suspended, not valid — selection by one of the specific statuses.**



Email Subscription

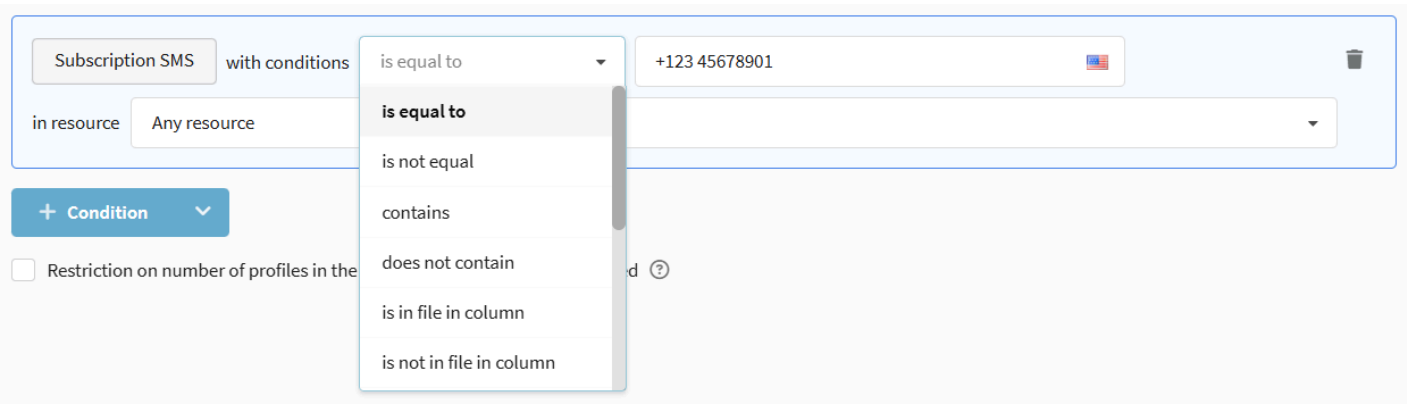
Allows isolating groups of subscribers by email provider or finding specific emails. The set of operators is similar to the main Email field:

- **Is equal to / Is not equal to — exact match of the address.**
- **Contains — search by part of the email address.**
- **Domain group is / Domain group is not — search by domain group (mail.ru, yandex.ru).**
- **Domain is / Domain is not / Domain contains — search by domain or part of it.**
- **Standard operators for searching data in lists and external sources.**



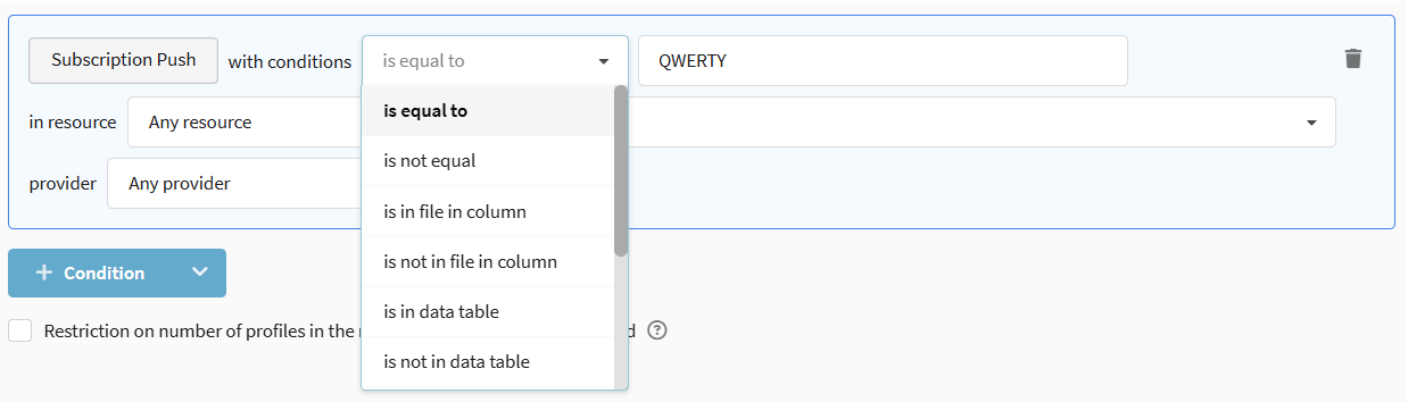
SMS Subscription

Segmentation by data contained in the SMS subscription. Selections by exact or partial match are available: Is equal to / Is not equal to, Contains / Does not contain, as well as checks in lists and external sources.



Push Subscription

Segmentation by data contained in the Push subscription. Selections are available only by full match Is equal to / Is not equal to and operators for searching data in lists and external sources. Additionally, you can specify the push notification service provider.



TIP

Detailed information about field types in the database and permissible values can be found in the separate article.

Segmentation by Interactions with Entities

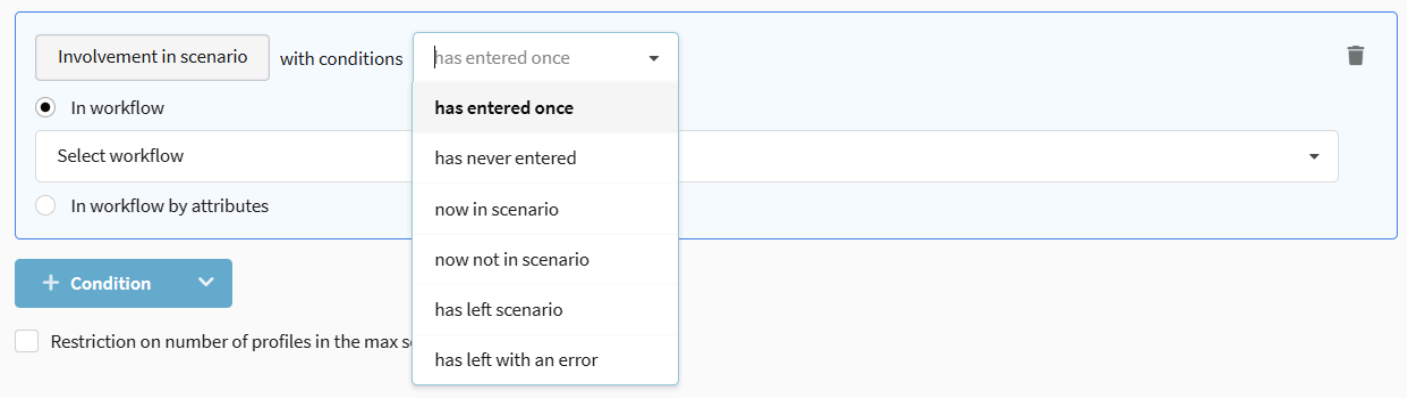
This section contains rules for segmenting profiles based on their interactions with various functional entities of the Altcraft platform, such as automation scenarios, forms, popups, segments, and loyalty programs.

Scenarios

Segmentation by participation in automation scenarios. Allows selecting profiles that are in a scenario, have completed it, or exited with an error.

Available operators:

- **has entered once — the profile has ever participated in the selected scenario.**
- **has never entered — the profile has never participated in the selected scenario.**
- **now in scenario — the profile is currently active in the scenario.**
- **now not in scenario — the profile is currently not active in the scenario.**
- **has left scenario — the profile successfully completed the scenario.**
- **has left with an error — the profile exited the scenario due to an error.**



Segments

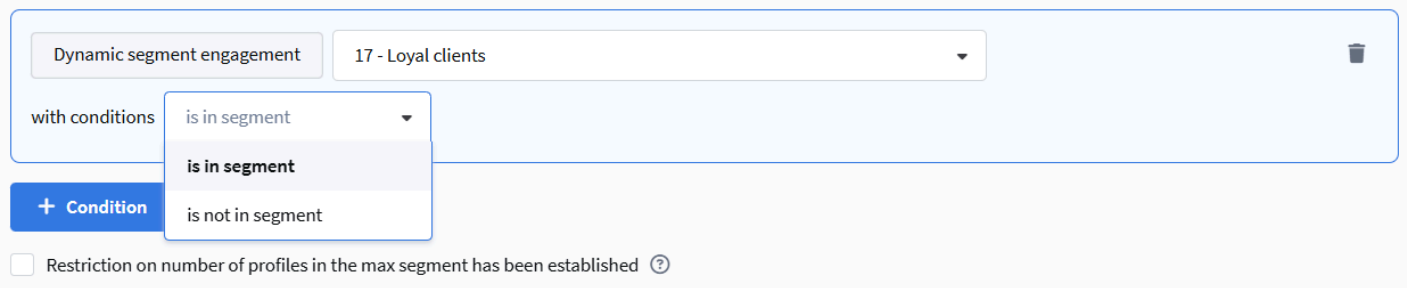
Selection of profiles based on their participation in other segments (dynamic or static). Allows creating complex multi-level selections.

Participation in a Dynamic Segment

The rule checks whether a profile is included in the calculation result of another dynamic segment.

Available operators:

- **Is in segment — the profile is included in the selected dynamic segment.**
- **Is not in segment — the profile is not included in the selected dynamic segment.**



CAUTION

Try to avoid the Is not in segment operator for large selections, as such a rule consumes significant system resources for comparing data arrays.

NOTE

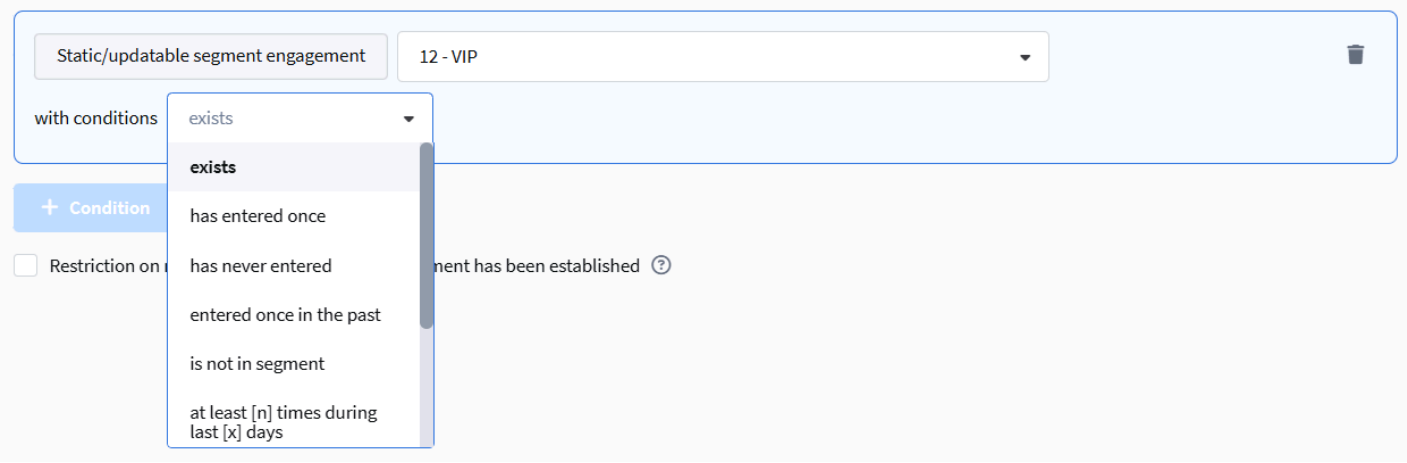
The dynamic segment for selection must be configured for the same database as the current segment.

Participation in a Static Segment

The rule checks whether a profile is in a static segment (control group) or has ever been in it.

Available operators:

- **exists — the profile is currently in the static segment.**
- **has entered once — the profile has ever been in this segment.**
- **has never entered — the profile has never been in this segment.**
- **entered once in the past — the profile is currently not in the segment but was in it previously.**
- **is not in segment — the profile is currently not included in the segment.**
- **entered at least [n] times during [x] days — the profile entered the segment the specified number of times in the last X days.**
- **entered at least [n] times during period — the profile entered the segment the specified number of times in the selected period.**
- **never entered during [x] days — the profile has not entered the segment in the last X days.**
- **never entered during period — the profile has not entered the segment in the selected period.**

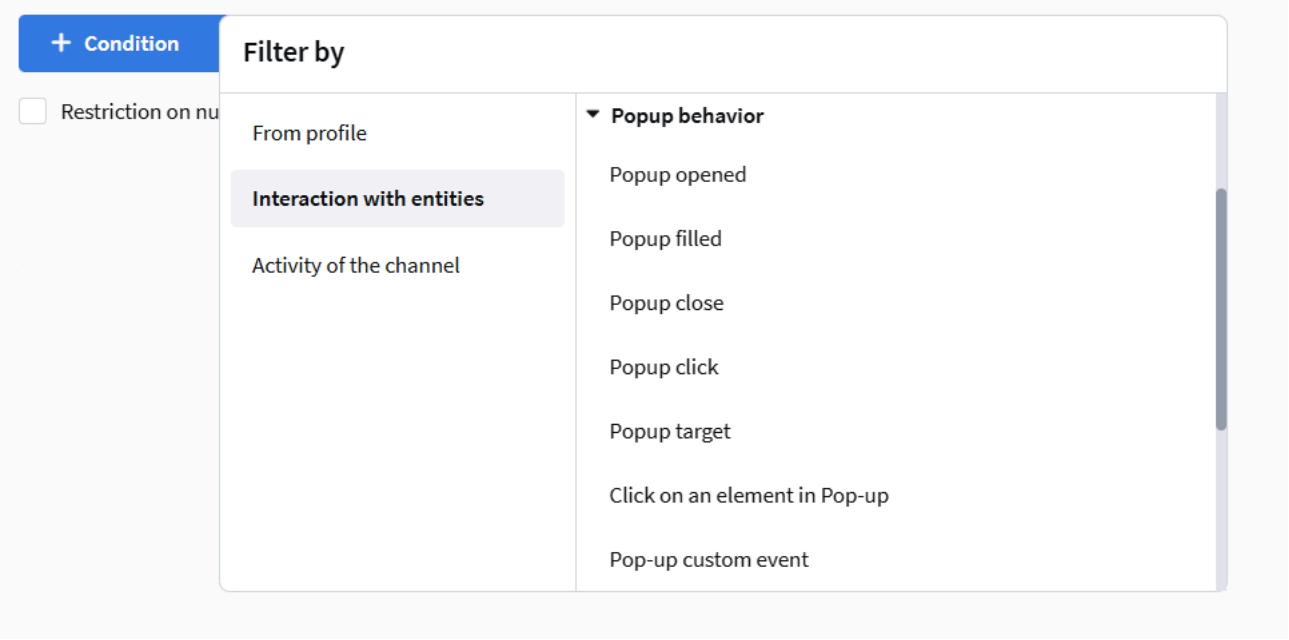


Popup Behavior

Segmentation based on user interaction with popups on your website.

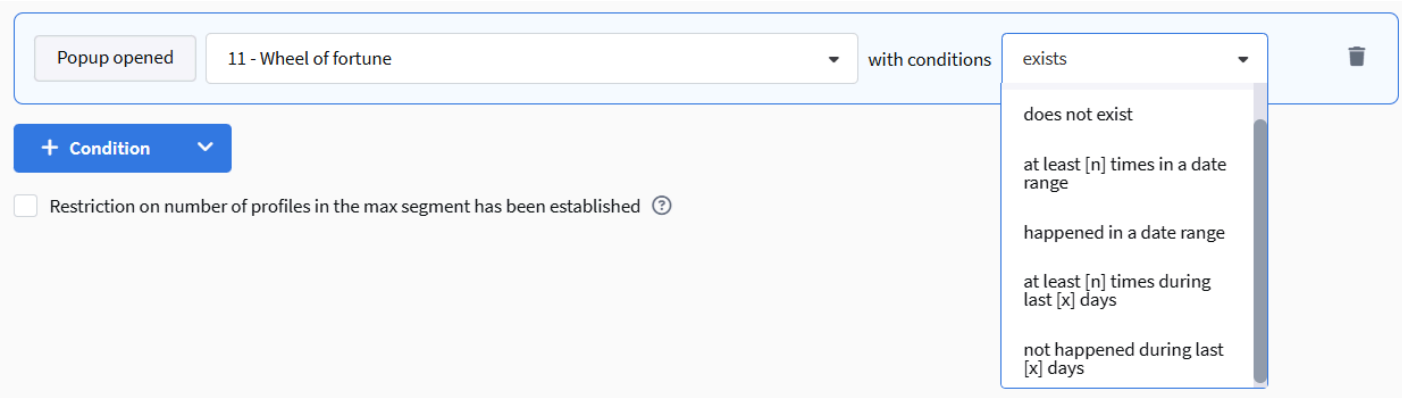
Available events for selection:

- **Pop-up opened — the popup was shown to the user.**
- **Pop-up filling — the form in the popup was successfully filled and submitted.**
- **Pop-up closed — the user closed the popup (click on the "Close" button).**
- **Click on pop-up — click on any element of the popup, except the close button.**
- **Pop-up target — the conditions for showing the popup were met (even if it wasn't shown due to limits).**
- **Click on an element in Pop-up — click on a specific HTML element with a given `name` attribute.**
- **Pop-up custom event — an event configured via JavaScript code.**



Available selection conditions:

- **Exists — the event was registered at least once.**
- **Does not exist — the event was never registered.**
- **At least [n] times in a date range — the event occurred at least N times during the specified period.**
- **Not happened in a date range — the event did not occur during the specified period.**
- **At least [n] times during [x] days — the event occurred at least N times in the last X days.**
- **Not happened during last [x] days — the event did not occur in the last X days.**



Loyalty

Promo code

Segmentation by participation in loyalty programs using promo codes.

Available operators:

- **Assigned / not assigned — selection of subscribers to whom a promo code was assigned by a loyalty program (manually or automatically).**
- **Activated / not activated — selection based on the fact of promo code activation (for example, when making a purchase).**

Loyalty Programs

Profile segmentation can use data based on actions in loyalty programs.

Available selection conditions:

- **Participation in a loyalty program — whether the profile is a participant in the selected loyalty program.**
- **Points balance on members' accounts — the number of points in the loyalty program.**
- **Points accrued in promotions.**
- **Points expiration in promotions.**

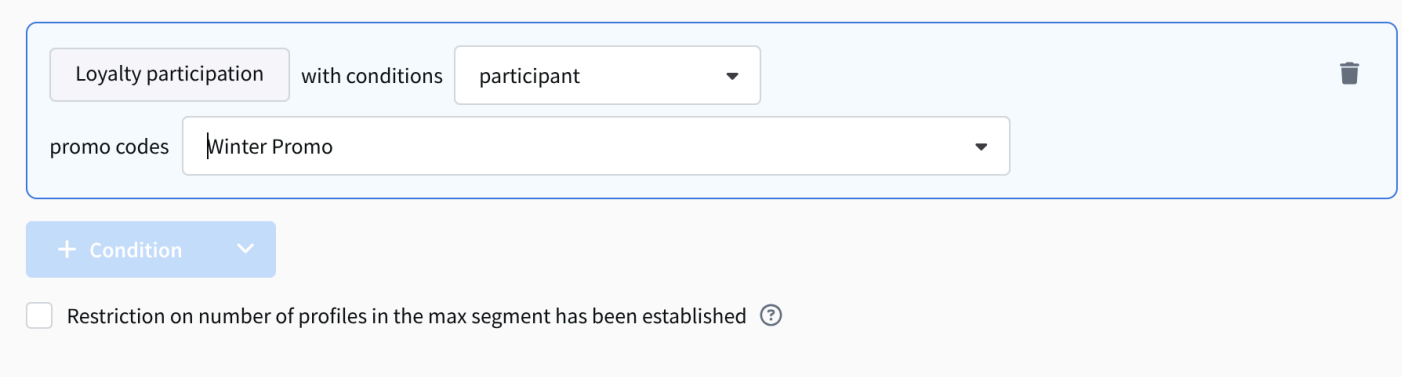
Available operators:

- **Member / Not a member.**
- **Points balance equal / not equal / is within points range.**
- **points amount from n to m within a period / points amount from n to m in the next x days.**
- **in the next x days / points amount from n to m in the next x days.**

To configure these conditions, you may need to select a loyalty program, a promotion, a balance type (active/final), and a point type.

TIP: Final and active balance

Accrued points can have an activation time. If it is not specified, the points become active immediately. If an activation time is set, the points cannot be used until that moment. The active balance is the number of points that can be used right now. The final balance is the total points balance, including inactive points.



Relations

Segmentation by the presence of relations between subscriber profiles.

Available operators:

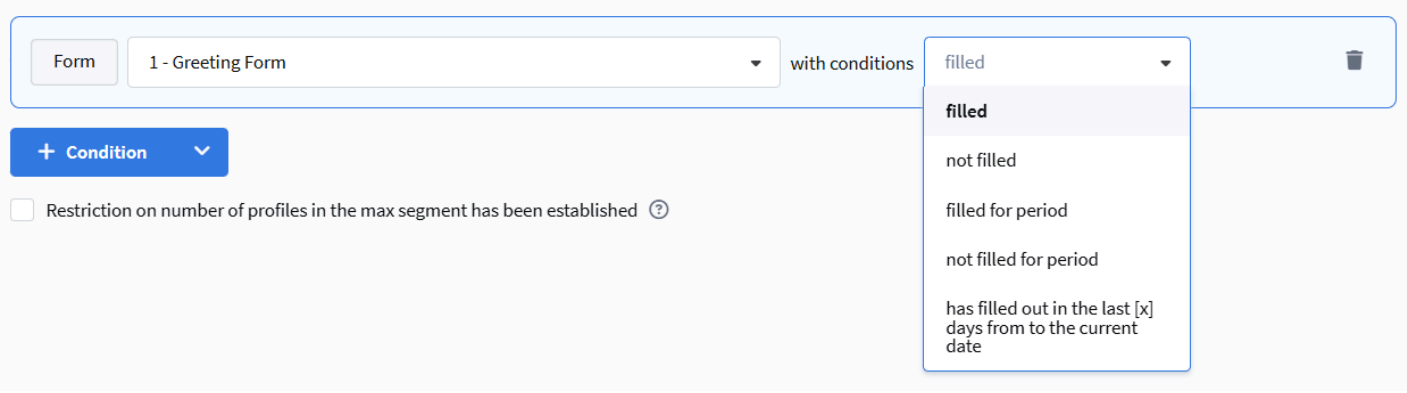
- **Is / Is not in direct relation — selection of subscribers who have or do not have a direct relation with another profile.**
- **Is / Is not in reverse relation — selection of subscribers who have or do not have a reverse relation from another profile.**

Forms

Segmentation based on interaction with forms placed on the website.

Available selection conditions:

- **filled out — the subscriber has ever filled out the specified form.**
- **did not fill out — the subscriber has never filled out the specified form.**
- **filled out for the selected period — the subscriber filled out the form during the specified dates.**
- **did not fill out for the selected period — the subscriber did not fill out the form during the specified dates.**
- **filled out in the last [x] days from the current date — dynamic selection for the last X days.**



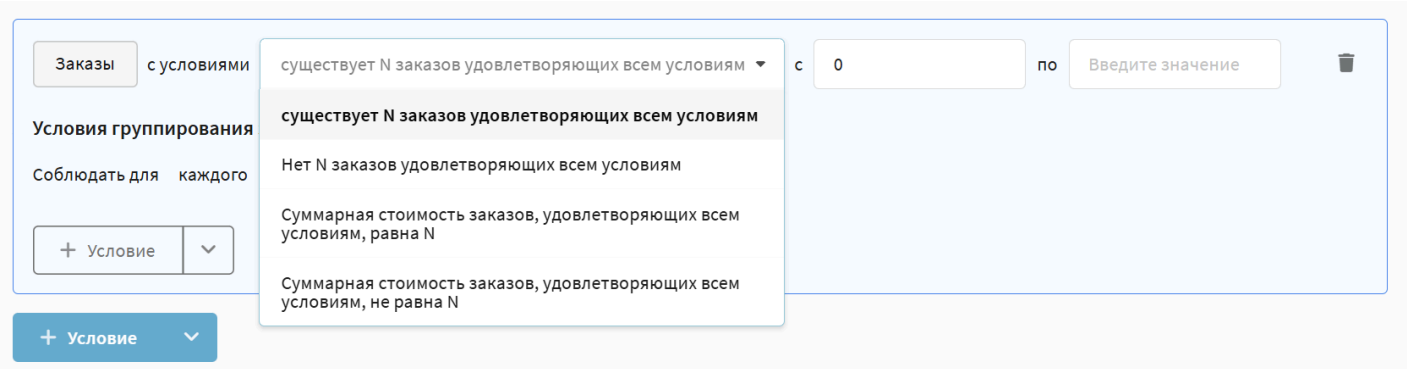
Orders

Segmentation based on orders registered in the Market module. Allows building selections by quantity, amount, average check, order composition, as well as by geography, sales channels, and statuses.

Basic Conditions

Rules of this type allow you to select profiles that have (or do not have) orders meeting the specified parameters. The following operators are available:

- **here are N orders that meet all the conditions — selection of profiles with the specified number of orders that fall under additional conditions.**
- **No N orders that meet all the conditions — selection of profiles that do not have the specified number of matching orders.**
- **Total cost of orders that meet all the conditions equals N — selection by total order amount taking into account additional conditions.**
- **Total cost of orders that meet all the conditions does not equal N — exclusion by total amount.**
- **Total cost of orders that meet all the conditions in range — selection of profiles whose total order amount falls within the specified numeric range.**
- **Total cost of orders that meet all the conditions is not in range — exclusion by total amount range.**



Purchase Analytics Conditions

For deeper analysis of purchasing behavior, rules based on average values are available. These conditions take into account all profile orders (or only those that meet additional parameters) over a certain period of time.

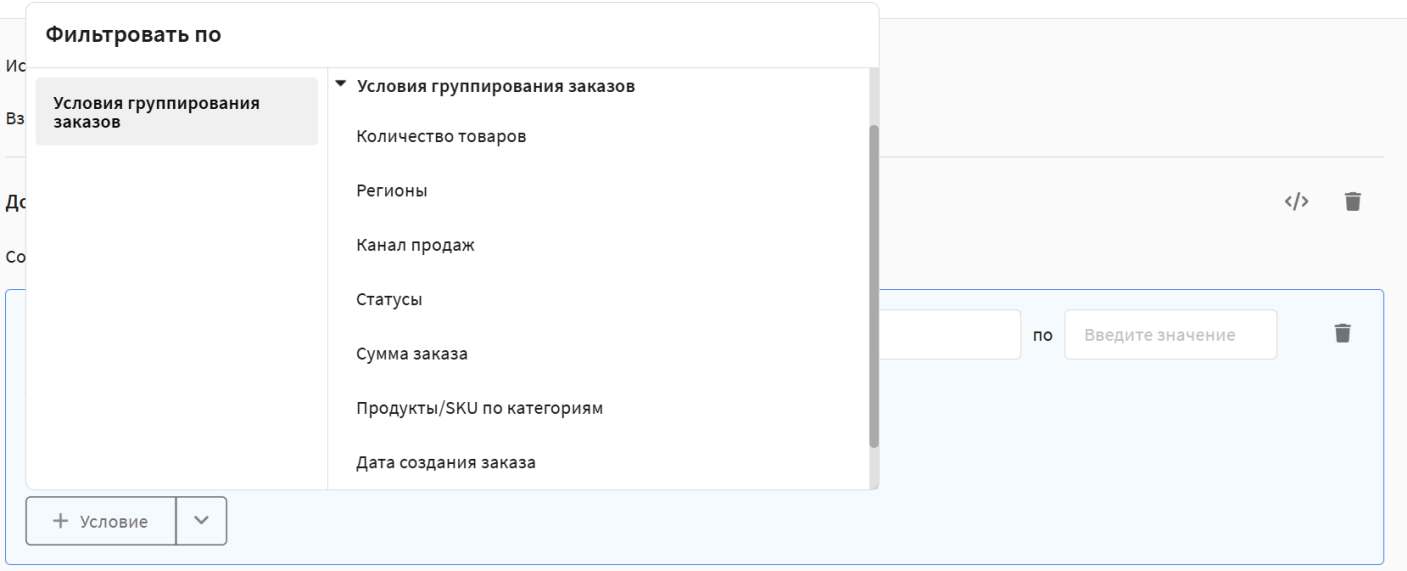
Available operators include checks for a period (specific dates) and for the last X days (relative to the current date). For each of these time intervals, you can set an “in range” or “not in range” condition:

- **Average check — the average cost of one order (sum of all orders / number of orders)**
- **Average check for period in range / not in range**
- **Average check for last X days in range / not in range**
- **Average sum of purchases — the average cost of one item in orders (sum of all orders / total number of items in all orders)**
- **Average check from segment A for period in range / not in range — the average cost of an order containing products from the specified product segment (sum of orders containing products from segment A / number of such orders)**
- **Average check for orders from segment A for last X days in range / not in range**

- **Average sum of purchases for orders from segment A for period in range / not in range — the average cost of one product item from the specified segment (sum of cost of all items of products from segment A / number of these items)**
- **Average sum of purchases for orders from segment A for last X days in range / not in range**

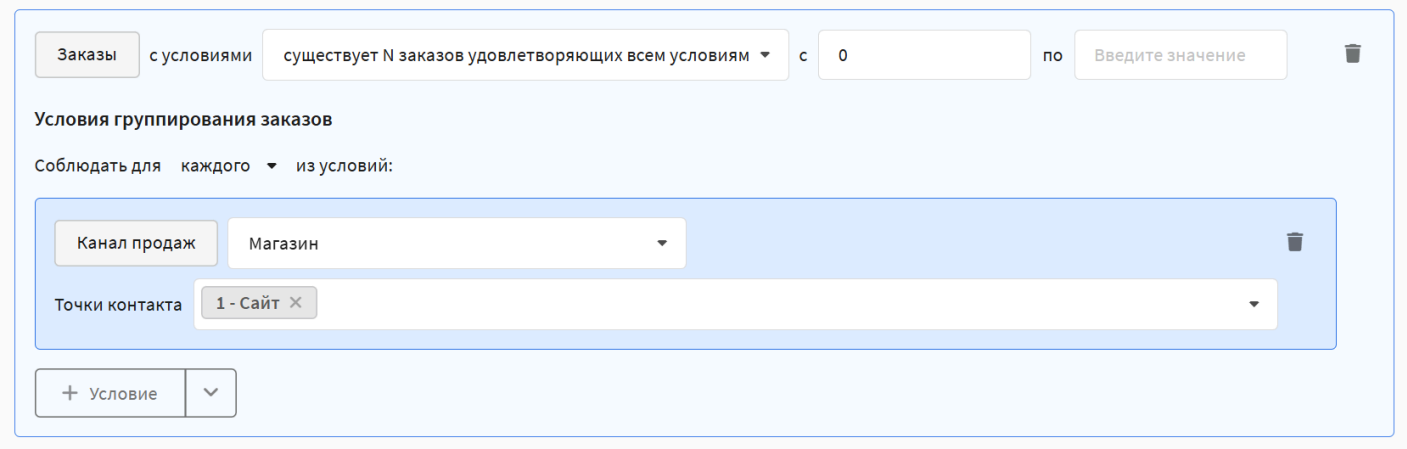
Order Filtering Terms

In any of the rules described above, you can set additional conditions to specify which orders to take into account. The following parameters are available:



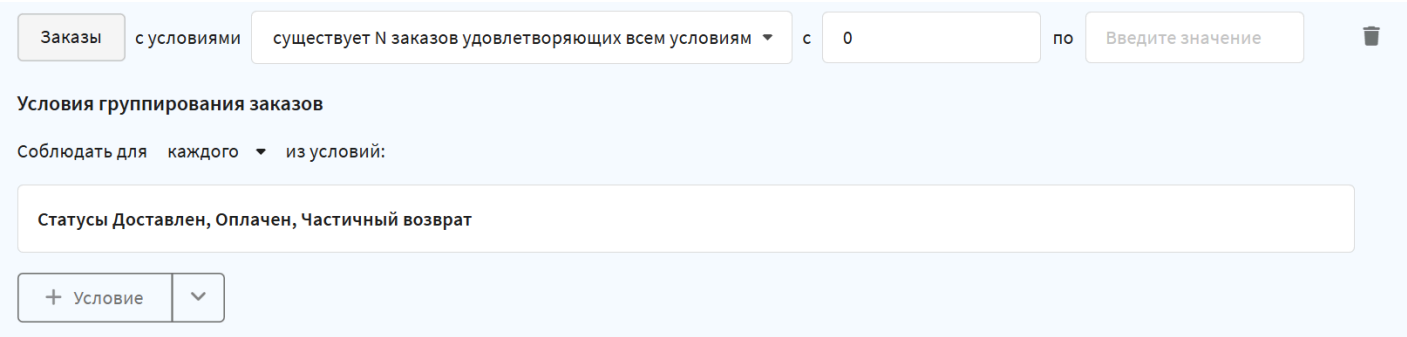
Sales Channel

Selects orders made in one [sales channel](). Multiple contact points can be specified.



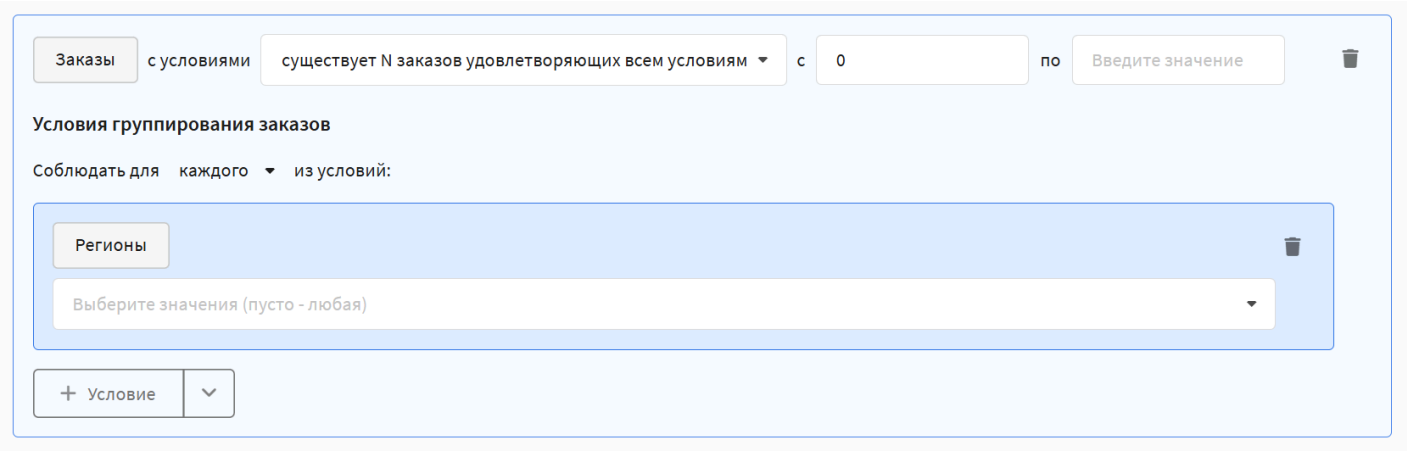
Statuses

Selects orders with the specified statuses: *new, delivered, paid, canceled, partial return*.



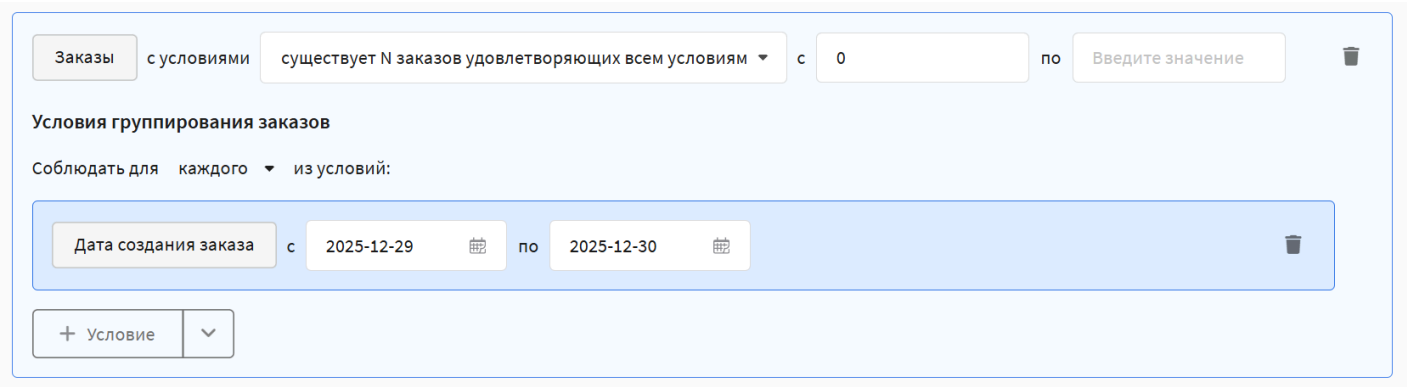
Regions

Selects orders made in specific [regions]().



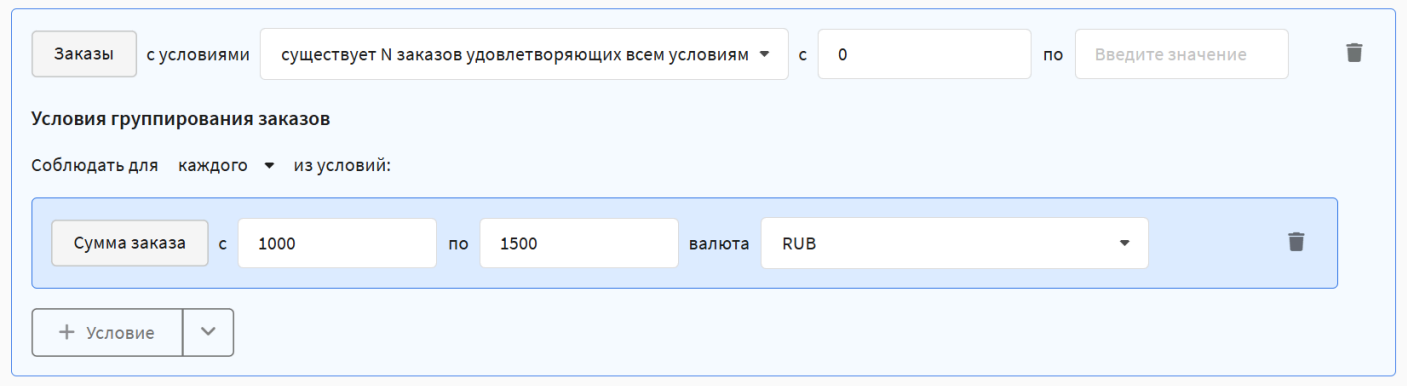
Date Created

Selects orders registered during the specified time period.



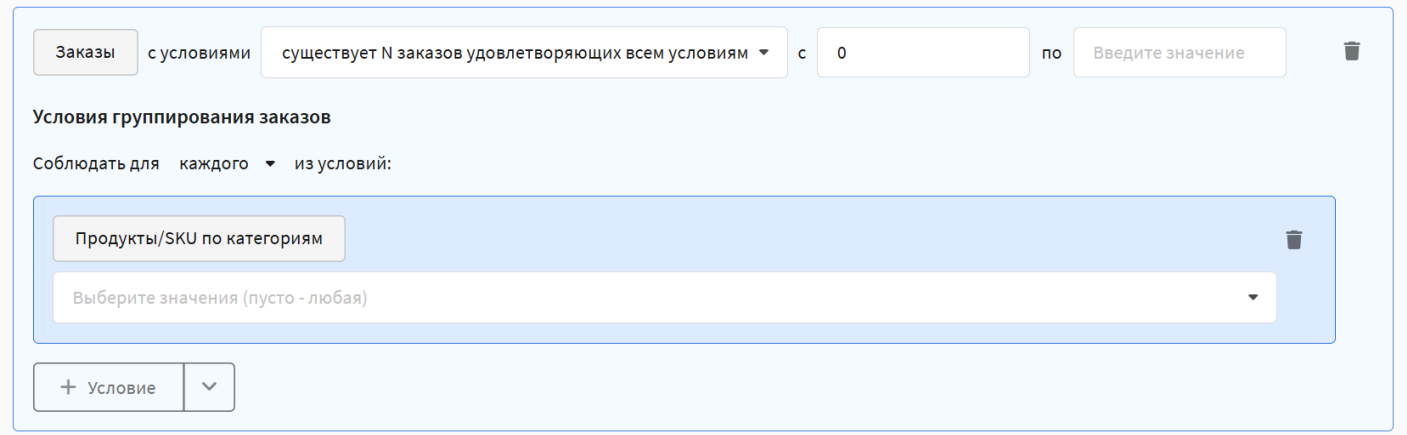
By Order Amount

Selects orders with a specific cost. Calculation currency can be specified.



Products/SKUs by Category

Selects orders containing products from specific categories.



Products by Name

Selects orders containing specific products — by name or SKU.

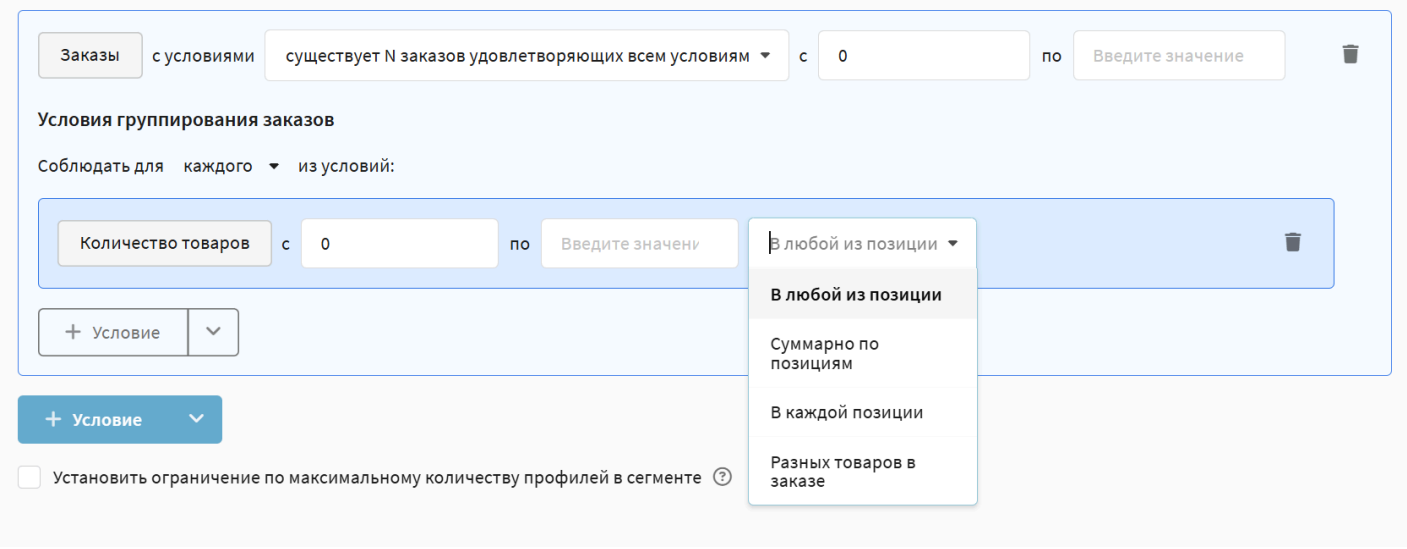
Product Segments

Selects orders containing products included in the specified product segment.

By Quantity of Goods

Selects orders with a specific quantity of products. Available calculation methods:

- **in any of the items — any order item has the specified quantity.**
- **summed across items — all order items together have the specified quantity.**
- **in each of the items — each order item has the specified quantity.**
- **different products in order — number of items in the order (1 product = 1 item).**



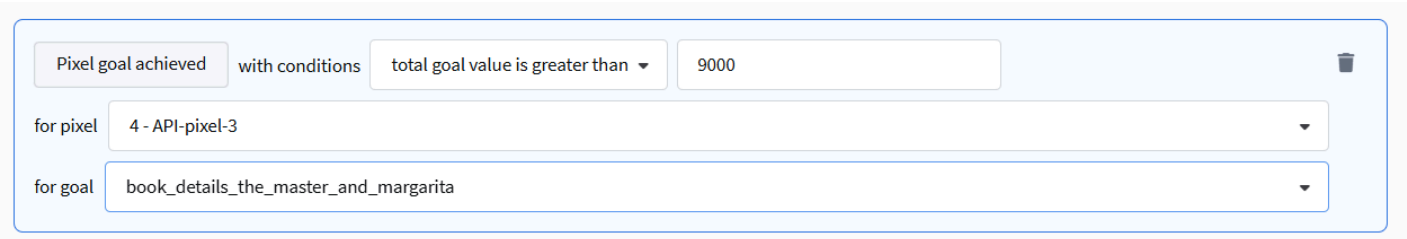
Goal/Pixel Behavior

Goal Achieved

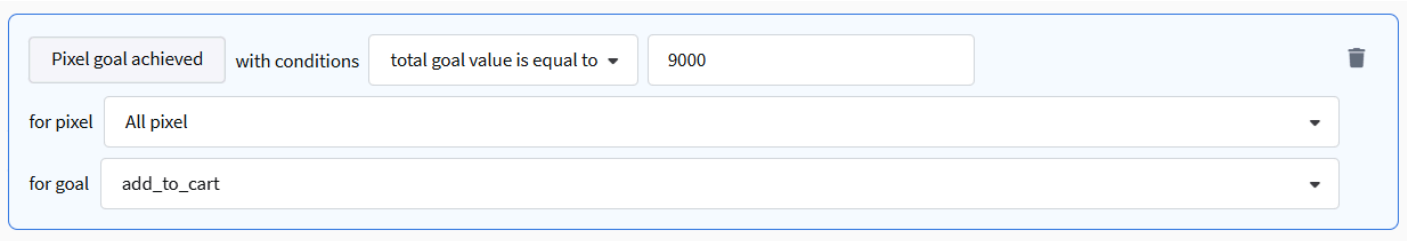
Segmentation based on subscriber actions on external resources (websites, apps) tracked using Altcraft pixels.

Available operators:

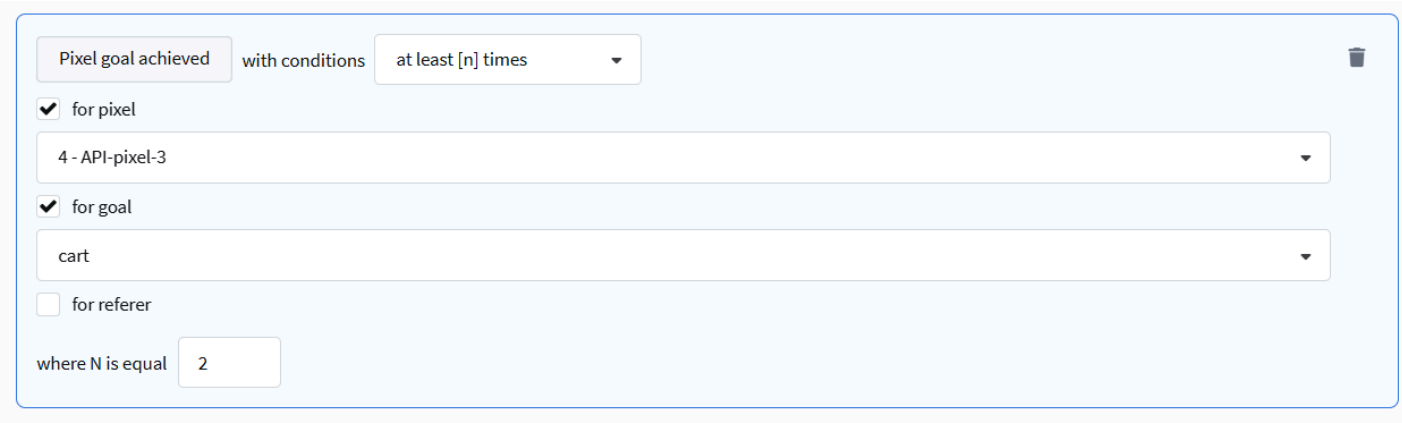
- **Exists / Does not exist — checks for the presence or absence of goal achievements.**
- **Total goal value — calculates the sum of the cost of achieved goals. Operators: greater than / less than / equal to. Can be specified for all time, for a period, or for the last [x] days.**



- **Average goal value — calculates the arithmetic mean of the cost of achieved goals. Operators: greater than / less than / equal to. Can be specified for all time, for a period, or for the last [x] days.**



- **At least [n] times / Not happened — selection based on the number and frequency of actions performed. Can specify goal, pixel, or referer (traffic source).**



- **Meets / Does not meet complex condition — extended selection based on additional data passed when registering a goal via API.**

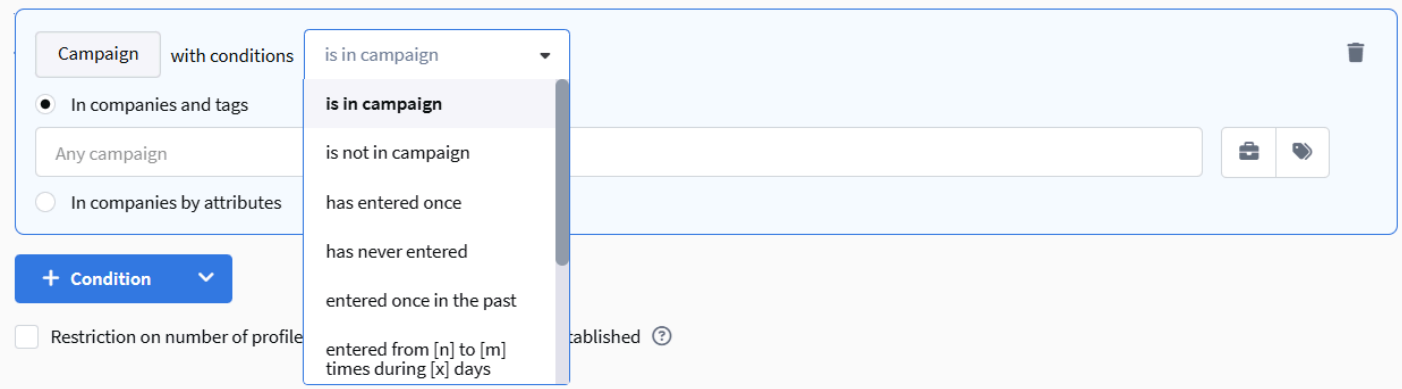
Rules can be configured for selection by:

- **Goal value (cost)**
- **Data from the `pixel_data` object (string, numeric, date, boolean)**
- **Profile device and geo-data (browser, OS, country, city, etc.)**
- **Sent messages and promo codes**
- **UTM labels**
- **URL Referer and Origin**
- **Profile order data**

Campaigns

Segmentation by participation in a campaign.

- **currently in campaign — the profile is currently in the campaign.**
- **not in campaign — the profile is currently not in the campaign.**
- **entered at least once — the profile has ever been in the campaign.**
- **never entered — the profile has never been in the campaign.**
- **currently not, but entered at least once — the profile is currently not in the campaign but was previously.**
- **entered at least [n] times in the last [x] days — the profile entered the campaign the specified number of times in the last X days.**
- **entered at least [n] times in period — the profile entered the campaign the specified number of times in the selected period.**
- **never entered in the last [x] days — the profile has not entered the campaign in the last X days.**
- **never entered in period — the profile has not entered the campaign in the selected period.**

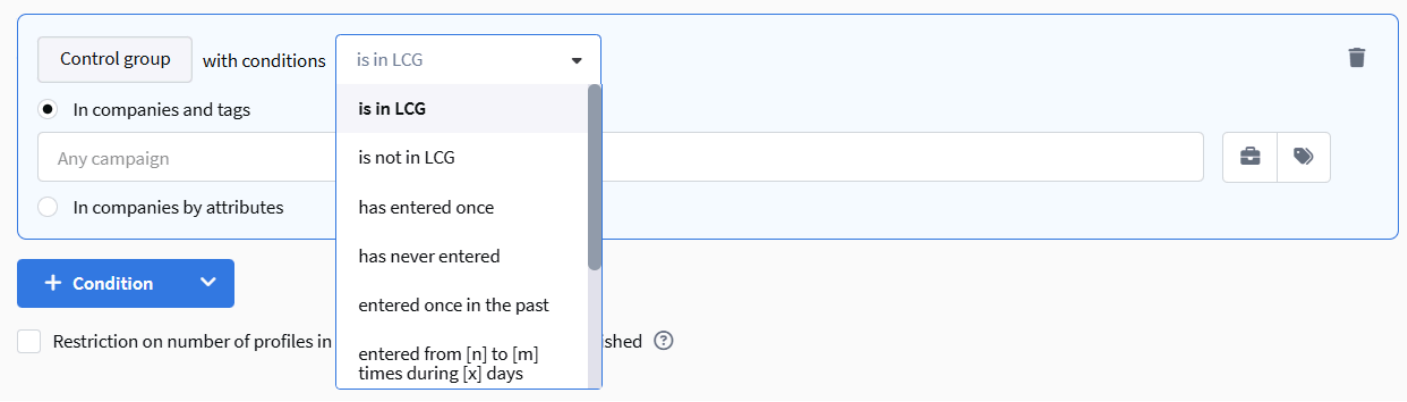


Control Group / Test Group

Segmentation by participation in test or local control groups.

Available operators:

- **is in GCG — the profile is currently in the global control group.**
- **is not in GCG — the profile is currently not in the global control group.**
- **has entered once — the profile has ever been in the control group.**
- **has never entered — the profile has never been in the control group.**
- **entered once in the past — the profile is currently not in the control group but was previously.**
- **entered at least [n] times during [x] days — the profile entered the control group the specified number of times in the last X days.**
- **entered at least [n] times during period — the profile entered the control group the specified number of times in the selected period.**
- **never entered during [x] days — the profile has not entered the control group in the last X days.**
- **never entered during period — the profile has not entered the control group in the selected period.**



Segmentation by Activity of the channel

The history of subscriber actions within sent messages is formed based on events in various communication channels. Selections based on this activity allow segmentation by engagement (e.g., by opens and clicks) and effective setup of retargeting mailings.

For most rules of segmentation by message activity history, the following types of selection operators are available:

- **Exists/Does not exist — selection based on the presence of at least one action of the selected type for subscribers.**
- **Not happened — selection based on the absence of at least one action of the selected type. Depending on the operator, you can additionally specify the campaign, message template, time period, or number of days from the current date.**
- **At least [n] times — selection based on the presence of the specified number of events for subscribers. Depending on the operator, you can additionally specify the campaign, message template, time period, or number of days from the current date.**

TIP

In all rules except Exists, you can select message templates or campaigns — within which the action was made. Then you can select several templates or campaigns in one rule. Action in any of these objects will count — they are combined by the OR operator.

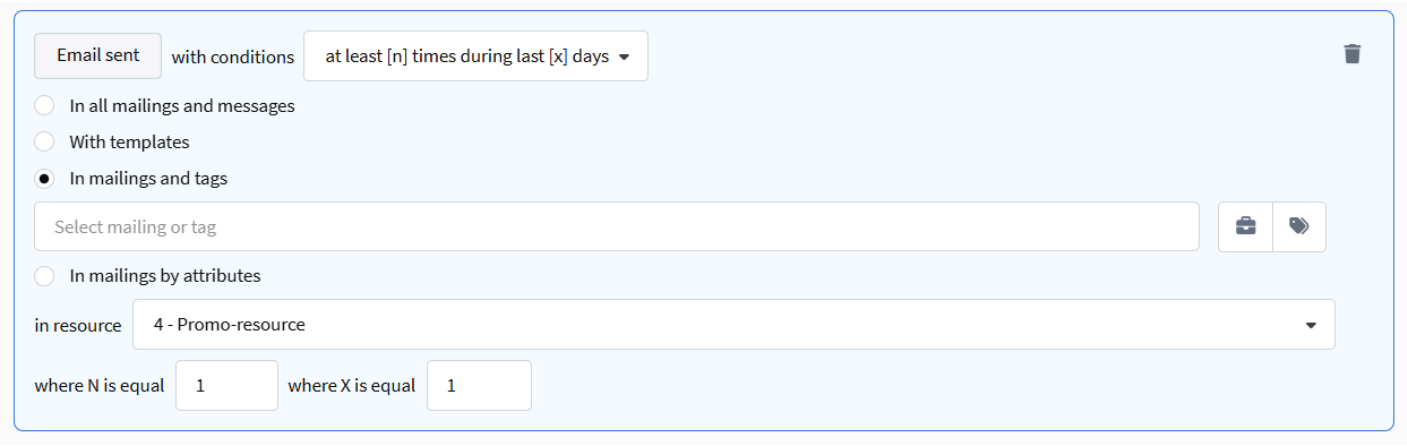
Common Events for All Channels

For most communication channels, similar event types are available. Below are common event groups with indications of the channels where they apply.

Message Sent

The event is registered when a message is formed and redirected to the sending module (mail server, SMS gateway, push or messenger platform).

- **Available in: Email, SMS, Push, Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, Viber, VK Notify.**



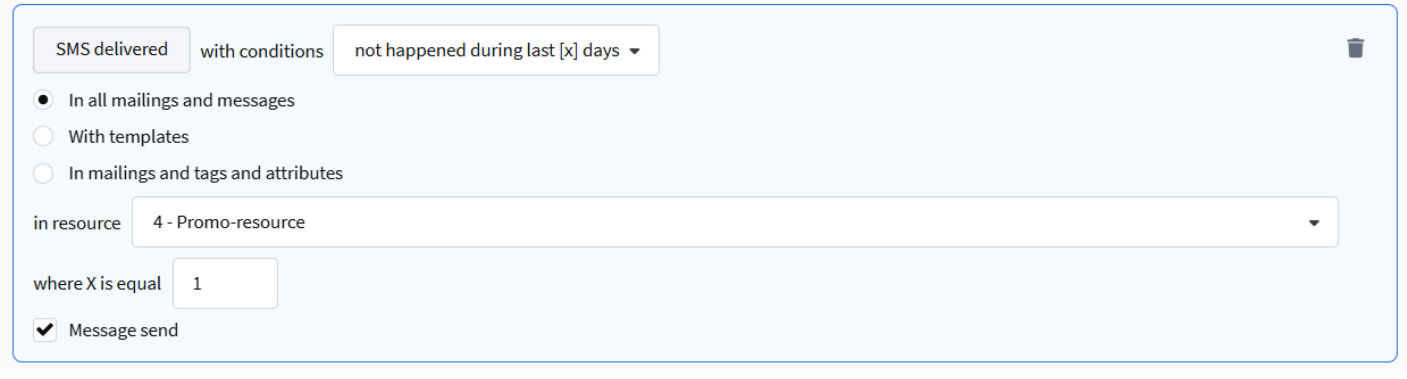
Message Delivered

The event is registered when the platform receives confirmation of message acceptance for delivery from the provider, gateway, or carrier.

- **Available in: Email, SMS, Push, Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, Viber, VK Notify.**

TIP

Allows determining which subscribers' messages were not delivered due to temporary channel unavailability.



Message Opened / Read

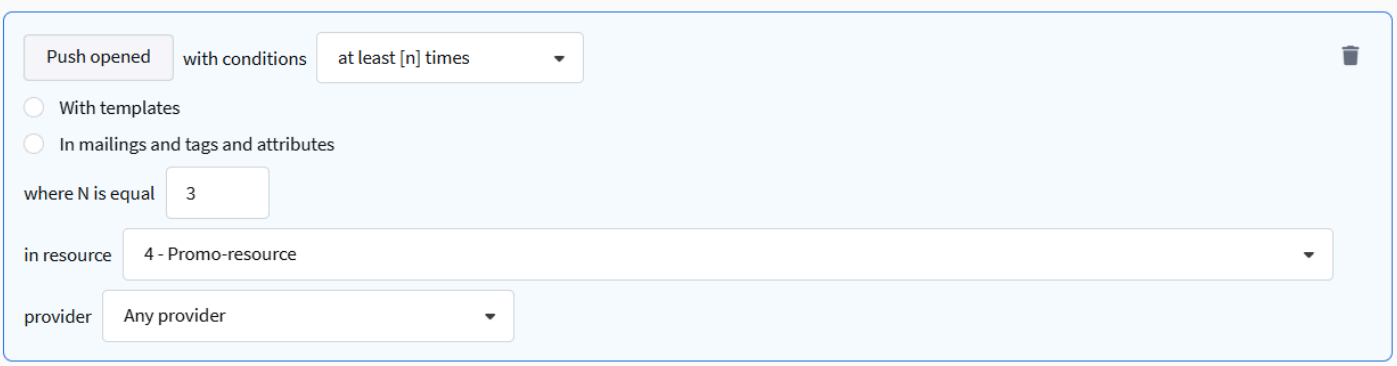
The event is registered when a user opens a message.

- **Opened: Email, SMS, Push.**
- **Read: Email, WhatsApp, Viber, VK Notify.**

For Email, opening is tracked using a pixel in the email body.

TIP

Allows obtaining the exact number of unique opens and calculating the Open Rate (OR) for a specific recipient group.



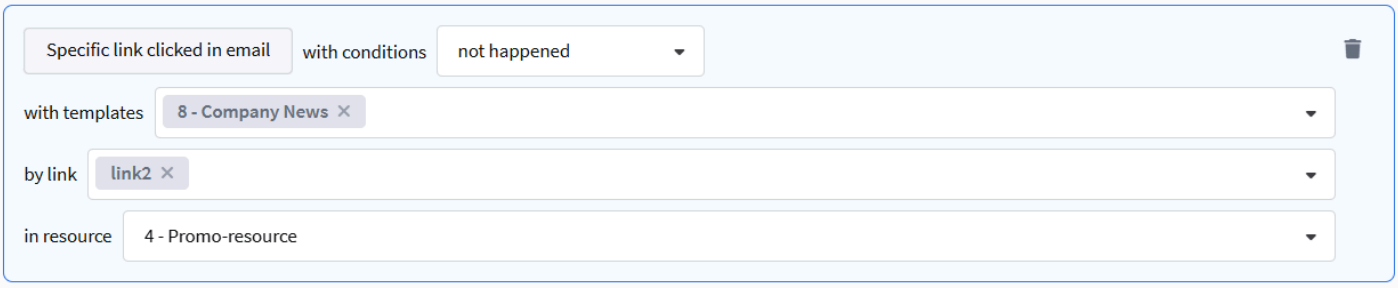
Clicks in Messages

Various events related to clicks on links within messages.

- **Any click in message: Email, SMS, Push, VK Notify. Selects recipients who clicked on any link in the message.**
- **Specific link click in message: Email, SMS, Push, Viber, WhatsApp, Telegram Bot, MAX Bot, VK Notify. Allows specifying specific templates and links for selection of clicks.**
- **Unsubscribe click in message: Email, SMS, Push, VK Notify. Selects recipients who unsubscribed via a special unsubscribe link.**

TIP

A click on an unsubscribe link has a separate event type and is not included in the selection by the rule “Specific link click in message”.



Unique Clicks in Messengers (Telegram, MAX, WhatsApp, Viber, VK Notify)

For messengers, detailed click statistics are available:

- **Click — each link click is recorded.**
- **Unique click — the event is registered if the recipient clicked on at least any link in the message (1 unique click per message).**
- **Unique link click — unique click on a specific link (one event is recorded for each link, even with multiple clicks).**

Non-delivery and Sending Errors

Events related to problems during sending or delivery.

- **Non-delivery: Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, Viber, VK Notify.**
- **Sending policy violation: Telegram Bot, Telegram Group, MAX Bot, MAX Group, WhatsApp, VK Notify.**
- **Hard bounced (permanent delivery errors): Telegram Bot, MAX Bot.**

Available operators for these events: Exists/Does not exist, Not happened, At least [n] times.

Subscription Management in the Channel

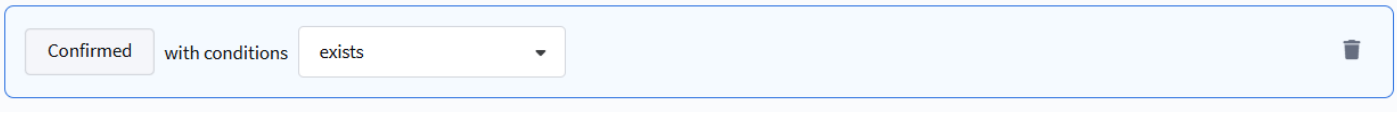
Events related to subscription and unsubscription from communication in a specific channel.

- **Subscribed to channel: Telegram Bot, MAX Bot, WhatsApp, Viber, VK Notify.**
- **Unsubscribed from channel: Telegram Bot, MAX Bot, WhatsApp, Viber, VK Notify.**
- **Subscription confirmation: Telegram Bot, MAX Bot, Email (Click on Confirm link).**

Channel-Specific Events

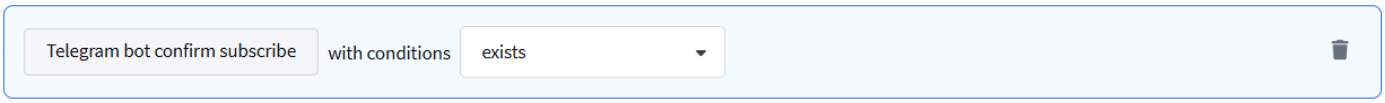
Email

- **Email Read — the `{read}` pixel triggers after 12 seconds of the email being open.**
- **Click on Confirm link — subscription confirmation via Double opt-in.**



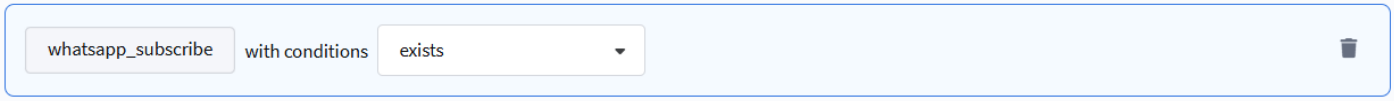
Telegram Bot

- **Telegram bot subscription confirmed — confirmation of subscription to the bot.**



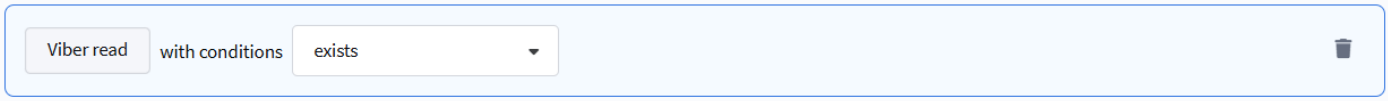
WhatsApp

- **WhatsApp read — message read event.**



Viber

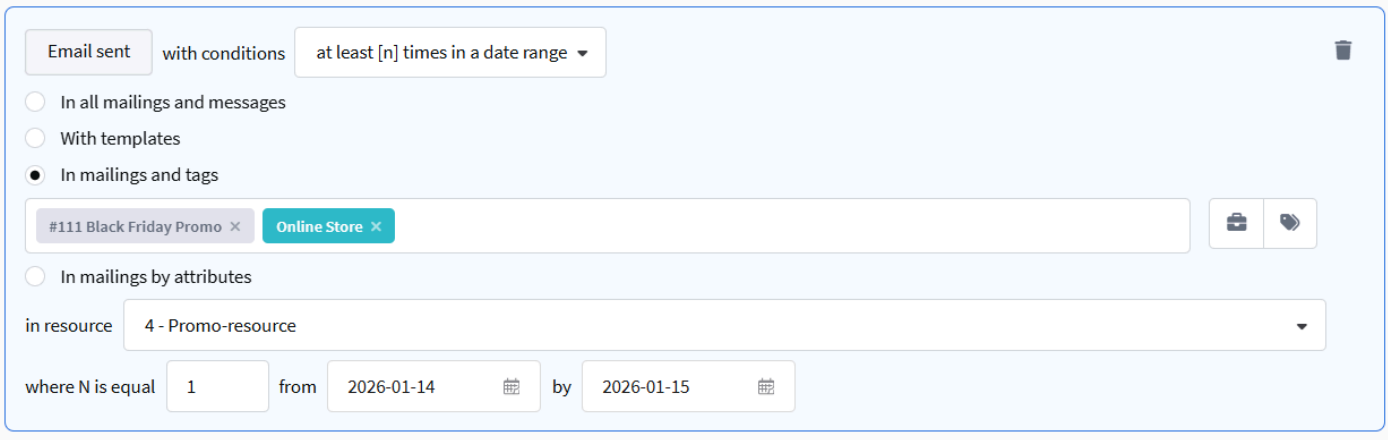
- **Viber read — message read event.**



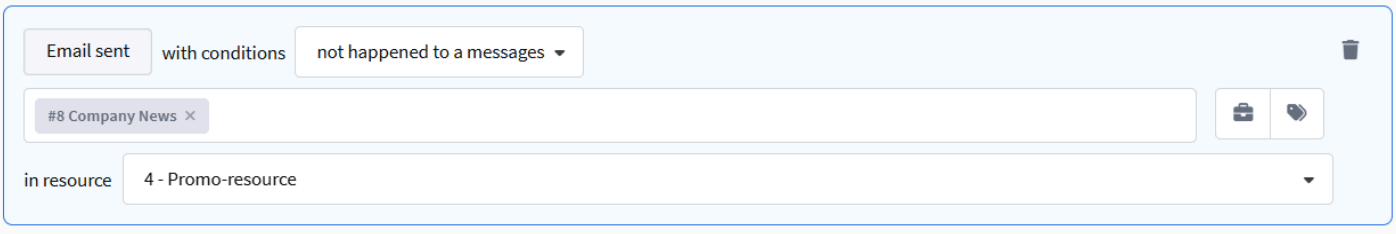
Checking an Event in a Message or Campaign

For precise checking of whether an event occurred in a specific campaign or message template, use conditions with object specification:

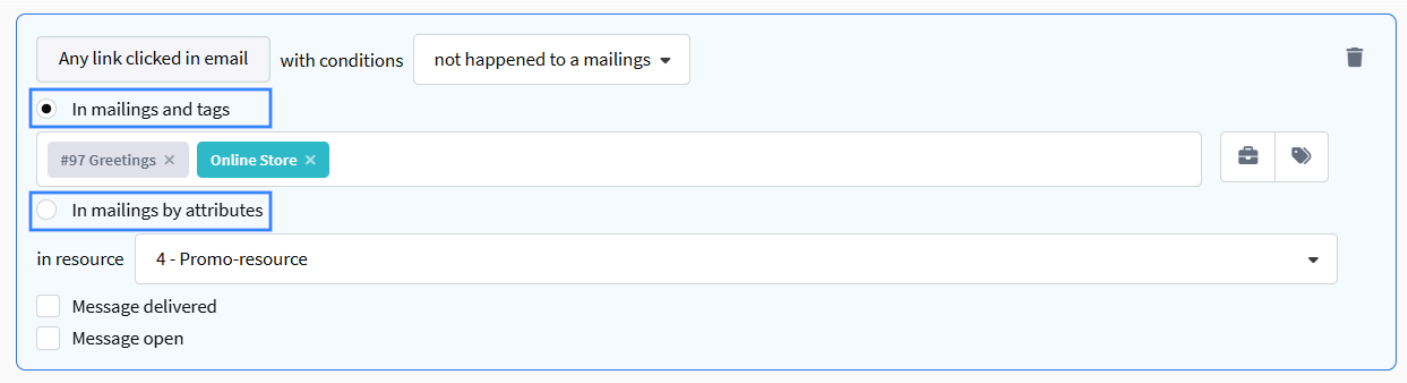
- **At least [n] times in messages / in campaigns — the profile will be included in the segment if the action occurred in any of the selected templates or campaigns.**



- **Not happened to messages / campaigns — the profile will be included in the segment if the action did not occur in each of the selected objects.**



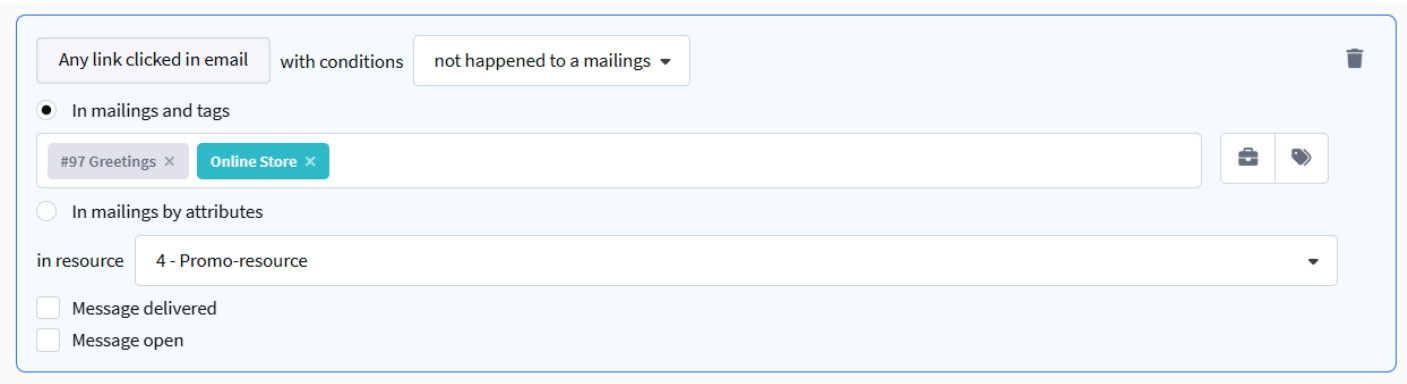
For reverse conditions, you can add a check for a preceding event (e.g., for a click — a check for delivery or opening).



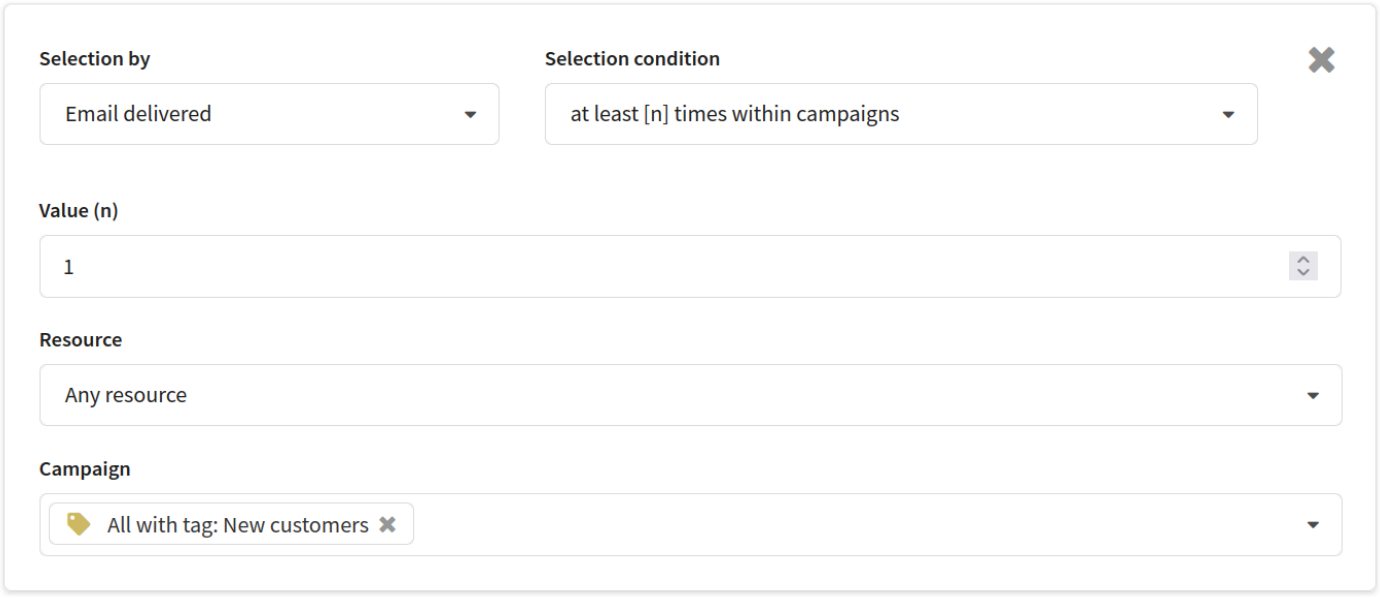
Selecting Campaigns, Messages, or Templates by Tags or Attributes

When segmenting by events in a campaign or message, you can also filter by tags and by attributes.

Selection of objects for checking is available not only individually but also by tags or by attributes.



In the example below, profiles that clicked in any campaign tagged “Online store” will be selected:



**** Meta, the organization that owns Instagram, Facebook, and WhatsApp, is recognized as extremist and banned in the Russian Federation.***

**** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

Segmentation by External data

Description

Segmentation rules can appeal to external sources for profiles selection data. This information can be stored it:

- **Web service API**
- **Uploaded file**
- **Data input form (in the rule)**

This allows you to make automatic segmentation by mane fields, and reduce similar conditions in the segment.

HTTP queries to external service API

If you select Is (not) in a field of post url responce operator in the rule, you can perform selection depending on answer from the service. Platform can take data in JSON, XML, and PIPE formats.

JSON segmentation rule example

Let's take situation, when information is given in JSON format. MIME header will be given, like:

`"Content-Type": "application/json"`

and data in this format:

```
{
  "segment": {
    "emails": [
      "john@example.org",
      "suzy@example.com",
      "rick@example.net",
      "andy@example.edu"
    ]
  }
}
```

It is necessary to set API endpoint URL, and path to necessary field in JSON, in the segmentation rule:

Segmentation rule example for XML

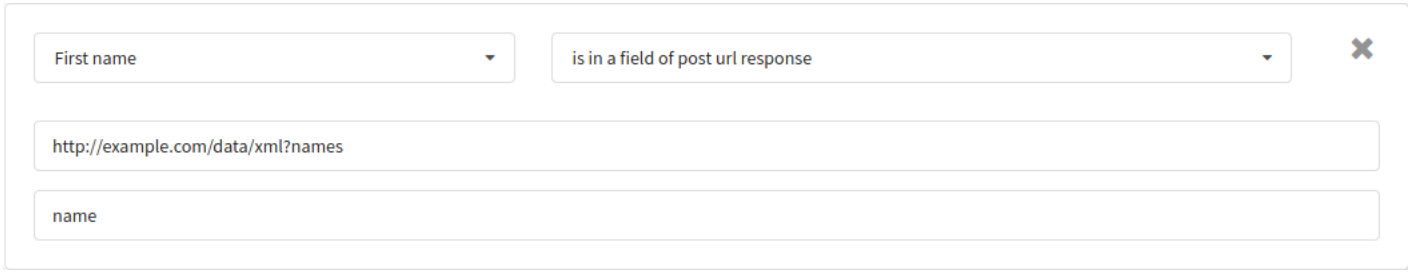
Let's take situation, when information is given in XML format. MIME header will be given, like:

`"Content-Type": "application/xml; charset=UTF-8"`

and data in this format:

```
<names>
  <name>Andy</name>
  <name>Dave</name>
  <name>John</name>
  <name>Suzy</name>
</names>
```

It is necessary to set API endpoint URL, and path to necessary field in XML object, in the segmentation rule:



Segmentation rule example for PIPE

PIPE is the format of data, divided by line breaks (written in one column). Let's take situation, when information is given in PIPE format. MIME header will be given, like:

`"Content-Type": "text/plain; charset=UTF-8"`

and data in this format:

```
+78002000100
+78004000200
+78006000400
+78008000600
```

It is necessary to set API endpoint URL in the segmentation rule:

Uploaded file

If you select Is (Is not) is file in a column segmentation rule, form for uploading CSV (comma separated values) file will appear:

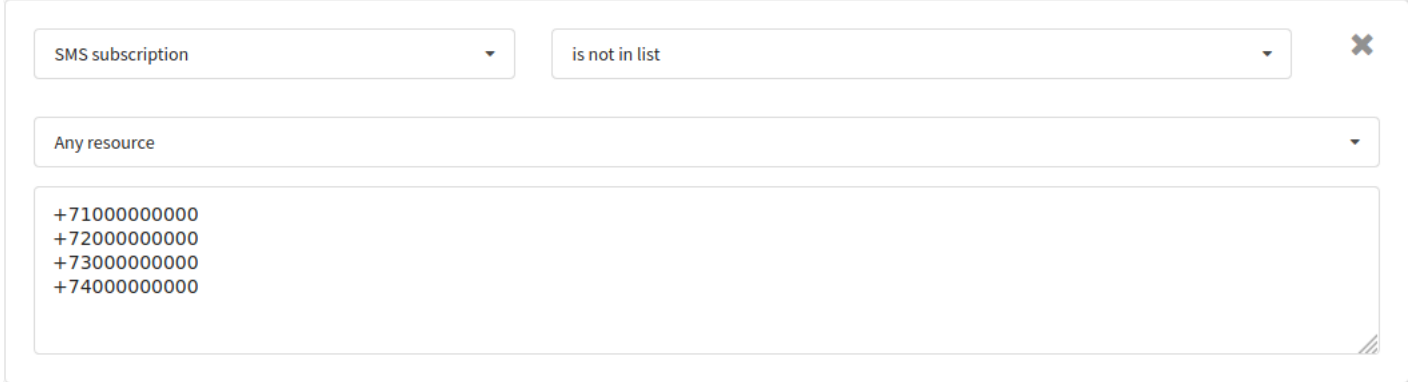
```
Email,Name
john@example.org,John
suzy@example.com,Suzy
dave@example.net,Dave
andy@example.edu,Andy
```

After uploading the file, you can select column in the drop-down menu:

Information from buffer

If you select Is (Is not) in list segmentation rule, data in PIPE format will be inserted manually into the special field.

PIPE is the format of data, divided by line breaks (written in one column).



Segmentation by External SQL tables

Description

The ability to connect external SQL databases allows you to perform flexible profiles segmentation. This type of data storage has the following features:

- **It is not necessary to store all data in Altcraft MP**
- **Regular synchronization is not necessary. Your data will always be actual**
- **You can use data types, that were not designed in Altcraft MP**
- **It does not matter, what platform and system you use to store information. You only need access to perform SQL queries**

Connectors to external databases are created in Altcraft administrative panel.

TIP

To connect to your database you would need an account with proper access privileges. The process depends on your version of Altcraft Marketing:

- **If you are a cloud account user — send access credentials to our support — support@altcraft.com. You can also contact this address for any questions related to connecting and configuring external services and databases.**
- **If you have an On-premise installation, platform administrator should create a database connector in Administrative panel — and assign it to your platform account.**

Creating query to the database

You can configure queries both in administrative and user panels. In user panels you can also watch profiles which were selected by your query.

You can restrict editing queries in main.json configuration, if it is necessary. To do this, you need to set parameter: `"DISABLE_QUERY_EDITOR": true` **.**

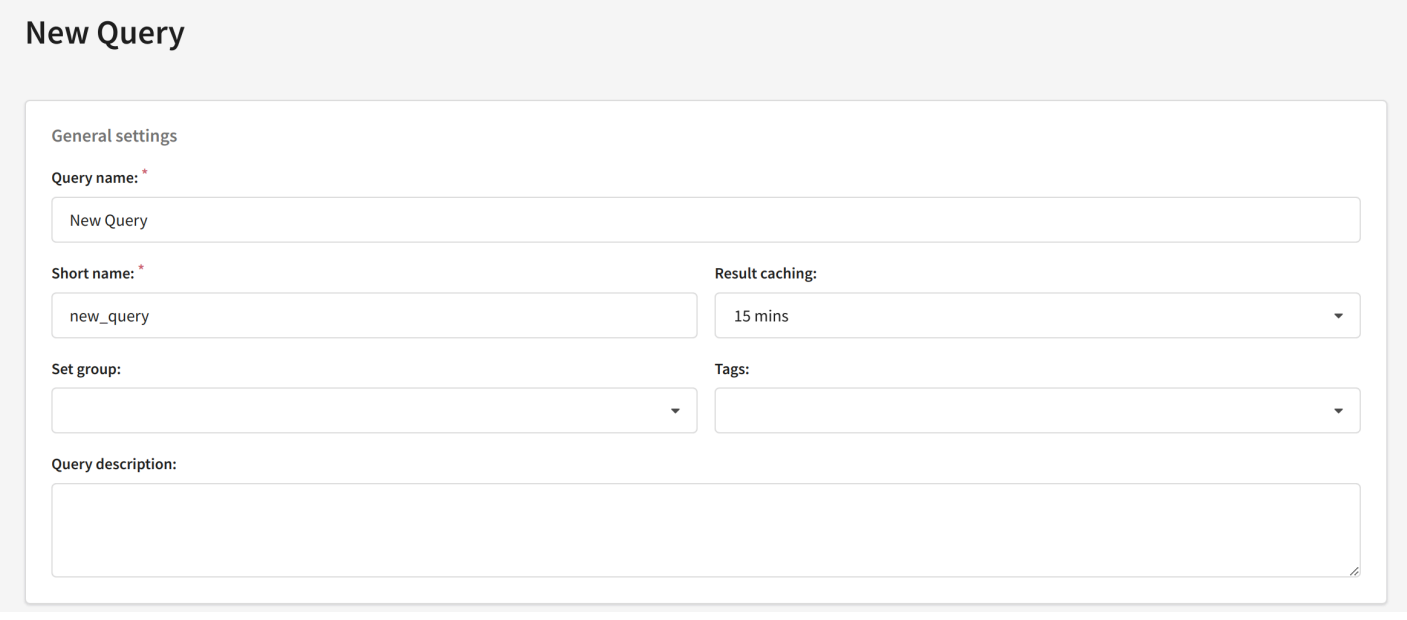
To configure queries, go to Data menu, and select Segmentation queries:

Go to this section, and press Create button.

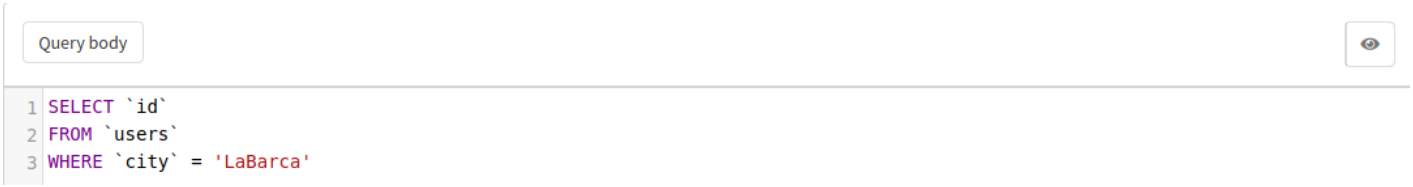
In General settings specify query name, short name (unique for each query, used in variables and API-requests), result caching time and also groups, tags and description:

INFO

Please note: the query result caching only works when calculating segments. Automation scenarios use a separate caching mechanism with its own settings.

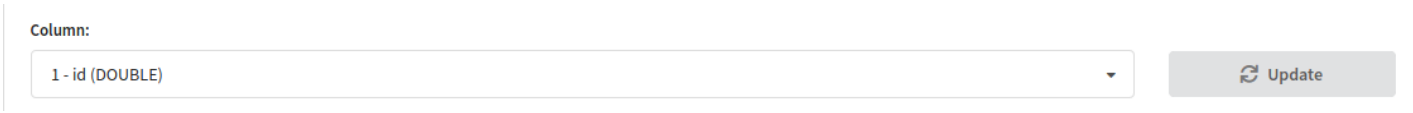


For correct query, it is necessary to select column with data, which are already contained in Altcraft MP. For example, if customer identifier is stored in Altcraft, you need to make a query, that has this ID at least in one column:

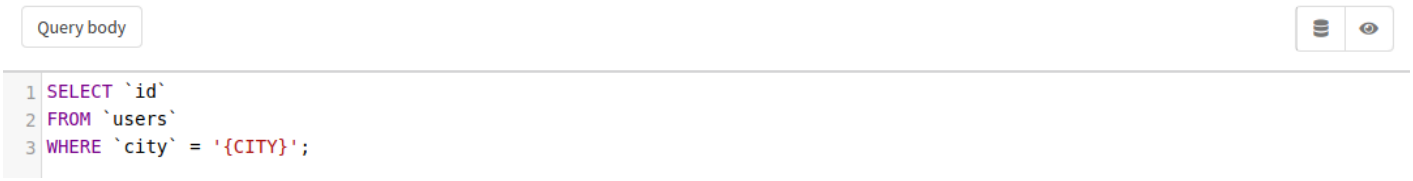


These queries will be used for segmentation. You can make unlimited number of these queries for segmentation. In every query, you make selection by one or several parameters. Also you can combine your queries using underselection, JOIN and UNION operators.

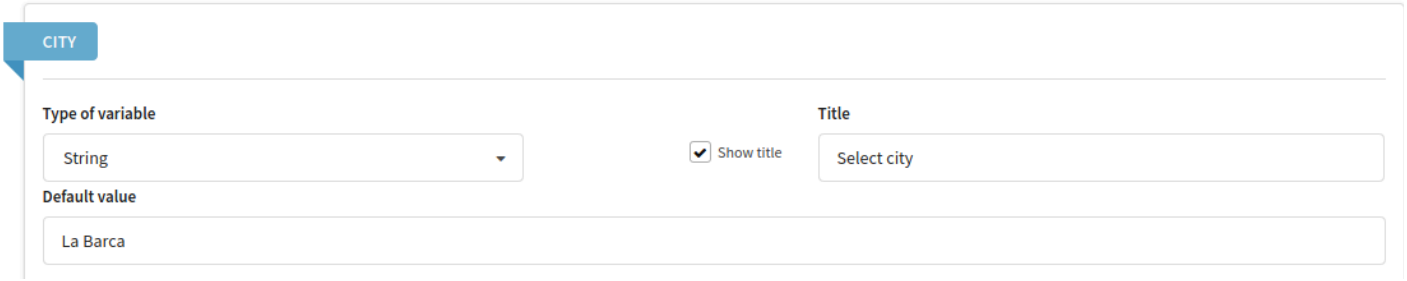
It is necessary to select the column which will be used to search profiles, and press Update button to get a new columns list for your query:



You can parameters into the query to change it during the segmentation. To do it, use random unique parameter name in braces, like `{CITY}`:

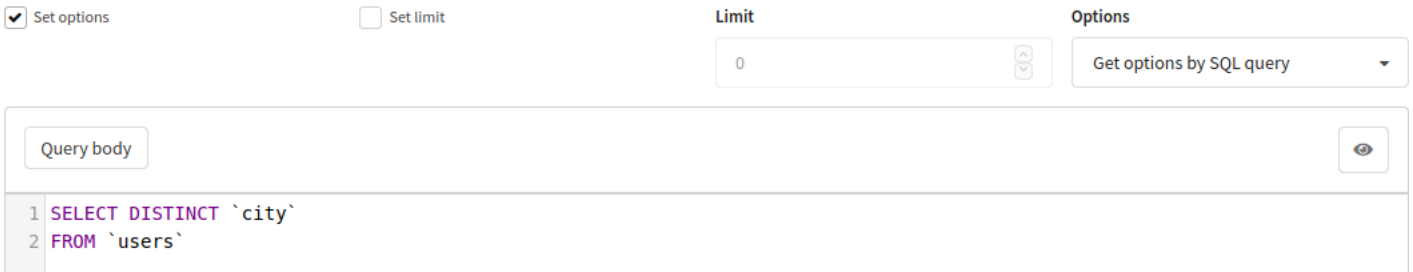


The form for configuring the parameter will appear, when you set it:

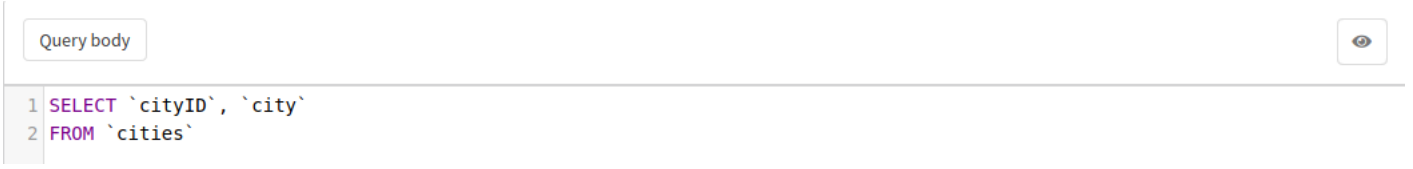


In this form, you can set Type of variable, its Default value, and Title, that will be displayed when you create the segment.

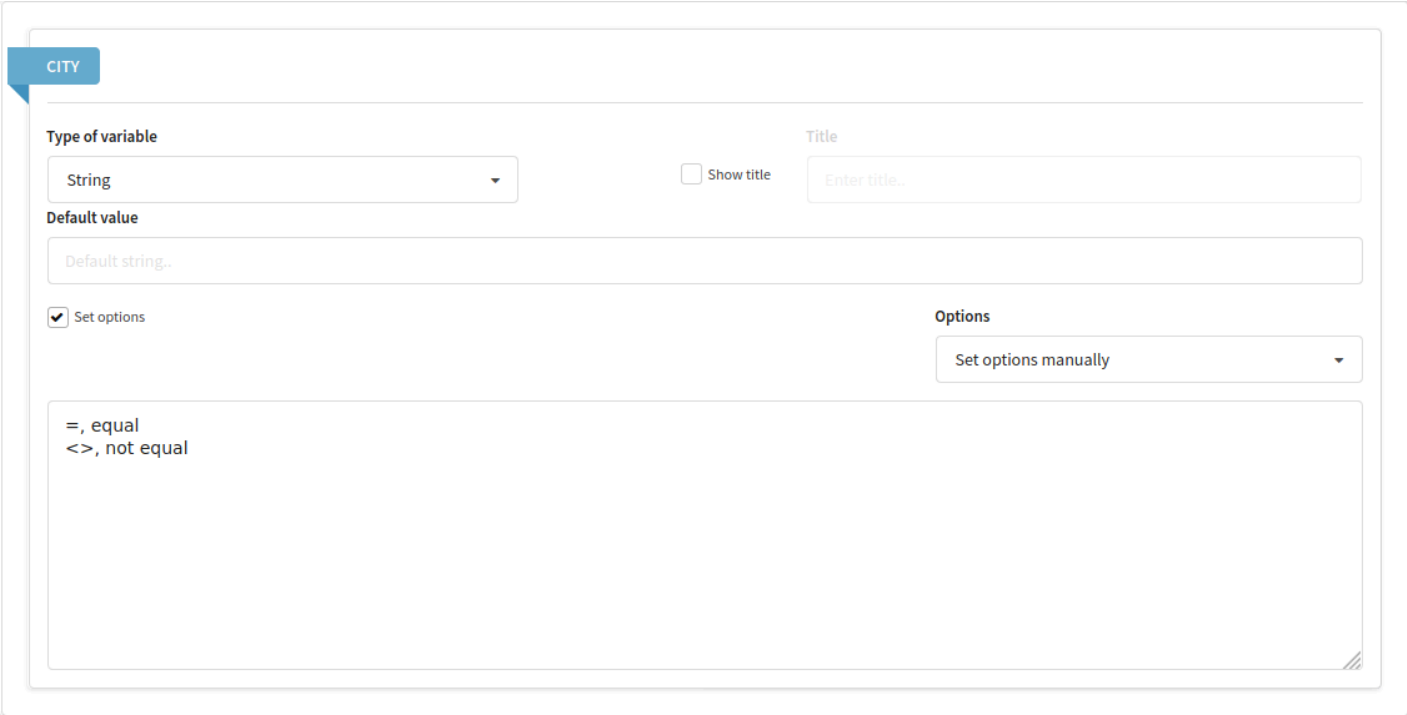
If you want to select field value from the list in the segment, use Set options checkbox. When you toggle it, you can set options list, or select them form SQL database:



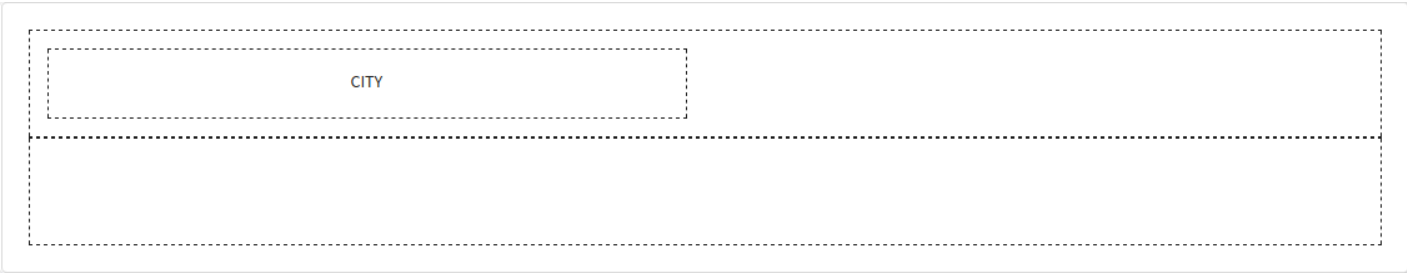
If you use identifiers, your SQL query must give two columns — the first one will be used for identification, and the second will be displayed for the user:



If you select Set options manually, you can enter identifier and options, separating them with commas:

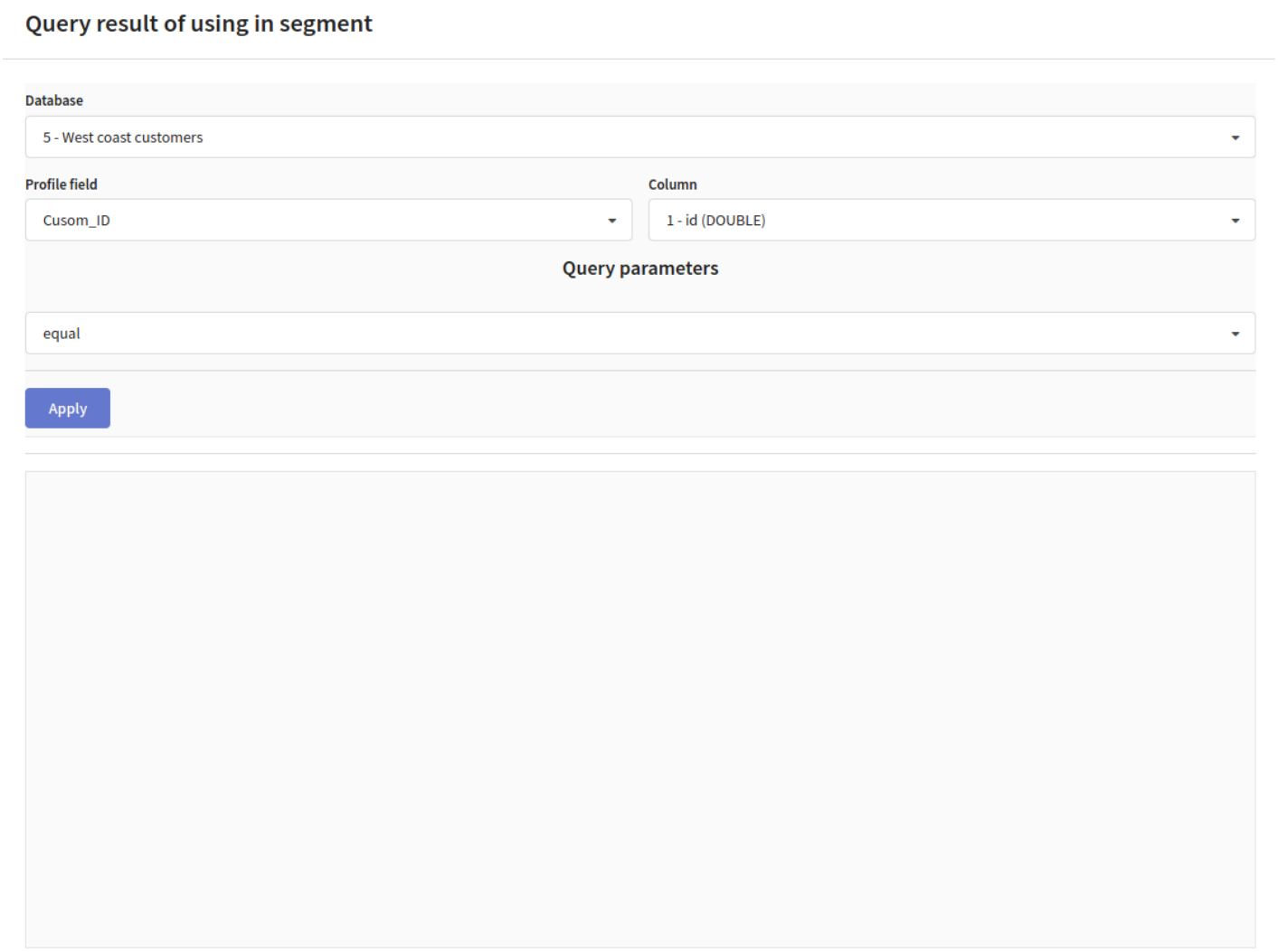


For all parameters, you can configure their display order during segmentation. Display order can be changed by dragging the elements. Headers will be displayed separately, for more convenience.



Use preview button to see how the query will be displayed in the segment, and what will be displayed in test selection:

Select database, profile field, and column for selection. In *Query parameters* section, the form will be displayed as it will be displayed in the segment. Press *Apply* button to perform the query, and profiles that fit query condition, will be shown:



After saving the query, you can use it in the segments.

Using the query to external database

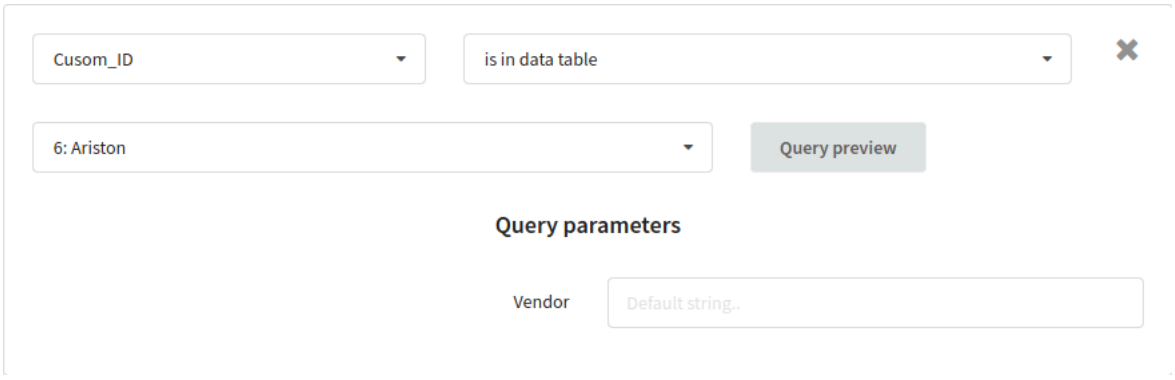
In the main menu, go to Data —> Segments section.

Create a new segment or edit existing one.

Add a new rule and select identifier field, or field that will be used for profiles search. Then, select the option:

- **Is in data table — if you want to select all profiles that fit the rule**
- **Is not in data table — if you want to select all profiles that do not fit the rule**

Than, select your query and fill the form with parameters:



Now, you can save the segment.

The rules can be applied more than once in the segment and in groups, combining them with AND and OR operators.

Any conditions can be combined in segments with queries: by Altkarft MP fields, by actions, by goals and others.

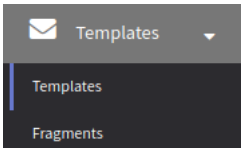
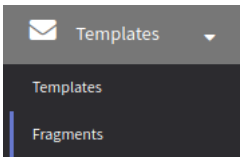
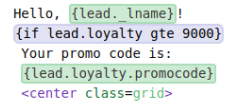
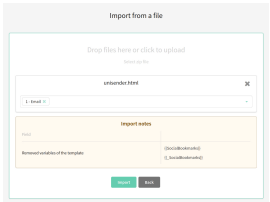
Message templates

Description

Message template is a form for creating content for your marketing mailings. Within a single template, you can customize content for different communication channels. Templates in the platform support several types of dynamic content, such as queries to external sources, websites and SQL databases, as well as data passed in API requests to launch mailings. The message editor allows you to add personalization variables, functions, logical expressions and cycles to your template. You can also customize the display of unique content for each specific group of subscribers.

You can attach files to the template to send to subscribers and stop lists to block email addresses. Templates can also contain special links to confirm subscription or unsubscribe.

Where to start?

	**Managing message templates** **Creating and editing message templates . Content editors for Email, Push and SMS versions. General and advanced settings. Personalization and live preview.**
	**Templates fragments** **Fragments are parts of message templates, which are stored separately and can be used for sections like header, footer, social links or loyalty program, - the repeatedly used ones. Fragment may contain plain text or HTML code for email.**
	**Variables and template functions** **You can use variables, logical expressions, loops and functions in the templates. In this section you can find description of the syntax and examples of message personalization.**
	Dynamic content in messages **You can insert dynamic content into your emails, such as HTML, APIs, JSON content, and data from external SQL tables. You can read more about each type of content in this section.**
	**Importing a template from a third-party service** **If you already have a message template that you're used to, you can easily transfer it to the Altcraft platform and use it in your marketing mailings. Learn how to do it by navigating to this guide.**

Managing message templates

Description

	Message template is a set of content for your marketing mailings. One template combines content settings for all communication channels. Select Templates in the main menu to start creating content for your mailings.

Managing templates

All the Message templates available in your account or your access group are displayed in the list. You can search them by name or template ID, or group templates by tags. You can also display only drafts or ready-to-send templates. Click on the template thumbnail to open the preview of the message. Here you can check how the message looks in different channels and on different devices.

> TIP
>
> **If you use Double opt-in — double subscription confirmation — such templates will have Confirmation tag on their thumbnails. You can read more about Double opt-in subscriptions in this article.**

For every template, quick actions menu is available, you can use it to edit, clone, remove template - or set access group. Press + Create button at the top part of the page to add a new template.

For each template, general information is collected. Whether there is a:

- **Confirmation link: a template with such a link is required when using a Double opt-in subscription.**
- **External content: the use of links to dynamic JSON content in the template.**
- **Suppression list when unsubscribing: if the subscriber clicks on the unsubscribe link in the message template, he will be added to the selected suppression list. A mailing that uses the template will block sending an email to a subscriber whose contacts are in the list.**

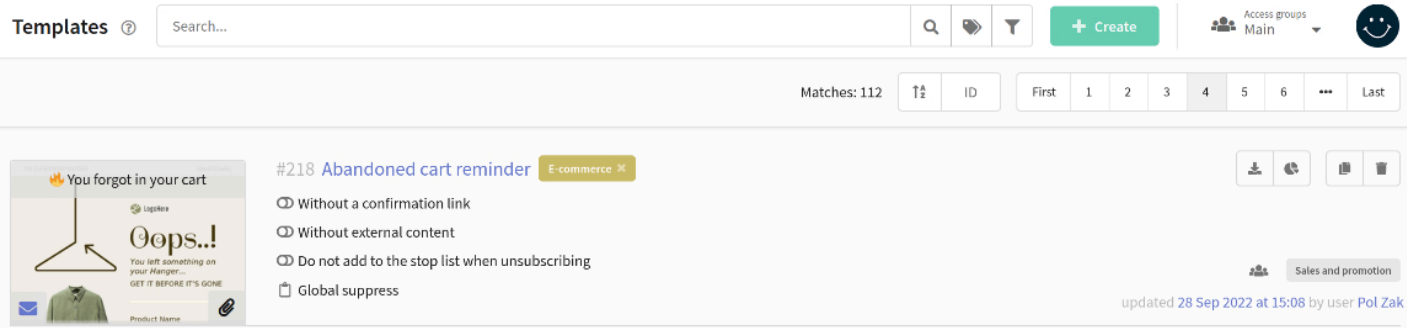
On the right there is a menu of quick actions:

— go to the summary report by template.

— clone the template.

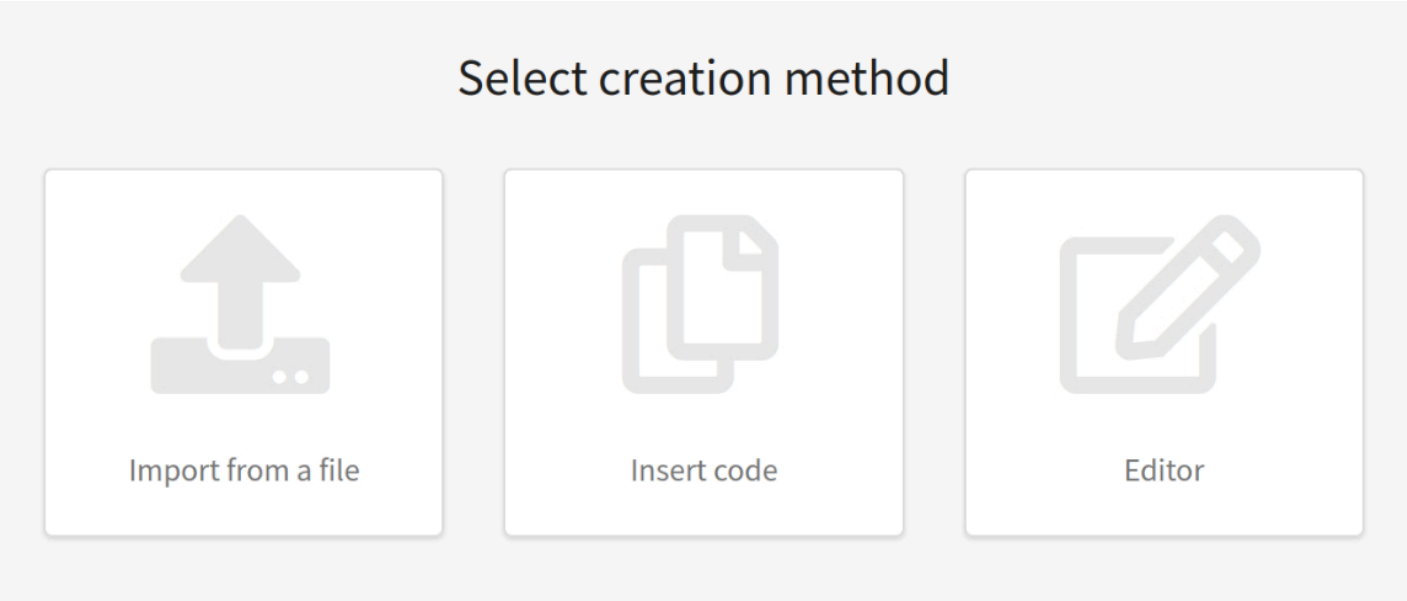
— delete the template.

To add a new template, at the top of the page, click the + Create button. If you need to edit an existing template, click on its title.



Next, you need to choose how to create a template:

- **Import from a file: download a ready-made template as a zip archive or html file.**
- **Insert code: Adds the copied html code to the text box.**
- **Editor: create a template in the editor.**



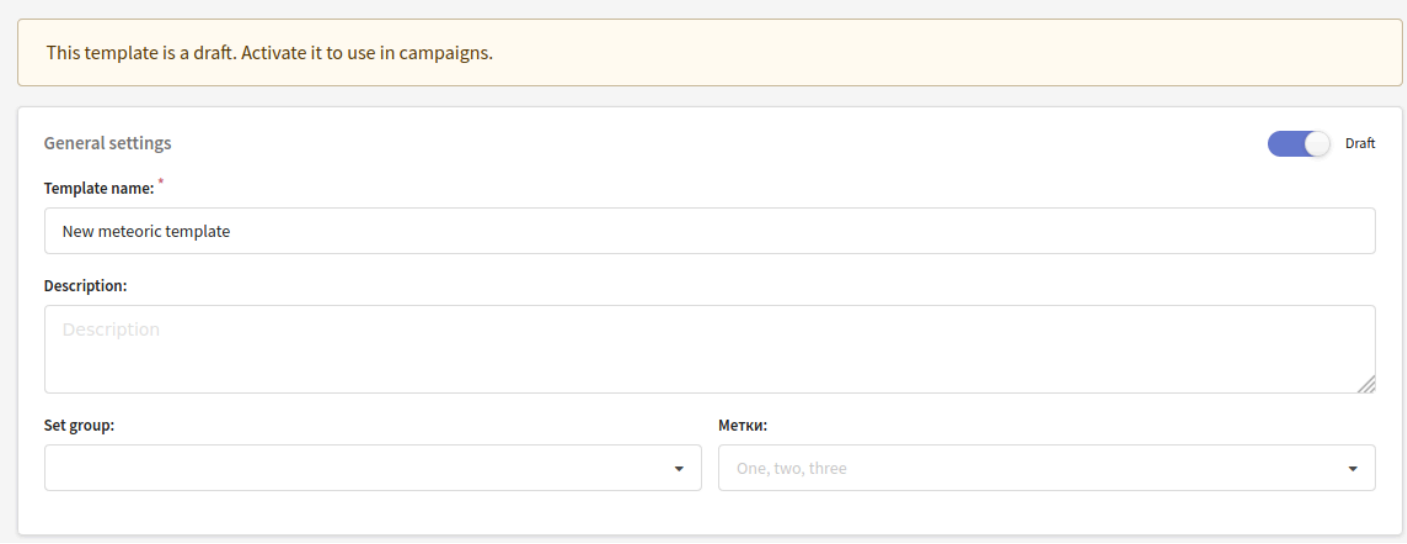
TIP

If you need to move a template from a third-party service, use import from a file or paste code.

General settings

In general settings set template name and description, add Tags and, if necessary, set access groups.

If you want to organize content moderation, you can mark template as a Draft. Thus junior and outsource employees can create content — and their senior colleagues can then verify it by lifting the Draft restrictions.



Content editors

Message template combines message editors for all available communication channels. It will help you create good-looking and convenient messages, that make content personalization easier. Press button with “plus” sign to add content for the certain communication channel:

Content editor for the selected communication channel will open in a new tab:

Email

For email channel fill in the following fields:

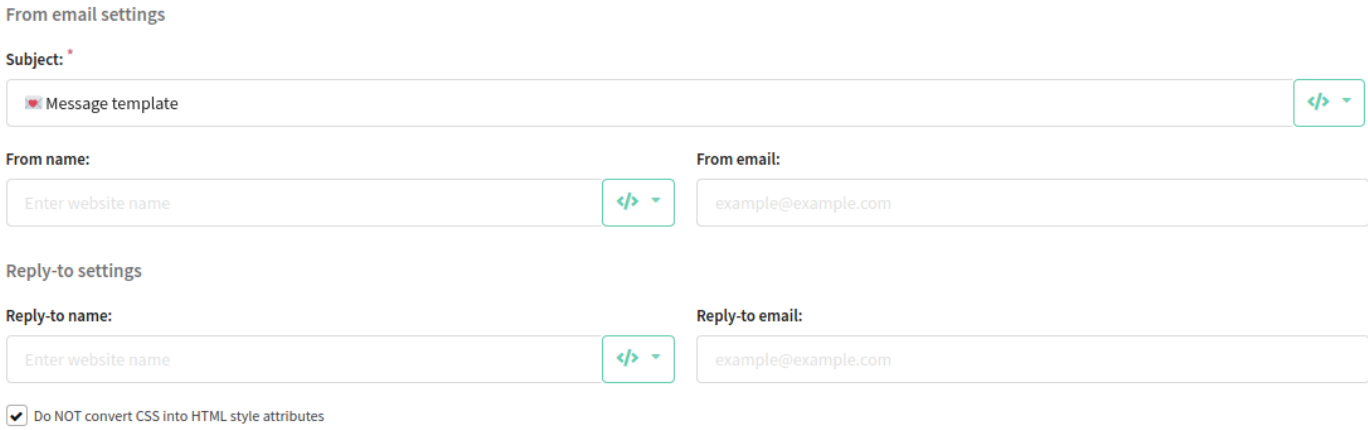
- **Subject — letter subject, that will be displayed to the customer. You can use personalization variables ({``}) and emoji.**
- **From name — your company, department or mailing name. Here you can also add personalization and emoji.**

- **From Email — your sender domain address, from which letter will be sent to the customer.**
- **Reply-to name — name for replies to the message. Personalization is available: for example, you can insert customer's personal manager name.**
- **Reply-to email — Email address for replies.**

By default, Altcraft does not convert styles, included into HTML version of the message, into line tags (Do NOT convert CSS into HTML style attributes). But, if you remove this option, styles will be added as attributes to message elements:

```
<h1 style="font-size:36px;">Hello!</h1>
<h2 style="font-size:28px;">Nice day, isn't it?</h2>
```

This will provide better compatibility with Outlook and other mail clients, that do not support `<style\>` tags.



CAUTION

From name and From Email fields overwrite values, configured for Email channel both in Resource and Profiles database.

HTML version editor

Most people see HTML version of the email in their inboxes. Editor supports HTML, CSS and personalization variables. These functions are available to make formatting and work with objects easier:

Formatting:

B

— bold

I

— italic

~~S~~

— ~~strikethrough~~

— bullet list

— numbered list

— table

Service functions:

— pixels for tracking of letter opening and reading. If you do not add pixel during message editing, the system will warn you about this.

— external HTML content. You can also set variable source address manually `{{http://example.com}}`.

— links builder (see more on the email editor tab).

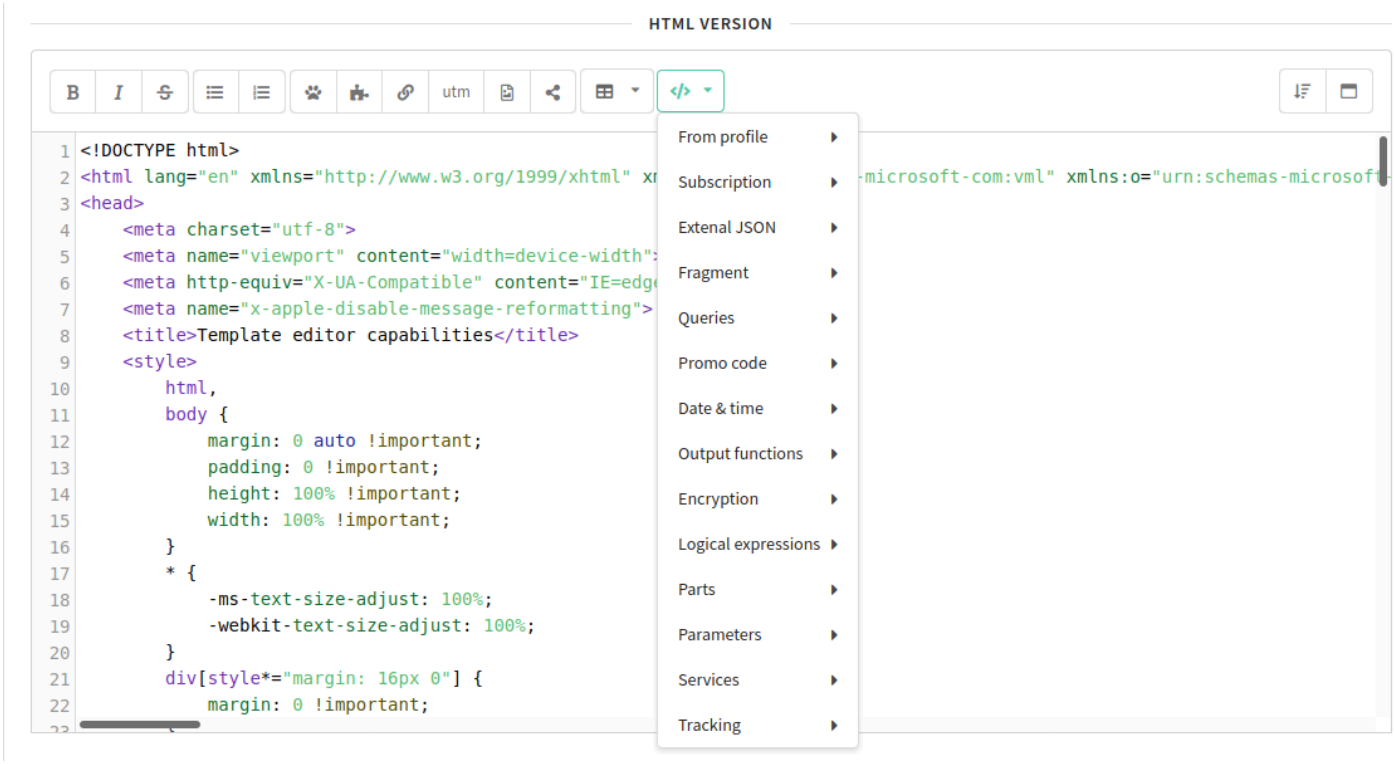
utm

— UTM tags.

— images.

— social buttons.

— variables and editor functions menu. Using this button, you can add Customer profiles' data, promo codes, loops, logical expressions, dynamic content and service parameters into the message.



TIP

You can also toggle word wrapping or enable fullscreen mode.

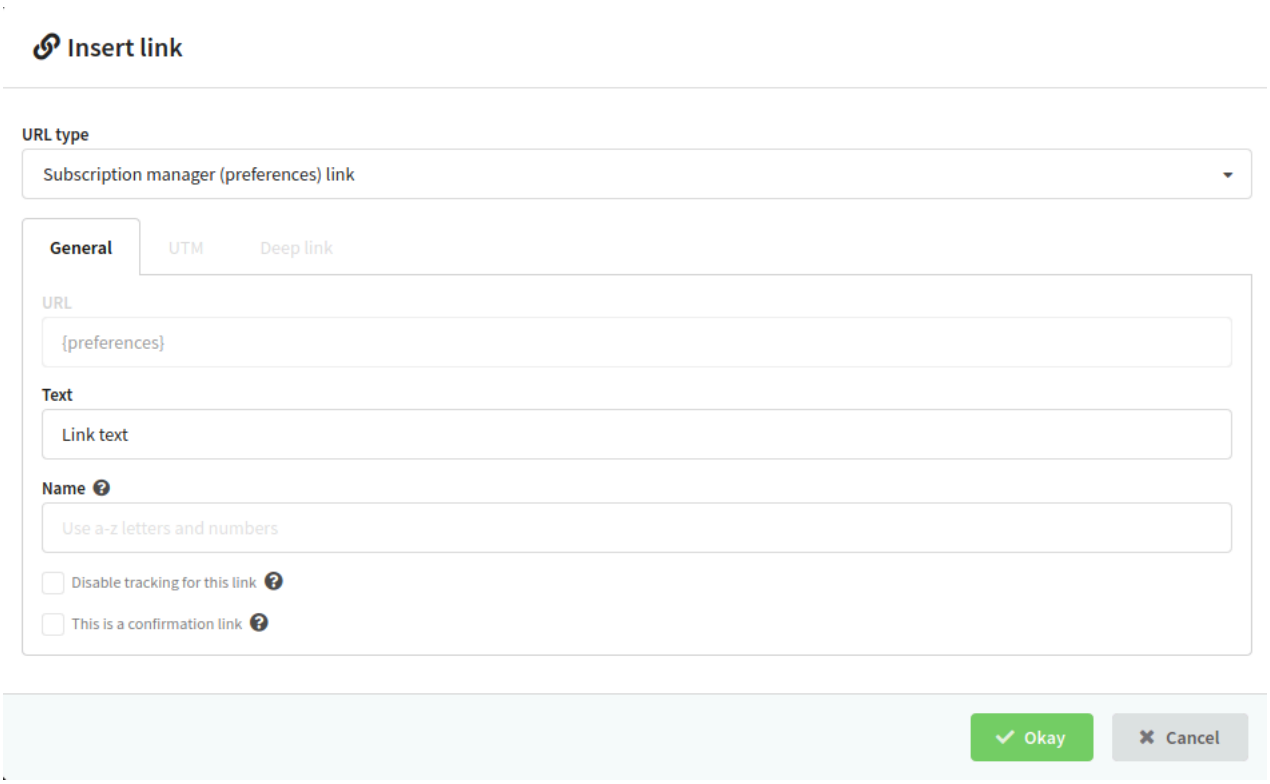
Link builder

Link builder

Press Insert link button in HTML or plain text version editor to insert a new link. If you want to edit an existing link — select it and then press the link editor button:

You can choose URL type depending on its purpose:

- **Enter a custom link — link to any page, except unsubscription and confirmation links.**
- **Unsubscription link — link to a system or custom link to stop resource subscription. Subscription goes to Unsubscribed status. It does not affect global status profile and other subscriptions.**
- **Subscription manager (preferences) link — link to the system page of subscription management. Customer can suspend his subscriptions, resubscribe or unsubscribe to mailings.**
- **Suppression link — link for adding customer into suppression list. Using this link, the customer can block receiving certain content.**
- **Global unsubscription link — changes global profile status to Unsubscribed, and completely stops any communications.**
- **Web version link — link for a web browser copy of the letter, that was sent to the subscriber. If the customer has any troubles with showing the letter, it would be available via this link in any browser.**

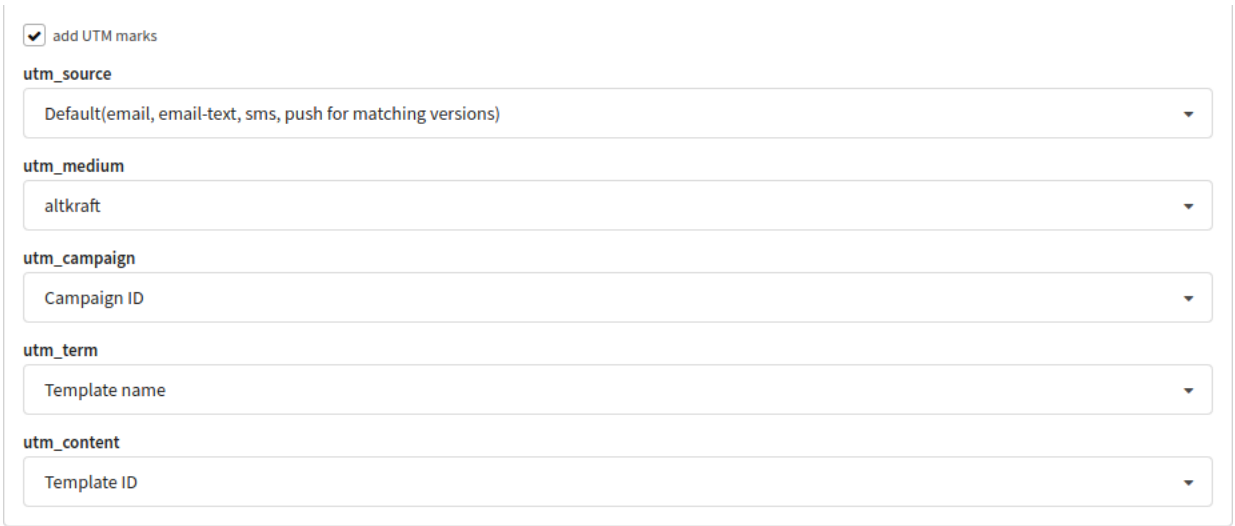


If you want to add your link, enter URL in general settings. This field will be pre-filled for system links. Also, enter Link text to be displayed and Link name — it is used in reports, segmentation and during triggers and scenarios launch.

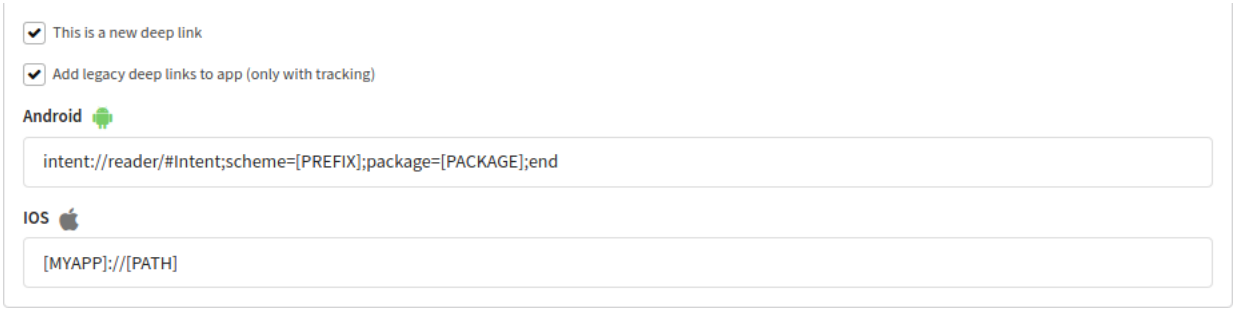
Also, you can disable redirection to tracking module when clicking on the link, or you can make any link a confirmation link.

You can set UTM tags for every specific link in the message. You can use a button in content editor to set marks for all links in the template.

You can use variables, like mailing name or database name, as UTM tags.



If you want to use the link for redirecting to mobile application, you need to activate This is a new deep link option in the appropriate tab. Legacy deep link can be added only if you use tracking.



Plain text version

Message text version is shown in email instead of HTML, if email provider — or email client — blocks HTML view.

TIP

Messages with text version have better delivery rate. Make sure you fill it in for the messages.

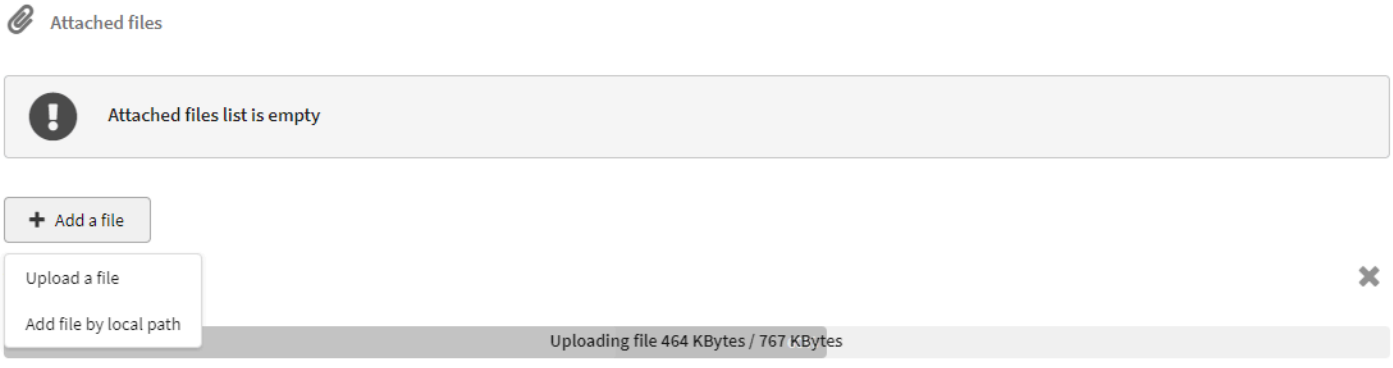
You can use

button to generate text version based on HTML. Text version also supports lists, UTM tags, links and variables.

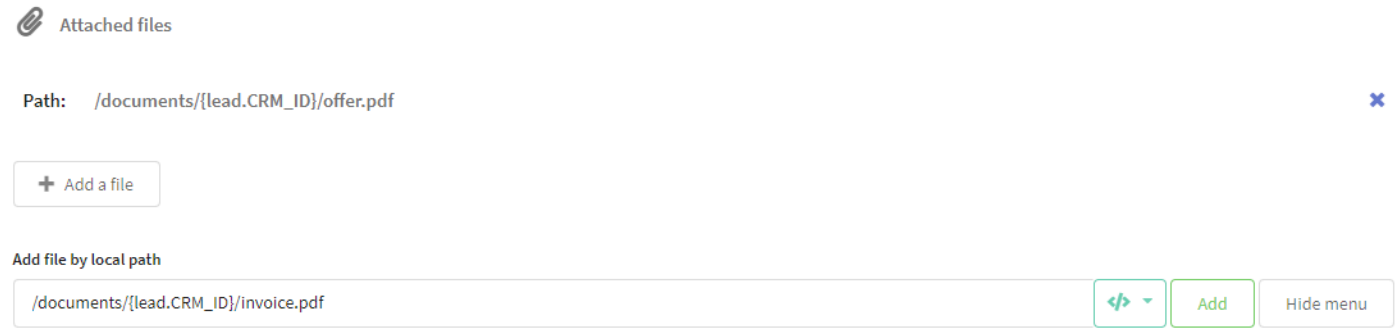
Email attachments

To attach files to your email message use the section in the lower part of the form. There are 2 ways:

- **you can upload a file from your computer file system:**



- **or specify a local (server, where the platform is installed) path to the file. Here you can use profile variables to send personal files. Enter file path and then press Add:**



To delete an uploaded file or a saved file path press an X in the right part of the form.

SMS

Enter SMS text into the text field. The editor will automatically count the number of symbols and messages. You can use links, UTM tags and variables in the message body:

</>

— variables and functions menu. Using this button, you can add customer profiles data, promo codes, loops, logic expressions, dynamic content and system links into the message.

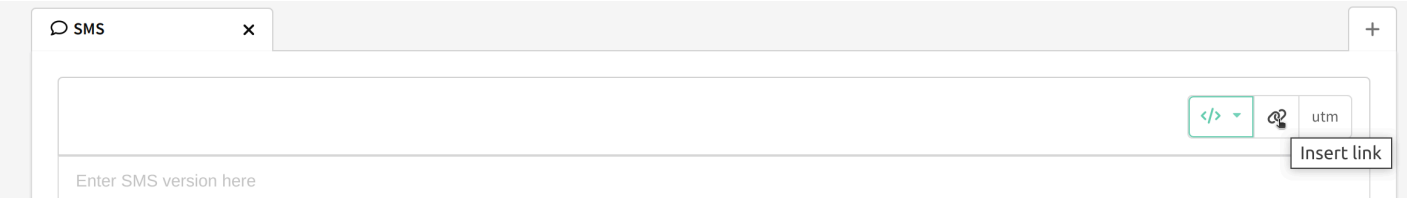
utm

— UTM tags settings.

— links builder.

Link builder

Press Insert link button to insert a new link. If you want to edit an existing link — select it and then press the link editor button:



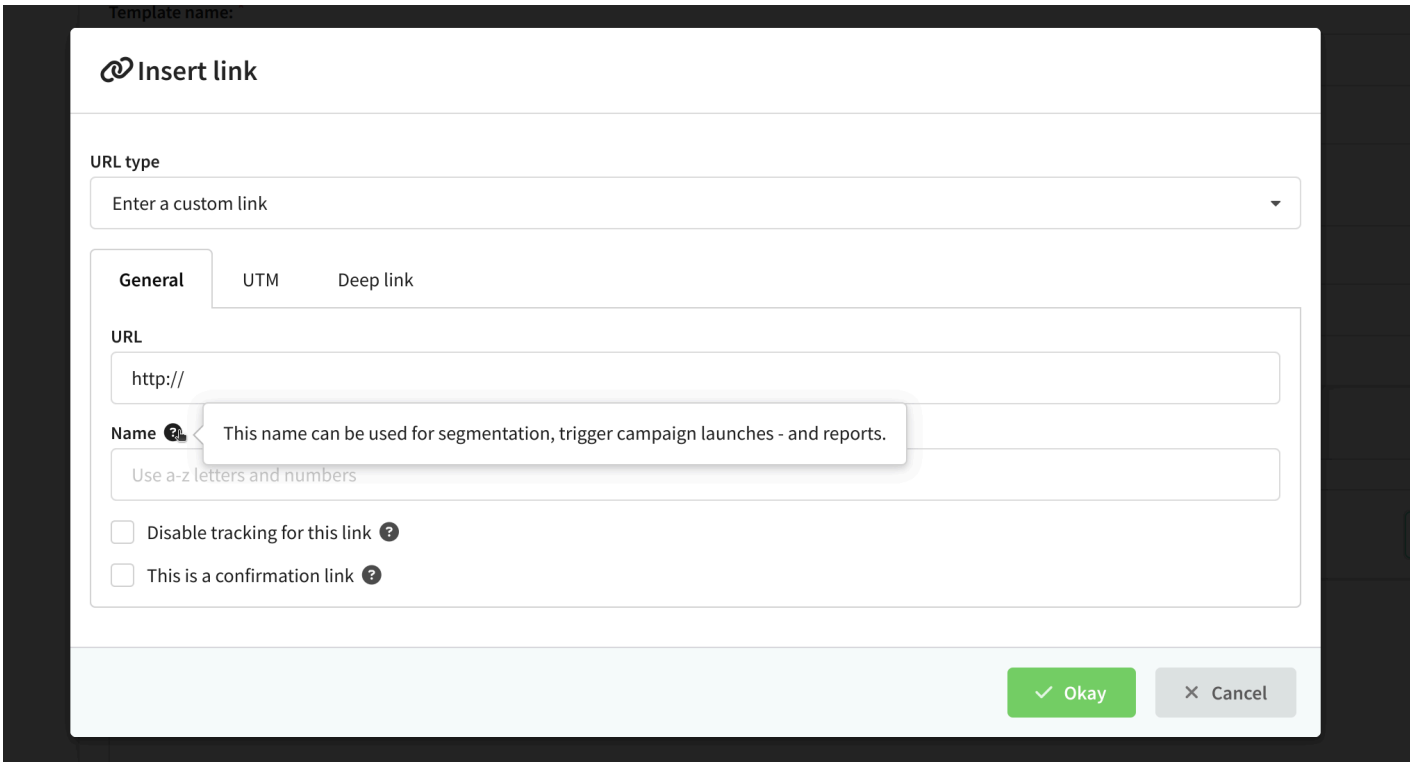
There is only one URL type for SMS. Press Enter a custom link to create a link to any page, except unsubscription and confirmation links.

To add your link, enter URL in general settings. Also, enter Link name — it is used in reports, segmentation and during triggers and scenarios launch.

CAUTION

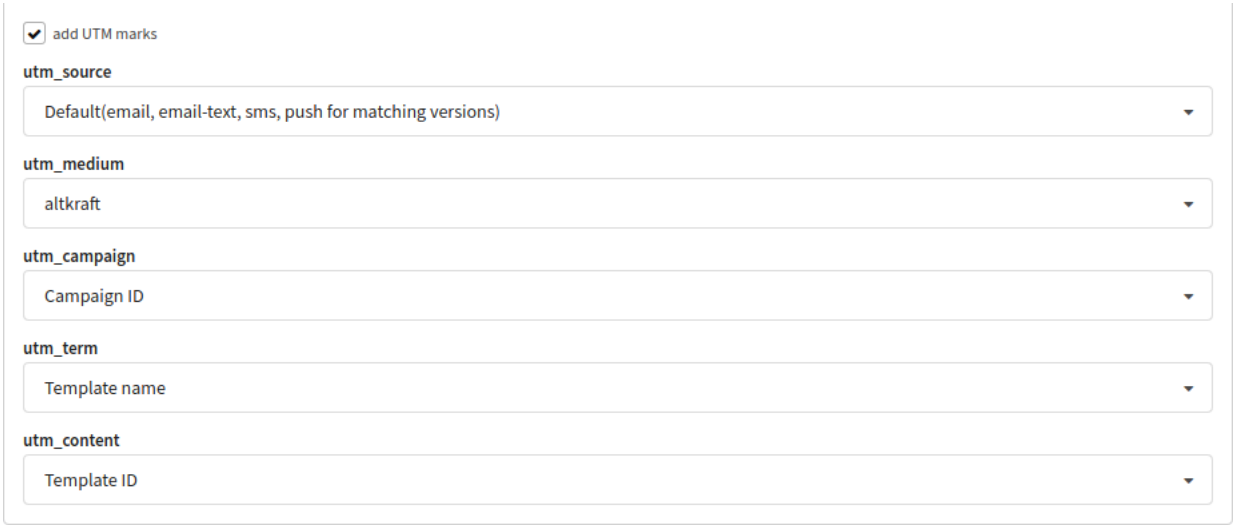
For further usage in scenarios and triggers, you must use the Link Name in the corresponding field, not the link itself.

Also, you can disable redirection to tracking module when clicking on the link, or you can make any link a confirmation link.

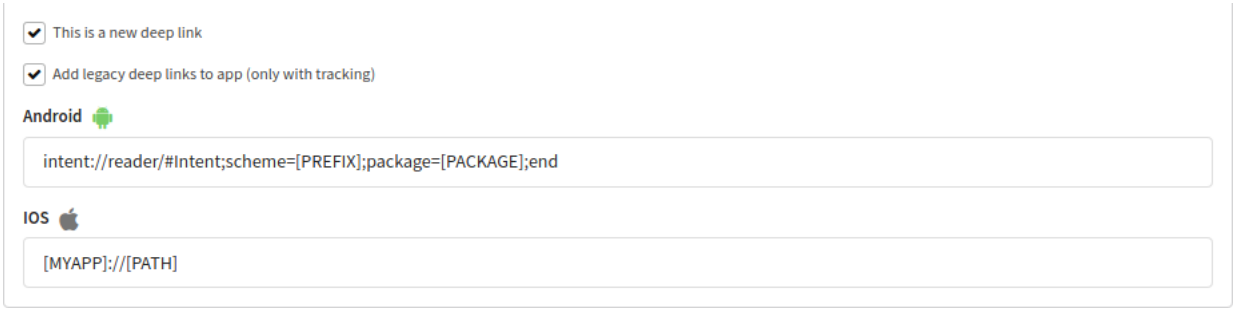


You can set UTM tags for every specific link in the message. You can use a button in content editor to set marks for all links in the template.

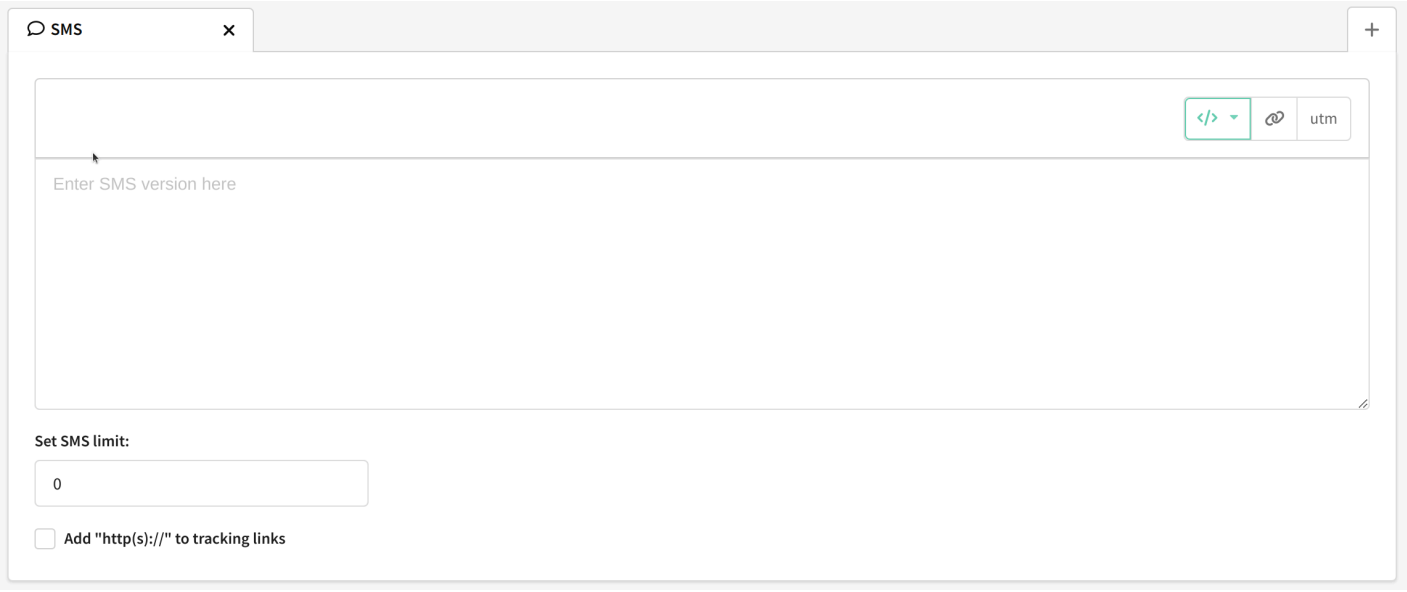
You can use variables, like mailing name or database name, as UTM tags.



If you want to use the link for redirecting to mobile application, you need to activate This is a new deep link option in the appropriate tab. Legacy deep link can be added only if you use tracking.



Also, you can limit the amount of SMS sended and add “http://” into the URLs:



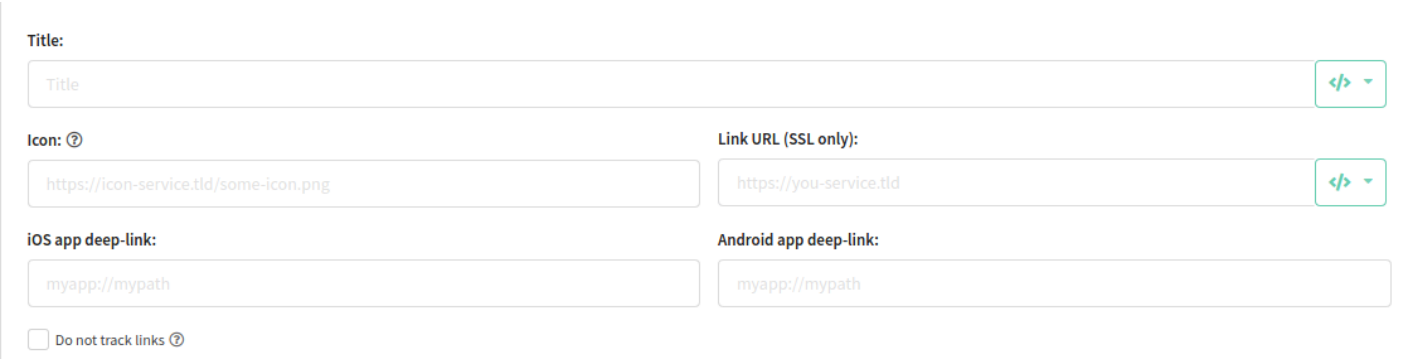
CAUTION

In SMS messages, angle brackets are used to denote parameters (for example, the name parameter would look `<name="">`). Anything you put in angle brackets will not be shown in the message.

Push

For push notifications, you can configure:

- **Title — variables menu is available for the title of your notifications.**
- **Link URL — link to follow when a customer clicks your push. You can use only encrypted protocol (SSL) for these links. Variables menu is available: for example, you can insert promotion codes. When customers clicks on this link, he will be redirected to the page with discount calculation.**
- **Icon — Image for display in the notification. Push notifications in Apple Safari support only one image for all notifications. It is configured in Push sending resource.**
- **iOS app deep link — link for iOS applications.**
- **Android app deep link — link for Android applications.**



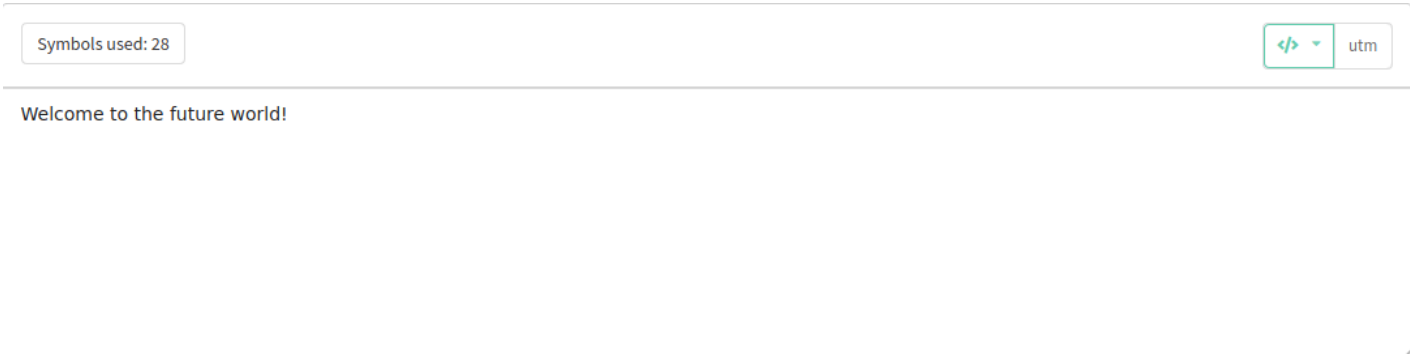
Enter push message text into the text field. The editor will automatically count the amount of symbols. You can use variables in the message body. Also, you can set UTM tags for the link in the message.

utm

— UTM tags settings.

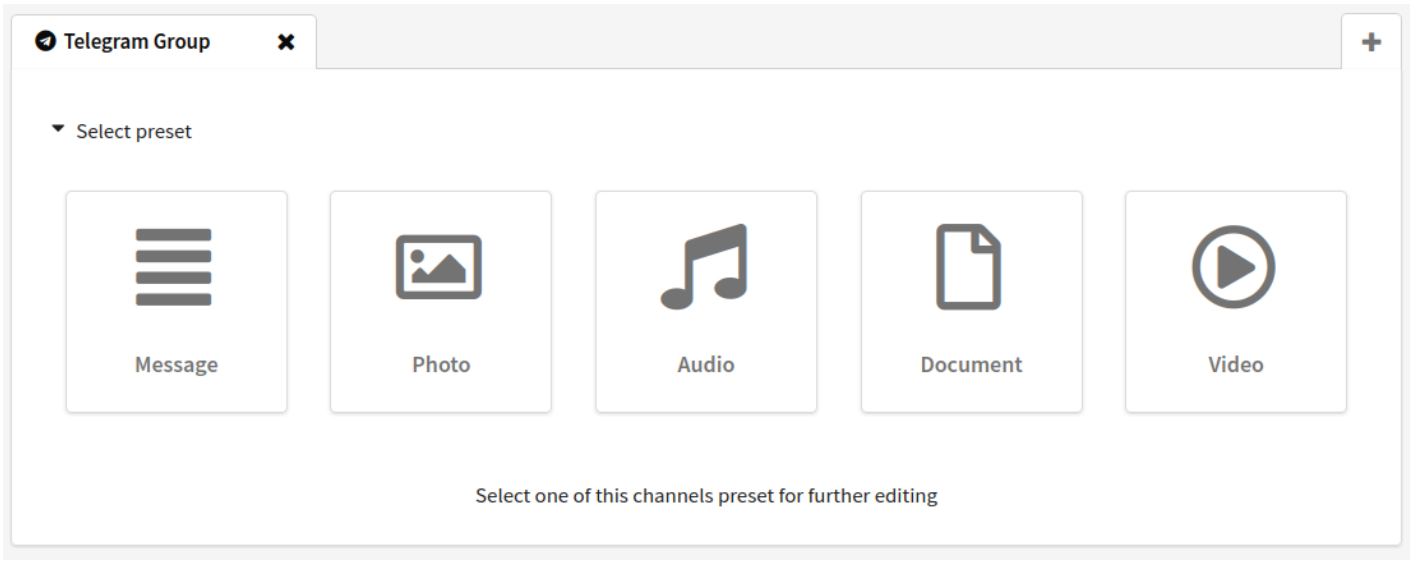
</>

— variables and functions menu. Using this button, you can add customer profiles data, promo codes, loops, logic expressions, dynamic content and system links into the message.



Telegram

Select a content type:

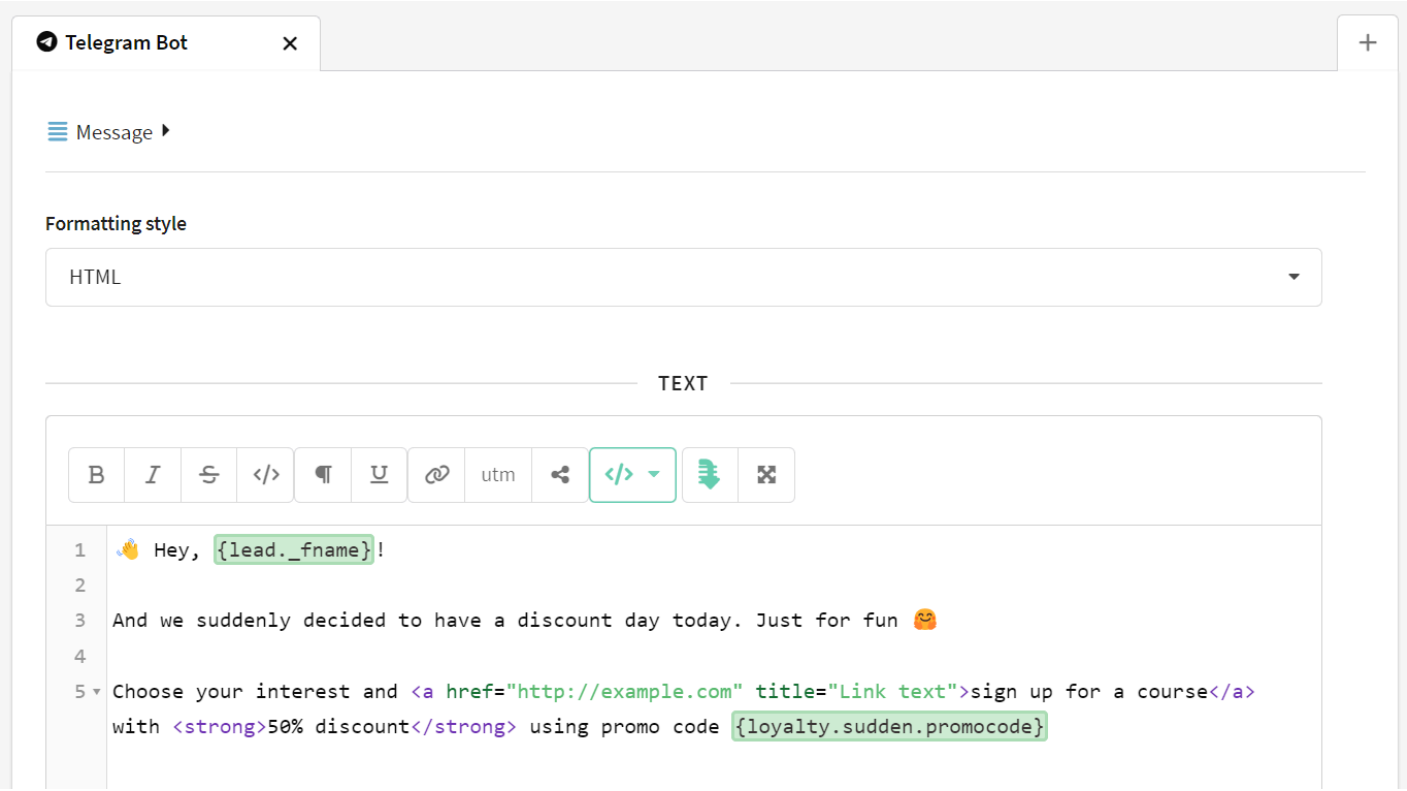


Message

Select a formatting style. For fine-tuning conten it's recommended to use HTML. Enter the text of your message.

TIP

The maximum length of one message is 4096 symbols. You can also add dynamic variables to the message — { </> }. Please note that data pulled from dynamic variables may exceed this limit. To ensure reliable delivery, we recommend monitoring the message length.



If you need to add a button to your message, enter a name and specify the link that the user will follow when clicking on the button. You can add only two buttons.

Photo

To create engaging content, you can add an image—click the “Upload image” button and choose an image from your gallery. Requirements for images in Telegram:

- **Formats: PNG, JPG, JPEG, GIF.**
- **Maximum size: 10 MB.**
- **The sum of the width and height of the photo should not exceed 10,000.**
- **The aspect ratio of width to height should be no more than 20.**

You can also add a description to the image (up to 1024 characters).

Audio, Video, Document

You can add files in two ways:

- **Upload a file from your device.**
- **Add a file via the local path. Choose the directory where the file is stored on the server where the platform is installed. In the directory path, you can use client profile variables. Enter the file location and click “Add.”**

Requirements for audio files:

- **Formats: MP3 and M4A.**
- **Maximum size: 5 MB.**

Requirements for documents:

- **Formats: Any document format.**
- **Maximum size: 5 MB.**

Requirements for videos:

- **Formats: MP4.**

- **Maximum size: 5 MB.**

You can also add a description to files of any type (up to 1024 characters).

TIP

All links in templates for Telegram Bot and Telegram Group are shortened.

WhatsApp

WhatsApp* uses 2 types of templates to communicate with users:

- **Typical template is an HSM (Highly Structured Message) template pre-approved by Facebook*. This type of template is required to start a business-initiated conversation. You have to use such templates to initiate a conversation with an end-user for the first time or after a period of 24 hours since the last interaction.**
- **Custom template is a free-form message. This type of template is required to start a user-initiated conversation. This is a conversation where a user messages a business. In this case, a business can reply to a user within the 24 hour customer service window.**

CAUTION

Currently, only typical templates (HSM) can be sent via Altcraft Platform.

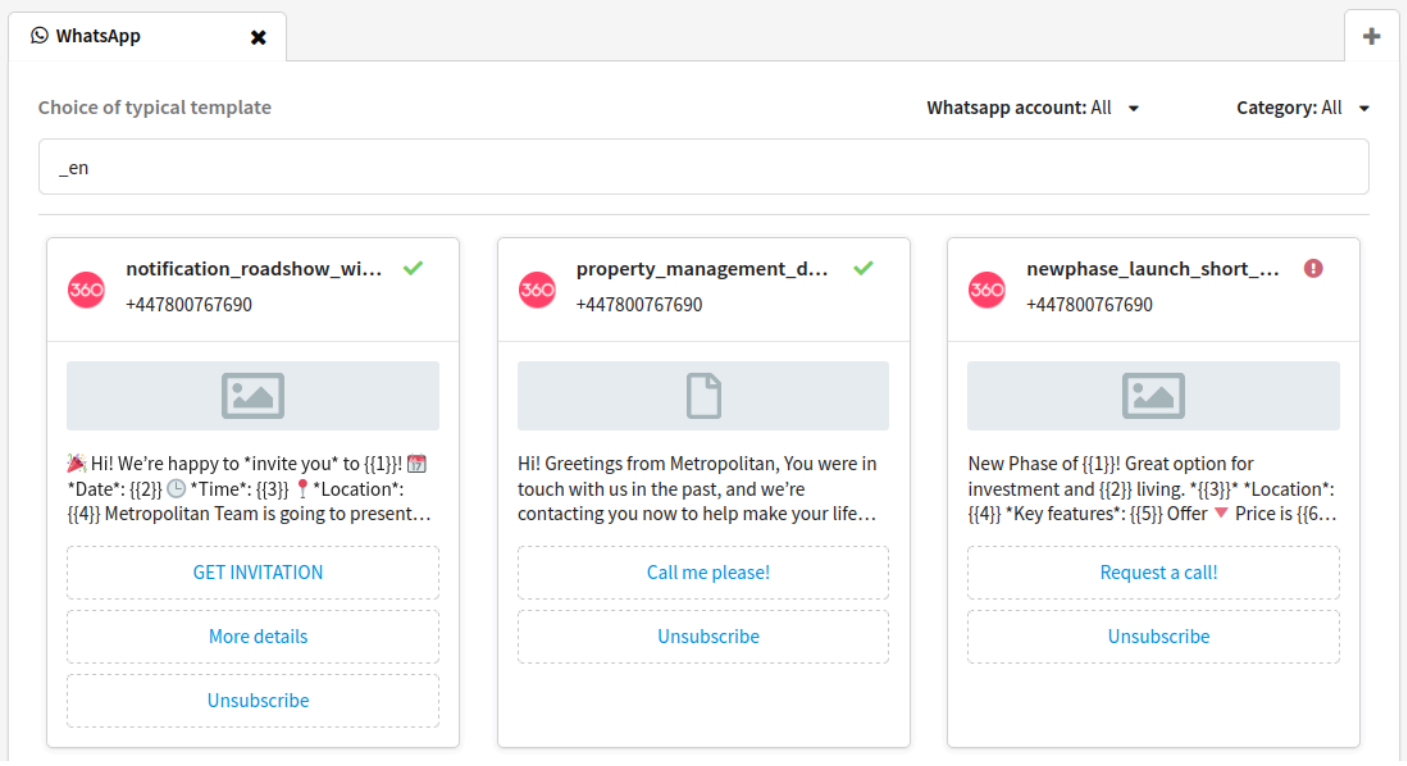
You must first create message templates in your personal provider account.

TIP

You can find more information about creating templates in your provider's documentation:

- **Edna: https://pulse-alpha-six.vercel.app/docs/template/whatsapp/operator-template**
- **360dialog: https://docs.360dialog.com/docs/360-client-hub/template-management-ui**

All templates available in connected WhatsApp* accounts through edna or 360dialog are automatically available in Altcraft Platform and displayed in the editor. To search for the desired template in the platform, you can enter a query in the search bar or use the account/category filter in the upper right corner.



CAUTION

Mailings can only use templates approved by Facebook*.

The template status is displayed to the right of its name:

✔

— template approved;

— template rejected;

— the template is being moderated;

— the template has not been sent for moderation.

Select the template you want to use in the mailing. To start creating a message based on a WhatsApp* template, click on it. The editor will display the template structure:

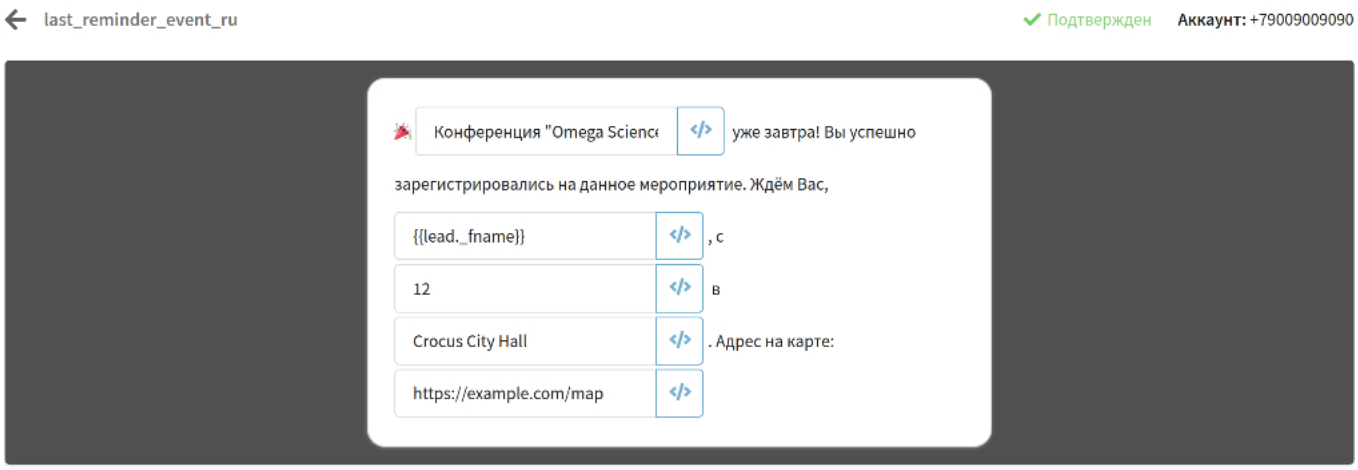
- **Header — the block can contain a welcome text or a media file;**
- **Body — the main text of the message with dynamic variables (required block);**
- **Footer — additional text;**
- **Button — block with buttons to add interactivity.**

You can edit template elements such as variables, media files, and buttons.

Variables

A text block consists of fixed text and variables. Variables are template text that changes for each new message (name, address, event name, date and time, link, etc.).

You need to manually enter variable values or add Altcraft variables. To open the list of variables, press `</>`. When sending a message, the entered text will be automatically added to the template.



TIP

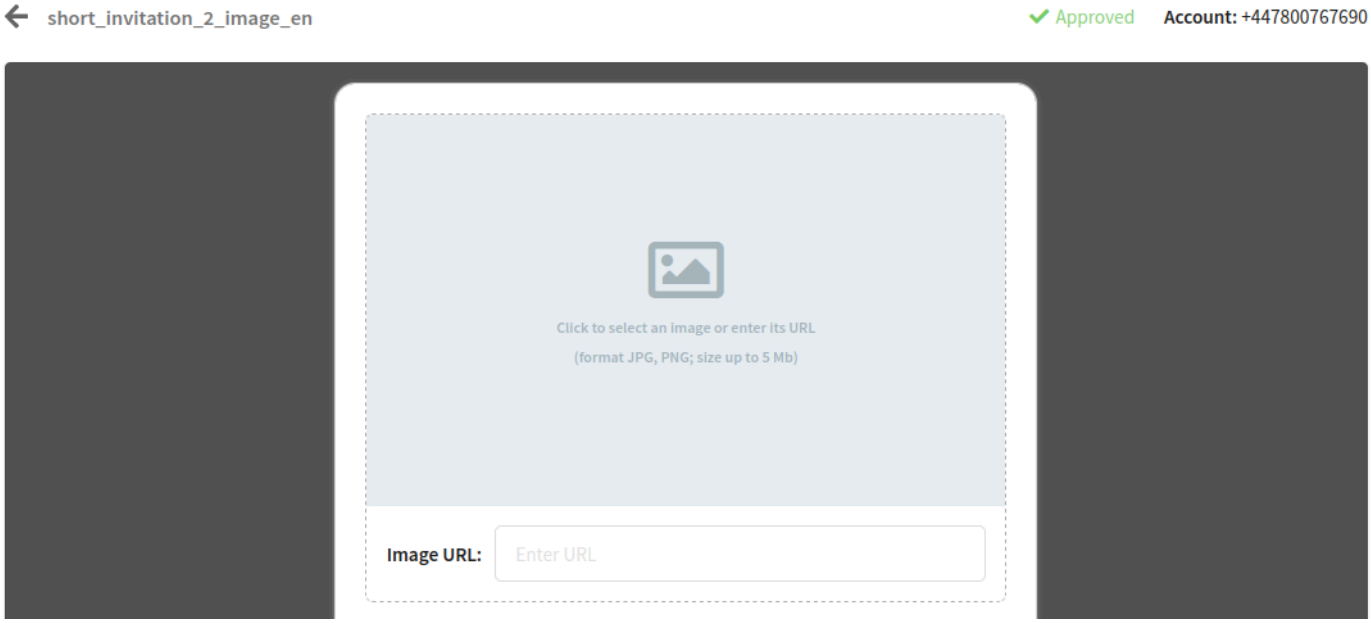
Links embedded in WhatsApp* variables are shortened.

Media files

Available media files:

- **image (JPG/PNG, up to 5 MB, recommended image size 640x640)**
- **file (any valid MIME type, up to 16 MB)**
- **video (MP4, 3GPP, up to 100 MB)**

To add an image, click on the upload area and select an image from the platform gallery or enter the URL:



If you need to add a file or video, click on

and select the file on your device or enter an URL.

Buttons

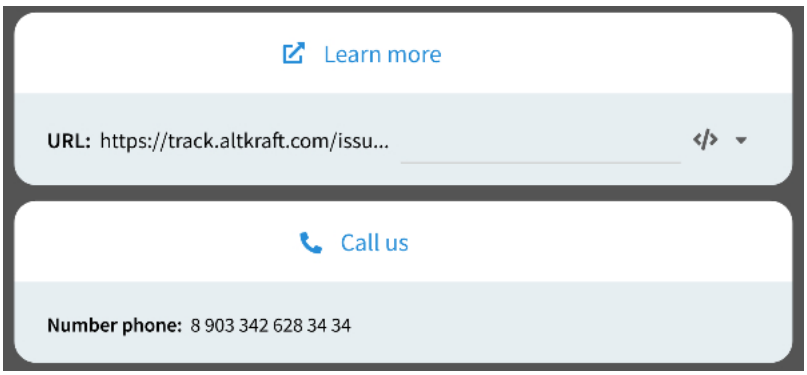
WhatsApp* has two types of buttons for HSM templates: Call-to-action and Quick-reply.

Call-to-action button

The CTA can be a URL button or a Phone button. There can be only two buttons of this type in the template.

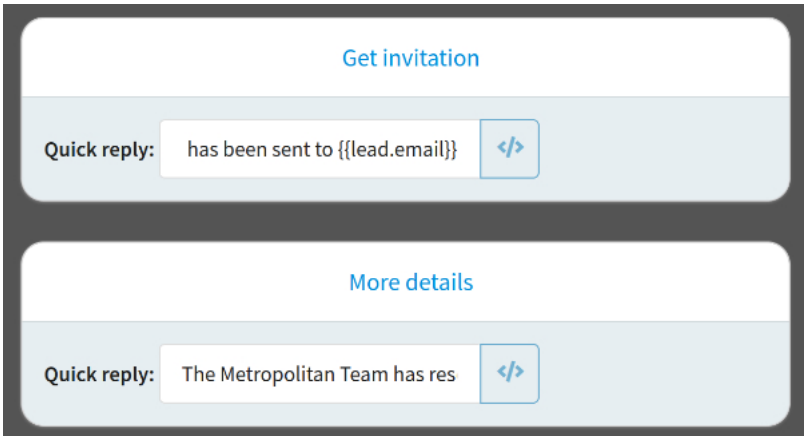
When clicking on the URL button, the user goes to the specified page. If a parameterizable URL is specified in the WhatsApp* template, it must be filled in.

When a user clicks on the Phone button, the number is dialed. The phone number cannot be edited.



Quick-reply button

When clicking on such a button, the user receives a quick response specified in the template. A template can contain no more than 3 buttons of this type.

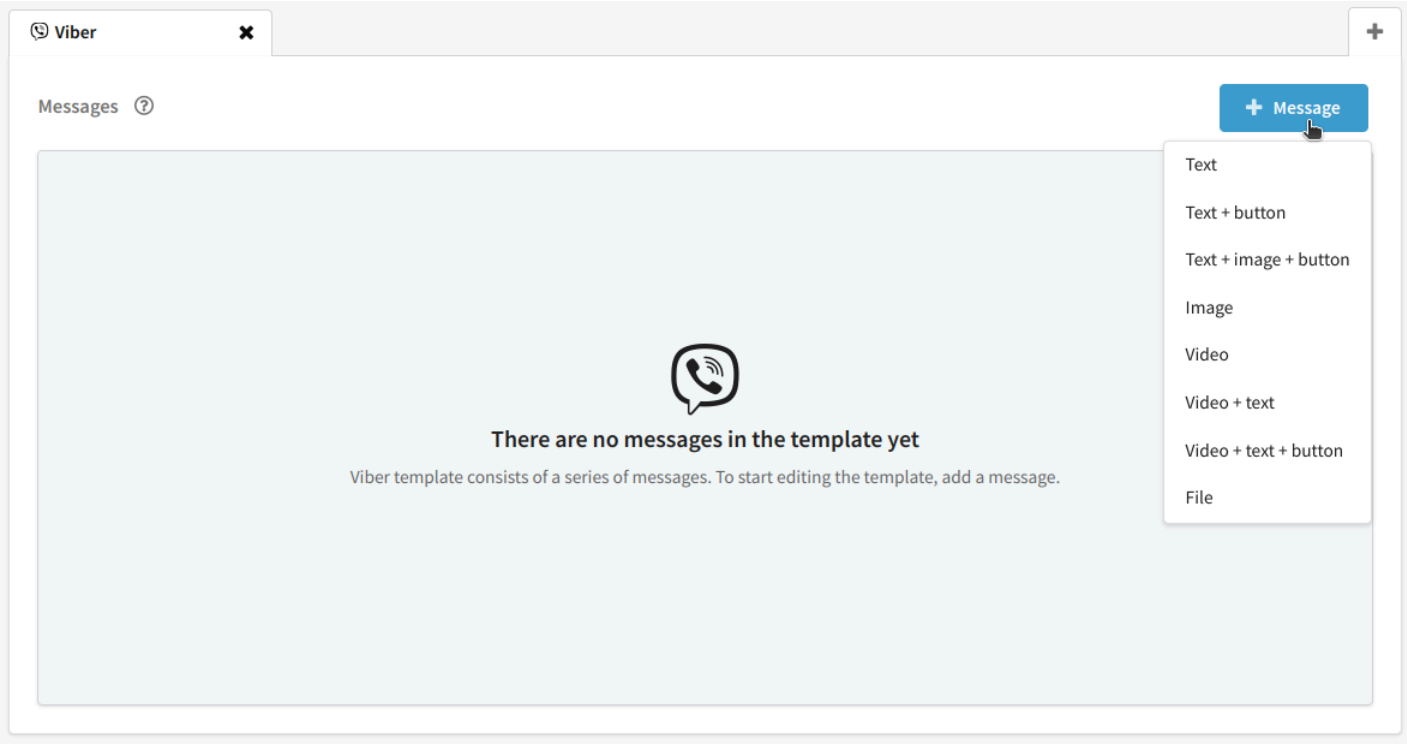


TIP

It is possible to save the template only after filling in all variables, buttons, and adding a media file.

Viber

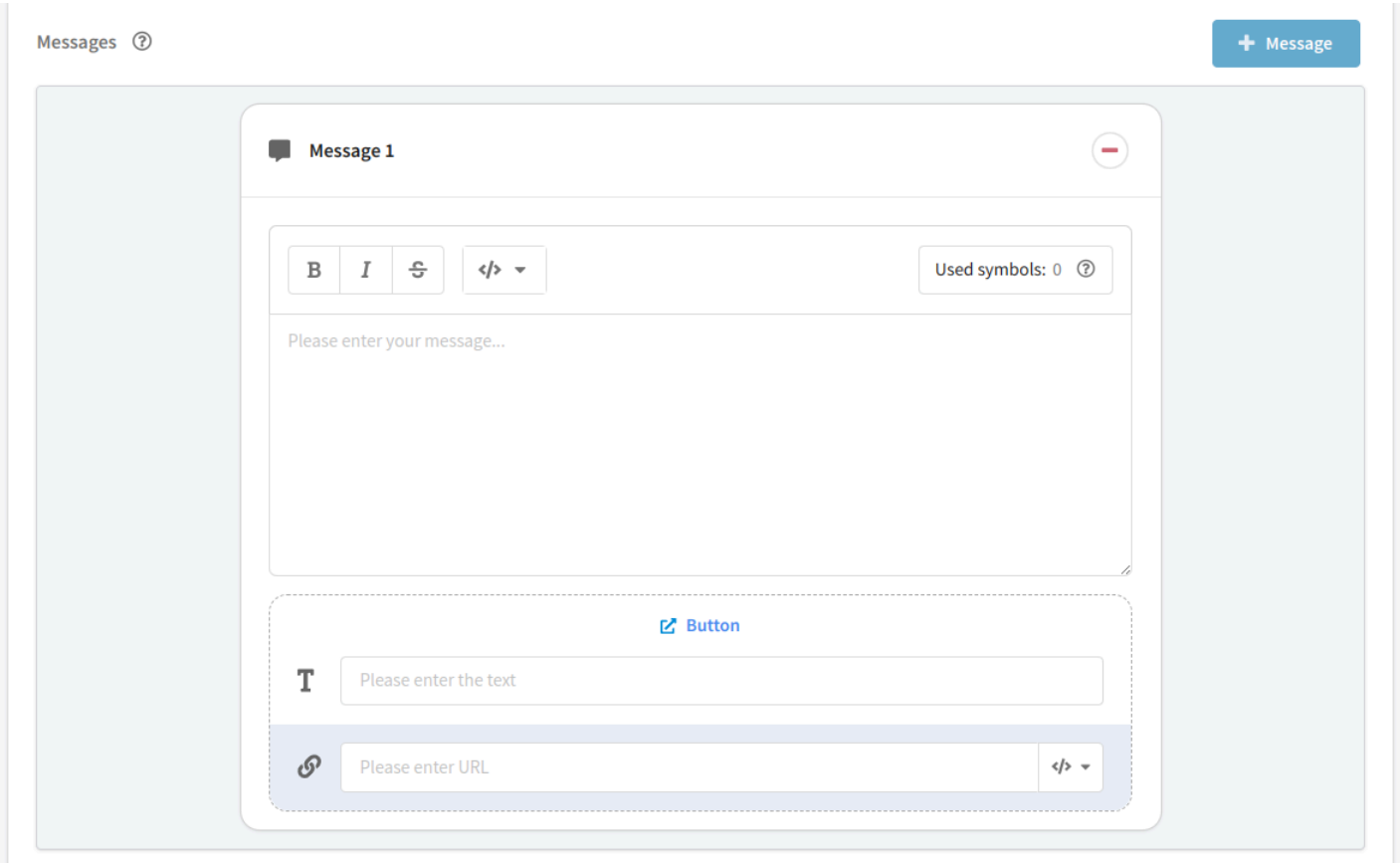
A Viber template consists of a series of messages. To begin editing a template, add a message using the + Message button.



The maximum number of messages in a series is 5. If multiple messages are used in the template, each will be sent with a 2-second interval.

To remove a message, click

.



The following types of messages are available:

- **Text**
- **Text + button**
- **Text + image + button**
- **Image**
- **Video (not available with MTS OmniChannel)**
- **Video + text (not available with MTS OmniChannel)**
- **Video + text + button (not available with SMS Traffic and MTS OmniChannel)**
- **File (not available with SMS Traffic and MTS OmniChannel)**

Text

Enter your message text in the editor using Markdown format. For convenient formatting, the following functions are available on the top panel.

B

— bold,

I

— *italic*,

~~S~~

— ~~strikethrough~~,

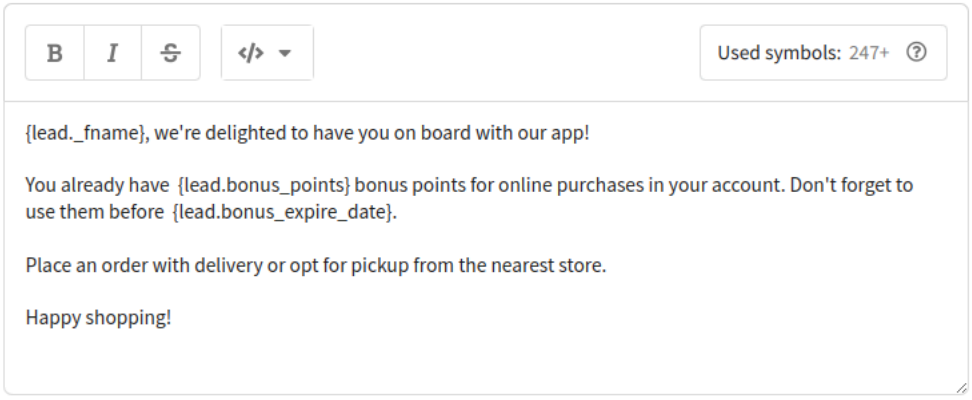
</>

— a list of variables and functions to personalize your message.

CAUTION

Please note that Markdown formatting (e.g., `"**text**`) is displayed in the preview mode, but the message recipient will see the correct formatting (text").

Also, you cannot add links in Markdown format in the “Text” block.



TIP

You can add emojis to Viber messages. To insert an emoji, you need to use its code. For example, to add 😊, enter `(smiley)`. You can find codes for all available emojis at this link.

Please note that when counting the number of symbols in a message, each symbol in the emoji code is considered.

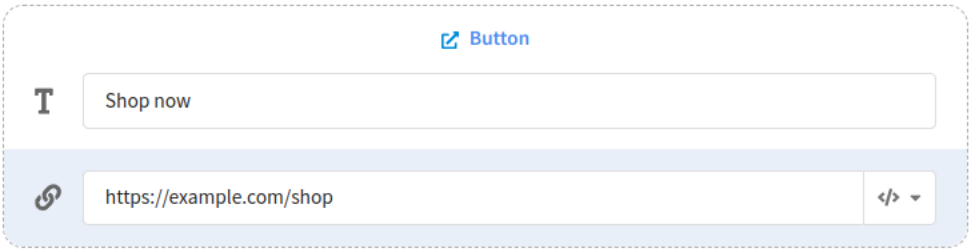
Emojis are not displayed in preview mode.

INFO

On the right-hand side, you’ll find a symbol counter. The maximum length of one message is 1000 symbols. Please note that data pulled from dynamic variables may exceed this limit. To ensure reliable delivery, we recommend monitoring the message length.

Button

Specify the text that will be displayed on the button and the link for redirection when clicked. You can also add variables to the URL.

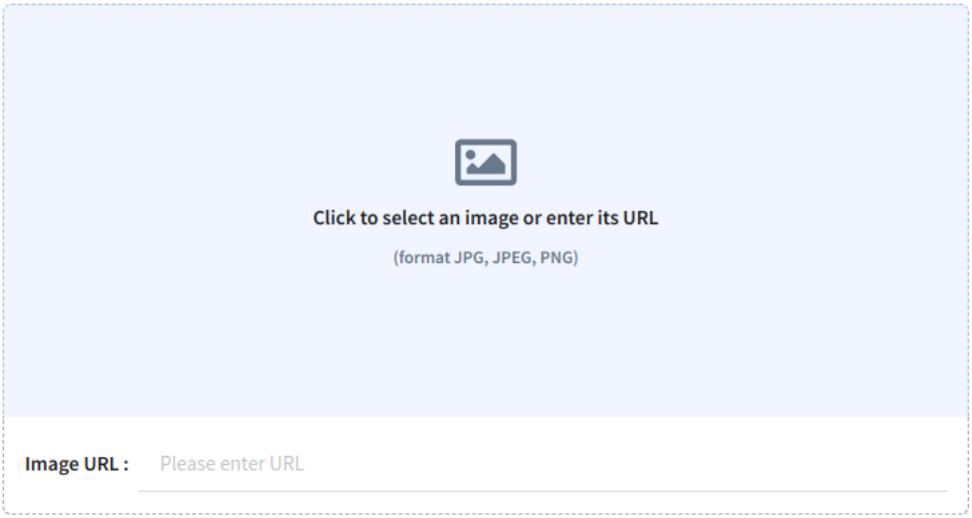


Image

Supported formats are jpg and png, with a maximum size of 5 MB.

You can add images in two ways:

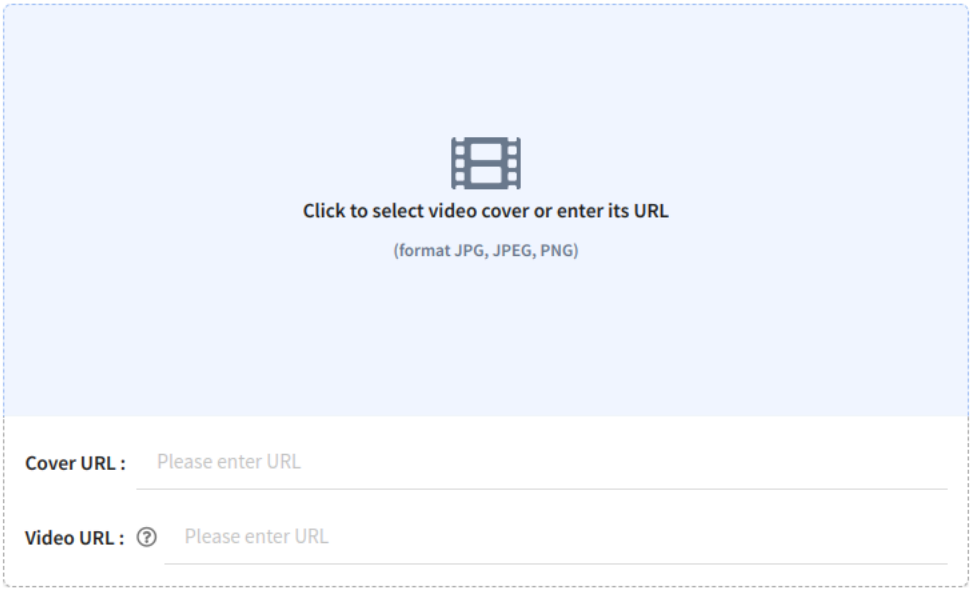
- **From the platform's gallery. Click on the upload area to open the gallery. Choose the desired image or upload a new one from your device.**
- **From an external source. Provide a link to the image.**



Video

Supported formats are mp4, 3gpp. Codec: H.264. Maximum size: 5 MB.

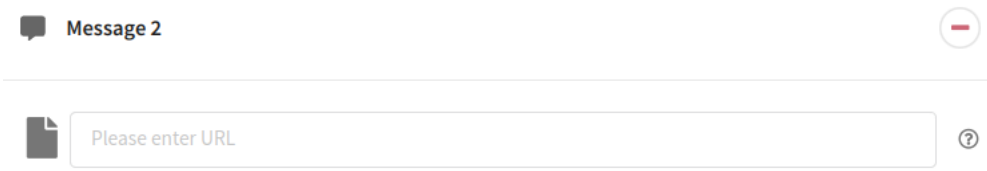
To add a video, specify its URL. You also need to set a cover for the video. You can choose an image from the platform's gallery or insert a link to an image.



File

Supported formats are doc, docs, rtf, odt, odf, txt, pdf, eps, xls, xlsx, ods, csv. Maximum file size: 200 MB. Symbol limit for file names: 25 (including the extension).

To send a file, provide a link to it.

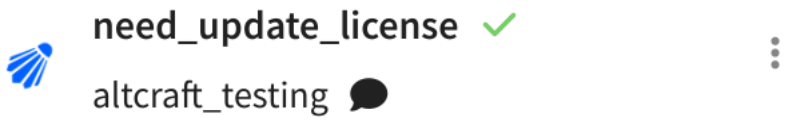


Notify

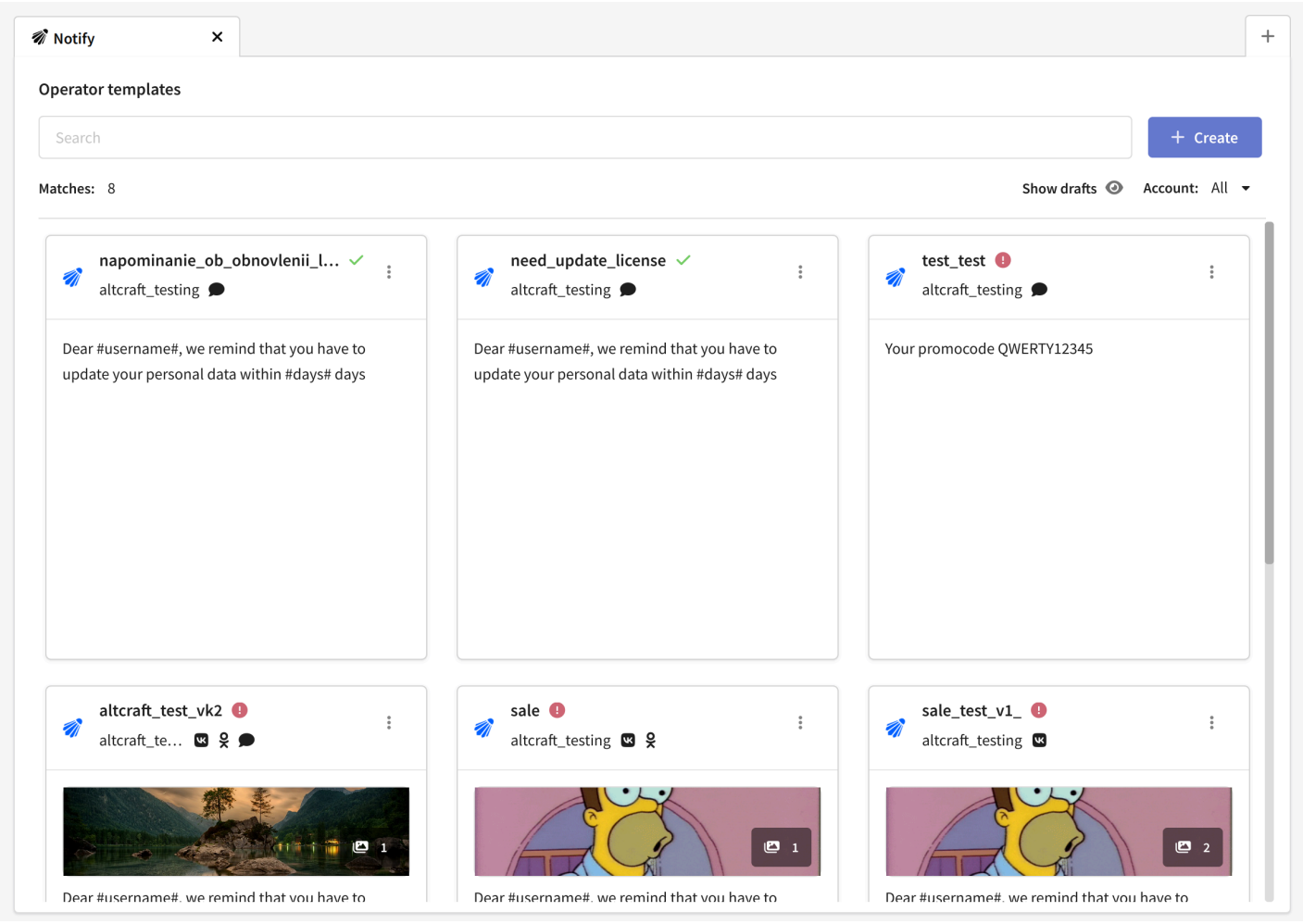
Using the Notify channel you can send messages via Vkontakte, Odnoklassniki or SMS messages. Vkontakte and Odnoklassniki messages and SMS messages are customized in two different sends that follow each other.

CAUTION

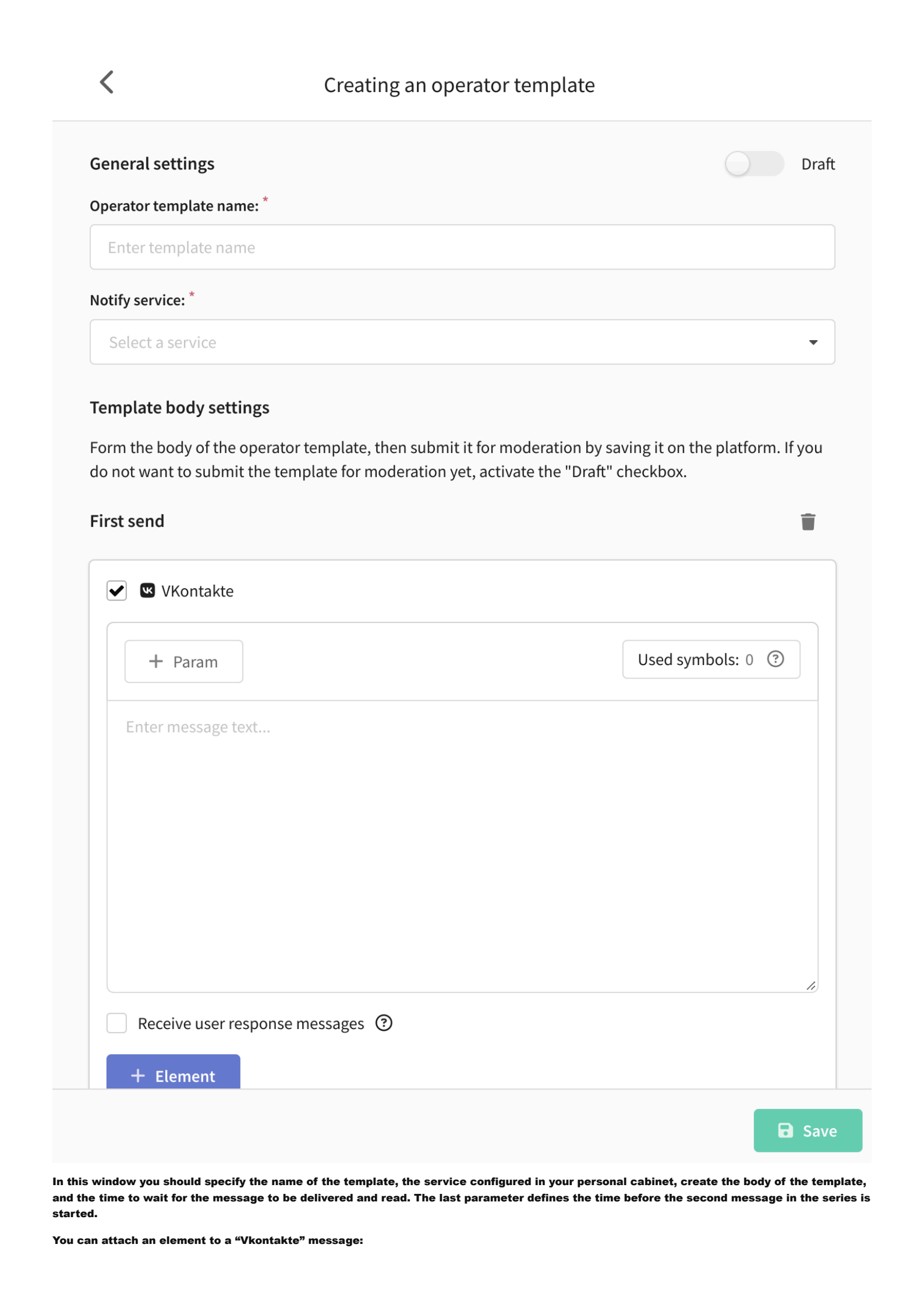
Only templates that have passed moderation can be used for mailing. They are marked with a special sign:



When you open the template editor for Notify, you will be shown the Operator templates of your account:

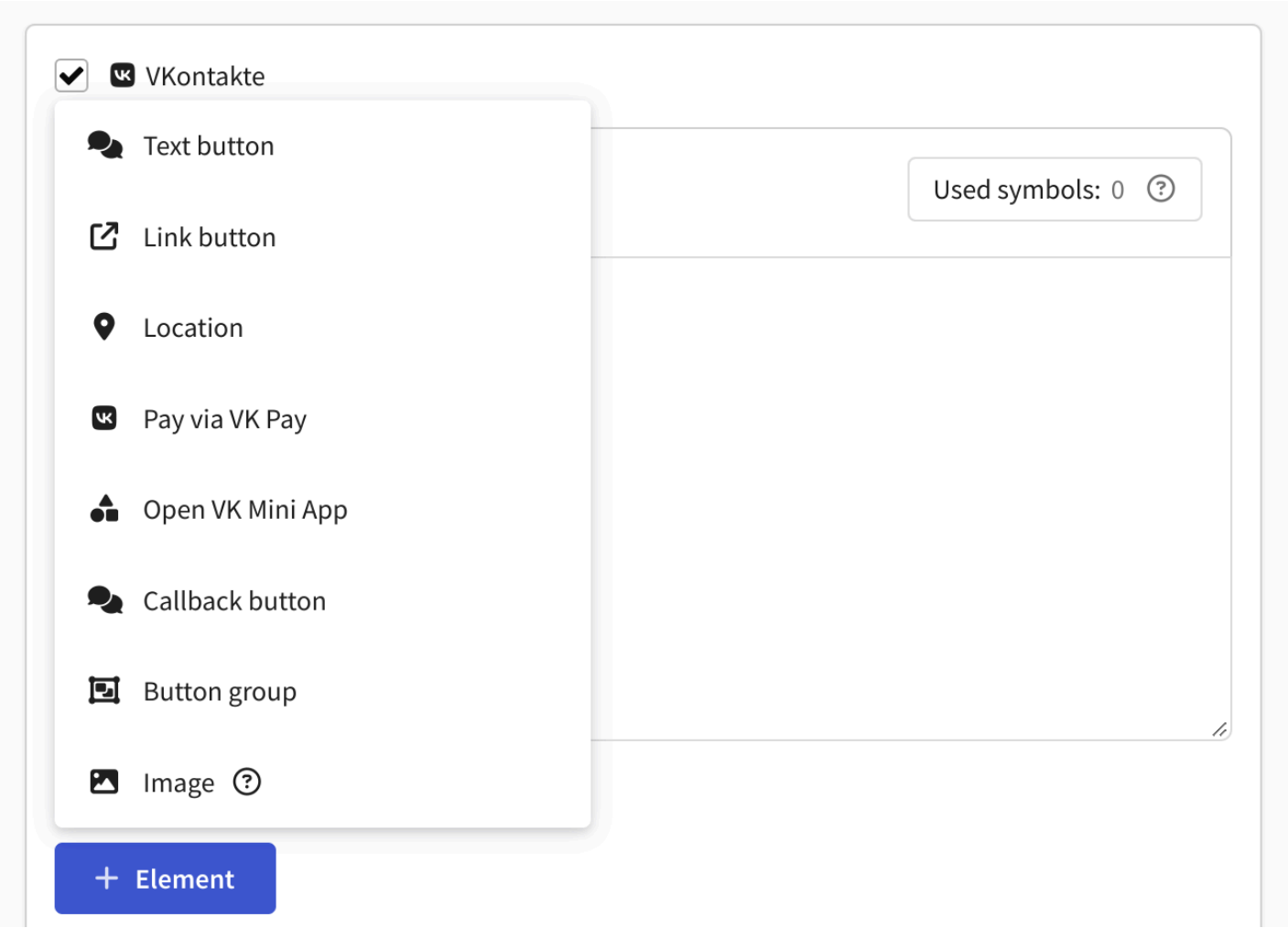


You can use one of the existing ones or create a new one. To create a new template, click the appropriate button. The template creation window will open:



In this window you should specify the name of the template, the service configured in your personal cabinet, create the body of the template, and the time to wait for the message to be delivered and read. The last parameter defines the time before the second message in the series is started.

You can attach an element to a “Vkontakte” message:

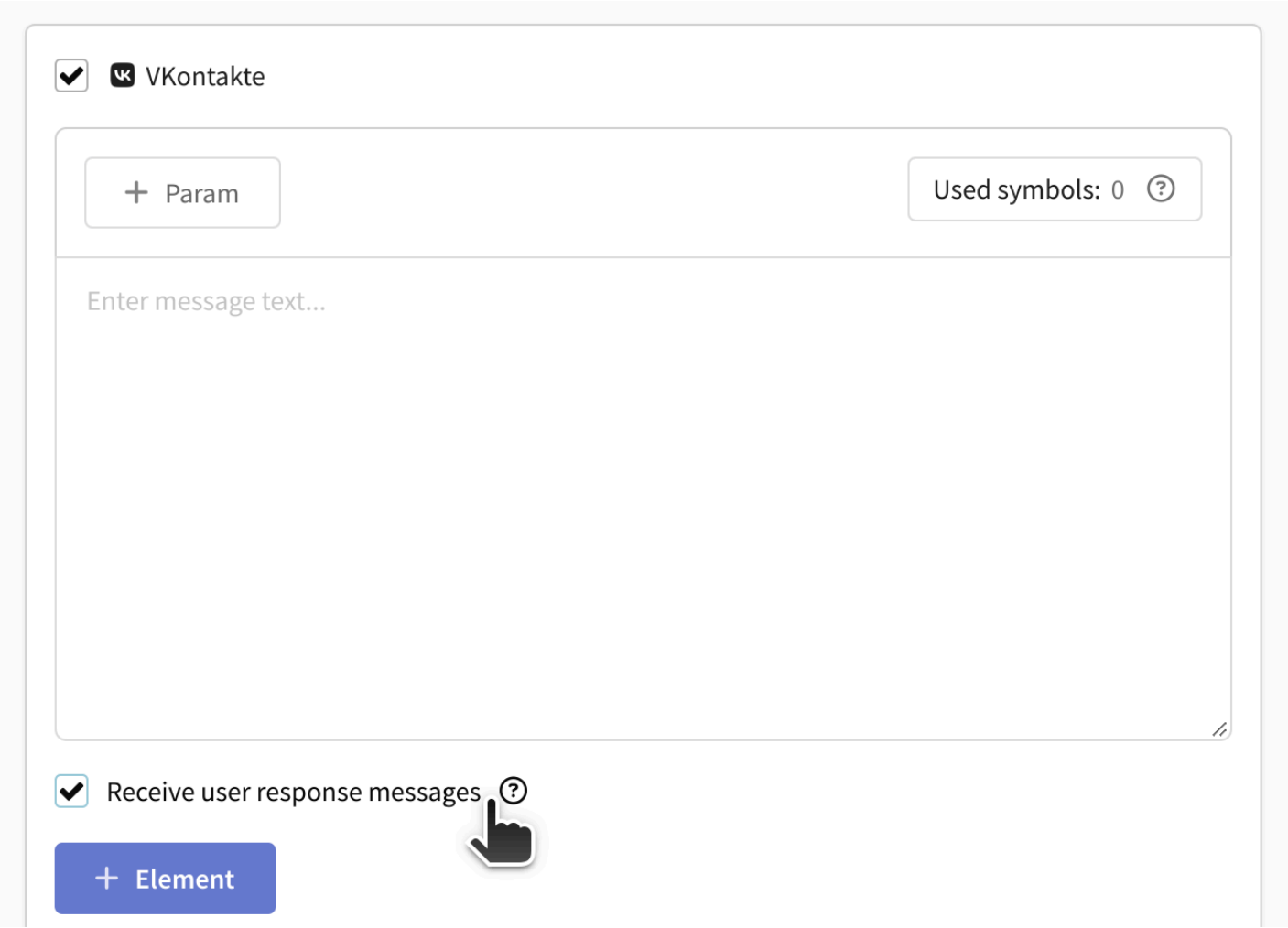


- **Text button - sends a message with the text.**
- **Link button - opens that link.**
- **Location - by clicking, it sends the location into a dialogue with the bot/conversation.**
- **Pay via VK Pay - opens the payment window VK Pay with predefined parameters.**
- **Open VK Mini App - opens the specified application VK Mini Apps.**
- **Callback button - allows without sending a message from the user to receive a notification about pressing the button and perform the necessary action.**
- **Button group - cobmination of the buttons.**
- **Image - up to 5 images with a total weight not exceeding 50 MB.**

TIP

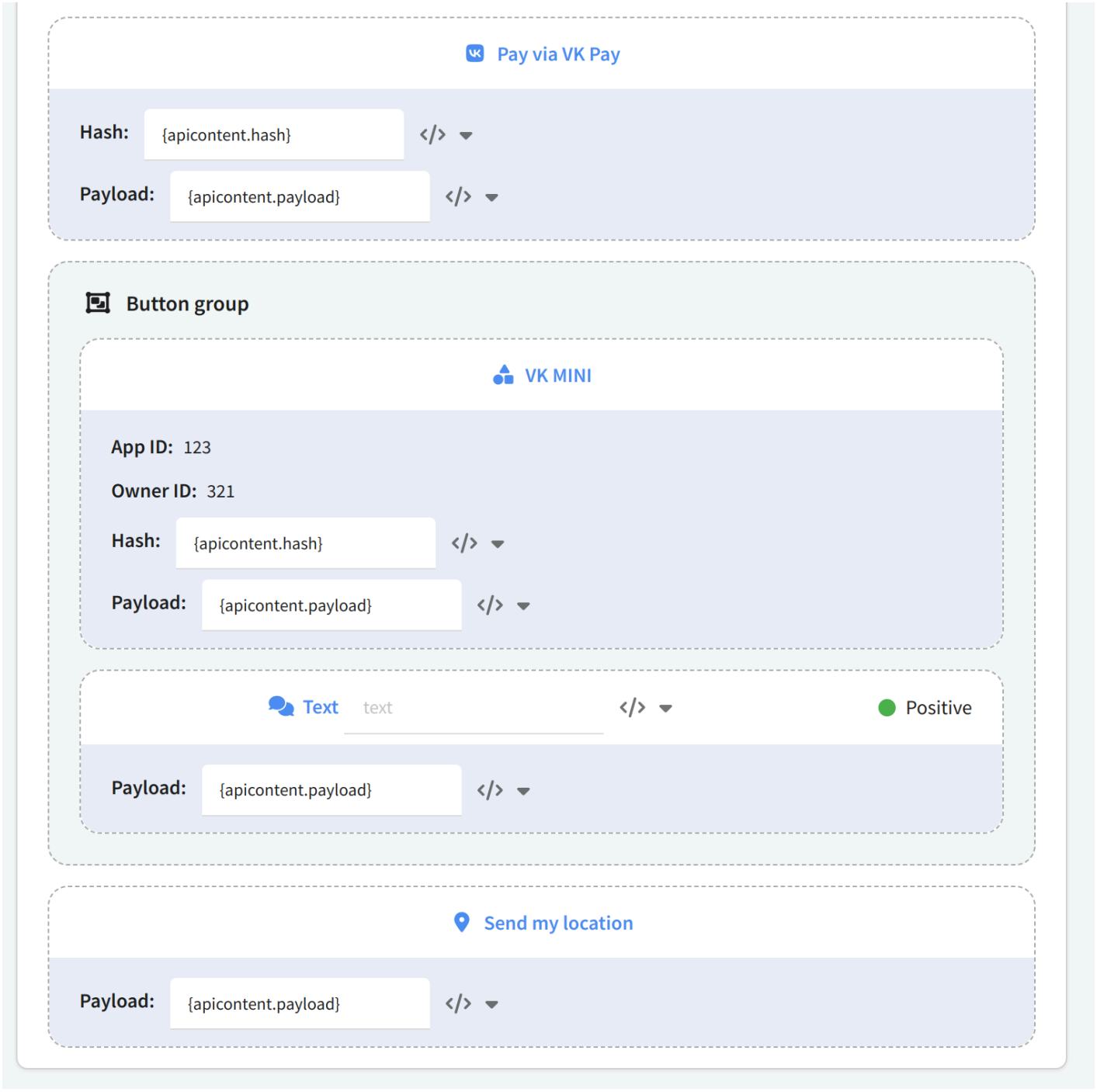
Read more about elements in Notify documentation.

You can also enable the customer to send reply messages in chat with the company. To do this, activate the corresponding item in the settings:



Once you save the template, it will be sent to moderation in the service. If you don’t want to send it for moderation, activate the Draft switch.

After selecting an approved template, you can insert platform variables into it using the drop-down menu:

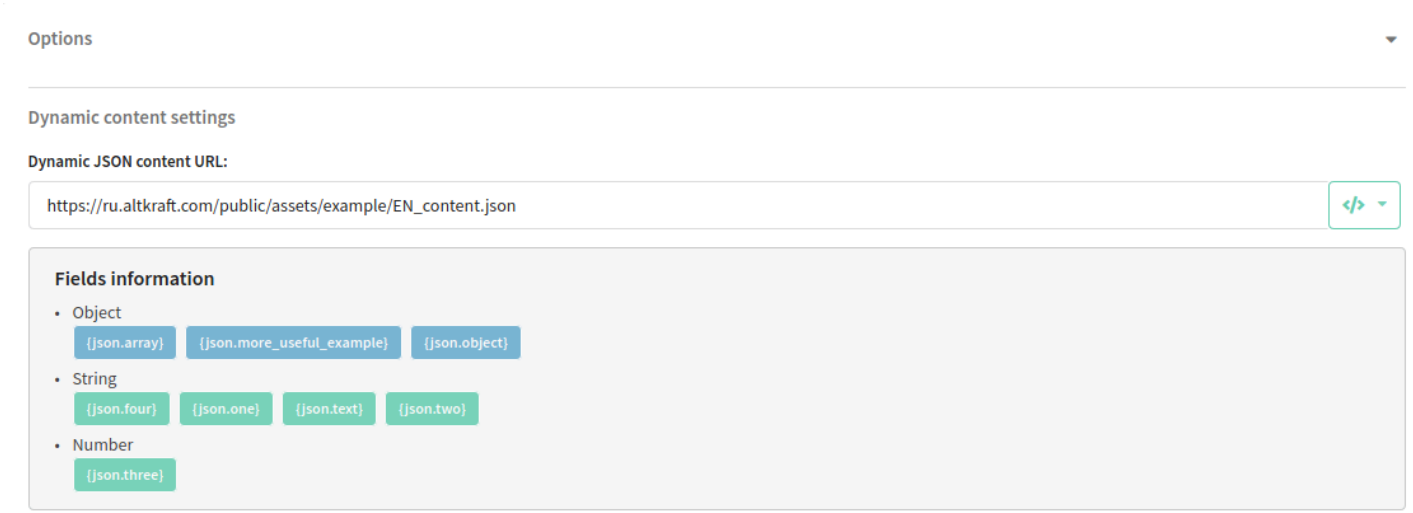


Don’t forget to save the template in the platform after making all the changes. You can now use this template to send out Notify distribution.

Message options

In Options block you can set some extra parameters for the template.

In Dynamic content settings you need to set data source for JSON content in the messages. If you have valid JSON, first level objects preview will be shown:

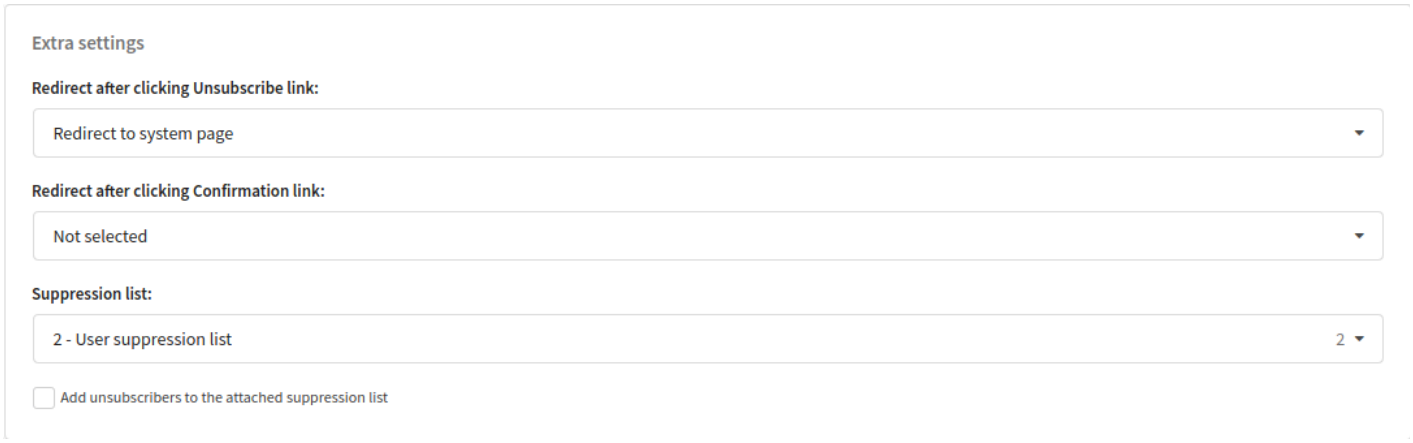


In Extra settings, you can select custom unsubscription and confirmation links for your message. For example, if you want to redirect customers to a personal page or subscriptions management page.

TIP

Custom links for unsubscription and confirmation can be created by master user of the account. You can select only previously created links in the template options. More information is available in Settings section.

Also, in Extra settings, you can select a custom suppression list, that you would like to use with this template. Using this function, you can easily restrict messaging for those customers, who unsubscribe from similar mailings. Also, you can toggle Add unsubscribers to the attached suppression list option to automate the process.



Personalization placeholders and Parts

In the Content tab you can set default values for profile variables you use in the template. If you use customer name for instance, but the field can be empty for some profiles, a default value like “Dear customer” will be used.

TIP

Content tab appears, when you use variables with profile data or parts of the message into the template.

Message Parts are similar to fragments, but they can be used only in templates wherein they were created. Fragments on the contrary can be used in several templates - e.g. for a header and a footer.

TIP

To get started with parts, add `{part.example_part}` to your template code. After that, the “Content” tab will appear at the bottom of the page, where you can customize parts of the template.

TIP

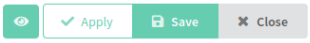
It is convenient to use parts with logical expressions for more compact message code. For example, here one of the parts will be selected, depending on customer’s score (custom integer field):

```
{if lead.score gte 9000}
{part.part_1}
{else}
{part.part_2}
{end}
```

For every part, you can choose content type: HTML for email messages, or text — for text version of email, and for Push and SMS content. Depending on content type, appropriate editor with necessary functions will be opened.

Message preview

Message preview allows you to check HTML layout, and to see content personalization result, based on customer data. Press this button to activate preview:



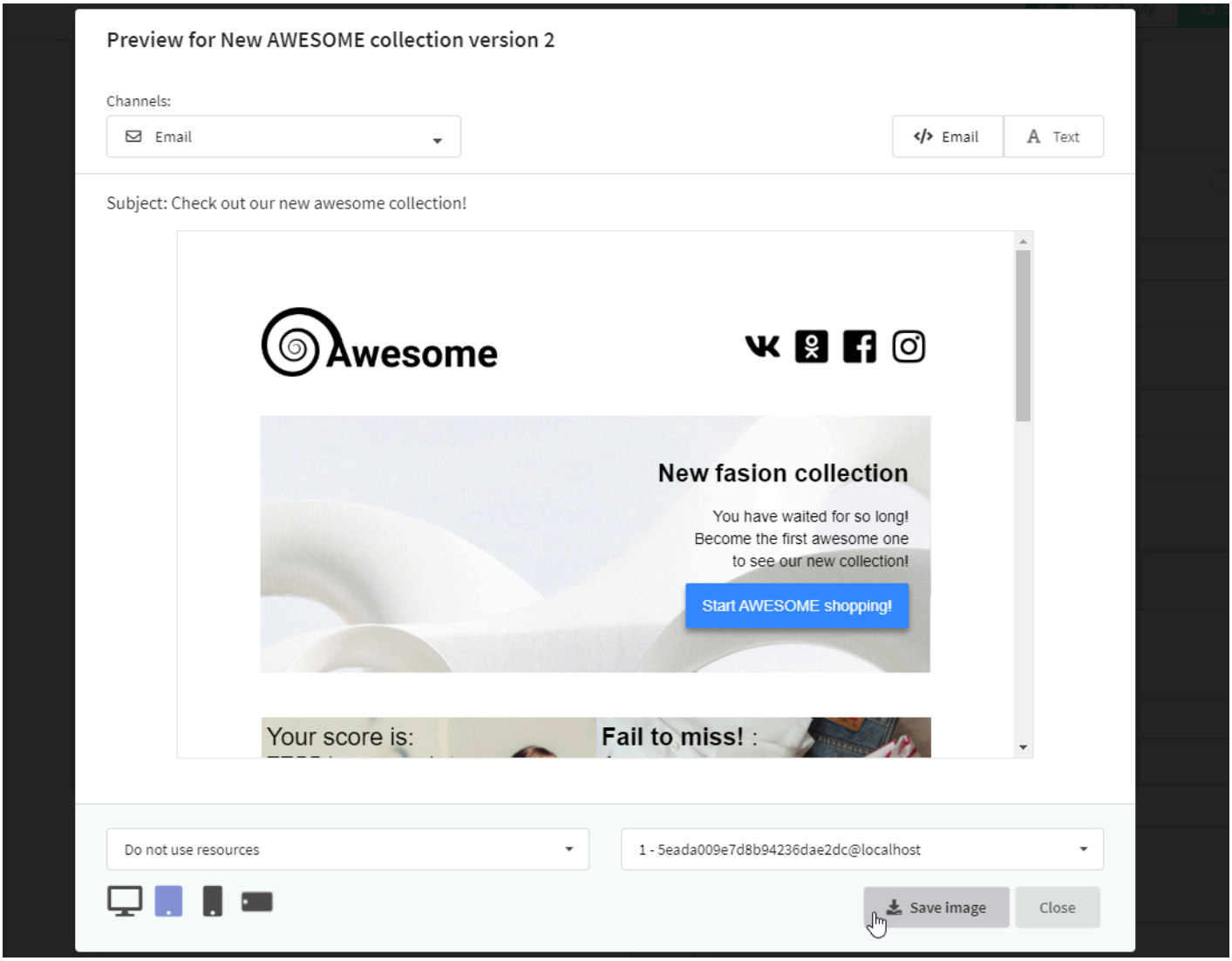
In preview window, you can select communication channel, resource and customer profile you want to use for preview. For email both HTML and text version preview is available.

In the bottom part of the page, there are buttons for switching desktop/mobile display mode, and personalization options.

TIP

Choose the resource, within which you want to send the message - and select the profile in the drop-down list. The search is similar to search in the database: segmentation rules are used, and you can immediately choose the profile with necessary data.

The preview is available for desktops, tablets and cellphones both in portrait and landscape orientation:



You can save previewed message as an image to use it in any way you need. To download the image to your device, click the Save image button.

Follow these rules for correct email template preview export:

1. **CSS styles such as `text-decoration`, `text-decoration-line`, and `text-decoration-color` should be set directly in the parent text element.**

```
<div>
    <span style="text-decoration: line-through;">Text</span>
</div>
```

2. **If you are using fonts imported from an external source (e.g., Google Fonts), it is recommended to replace them. Set the `font-family` property to fonts installed on the system.**

```
<!-- Correct display in the preview: -->
<span style="font-family: Roboto, Arial, Tahoma, sans-serif">Text</span>

<!-- Incorrect display in the preview: -->
<link href="https://fonts.googleapis.com/css2?
family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap" rel="stylesheet">
```

---

* ***The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.***

Template fragments

Description

	Fragments are message blocks, which are used throughout several messages. For example, a fragment can be used as header or footer in the message, or block with social networks links — any content, that you are going to send within several mailings.

TIP

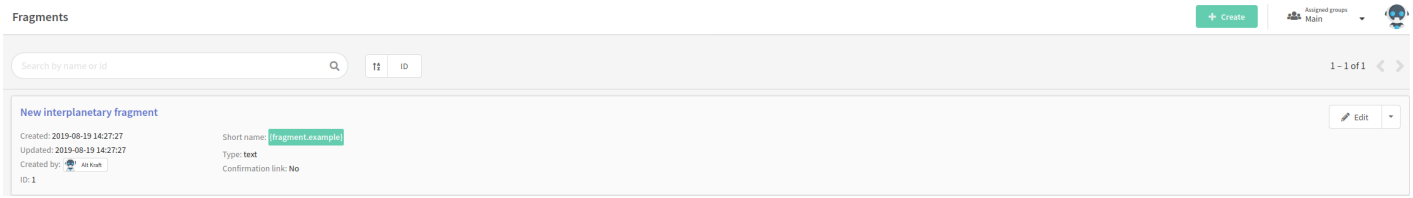
Fragments are similar to template parts, but they can be used throughout several templates, when parts can be used only in the template where created. You can read more about parts in templates managing article.

For example, emails often contain links to social networks pages:

Fragments have been developed to make repeated message sections easy and fast to use — and thus to avoid mistakes.

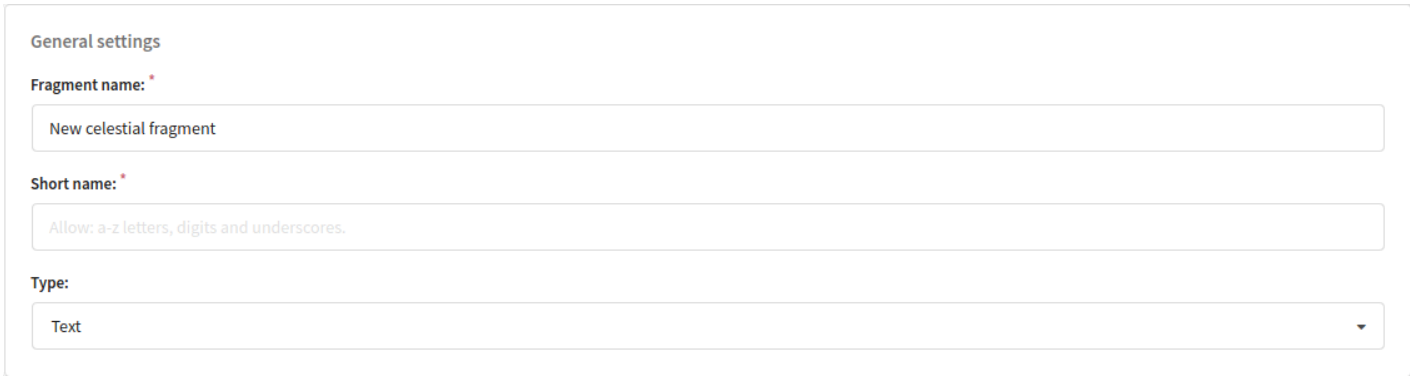
Managing fragments

Select Templates —> Fragments in the main menu to start working on fragments. A list of existing fragments will open. Using context menu, you can edit, clone or delete the fragment, — or set access group for it. If you need a new fragment — press + Create at the top part of the page:

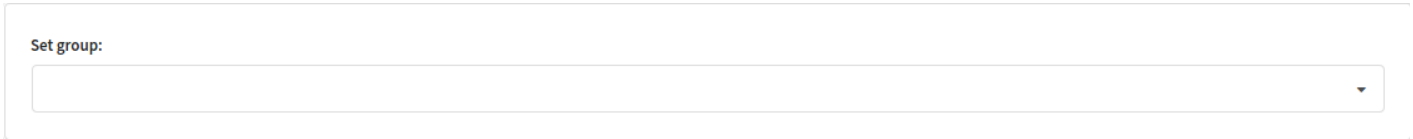


General settings

In general settings, you need to set Fragment name, that will be shown in the list, and Short name. Short name is used when you insert fragments into the templates manually, like `{fragment.<short name>}`, or using variables menu (`</>`). Also, you need to select fragment Type: HTML or text:



At the bottom part of the page, there is a block, using which you can set access group for the fragment:



HTML fragments

HTML fragments are suitable for email messages, and custom channels that support HTML. The editor supports HTML and CSS, and personalization variables. To make formatting and working with objects easier, next functions are available:

Formatting:

— bold

— *italic*

— ~~strikethrough~~

— bullet list

— numbered list

— table

Service functions:

— pixels for tracking of letter opening and reading. If you do not add pixel during message editing, the system will warn you about this.

— external HTML content. You can also set variable source address manually `{[http://example.com]}`.

— links builder.

utm

— UTM tags.

— images.

— social buttons.

— variables and editor functions menu. Using this button, you can add Customer profiles' data, promo codes, loops, logical expressions, dynamic content and service parameters into the message.

```
  .fa instagram {
  background: #f09433;
    background: linear-gradient(45deg, #f09433 0%,#e6683c 25%,#dc2743 50%,#cc2366 75%,#bc1888 100%);
  color: white;
  }
/* Container */
  .cont {
  padding:3rem;
  background: linear-gradient(to right, #00d2ff, #3a7bd5);
     }

</style>
<!-- Add font awesome icons -->

<center class=cont>
  <h2>Follow us in social netwoks</h2>
  <center>
<a href="#" class="fa fa-facebook" name="link1"></a>
<a href="#" class="fa fa-twitter" name="link2"></a>
<a href="#" class="fa fa-vk" name="link3"></a>
<a href="#" class="fa fa-instagram" name="link4"></a>
  </center>
</center>
```

Plain text fragments

Text fragments can be used in all types of content. The following functions are available in the editor:

— bullet list

— numbered list

— links builder.

— [variables and editor functions menu]() .

Using fragments in templates

Fragments can be added into templates using variables menu (</>). Also, you can add them manually using {fragment.<short name>}. Fragments can be displayed depending on some conditions and subscriber data.

TIP

If you want to display one of several fragments in a random order, choose Random fragment and specify the set of fragments you want to use:

```
{randomfragment[fragment.header_1 fragment.header_2 fragment.header_3]}
```

Image gallery

Description

Gallery provides an opportunity to save and organize images for using while working with marketing templates:

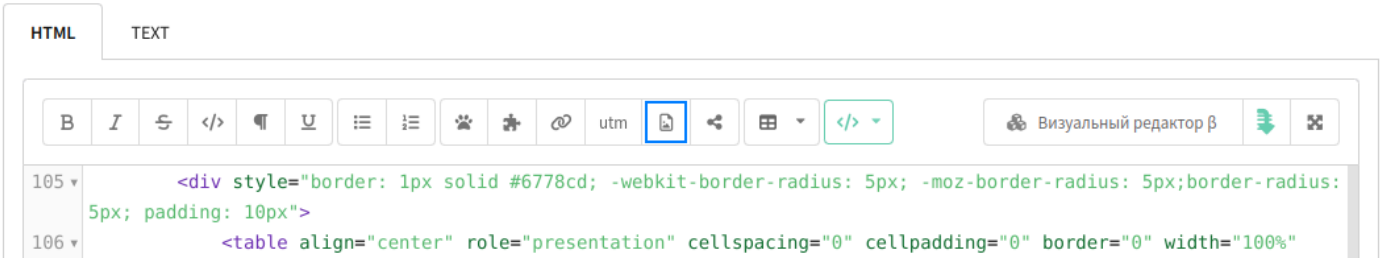
- **HTML and AMP versions of Email templates;**
- **Push templates (banners and icons).**

Images are loaded into messages and pop-ups from the platform server through the tracking domain.

Work in the gallery

To open the image gallery, go to the template editor and click on

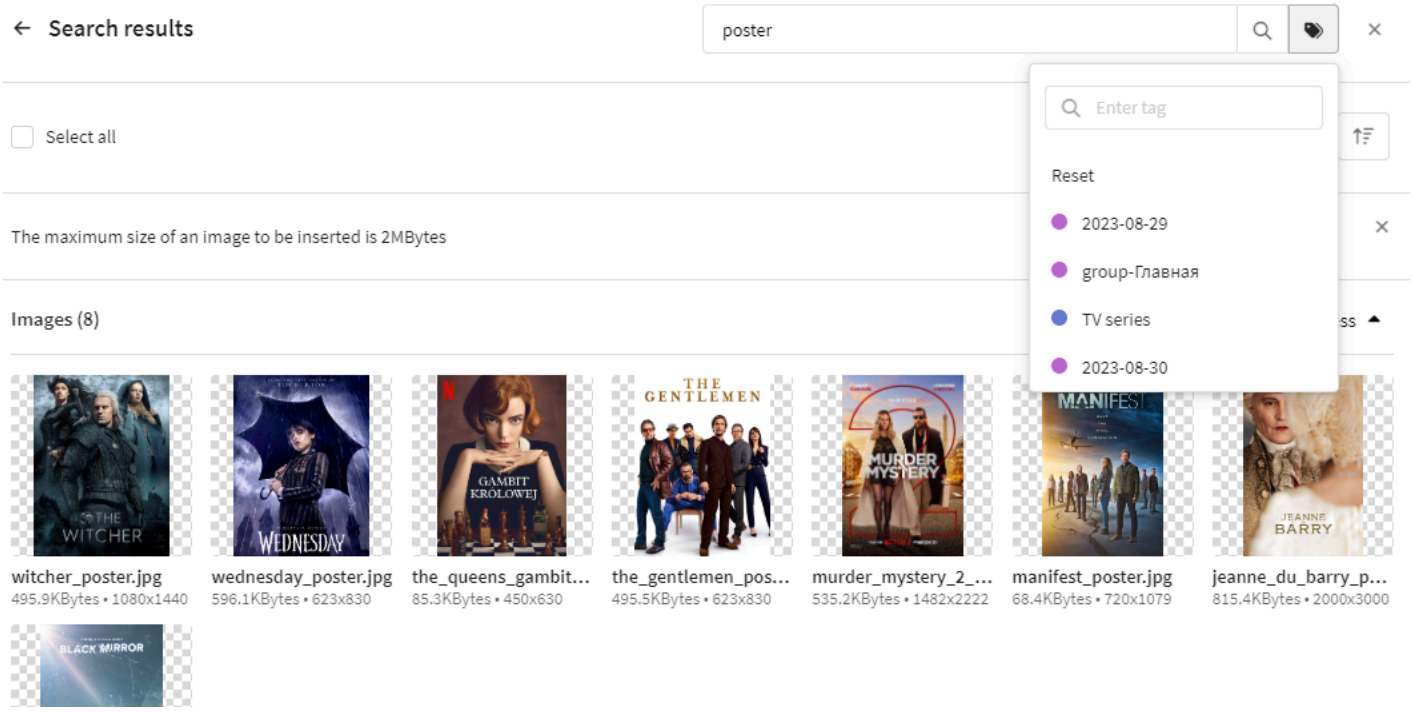
the icon. For example, in the Email message editor, you can go to the gallery using the button in the toolbar:



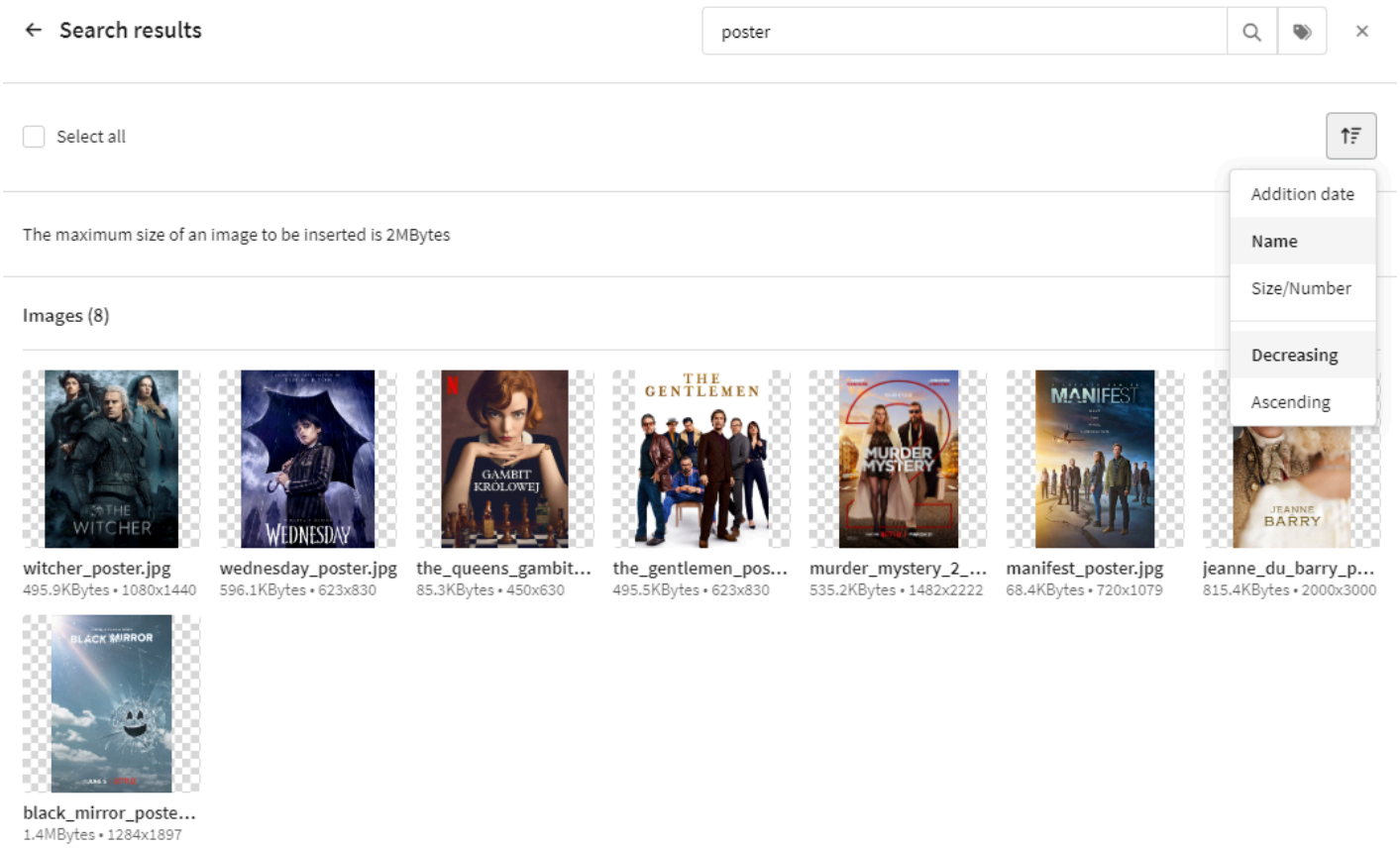
The gallery is a modal window, visually divided into 3 blocks:

- **recent (all images added in the last month);**
- **directory;**
- **images.**

You can search by file/directory name or attached tags. To exit search mode, press ← Search results:



You can also sort objects in the gallery by date added, by name, or by size:



You can select one, several, or all objects in the gallery and do one of the following:

— move the object to another directory.

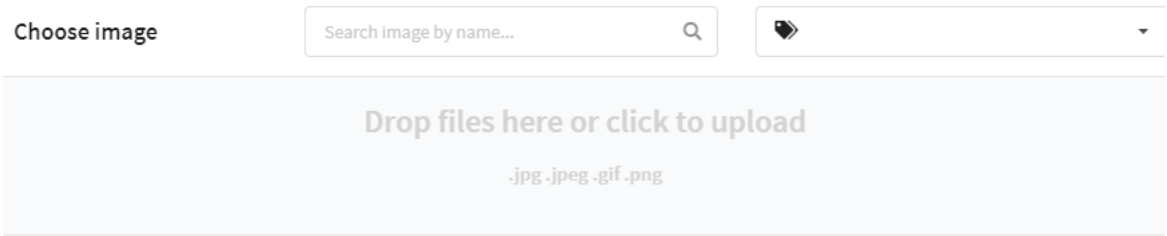
— download an object (only for images).

— delete an object.

Working with images

Uploading an image

To add an image to the gallery, tap + Image and select the file on your device. The second way is to drag the image into the gallery area:



NOTE

You can upload multiple images at once. The size of one image should not exceed 2 MB. Available formats: jpeg, jpg, png, gif.

The upload status is displayed in the lower block. You can interrupt upload by clicking on the cross. If you encounter an error while adding an image, hover over the question mark to find out the reason.

If you close the gallery before all files have been imported, the import process will be interrupted, but all files that have been successfully uloaded to that point will remain in the gallery.

Actions with images

To work with downloaded images, move the cursor over the image and click

. A list of available actions will open:

— insert a link to the image into the template. Convenient when you don’t need to change the formatting, but change the src attribute of the img tag.

— insert an image into the template in HTML format (new img tag).

— download the image to your device. Images are uploaded to a zip archive.

— move the image to another directory.

— rename the image. When you change the name, the link to the image does not change.

— remove an image from the gallery.

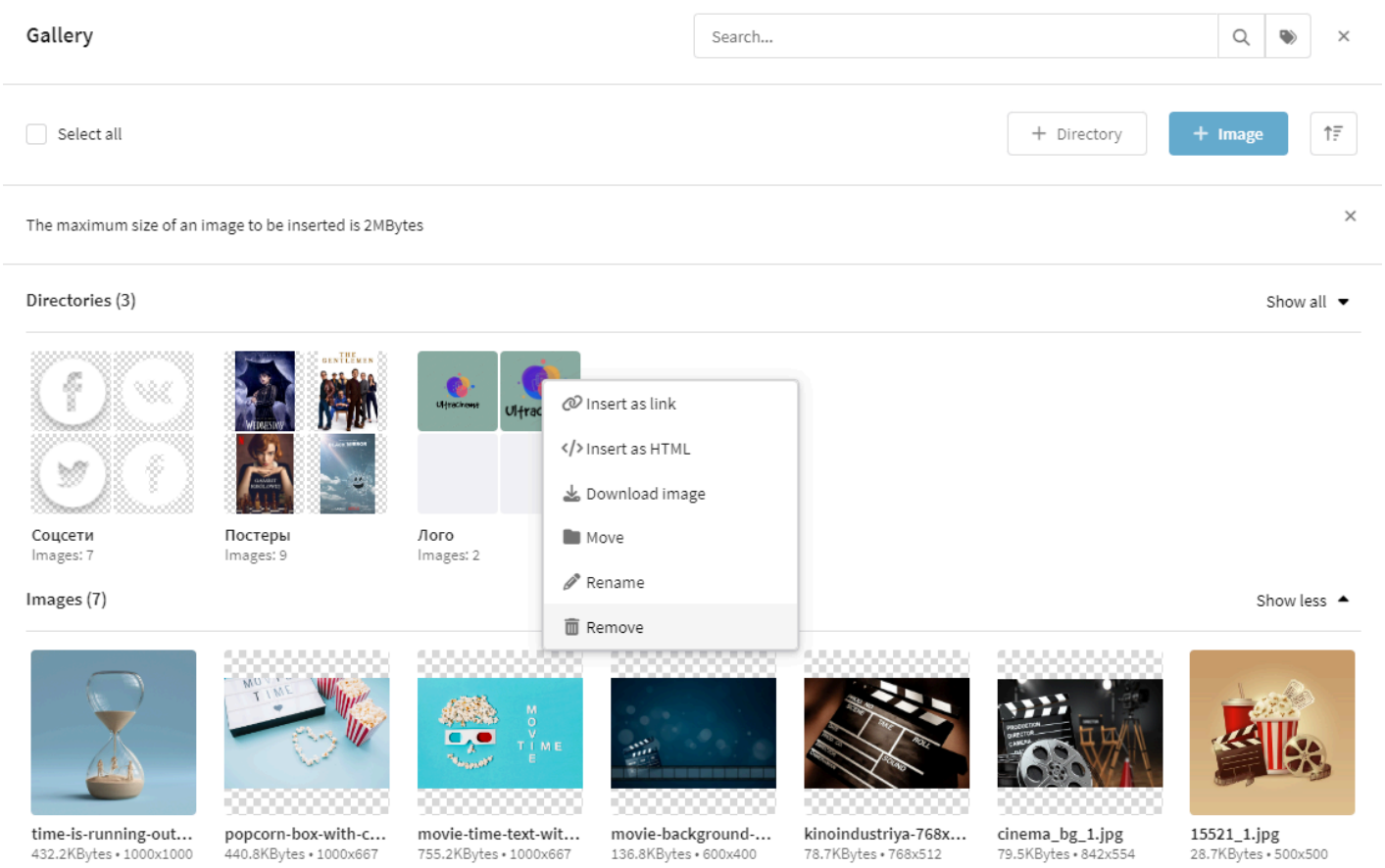
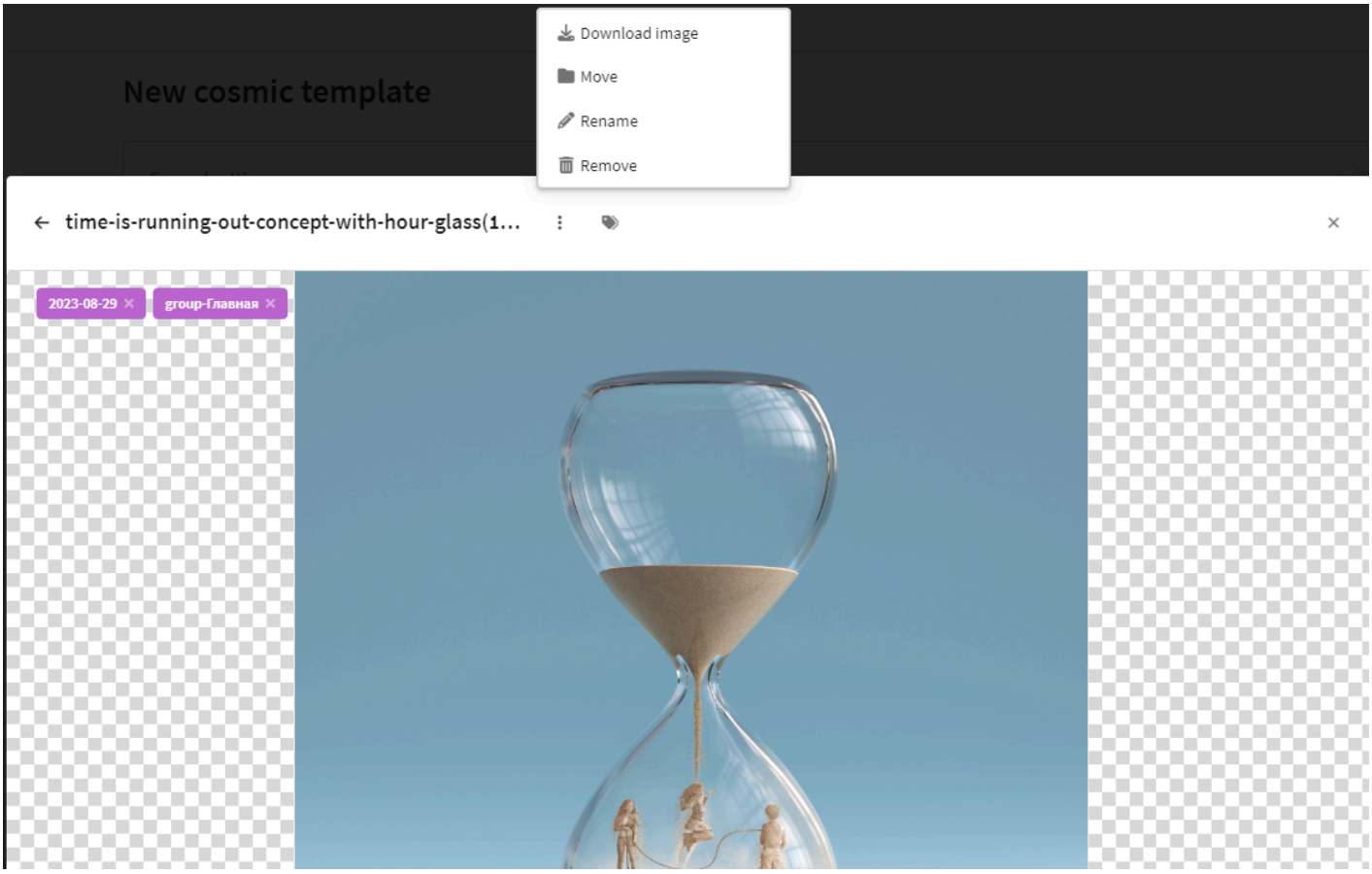


Image view mode

If you want more information about image, click on it. You will enter image viewing mode. All the actions, which you have known already, are available here. To open the list of actions, click on

:



You can also copy URL link of your image by clicking on Copy

. This link can be used, for example, when setting up an icon or banner in a push template.

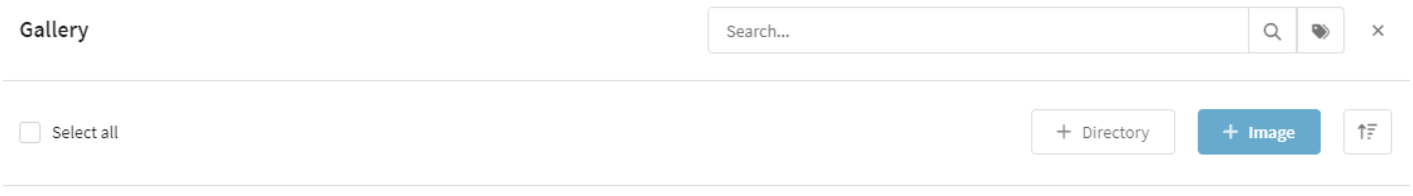
Besides, in this mode you can view tags attached to the image, as well as add new ones or remove existing ones. Some tags are created automatically, for example:

- `2023-10-25` **(date tag when an image was added to the gallery);**
- `group-Home` **(name of the group in which the user worked at the time the image was added).**

To exit image viewing mode, press ← <image_name>.

Working with directories

To more conveniently organize images in the gallery, use directories. To create a new directory, click "+ Directory":



Enter a name for your directory. You can use any characters in the name, but no more than 244.

To work with a directory, move the cursor over it and click

— a list of available actions will open:

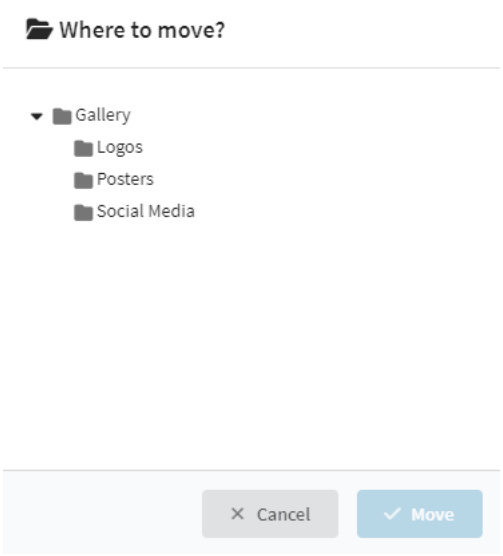
— move a directory to another directory. You cannot move a parent directory to a child directory.

— rename the directory. The directory name must be unique.

— delete the directory. All items in the directory will also be removed from the gallery.

If you want to upload images to a specific directory, go to it and click + Images or drag and drop files into the gallery area.

To move already added images to a directory, select the images, click Move in the list of actions and select the desired directory:



Sub directories is also available. To create a sub directory, go to the parent directory and click + Directory. The level of nesting is not limited.

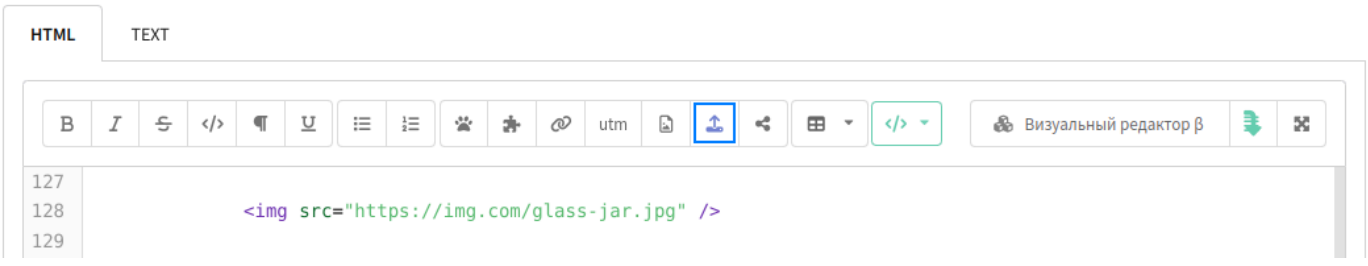
To return to the previous directory, press ← <directory name>.

Uploading images from external sources

Often templates use links to images from external sources. For example, if you designed a template in a third-party service and then transferred it to the Altcraft platform. In case of failures in the external service, such images may become unavailable. To avoid this, it is recommended to upload them to the platform's gallery.

Click on the button

in the editor toolbar (the button appears only after adding the URL to the external image to the `src` attribute):



After uploading to the platform server, the value of the `src` attribute for all external images will change, for example, `<img class="img" src="https://img.com/glass-jar.jpg">` will be replaced by `<img class="img" src="http://{trkdomain}/383848228/1/64eefca84a8f9017e5/glass-jar.jpg">`. Now these images will be loaded into messages by the tracking domain.

Content personalization

Description

Customer profiles data can be used for creating dynamic personal messages, that do not need constant changes. The template will take profile information from the database. It allows you to make personal human-like mailings.

TIP

For example, this is the message with a daily horoscope:

Hello, Eddie!

Daily horoscope for Scorpio.

This is the greatest day to consider your options, study economic trends, and analyze what you want from investing. You can create this message, using profile data and the link to a server generating horoscopes. This is called dynamic HTML content.

```
Hello, {lead._fname}!

Daily horoscope for {lead.Sign}.

{[http://127.0.0.1/horoscopes?sign={lead.Sign}]}
```

This way, you will make one letter for all 12 zodiac signs, that does not require any time afterwards.

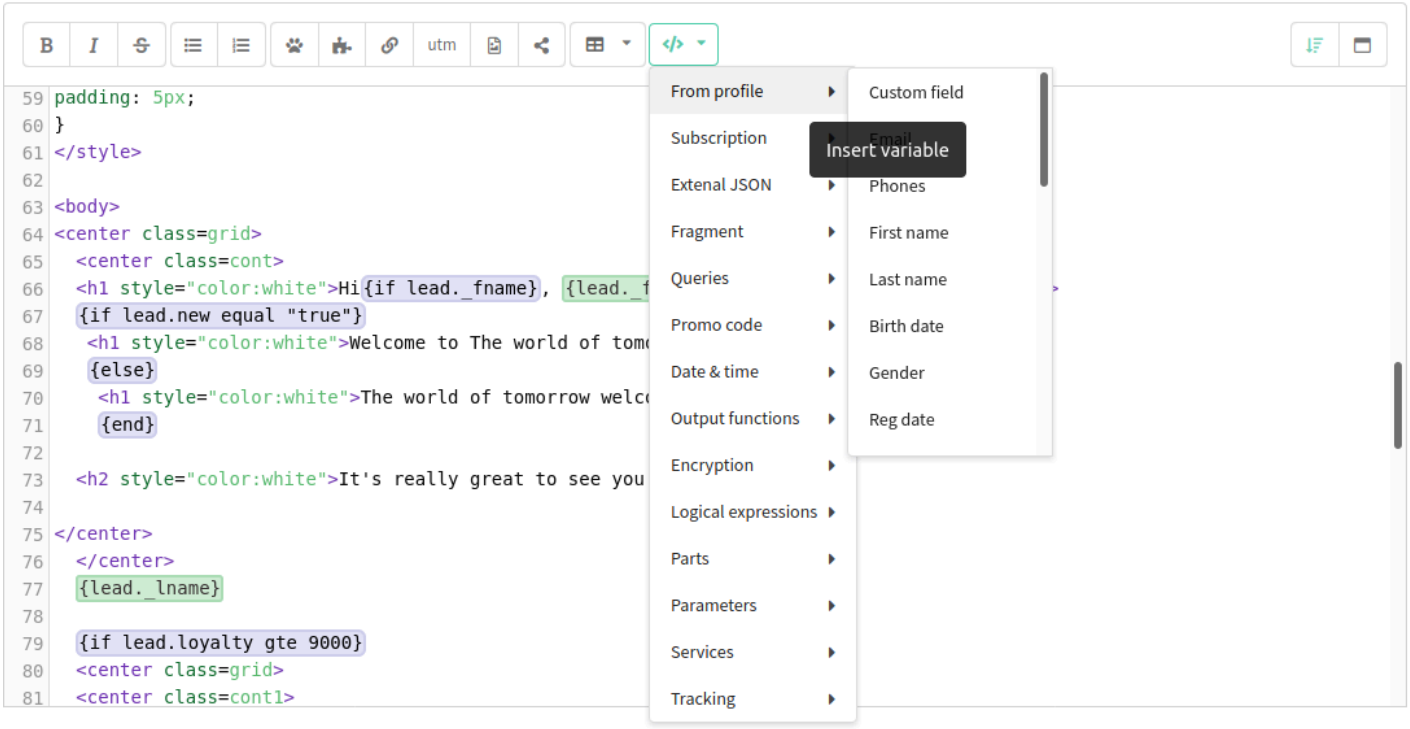
Customer data in message templates

To add data from customer profiles, open variables menu (`</>`) in template content editor, and select one the following:

- **From profile — customer data from profiles database.**
- **Subscription — information about profile subscriptions to different resources.**
- **Promo code — promo code from a certain loyalty program.**
- **Parameters — additional data, like Database, Mailing, Template, Resource name.**

To add data from additional (custom) fields of the client profile, add `{lead.Custom_Field_Name}` into the code, where `Custom_Field_Name` is the name of the additional field that matches the one in the profile database. In our example (see above), it is `{lead.Sign}` .

You can also add data from the additional field of the "Object" type. Nesting can be of any type. For example, `{lead.Custombject.fieldname}` .

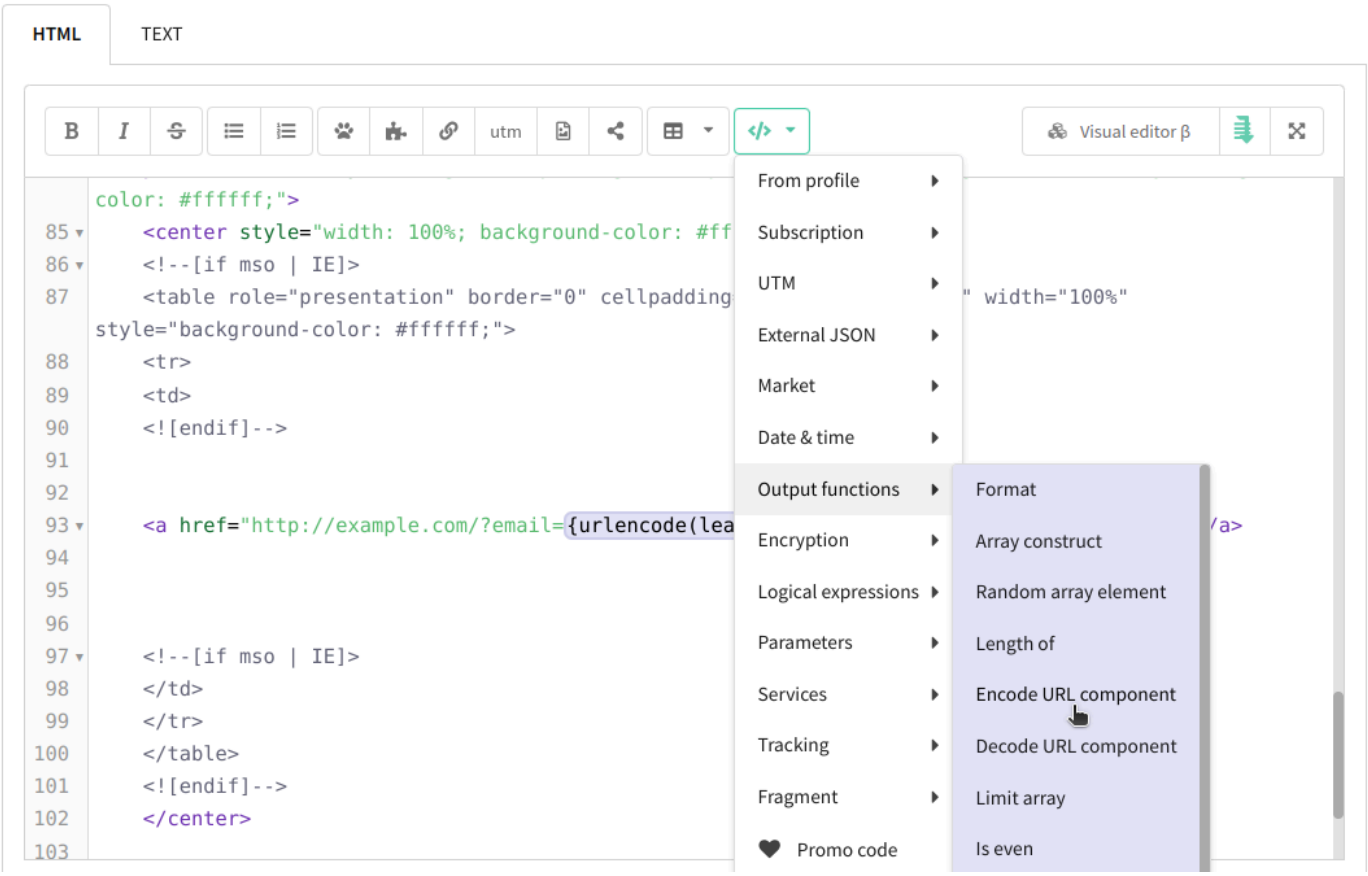


Using variables in URL parameters

URL addresses only allow Latin letters, numbers, and a limited set of special symbols. However, when using variables that pull data from profile fields into a template, certain characters may be invalid for URLs, such as spaces or non-Latin characters. In such cases, the link in the template will not work.

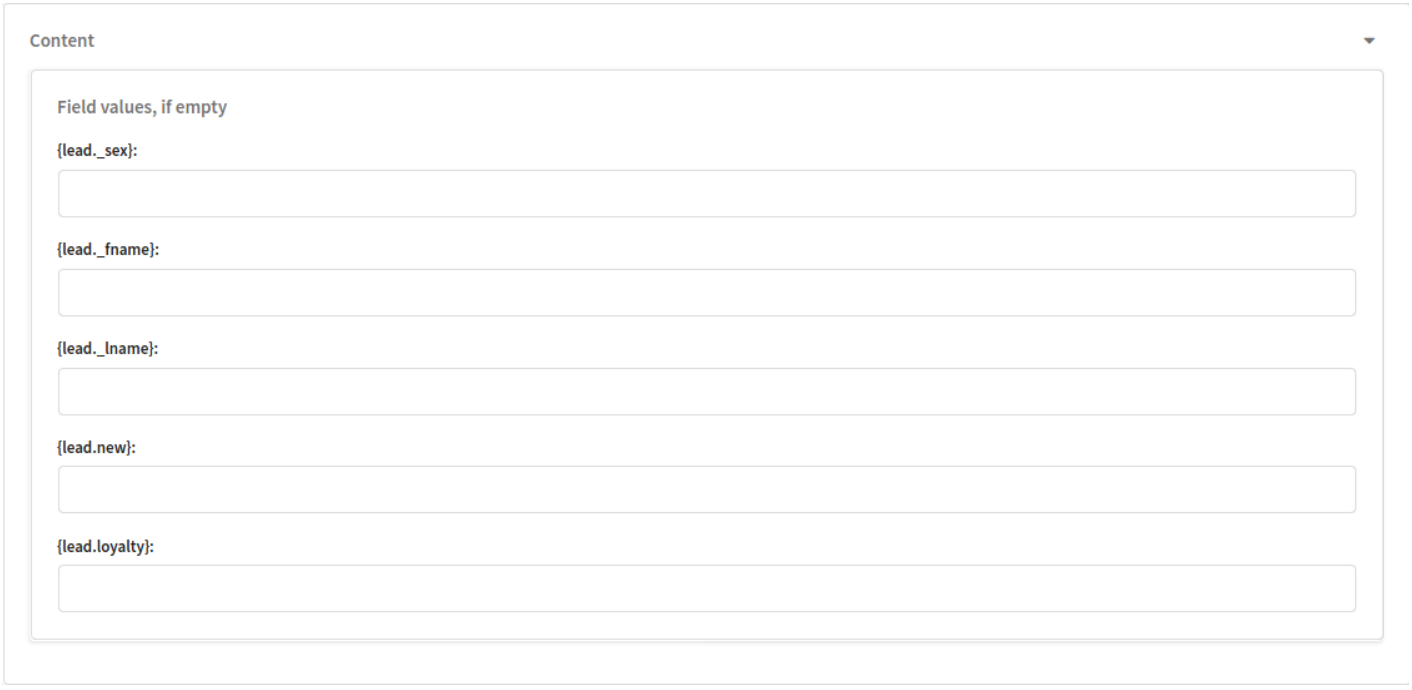
To avoid this situation, it is important to encode the parameters in the URL. This can be done using the `{urlencode}` function. For example, if you're inserting the variable `lead.email` into a link, use the following format:

`https://example.com/?email={urlencode(lead.email)}`.



Placeholders for missing data

In the template editor Content tab you can set default values for the profile variables used. They will be inserted when there is no necessary data for a customer.



On-click customer data update & subscription

Customer data can be also updated dynamically whenever your customer clicks a specific link in your message. Add the following attribute to any link: field_name should be written the same as in profile database:

```
update-field_name="value"
```

When a customer clicks such link, the value of the customer profile field in the Database will be changed to the specified one. For instance:

```
<a href="http://www.example.org" name="link" update-clicked_link="true">Update clicked_link!</a>
```

To subscribe your customer to a different resource on click - use the following link attribute:

```
<a href="http://www.example.org" subscribe-to="1">Subscribe me to resource #1</a>
```

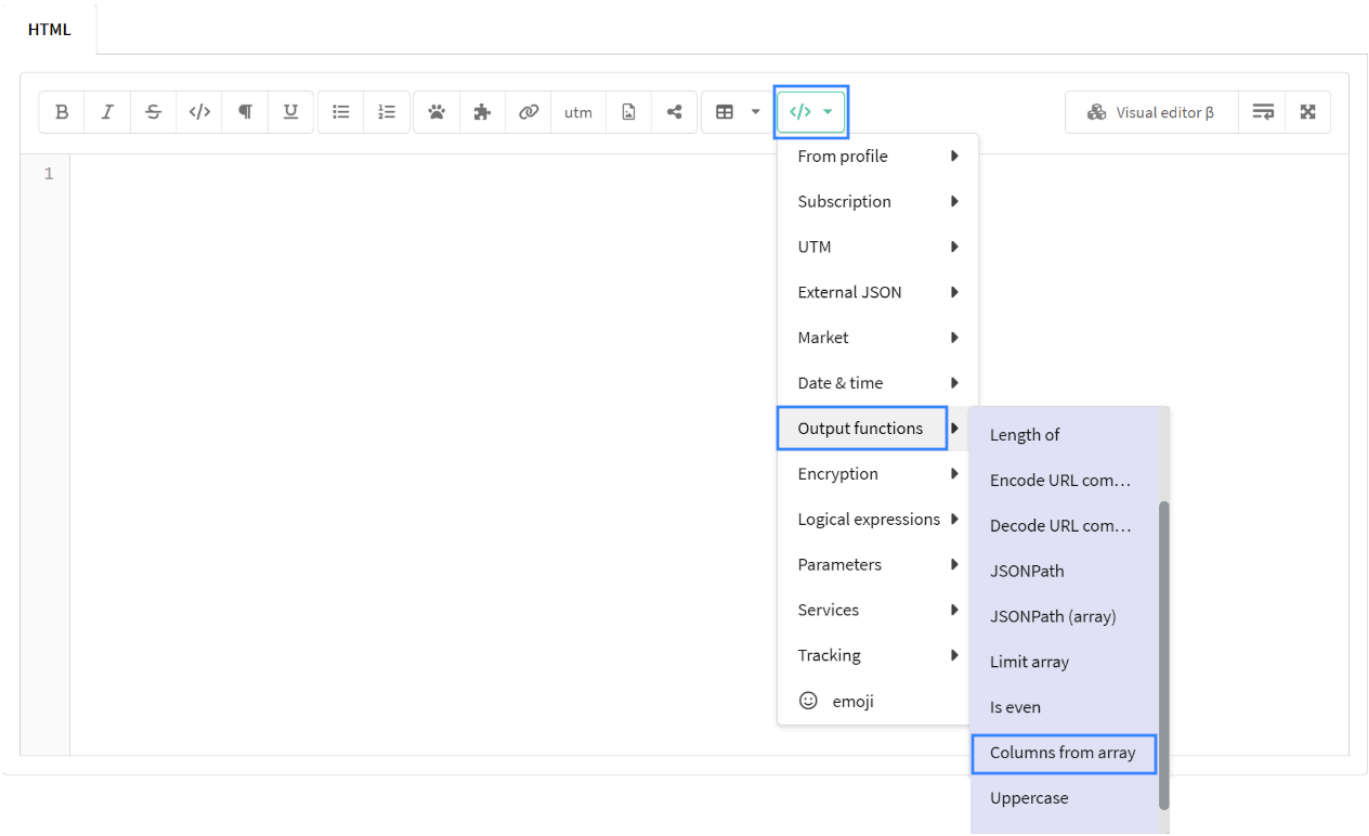
Creating tables based on array elements

Description

Creating tables based on array elements is one of platform features you can use if you transfer dynamic content. We can take an array of line objects from the database, and then distribute the elements of this array in the table as the situation requires.

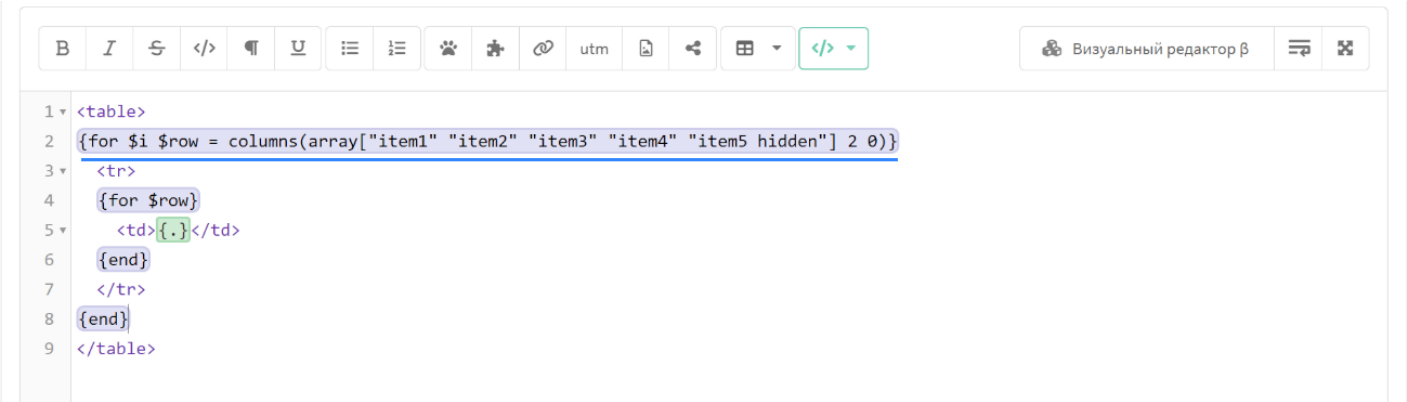
> TIP
>
> **This feature is available when transferring content in any way: API content, JSON content, content from a SQL database.**

If you want to create tables based on array elements, in the HTML version tab click: Insert variable —> Output functions—> Columns from array.



The code example will appear in the editor window. Look at the line of code:

```
{for $i $row = columns(array["item1" "item2" "item3" "item4" "item5 hidden"] 2 0)}
```



Array is an object from which we take content for our table. There is content for the certain table ceil in every element of the array (“item1”, “item2”, “item3” и etc).

“2 — it says how many columns there are in the table. 0” — it’s the amount of elements that will not be added to the table (we count from the first element of the array).

As a result, in our table all elements of the array will be distributed into two columns and none of them will be omitted.

Use cases

Let’s take a look at one of the cases for this option.

For instance, an airline’s website dynamically changes content for each user. The content depends on the history of his flights. Each user can see on his screen 5 flight offers that he may be interested in. These offers are shown in the interface as 5 blocks (item). Each of them has information on cities, dates and ticket prices. We want to not only display these blocks, but also add a separate block between them, where we will offer the user to choose the city of flight himself (offer):

	Item1	
	Offer	
	Item2	Item3
	Item4	Item5

Our steps:

- **Create an array where we add information about all five offers ("item1", "item2", "item3", "item4", "item5").**
- **Separately add the first element of the array (item1).**
- **Add a block with an offer.**
- **Add the remaining elements of the array, distributing them into 2 columns. The first element will not be added because we indicated that we want to skip one element.**

```
<table>
  <tr>
    <td colspan="2">{for limit(array["item1"] 1)}{.}{else}{cancel}{end}</td>
  </tr>
  {if lead.offer}
  <tr>
    <td colspan="2">{lead.offer}</td>
  </tr>
  {end}
{for $i $row = columns(array["item1" "item2" "item3" "item4" "item5" "item6 hidden"] 2 1)}
  <tr>
  {for $row}
    <td>{.}</td>
  {end}
  </tr>
{end}
</table>
```

So, creating tables based on array elements is a flexible mechanism that allows you to easily change the way how the user sees your message.

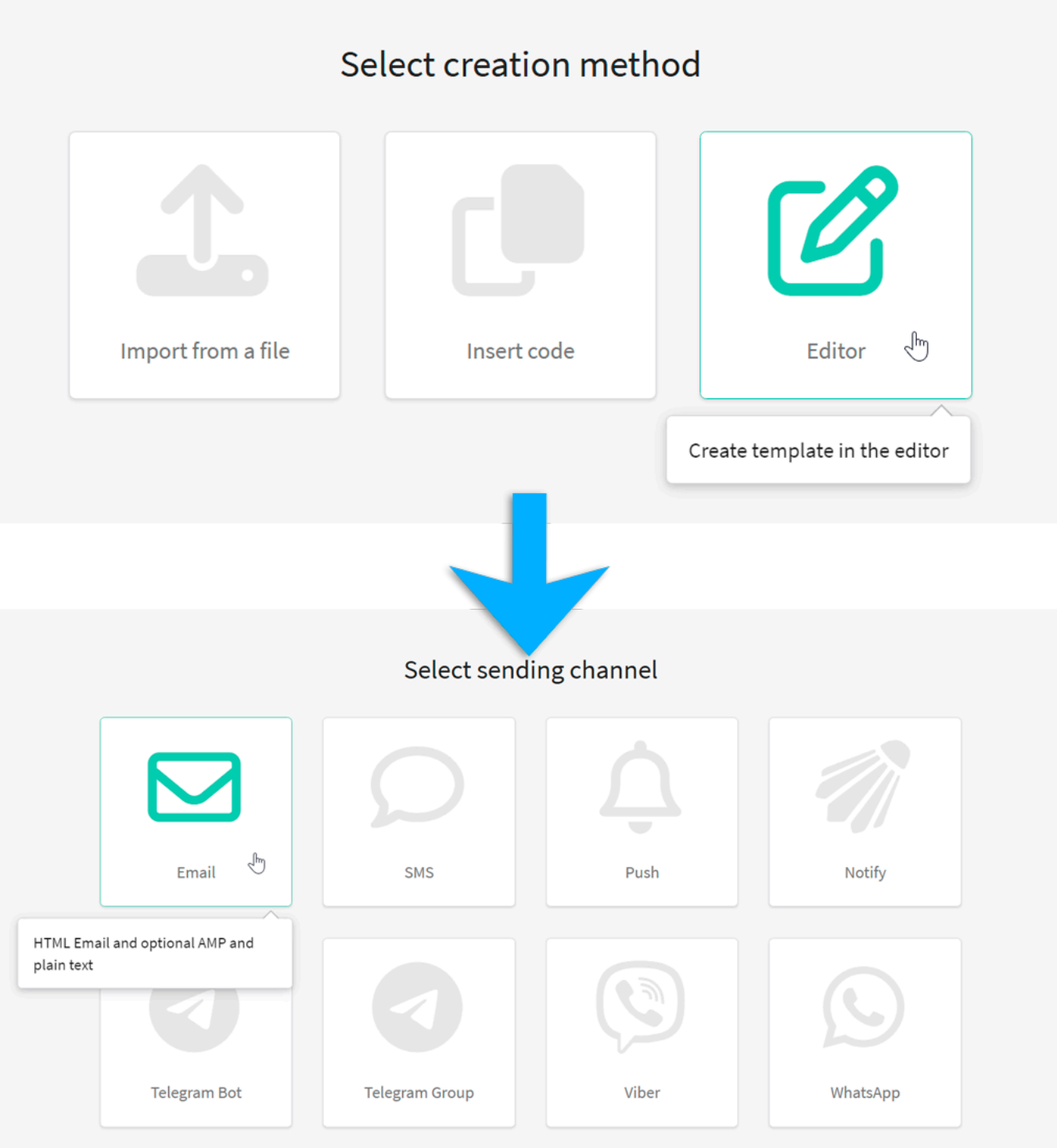
Block editor for email template

Block editor for email template is a tool for creating, editing and customizing marketing email templates and emails without writing code. It is designed for marketers and designers who want to quickly assemble professional emails with an easy-to-use interface using blocks, visual elements and design customizations.

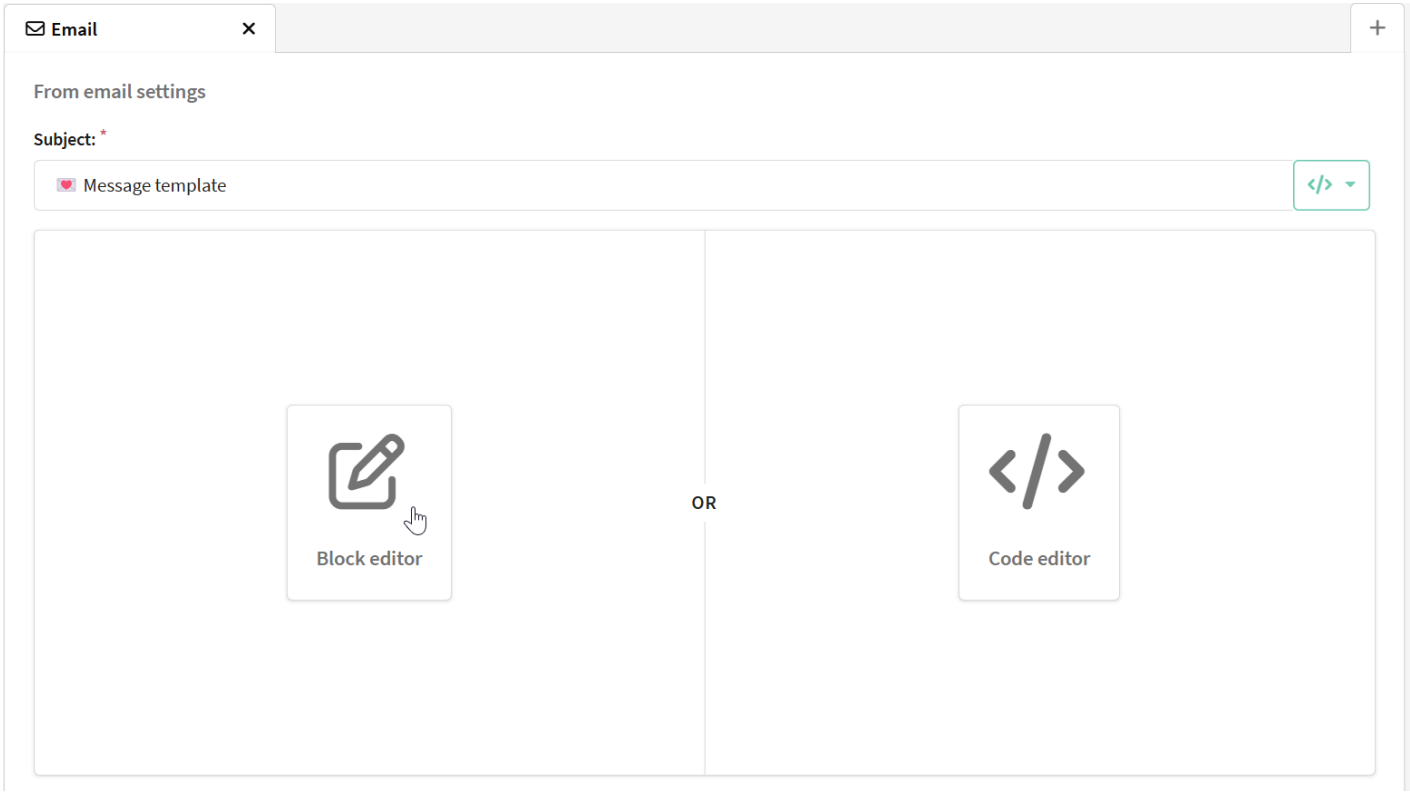
Detailed instructions, usage examples and tips on how to use the editor are available in a documentation.

Getting started

In the main menu, open the Templates section and create a new message template. Select Editor —> Email.



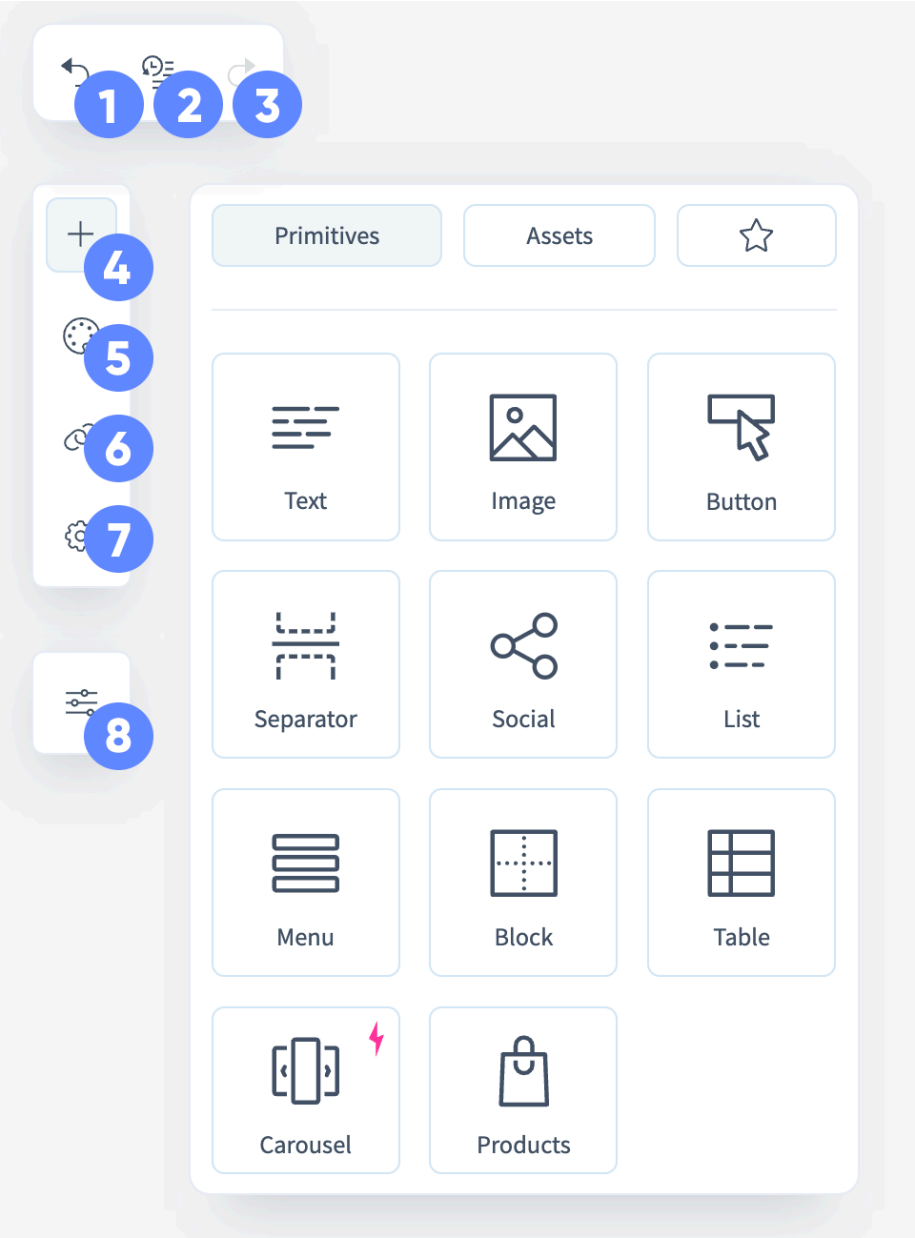
You will see an email template configuration window, select Block Editor.



Editing templates and interface

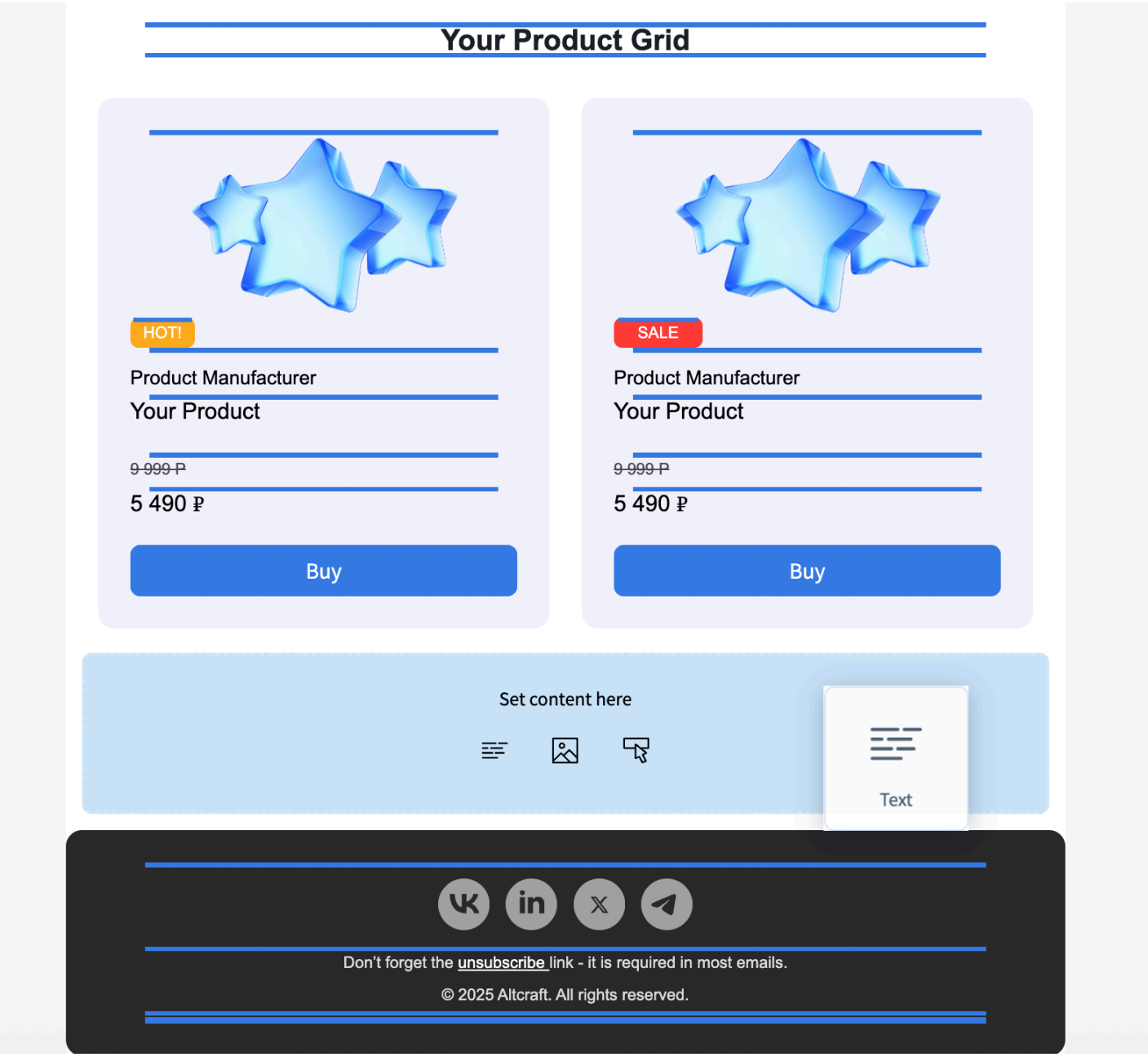
Toolbar

The toolbar contains the following controls:



1. **Undo — cancels the last performed action;**
2. **Action history — opens the list of all changes and allows you to revert to any previous version;**
3. **Redo — repeats the previously undone action;**
4. **Add — opens the menu of primitives and ready-made blocks for inserting into the template;**
5. **Styles — panel for customizing the design of the template;**
6. **Link Map — displays all links used in the template for review and editing;**
7. **Parameters — theme and preheader settings for the email;**
8. **Options — general editor parameters (interface theme and property panel position).**

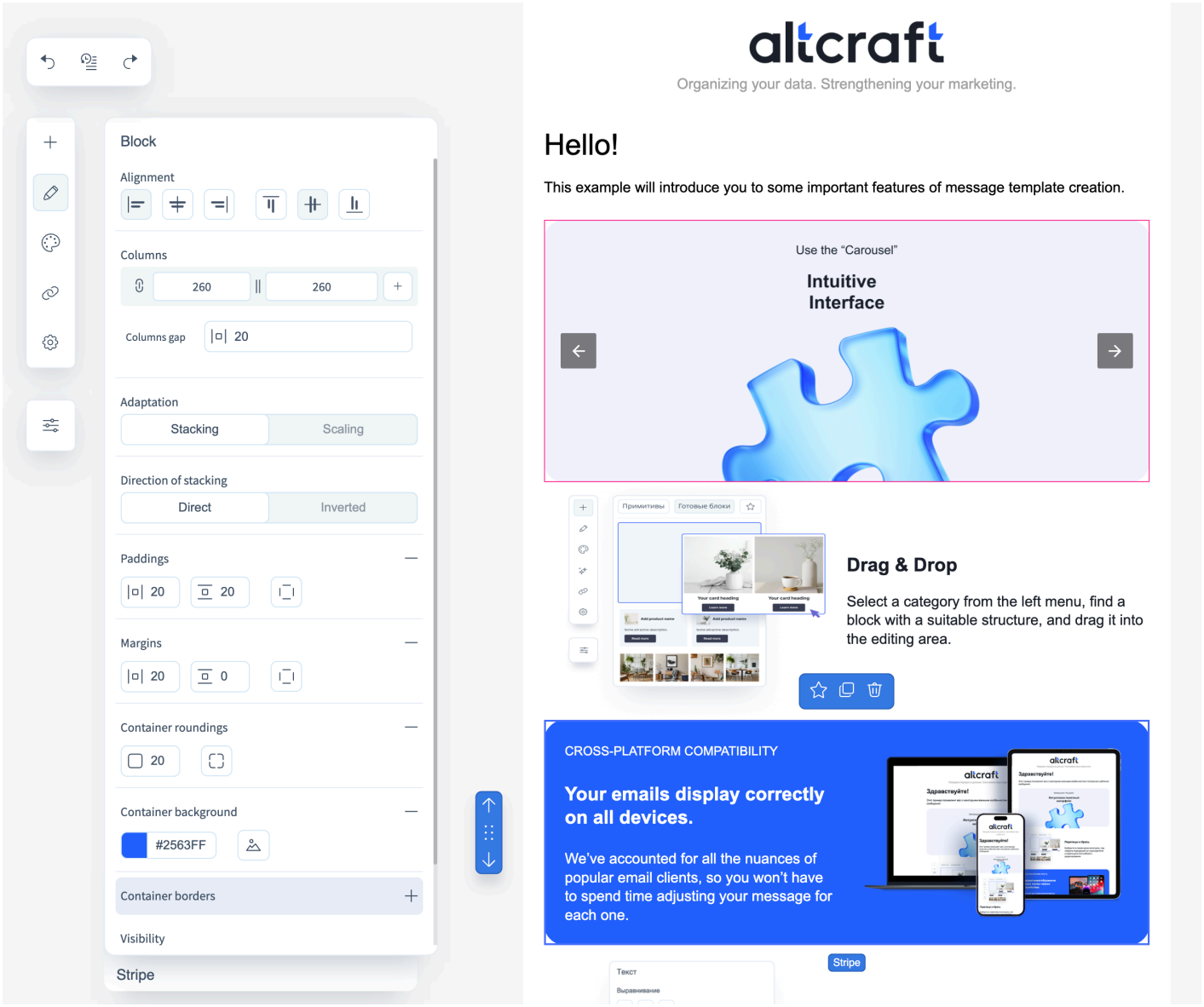
Editing Area



The main part of the editor where the email is assembled. Blocks can be dragged here from the toolbar. Each element is outlined and can be modified directly.

Blocks and Elements

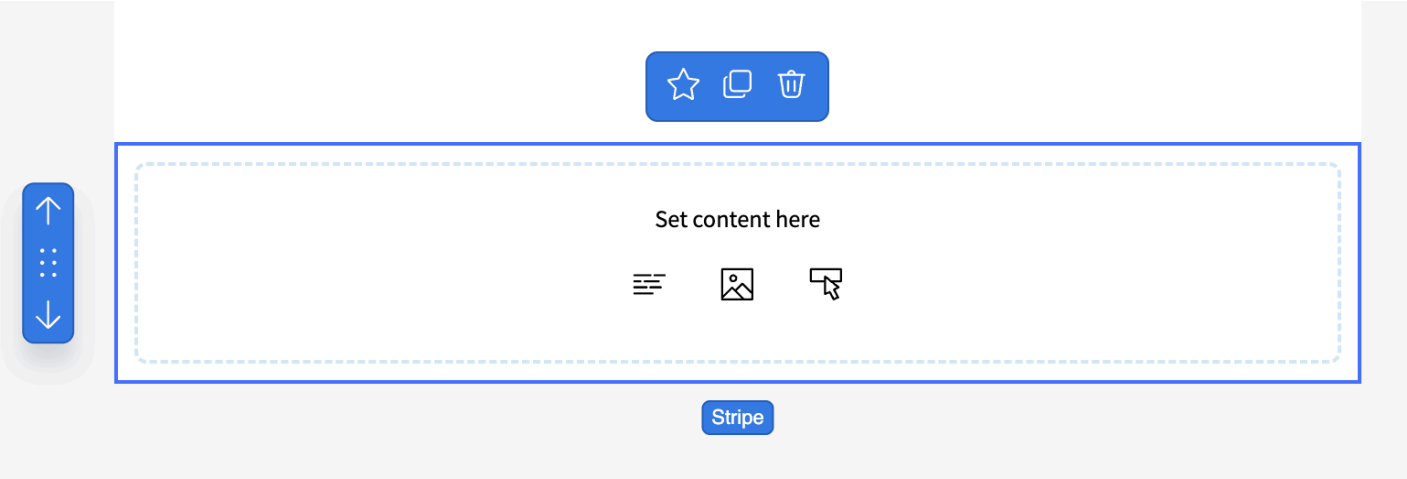
Block



For a content block, the following parameters can be configured:

- **Alignment — alignment of the block to the left, right, or center of the email;**
- **Columns — set the number of columns, their width, and spacing;**
- **Adaptation — available if the block has more than one column. Defines how the columns appear on mobile: shrink in size or stack vertically, maintaining size;**
- **Direction of stacking — sets the column order if vertical stacking is selected;**
- **Paddings — inner spacing between block content and its borders;**
- **Margins — outer spacing between the block’s borders and the email layout;**
- **Container roundings — controls corner rounding;**
- **Container background — choose a background color or upload an image;**
- **Container border — set the color, thickness, and border style;**
- **Visibility — determines visibility in different versions of the email (HTML/AMP/everywhere, desktop/mobile).**

You can add content to an empty block by clicking the respective icons inside the element:



Column properties can also be customized:

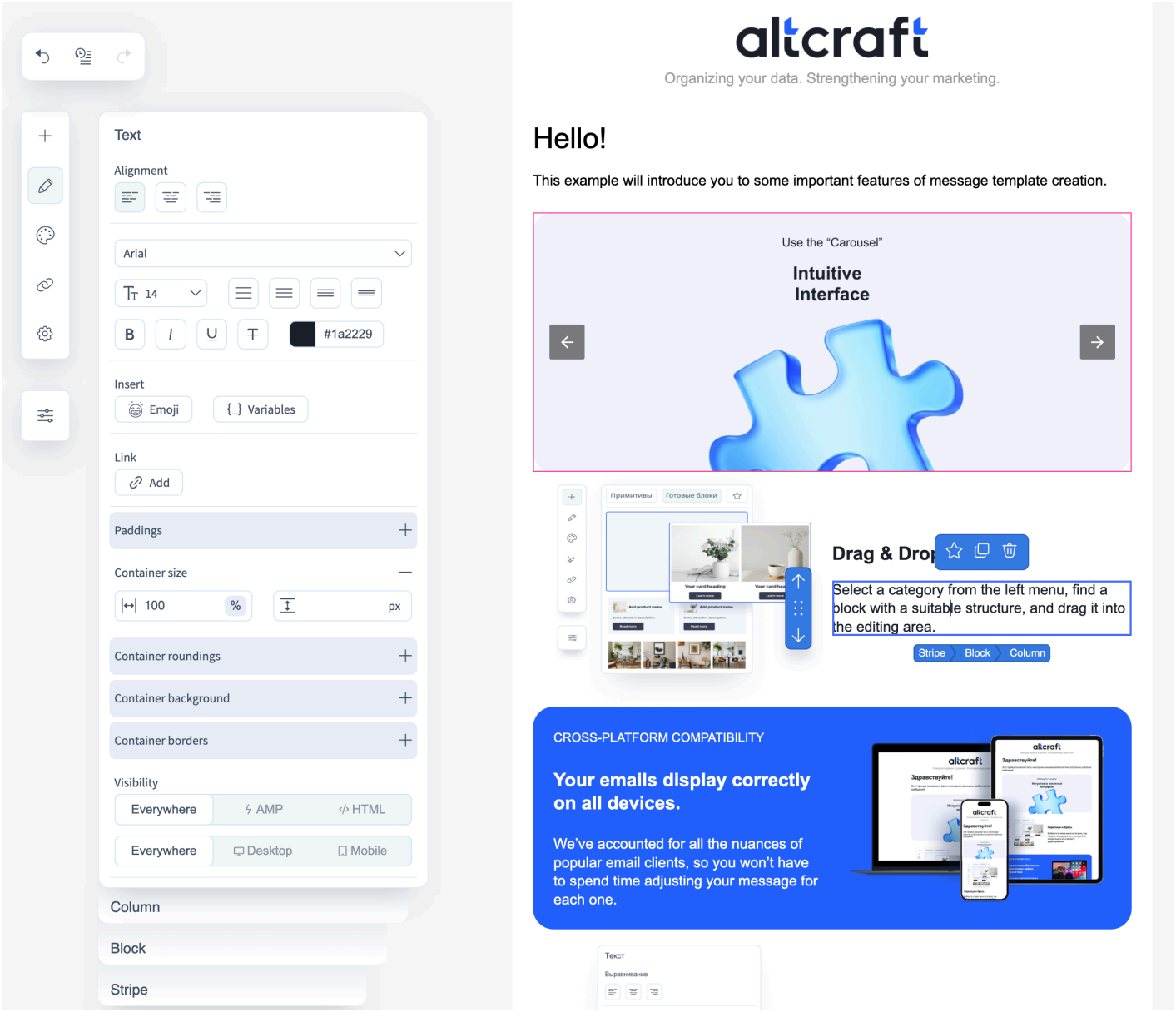
- **Alignment;**
- **Padding;**
- **Column background;**
- **Column rounding;**
- **Column border;**
- **Column size.**

For the row containing the block, you can configure:

- **Alignment;**
- **Background;**
- **Content width;**
- **Content rounding;**
- **Content borders;**
- **Padding.**

Some row parameters affect the appearance of the entire email.

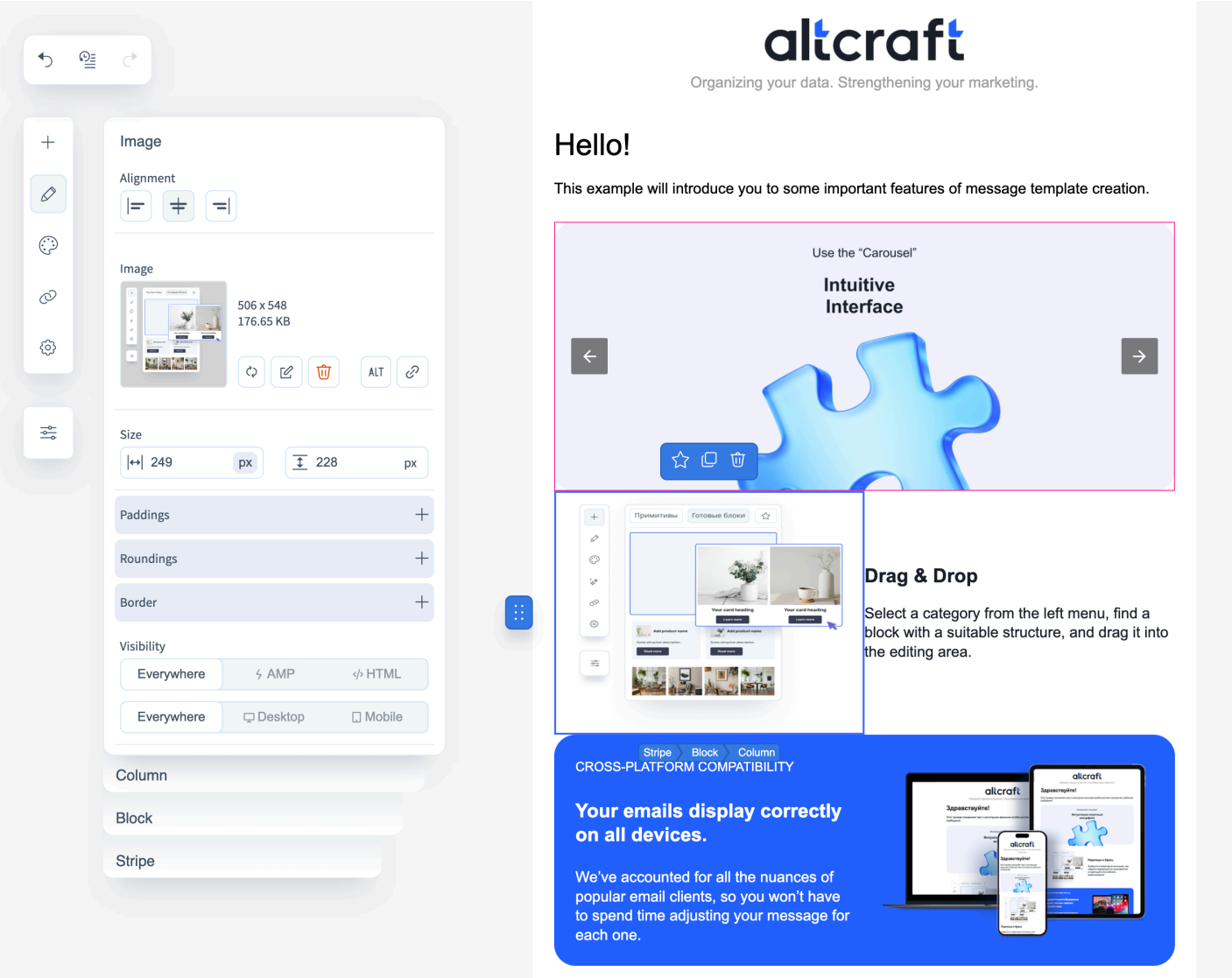
Text



Configurable properties for a text block:

- **Alignment of text within the container;**
- **Font, size, style, and color;**
- **Insertion of emojis and variables (variables supported by popular email services);**
- **Insert links in text;**
- **Padding inside the block;**
- **Size, rounding, background, and border of the container;**
- **Visibility — determines block visibility in different email versions (HTML/AMP/everywhere, desktop/mobile).**

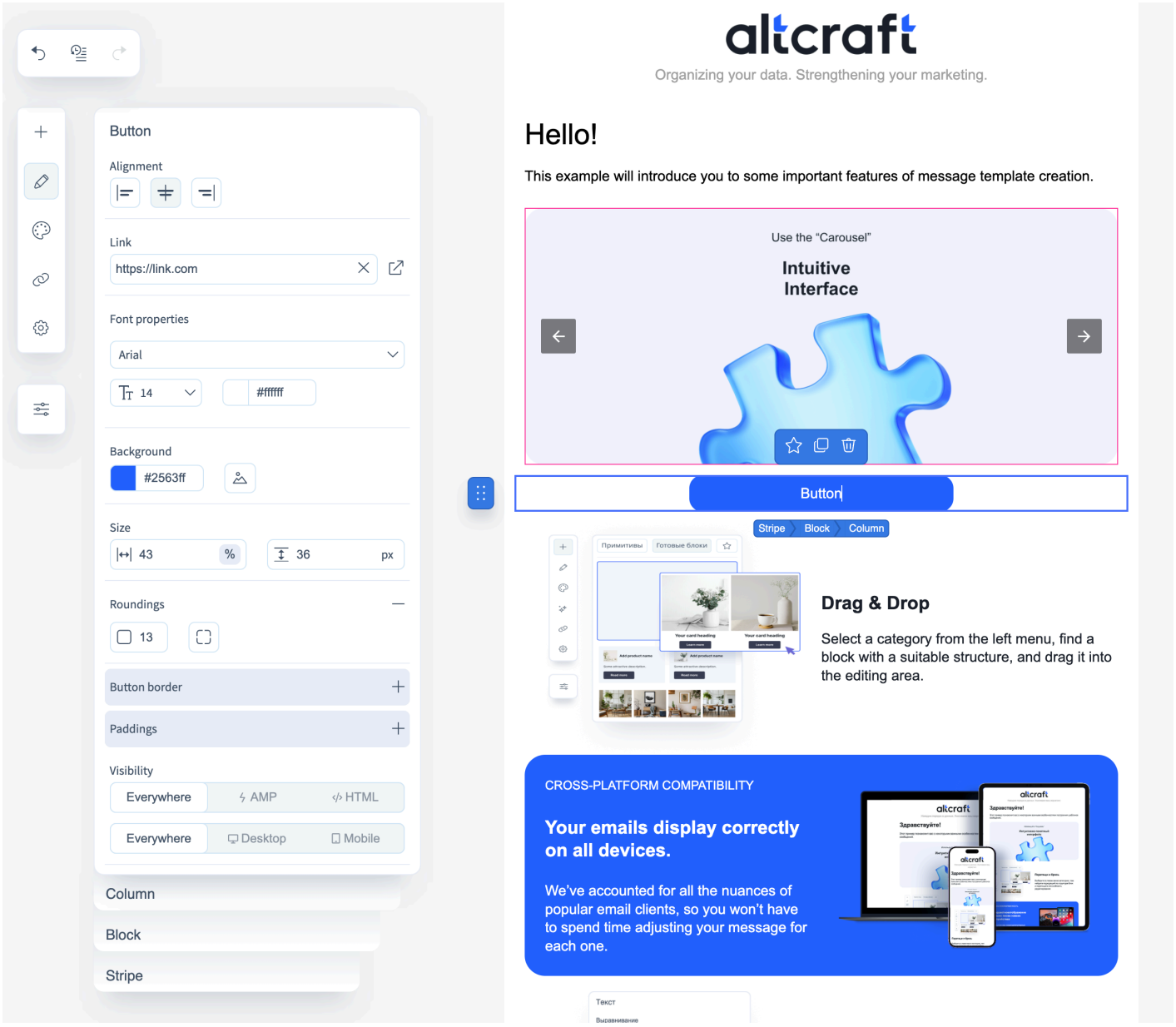
Image



Configurable properties for an image block:

- **Alignment of the image within the container;**
- **Image size;**
- **Padding within the block;**
- **Container rounding and its borders;**
- **Visibility — determines block visibility in different versions of the email (HTML/AMP/everywhere, desktop/mobile).**

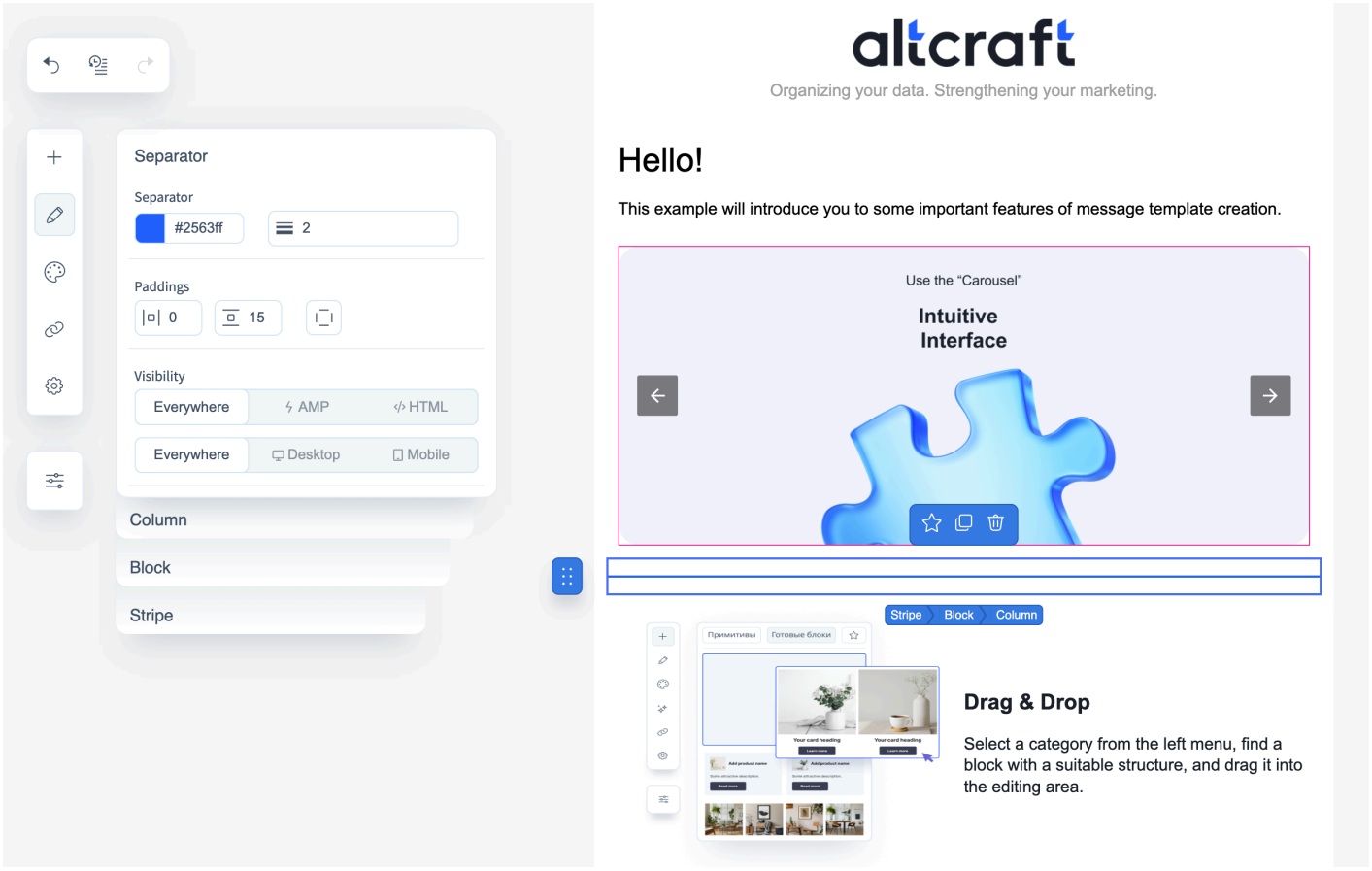
Button



Configurable properties for buttons:

- **Alignment of the button within the container;**
- **Link to follow when clicked;**
- **Font, size, and color of the text;**
- **Background of the button (you can upload an image);**
- **Size, style, and corner rounding of borders, as well as padding within the container;**
- **Visibility — determines block visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

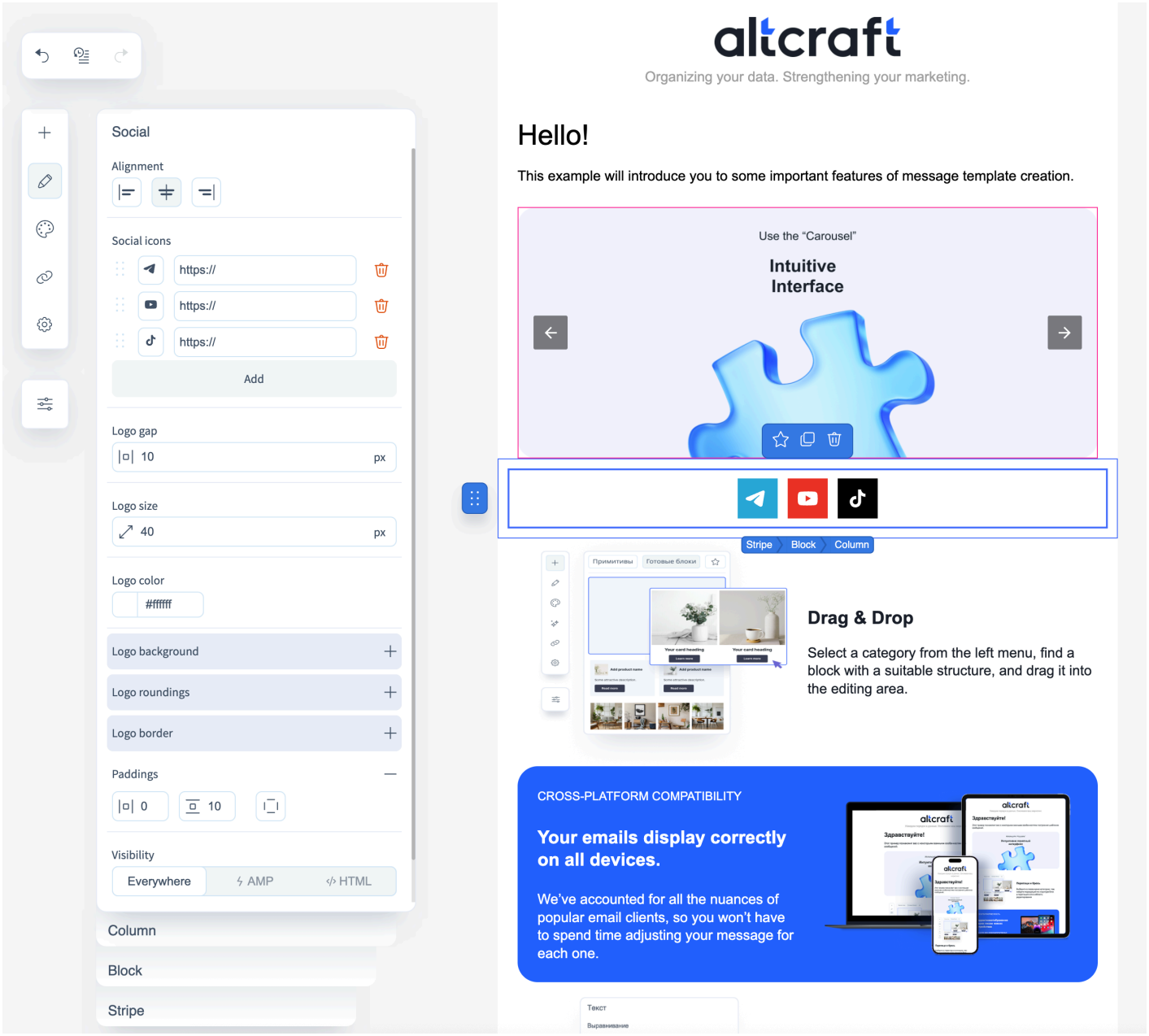
Separator



A separator is an empty block with a horizontal line inside. Configurable options:

- **Color and thickness of the separator;**
- **Padding inside the container;**
- **Visibility — determines visibility in different email versions (HTML/AMP/everywhere, desktop/mobile).**

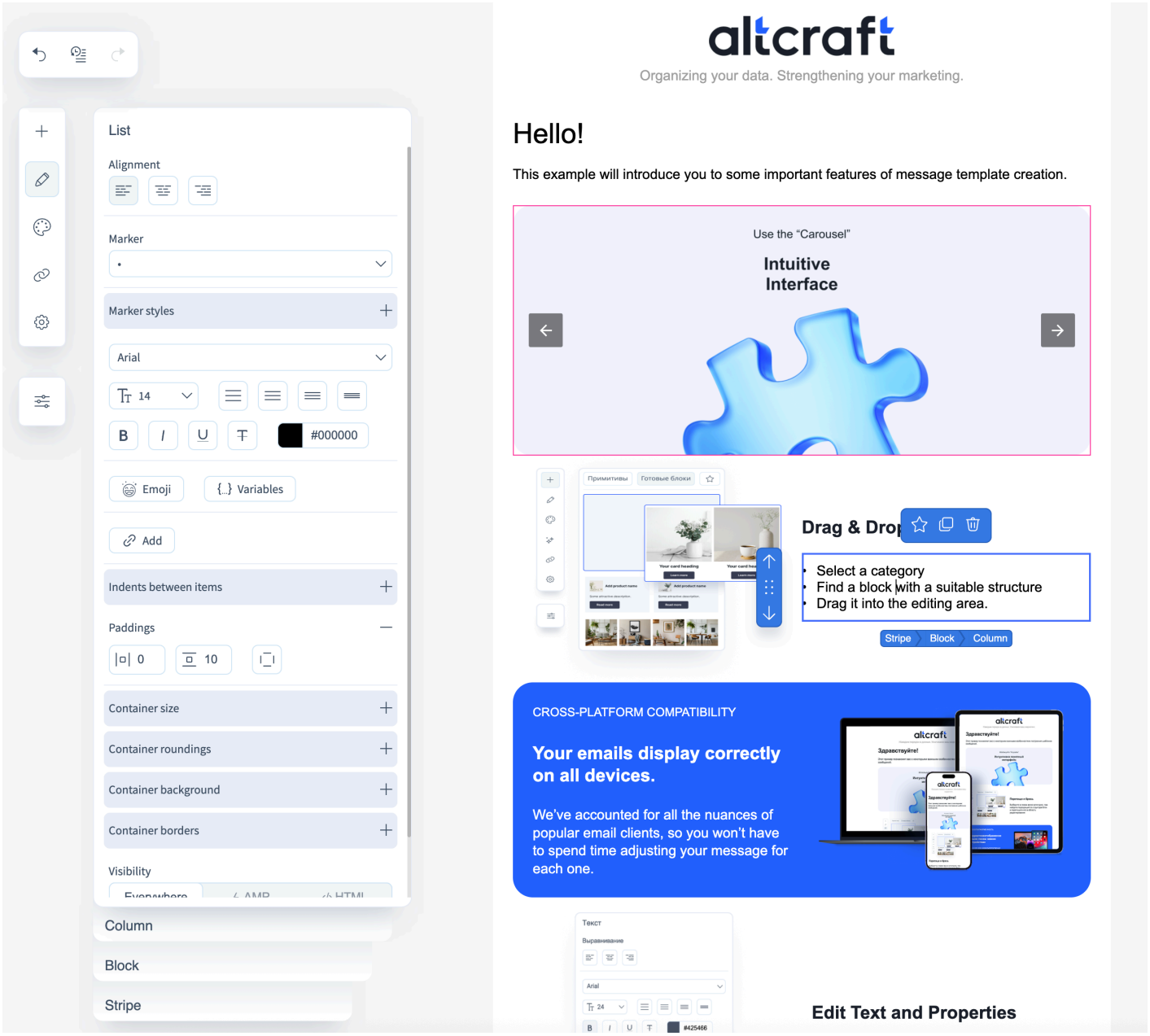
Social Networks



Social Networks — a block with clickable icons of popular social networks. Configurable options:

- **Alignment within the container;**
- **Social network settings — add or remove icons, change their order, and configure click links. You can also set individual background and logo colors for each network;**
- **Spacing between logos, logo size;**
- **Logo color and background color — customizable per network;**
- **Padding inside the container, corner rounding, and borders of logos;**
- **Visibility — determines block visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

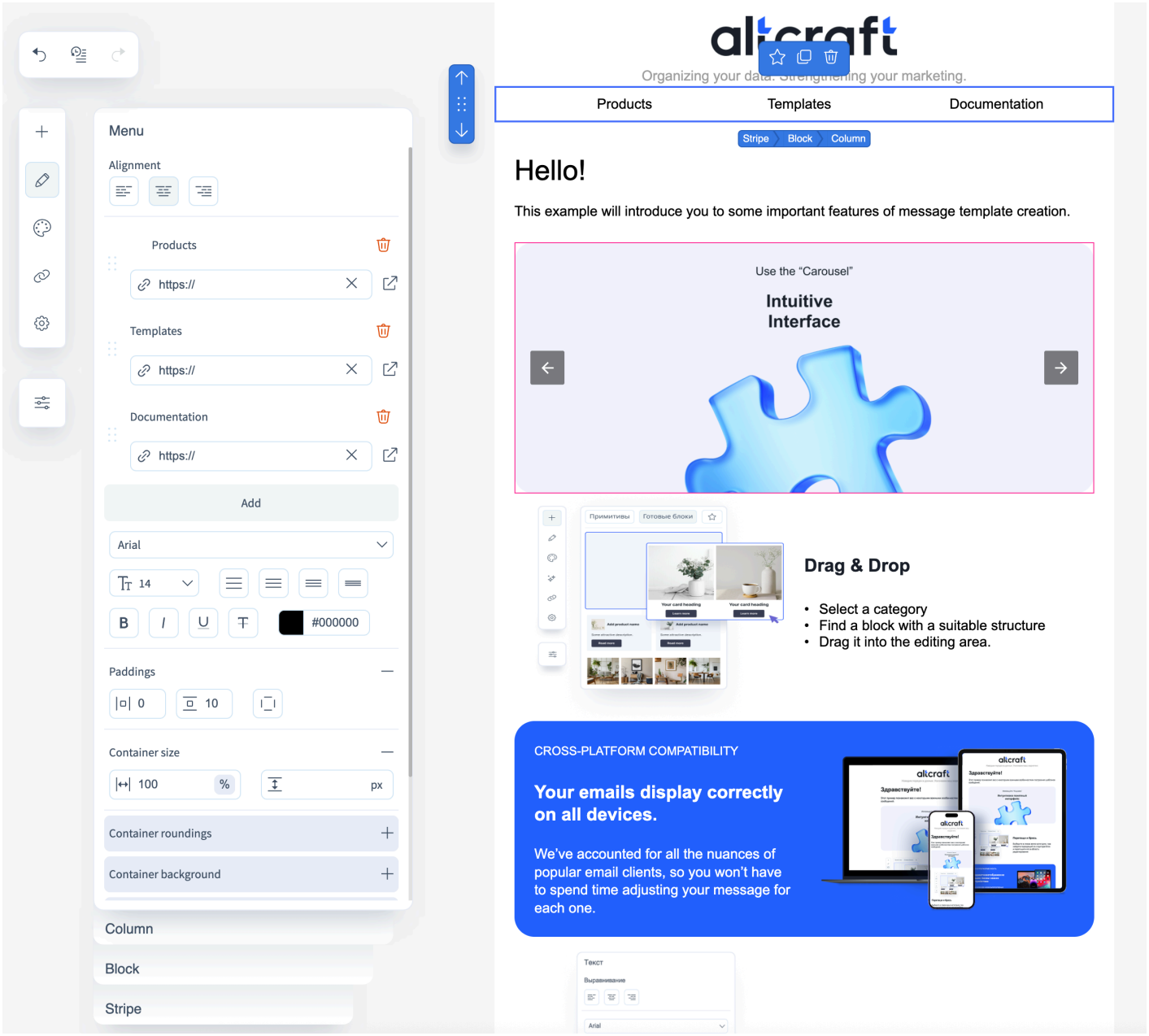
List



Configurable properties for lists:

- **Alignment within the container;**
- **Marker style, as well as size, color, and font (if a numeric marker is selected);**
- **Container size and background, border style and rounding, plus padding inside the container;**
- **Text properties — size, color, font, style, and insertion of variables and emojis. Links can also be inserted into the text;**
- **Visibility — determines visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

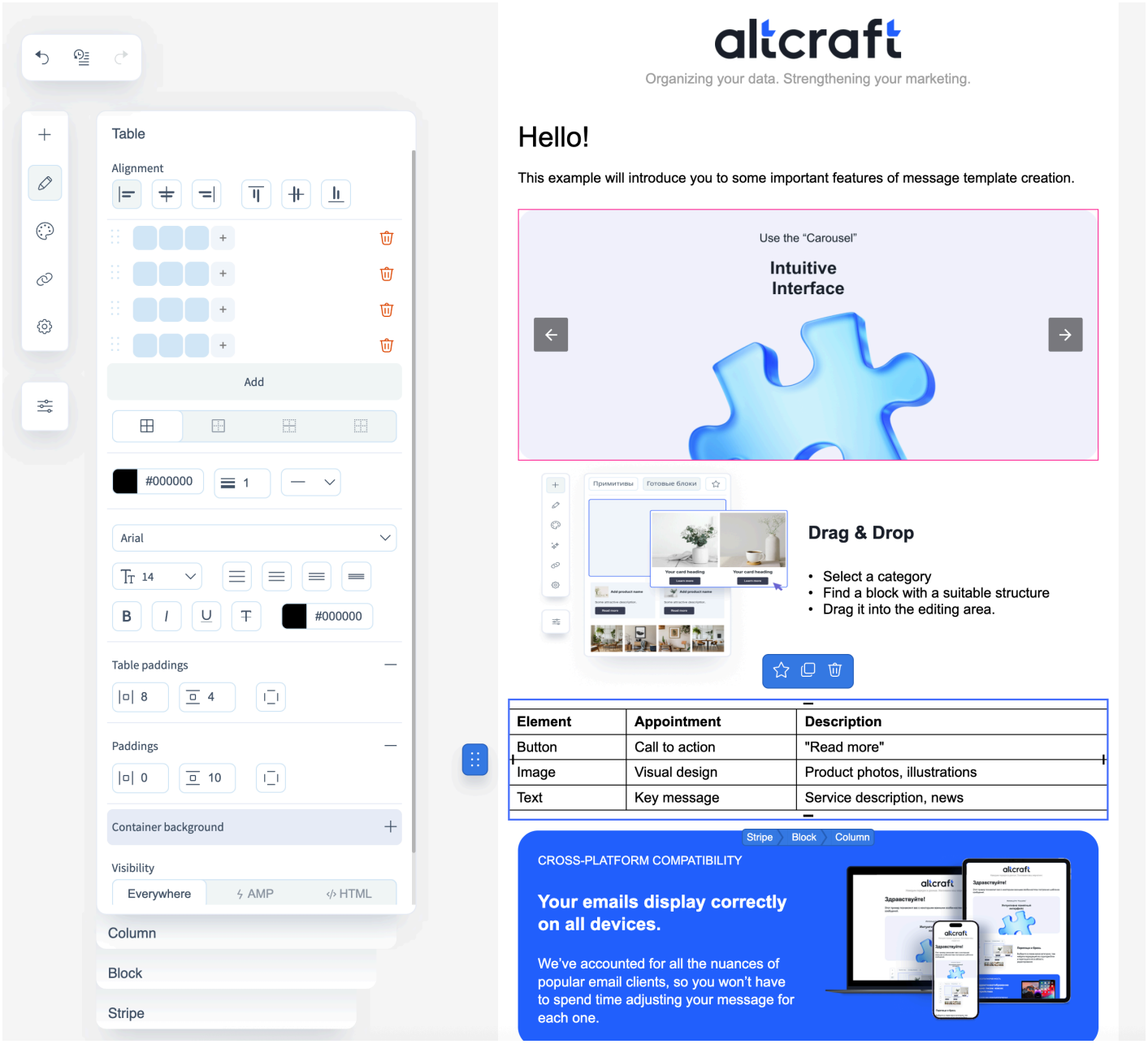
Menu



Configurable properties for menus:

- **Alignment within the container;**
- **Menu items and their order — you can specify both the text and a link for each;**
- **Text properties — size, color, font, style;**
- **Container size, background, border style, rounding, and padding;**
- **Visibility — determines visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

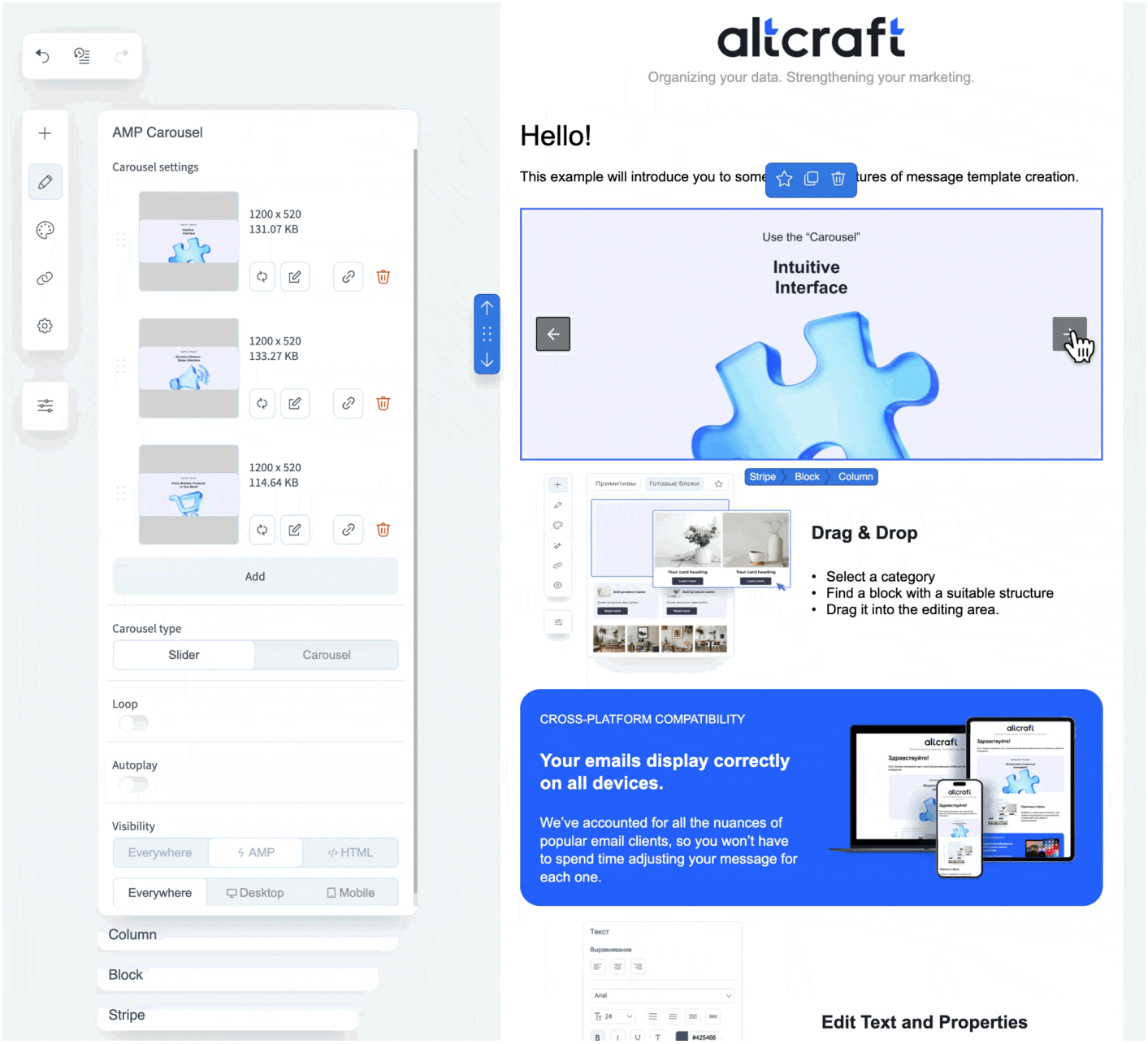
Table



Configurable properties for tables:

- **Text alignment within the table;**
- **Table size and border style/thickness between cells;**
- **Text properties — size, color, font, style;**
- **Padding inside cells;**
- **Padding inside the container;**
- **Container background;**
- **Visibility — determines visibility in different versions (HTML/AMP/everywhere, desktop/mobile).**

Carousel

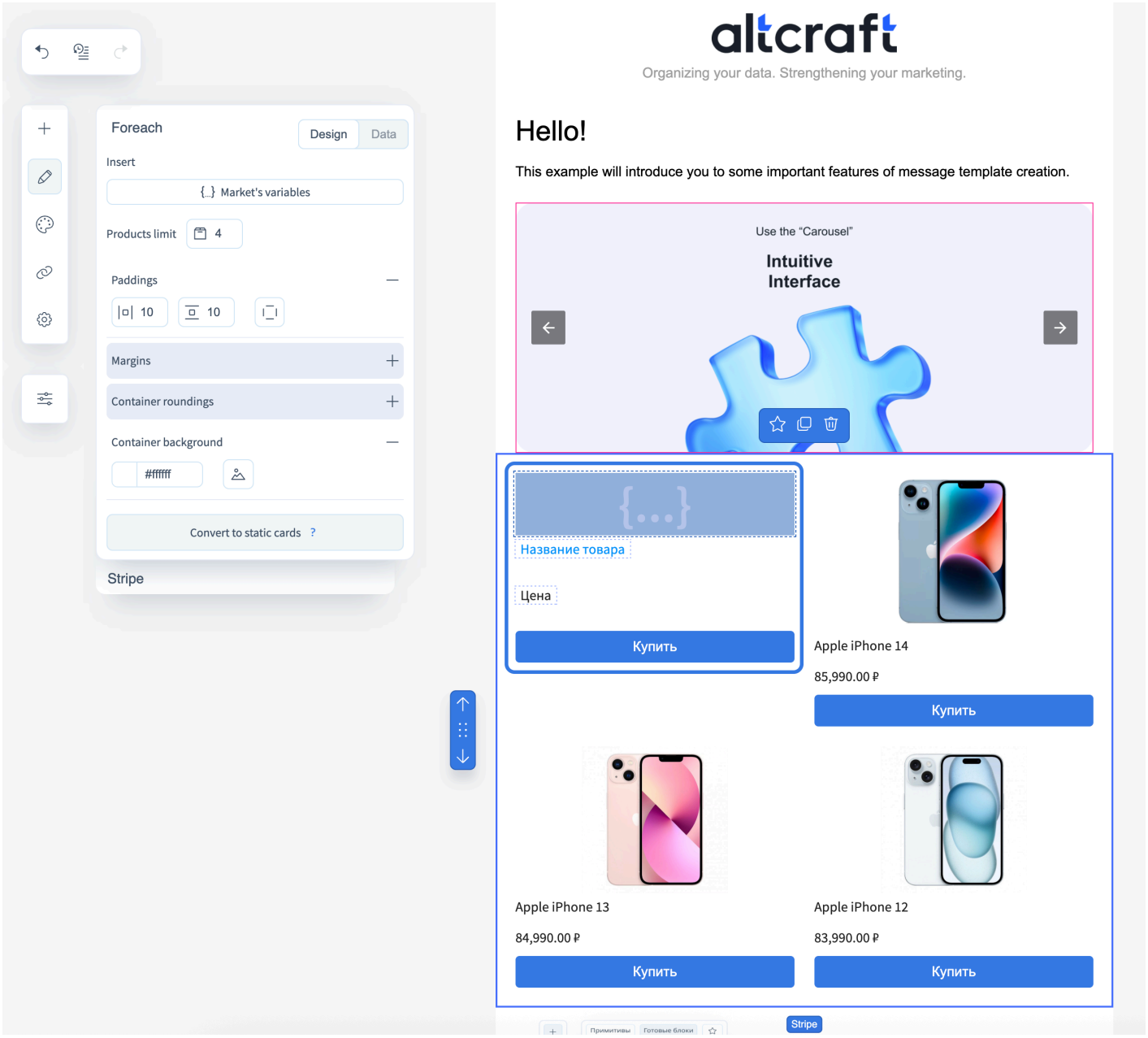


Carousel — an AMP component consisting of several scrollable images. Configurable options:

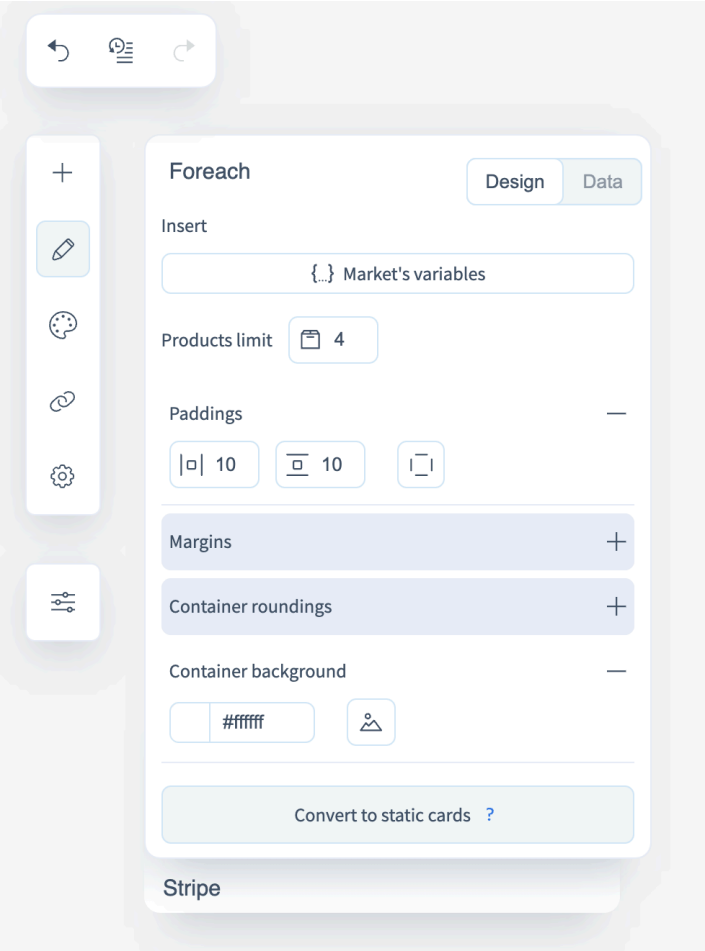
- **Images included in the carousel. You can upload from your device, choose from the Pixcraft gallery, or add click-through links;**
- **Carousel type — slider carousel (displays one image) or classic carousel (displays multiple images);**
- **For slider carousels, you can enable looping and auto-scroll after a set delay;**
- **For classic carousels, configure the height of the images;**
- **Visibility — determines visibility in different versions (desktop/mobile). Since this is an AMP component, it will not display in HTML emails.**

Products

The Products block is used for dynamically generating product sections in newsletters — for example, selections of popular, promotional, or recommended products synchronized with up-to-date data from the CDP.


Foreach
Design
Data
Insert
{...} Market's variables
Products limit
4
Paddings
10
10
Margins
Container roundings
Container background
#ffffff
Convert to static cards
Stripe
altcraft
Organizing your data. Strengthening your marketing.
Hello!
This example will introduce you to some important features of message template creation.
Use the "Carousel"
Intuitive Interface
Название товара
Цена
Купить
Apple iPhone 14
85,990.00 ₽
Купить
Apple iPhone 13
84,990.00 ₽
Купить
Apple iPhone 12
83,990.00 ₽
Купить
Stripe

Design

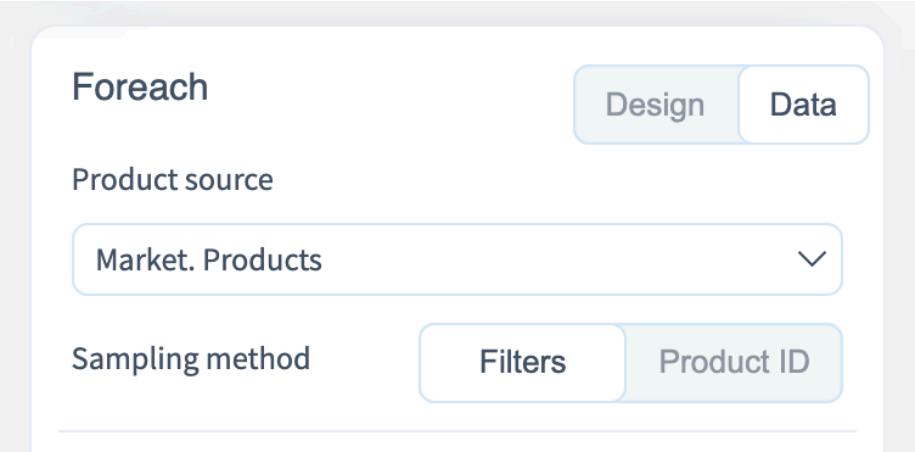


The Design tab controls the visual appearance of product cards and includes the following properties:

- **Market variables — define which elements are displayed in the card: name, price, image, button, etc.**
- **Number of products — sets how many product cards are displayed per row.**
- **Padding — defines inner and outer spacing around the cards.**
- **Container rounding — sets the corner radius of the cards.**
- **Container background — defines the color or image of the card background.**

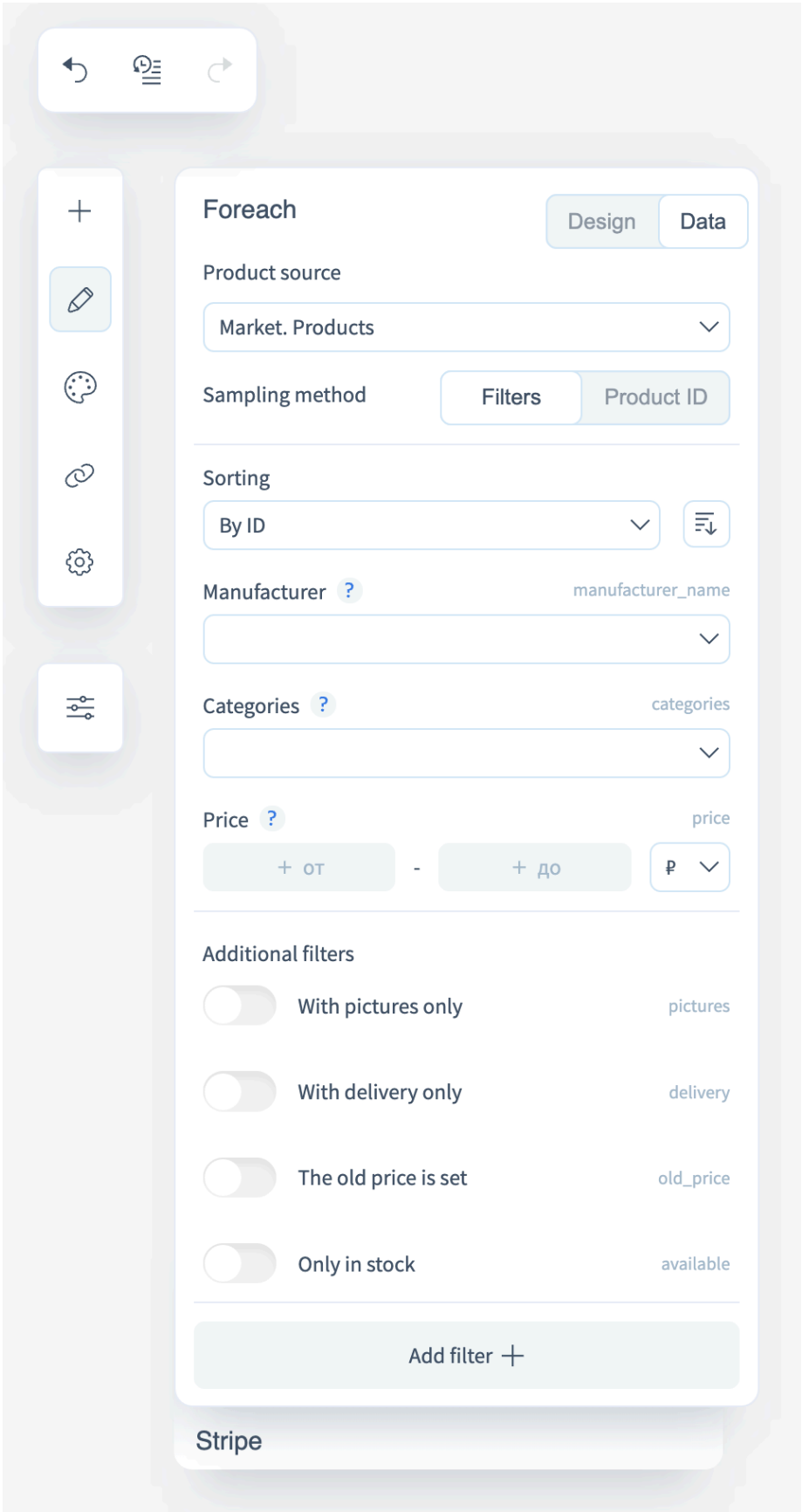
All changes made in this tab are applied to all cards within the block.

Data

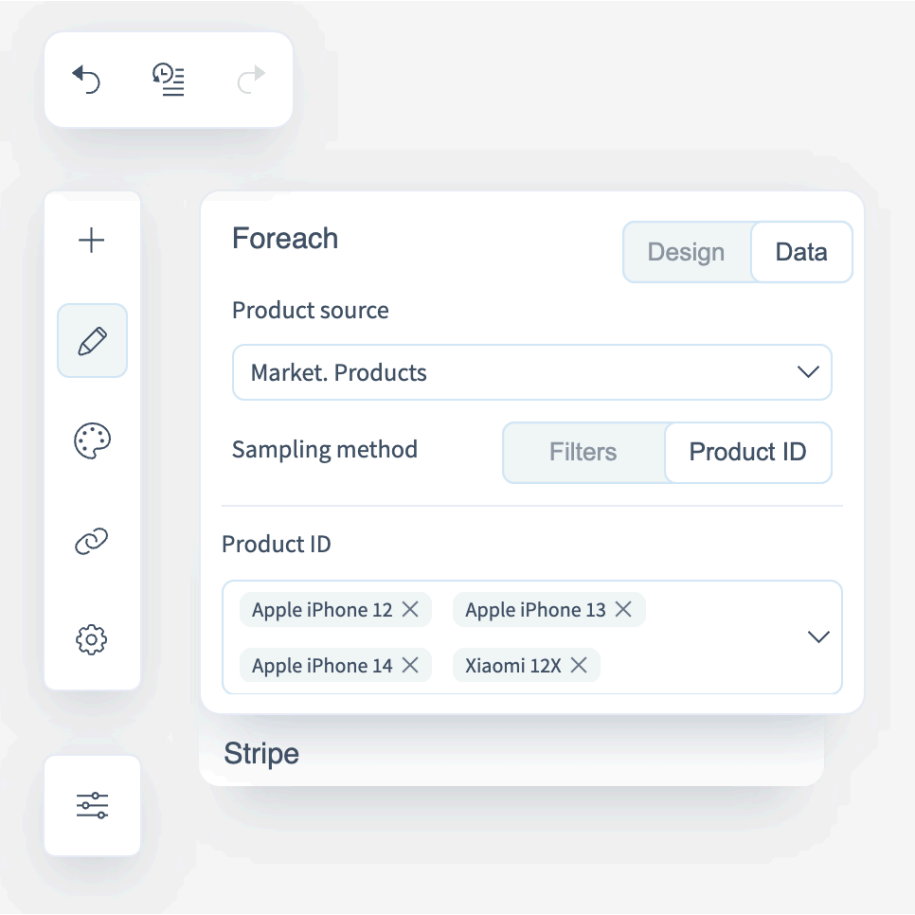


The Data tab is used to configure product sources and data selection parameters from Altcraft CDP. It includes the following settings:

- **Source — select the data source from which products are loaded.**
- **Selection method — defines how products are retrieved:**
  - **Filters — dynamic product selection mode based on defined criteria (sorting, manufacturer, categories, price). Additional filters are available: only with images, only with delivery, has old price, in stock only.**

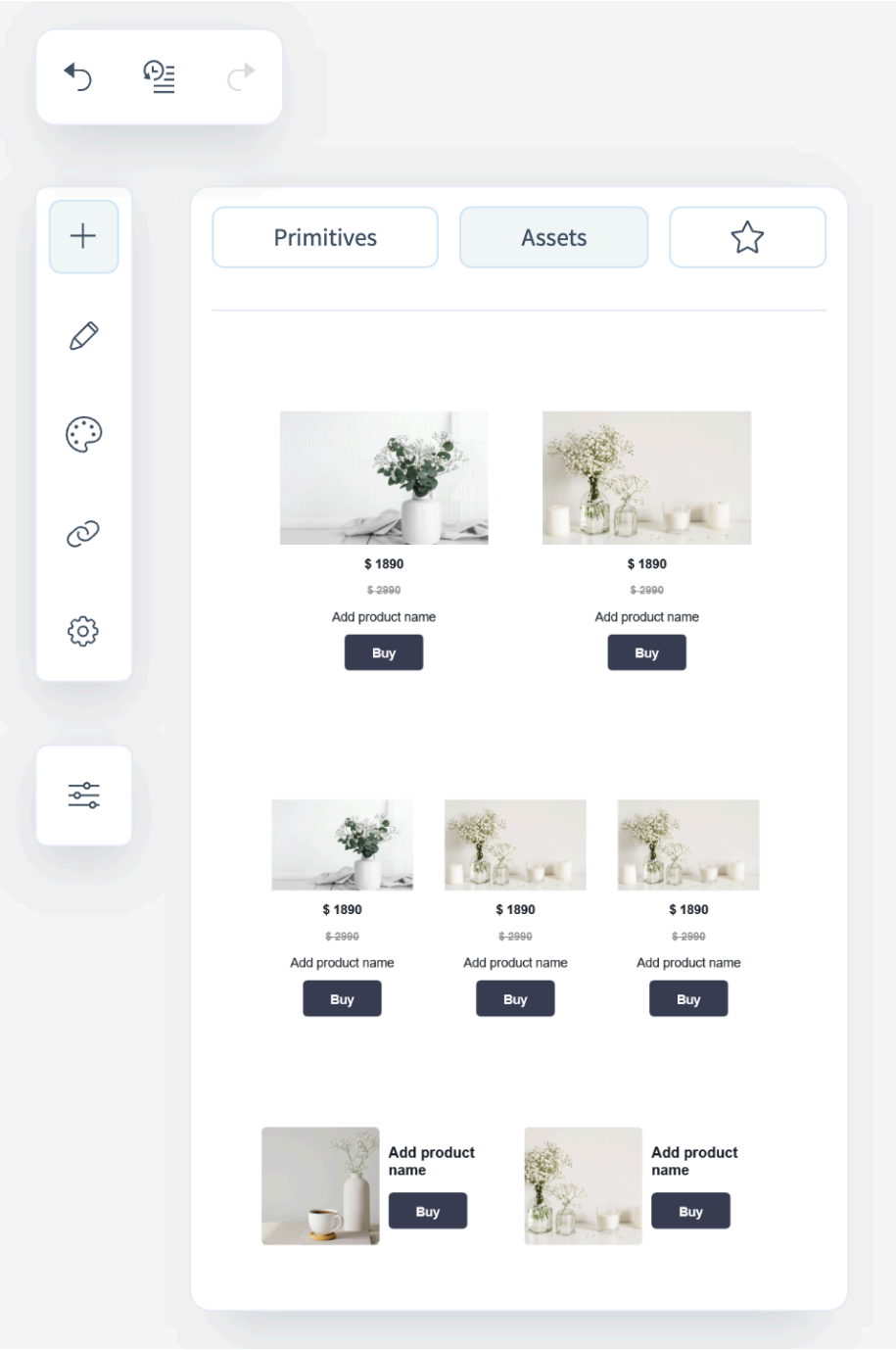


- **Product ID — manual selection mode by specifying specific product identifiers. Suitable for displaying fixed product sets that do not depend on inventory updates.**



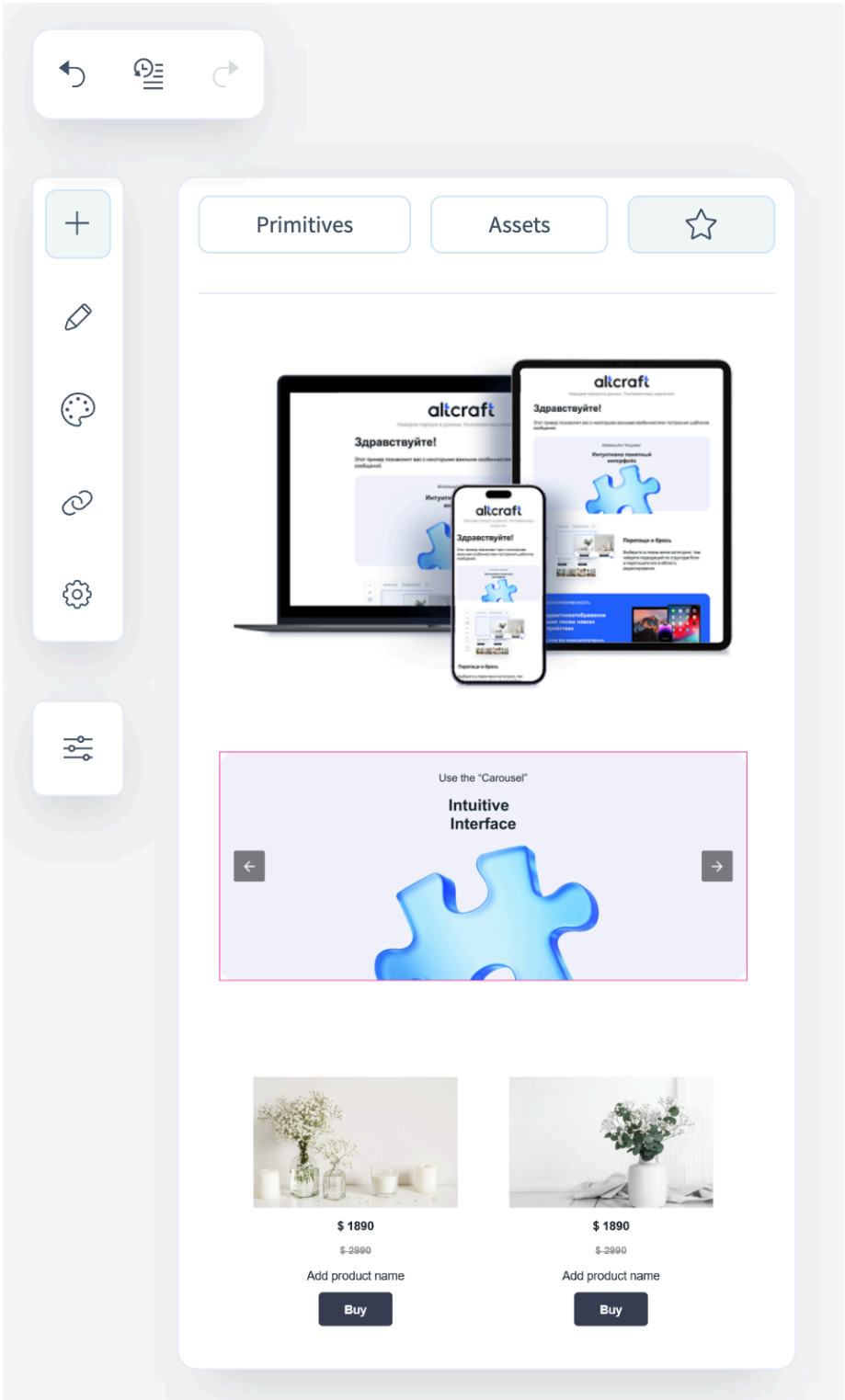
Assets

A collection of pre-designed blocks for various purposes. Using them, you can assemble a basic email layout in just a minute.



Saved Blocks

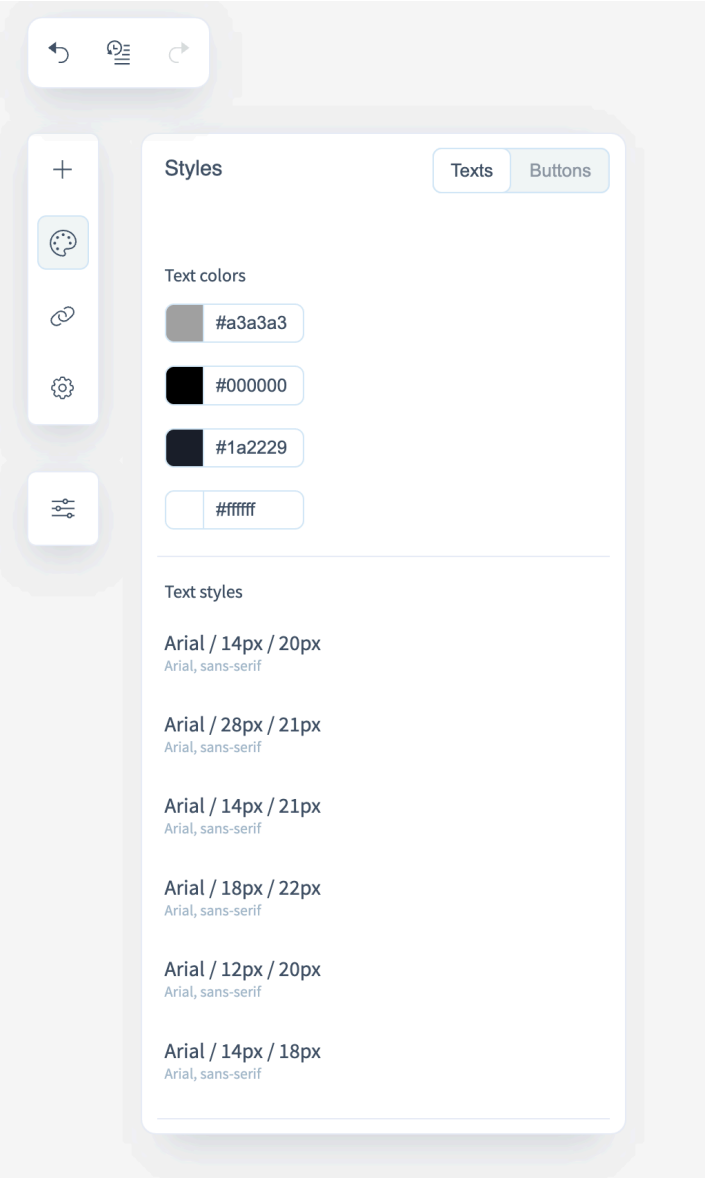
If you frequently use a specific component or want to apply consistent styled blocks across different emails, you can save it for later use.



To save a block, click the “star” icon.

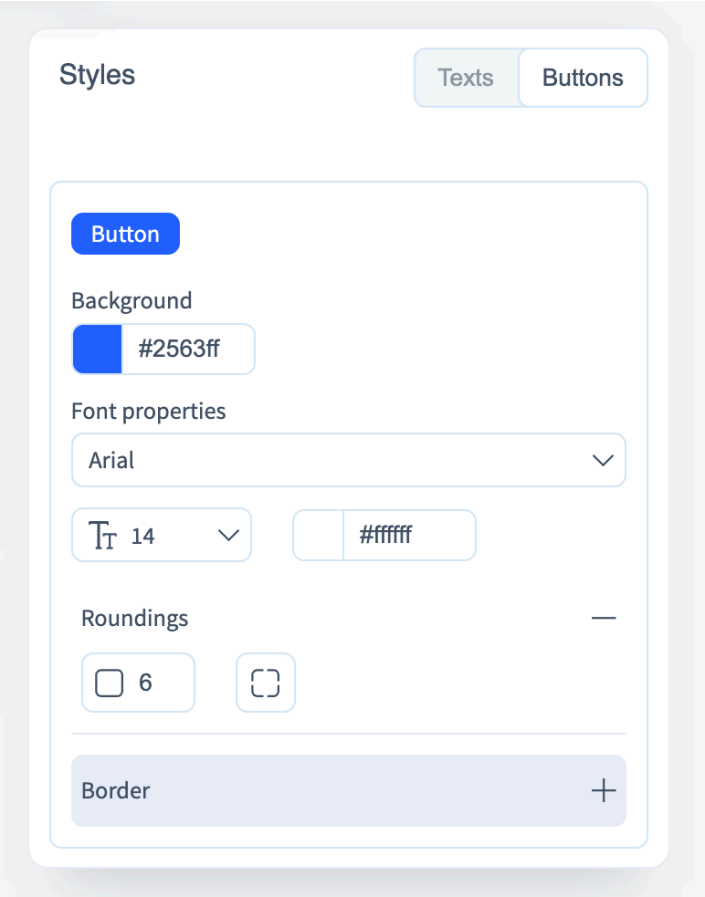
After saving, it will always be available in the editor on the respective tab (to the right of the “Ready-Made Blocks” section).

Styles



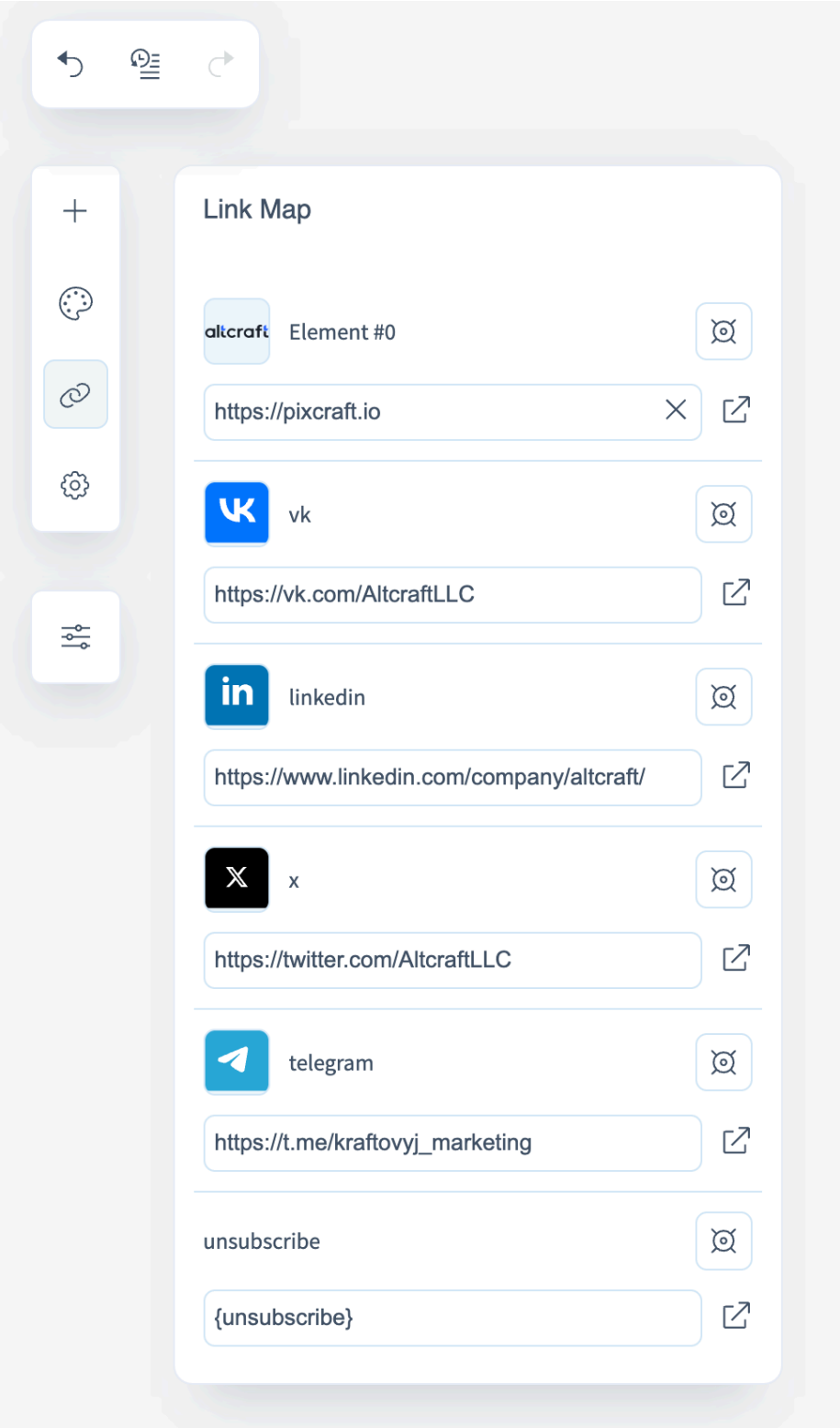
To maintain visual consistency and follow a unified design system, use the Styles section. Here you can monitor the colors and text styles used within blocks and buttons.

In the “Buttons” section, you can view a list of all buttons and their styles. From this menu, you can also modify fonts, text colors, and visual button properties:



Next to each style and color, there's a button that takes you to the corresponding element in the email where that style is applied.

Link Map

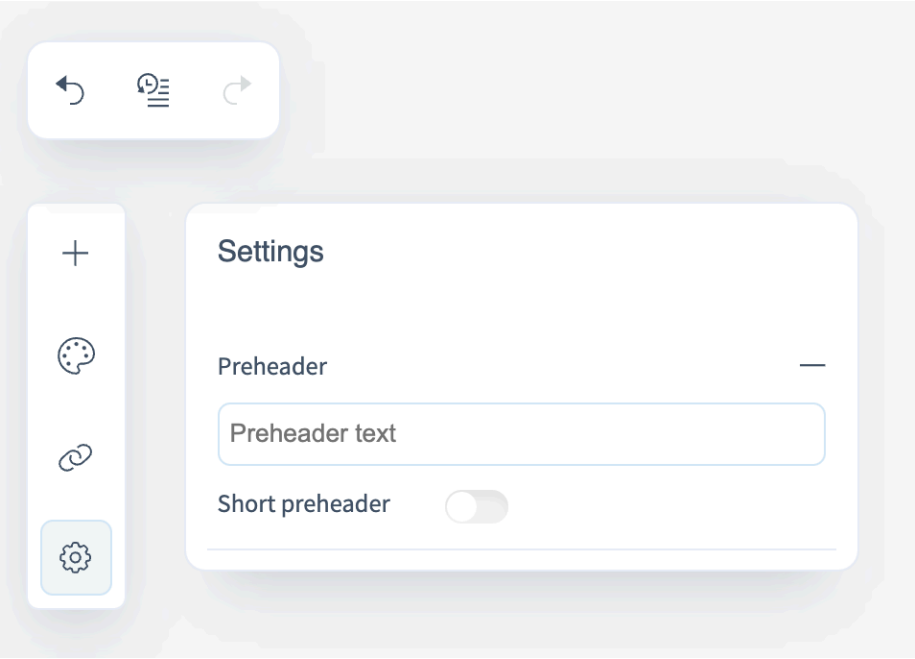


This section displays all links used in the current email. For each link, the element it's applied to is indicated.

The following control buttons are available:

- **Open Link — opens the link in a new browser tab to check the destination;**
- **Find Element — highlights the element in the email where the link is used;**
- **Delete — removes the link from both the list and the corresponding element on the canvas.**

Parameters

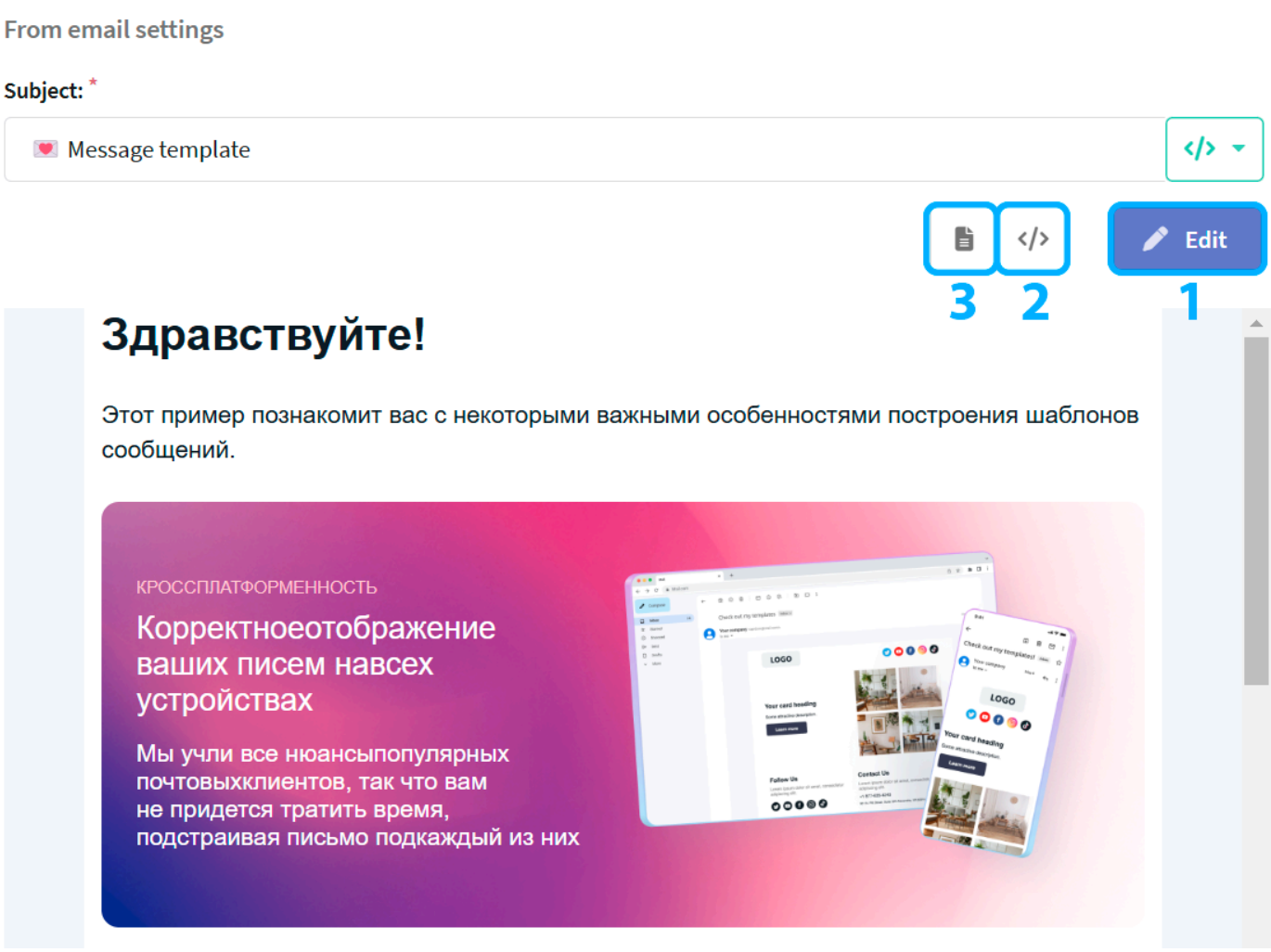


In this section, you can adjust the editor's appearance and layout according to your preferences.

- **Interface theme — choose between light and dark modes.**
- **Property panel position — defines where the properties panel appears. You can place it on the left or on the right, depending on your preference.**

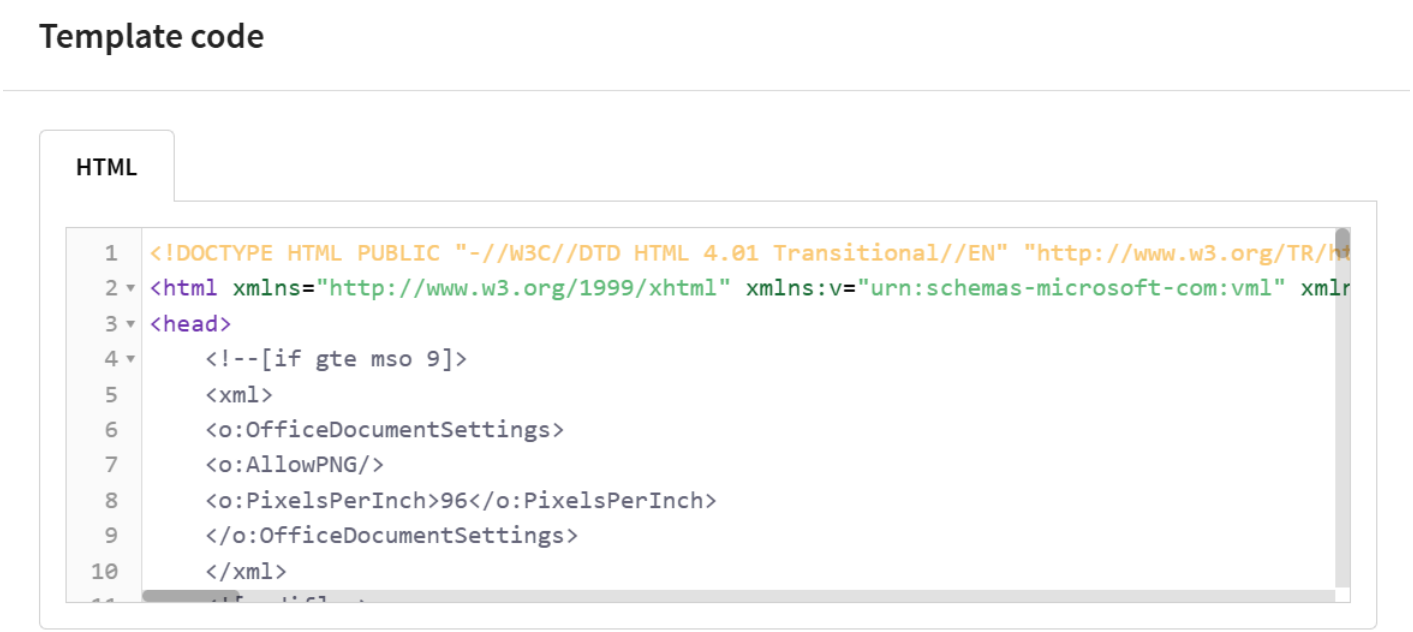
Working with the template after creation

After creating a template and applying the changes, you will have several actions available to you:



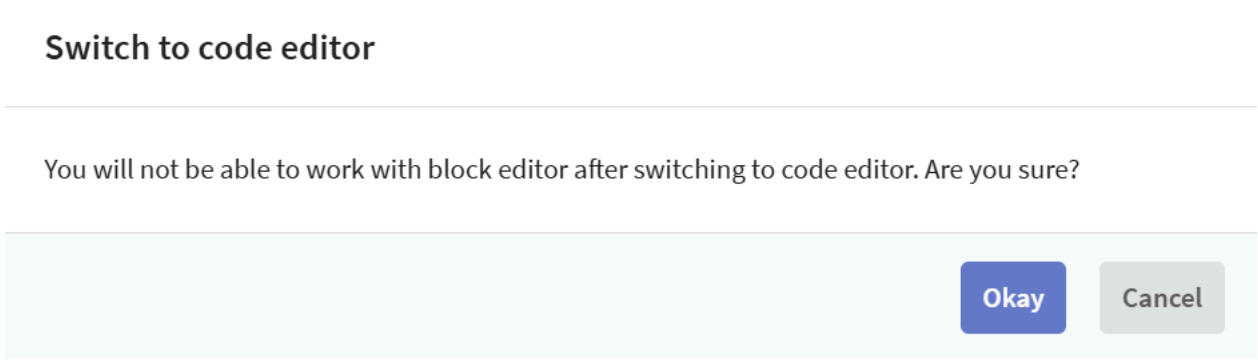
1. Edit button - allows you to return to the editor and make changes to the template;

2. Show template code button - shows HTML code of the created template:



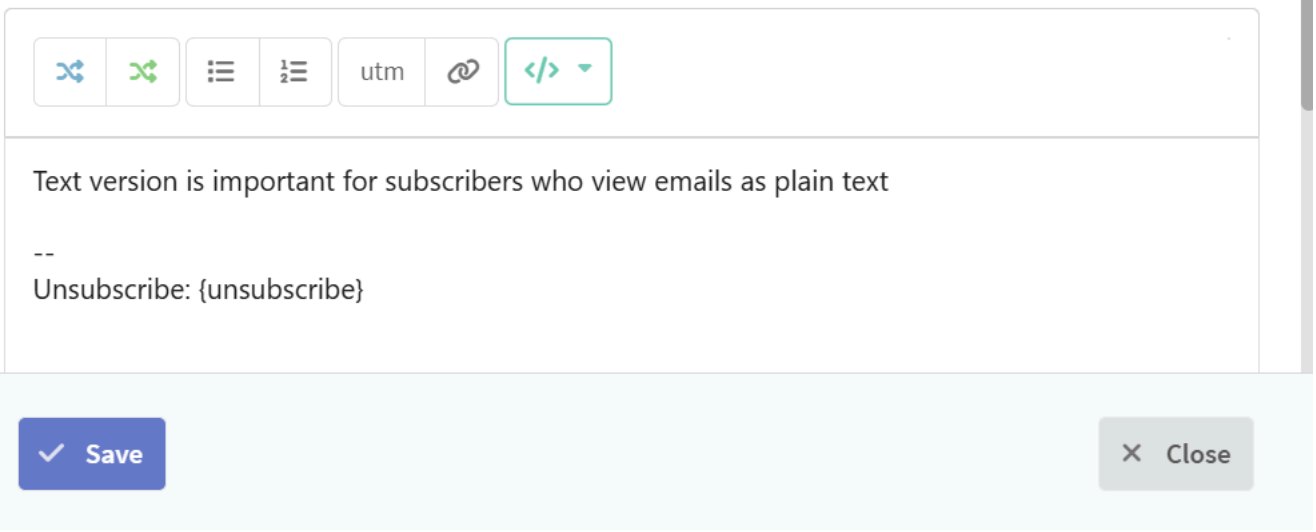
CAUTION

You can go to the code editor from the window that appears by clicking the “edit as code” button, but you will no longer be able to edit your template in the block editor.



3. Edit text version button — demonstrates the text version of your template, it also allows to edit it.

Text version



Web version of an email

Web version of an email is a version of an email message that a user can view as a web page in a browser in case it is not displayed correctly in email services. Web versions increase the chances that your message will be viewed even in case of unexpected email client errors. Also, having a link to the web version of an email allows you to quickly and conveniently share it if necessary.

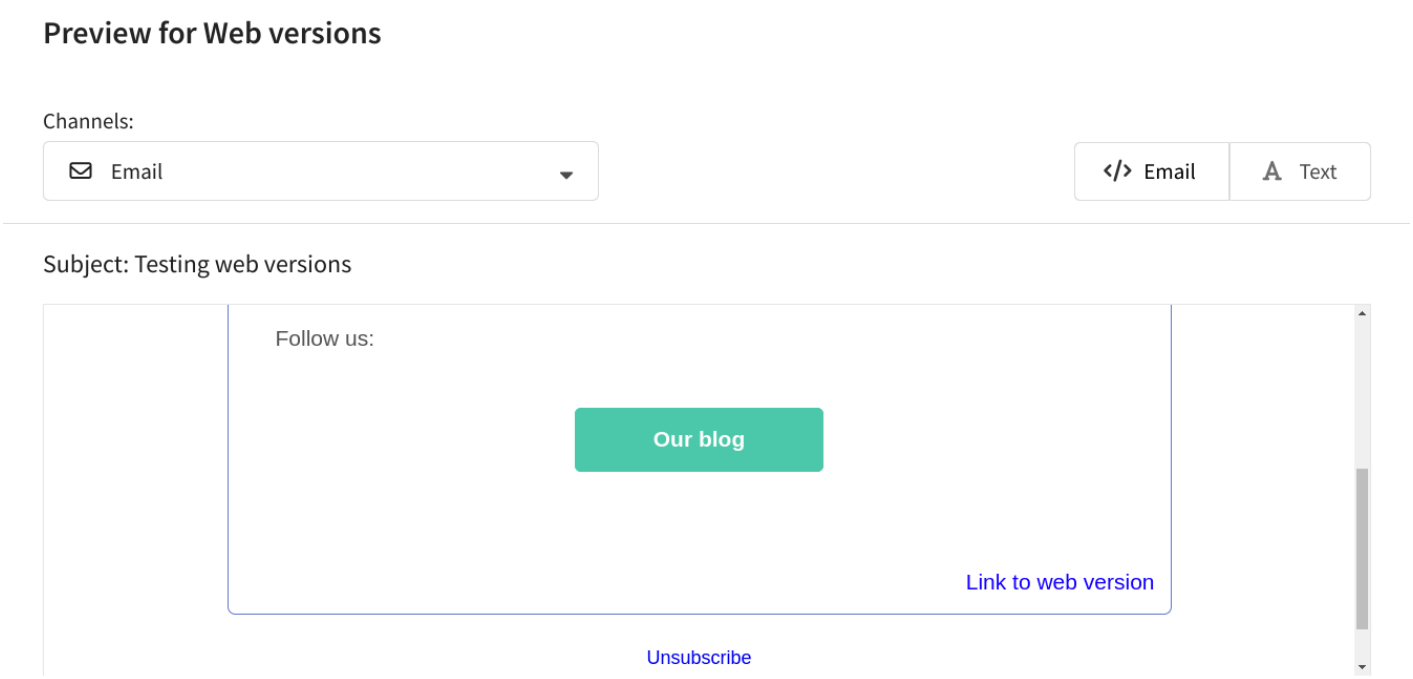
You can view the web version of an email sent to a client in the profile history. By default, the web version is stored for 30 days. The storage period can be changed by the platform administrator in the account settings.

INFO

To enable users to save web versions, the platform administrator must change the account settings, namely, activate the “Enable web versions” field. After that, links to view the web version of the sent message will appear in the profile history.

Create a link to the web version of the email

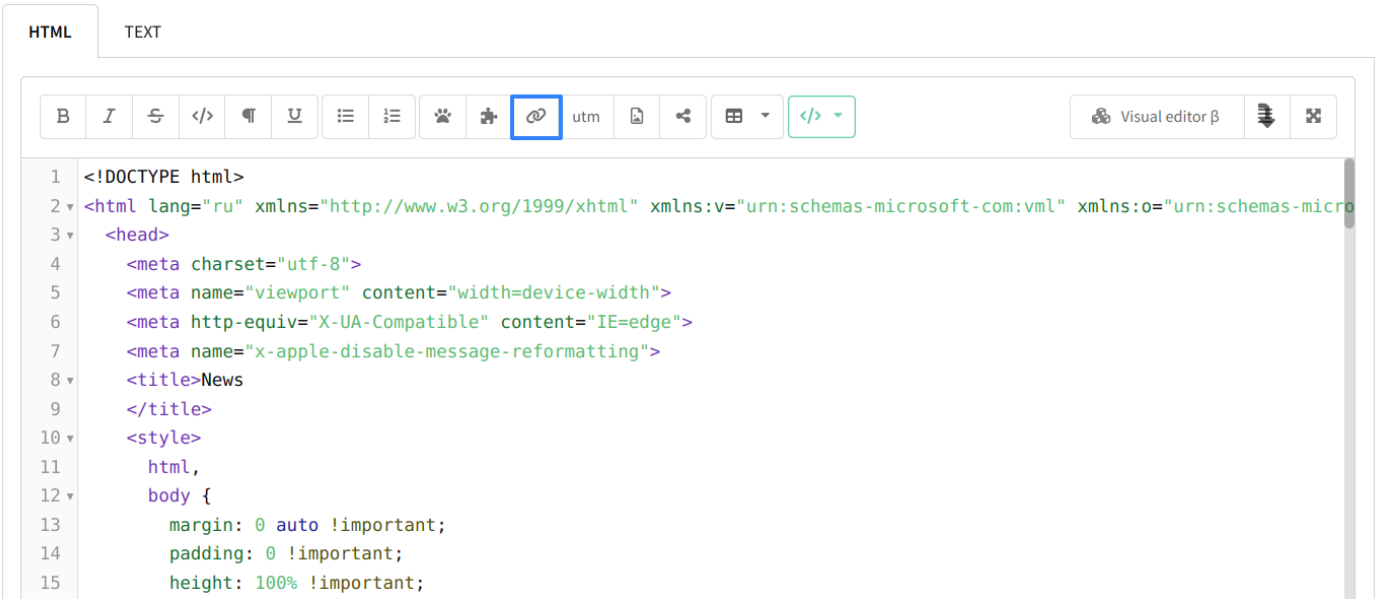
A link to the web version should be specified in the message body when creating or editing a message template:



There are several ways to create this link. Let’s take a closer look at them.

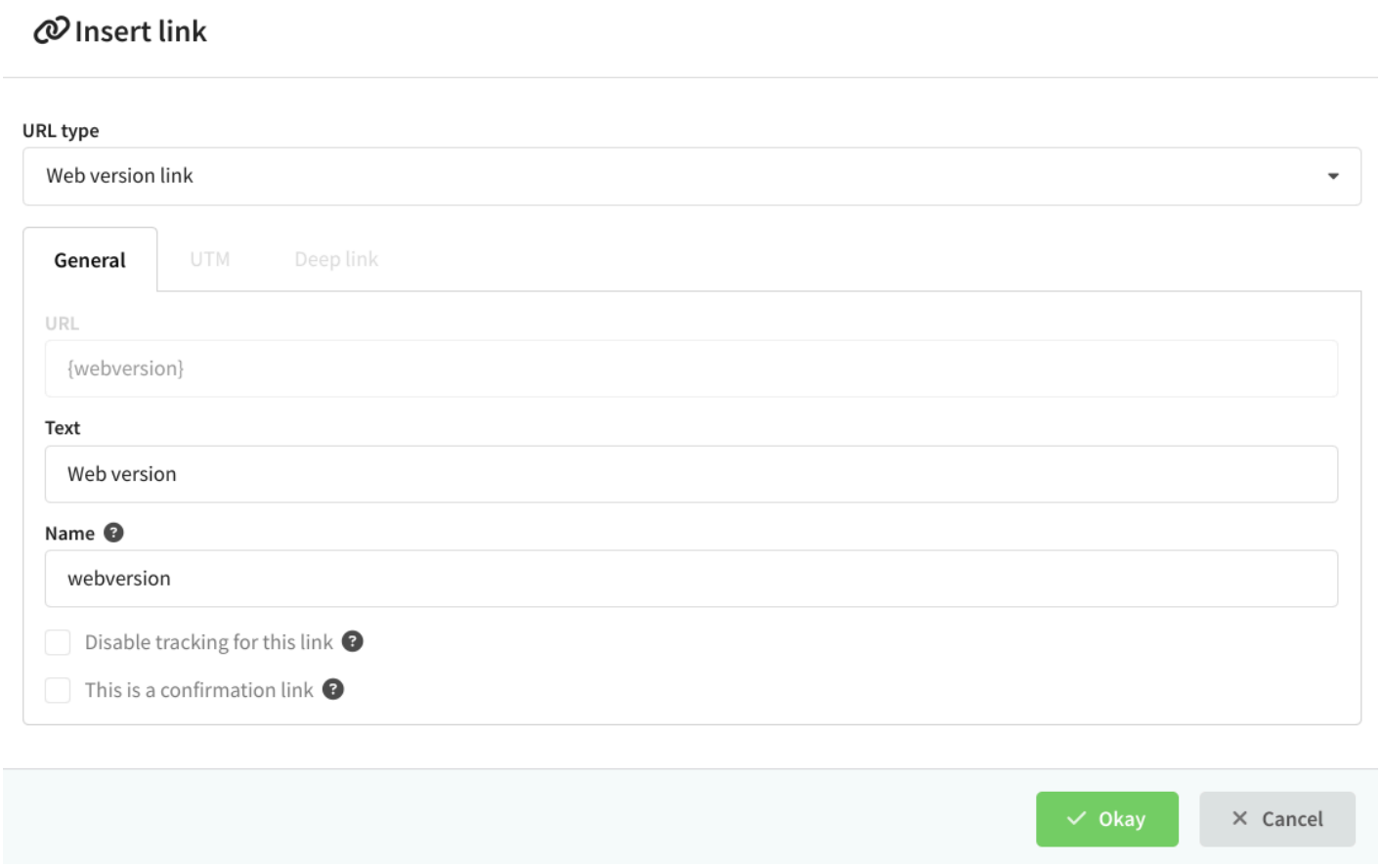
Using the link editor

Place the cursor where you want to add the link. In the toolbar, click the icon for inserting links:



In the window that opens, customize the link:

- **Select URL type - Web version link**
- **Enter link text - this is the text the user will see in the email. For example, “Open the web version of the email”.**
- **Enter link name. Later you will be able to track who used it by building a segment of the customers who clicked on the link.**



Click the Okay button. The link code will appear in the message template.

NOTE

Note that the link to the web version of the email cannot be used as a confirmation link and it is not possible to wrap a tracking link.

Manually

The link to the web version of a message is as follows:

```
<a href="{webversion}" title="Web version of a message" name="web_version"> Open this message in the browser</a>
```

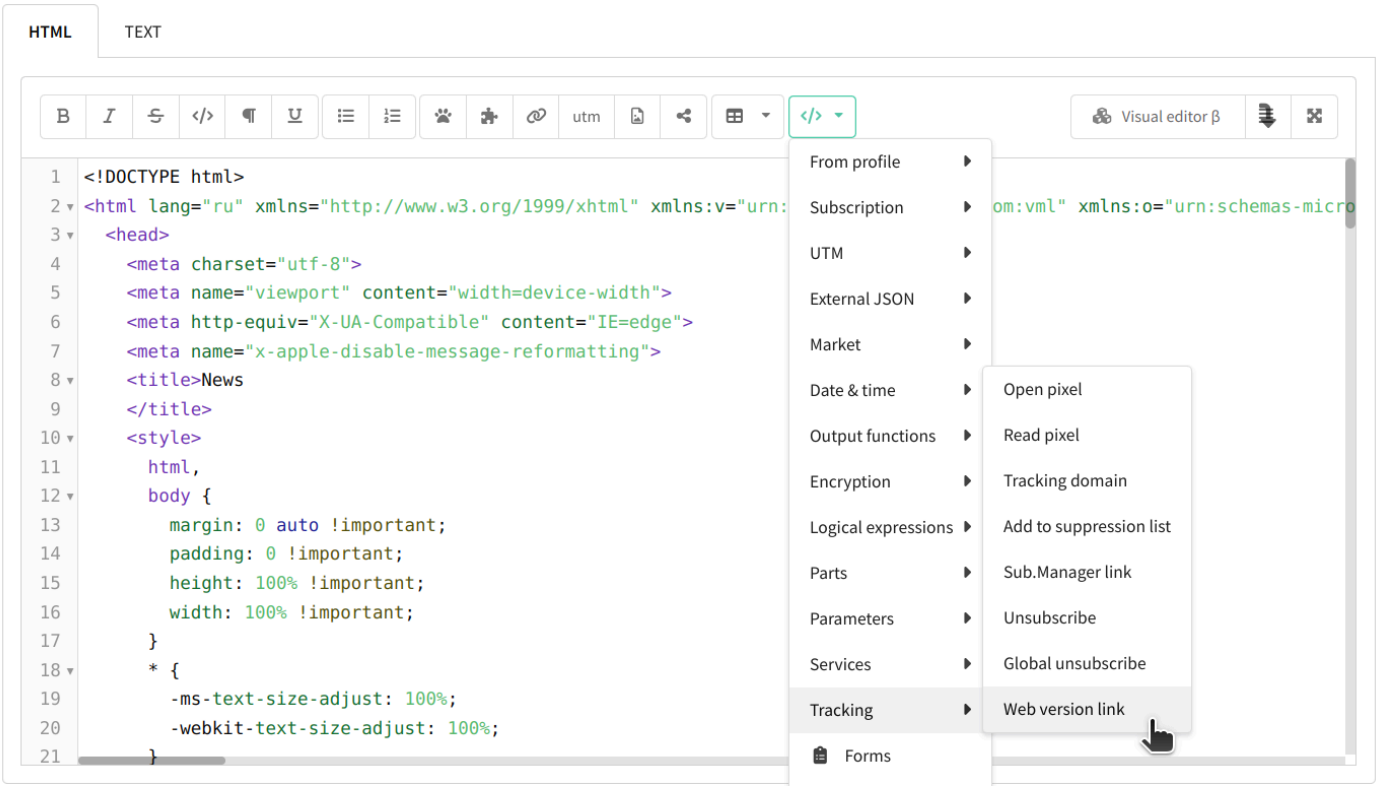
Let’s look at the parts of this link in more detail:

- `<a href="{webversion}"` **- system component;**
- `title="Web version of a message"` **- the title of the link;**
- `name="web_version"` **- link name, further used to get analytics on transitions;**
- **Open this message in the browser - text displayed when pointing at the link;**
- `</a>` **- closing tag.**

You can insert the link in the place you need it.

As a variable

You can create a link to the web version from the variables menu. Open it by clicking on the toolbar. Select Tracking - Web version link:



The `{webversion}` variable will appear in the body of the template:

Next, it is necessary to add additional parameters to this link. How to do this is described in detail in the previous paragraph of the article. As a result, your link will look like this:

```
<a href="{webversion}" title="Web version of a message" name="web_version"> Open this message in the browser</a>
```

AMP Email

Description

AMP technology allows you to add JavaScript components to emails and make them interactive. In an AMP email, a user can fill out a form, buy a product, or confirm registration without going to the site.

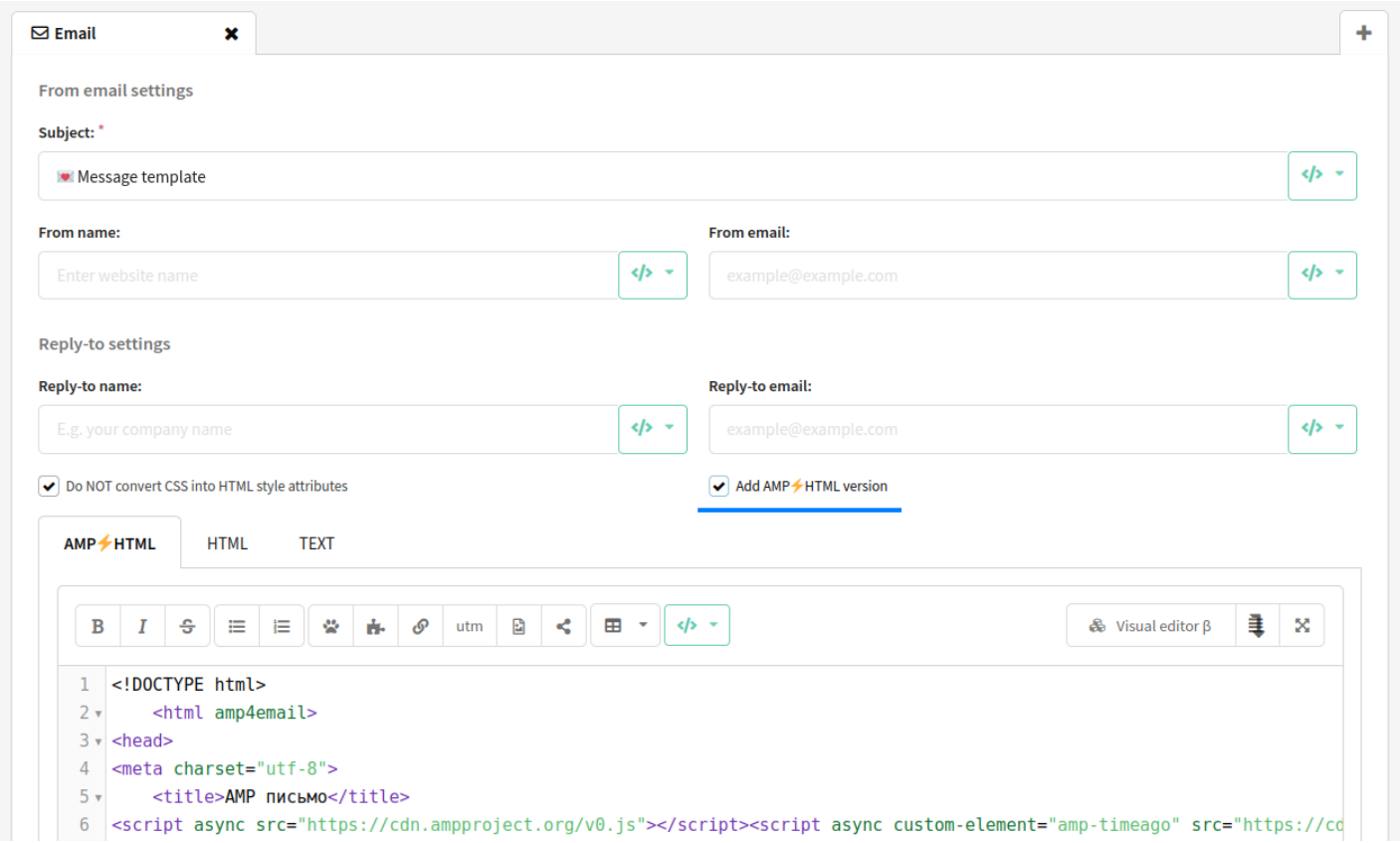
Email clients require sender registration to display AMP emails to users. Follow this guide to get your sender address allow listed on supporting clients.

TIP

This guide will give you everything you need to know to get started writing valid AMP Emails

Adding an AMP version

The standard email template includes HTML and text versions of the email. To send AMP emails, select the Add AMP⚡HTML version checkbox in the template editor. Below you will see an additional tab AMP⚡HTML where you can use AMP components.



TIP

AMP emails are currently supported by the following email clients:

- **Gmail;**
- **Yahoo;**
- **Mail.ru.**

If the email is delivered to an email client that does not support AMP technology, the recipient will see an HTML version.

Links in AMP emails

In the AMP version of the email, instead of the `name` attribute, the `data-name` attribute is used to identify links. When you save the template, the platform will automatically insert the attribute into all links.

```
<a href="https://example.com" title="Link text" data-name="link1">Link text</a>
```

Also note that if you add an AMP parameter to the link, you don't need to wrap the link with the tracking domain. To disable tracking, add the `no-track` attribute. Clicks on such a link will not be tracked.

```
<a href="${{exampleUrl}}" title="Link text" data-name="link1" no-track>Link text</a>
```

Escaping template expressions

AMP template syntax is the same as Altcraft template syntax. To avoid incorrect code parsing, AMP template expressions must be escaped with the $ symbol.

For example, if you are using the `<amp-list>` component, then replace the list element substitution variable `{{title}}` with `${{title}}`:

```
<amp-list id="myList" src="https://foo.com/list.json">
  <template type="amp-mustache">
    <div>${{title}}</div>
  </template>
</amp-list>
```

Importing and exporting a message template

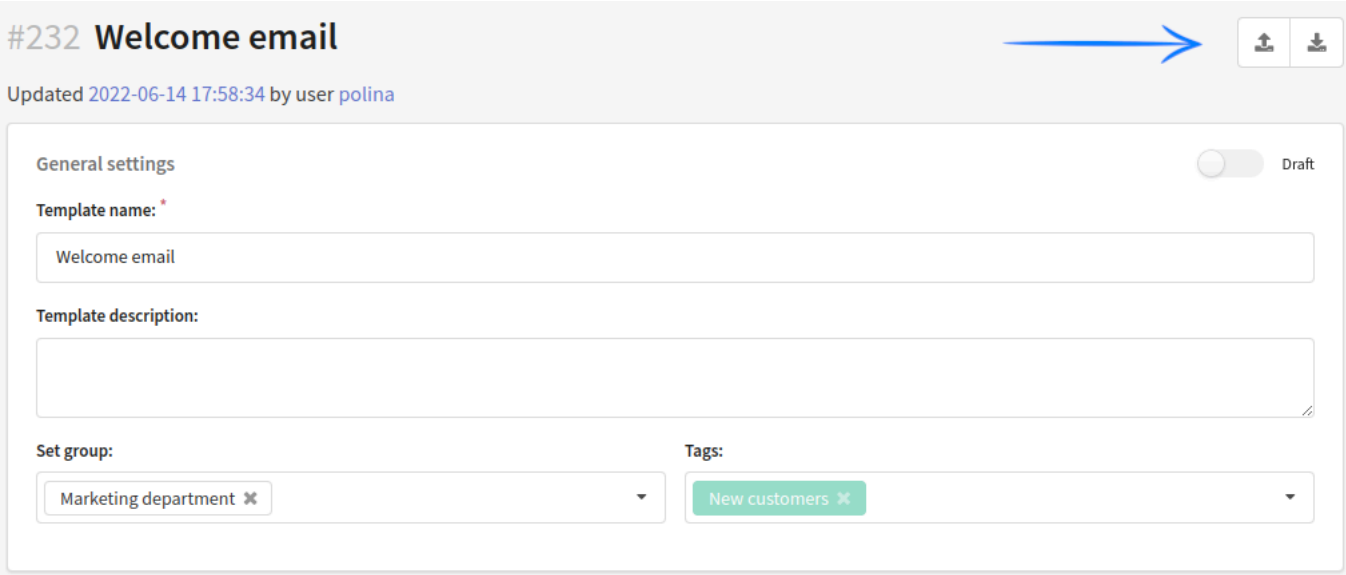
Description

This functionality allows you to upload a message template to the platform or download it to your device. This can be a template for one channel, or for several channels at once. A zip file is used for export and import.

The corresponding buttons are available in the template editing window in the upper right corner:

— template import

— template export



Template import

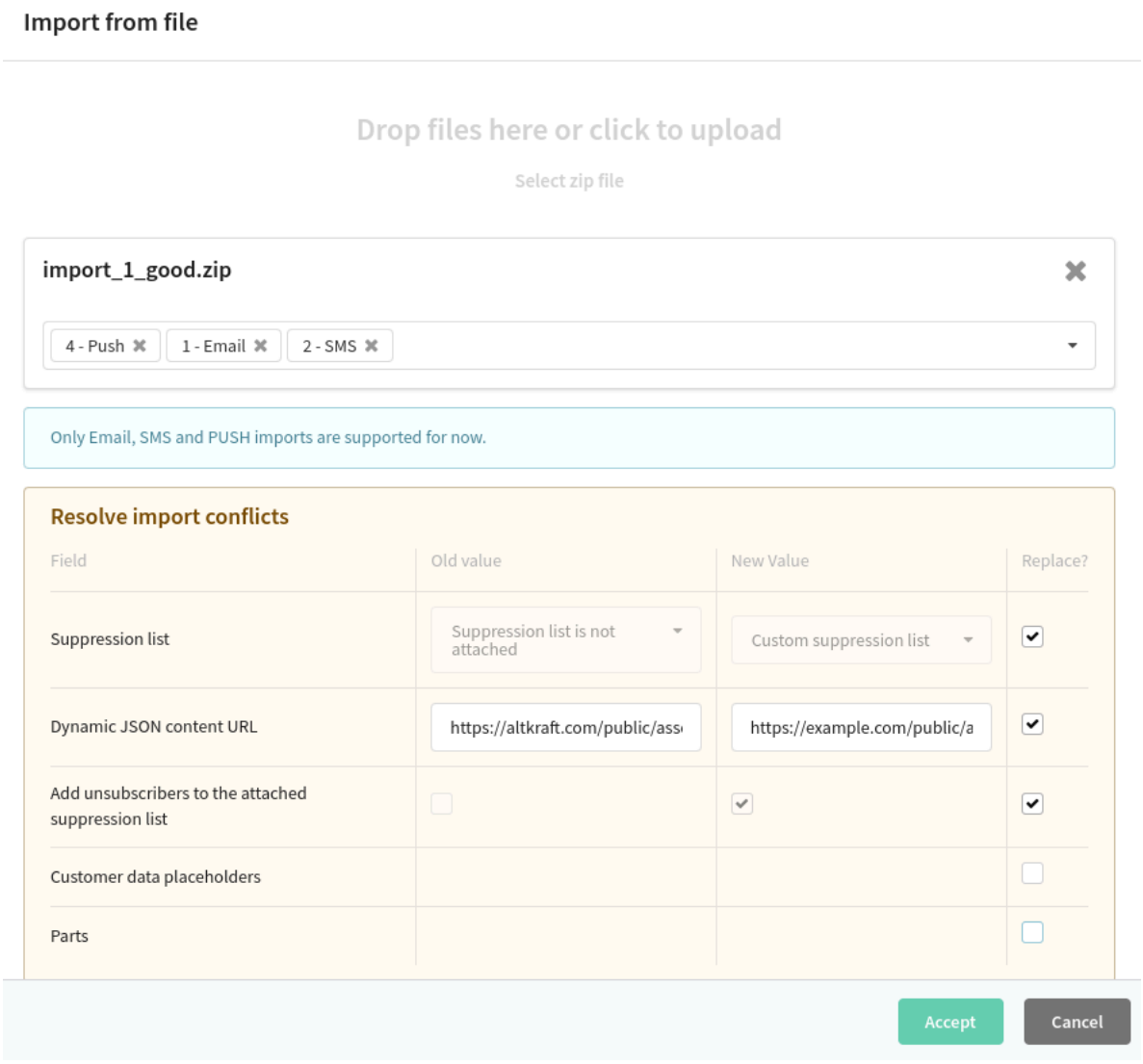
To import a template into the platform, click on the icon

. Next, click on the download area and select the zip file on your device.

Set up the import:

1. **Select the channels for which you want to import the template (Email, SMS, Push).**
2. **Resolve import conflicts. Conflicts arise if the data being imported and the data already stored in the platform are different. The table shows the old and new value. If you want to update the data, check the "Replace" column.**

Click Accept to start the import process.



Zip file structure

A zip file is used for import. The archive must have a certain structure:

```
<catalog_name>/
    <channel_name>/
        <fieldname>.<type>
        <attach>/
            <fileid>/
                <filename>
    images/
        <image_name>
    template.json
```

Structure description:

Element	Example	Description
<catalog_name>	**template_newsletter**	**Root directory. You can use an arbitrary name.**
<channel_name>	**email** **sms** **push**	**The directory contains template data for a specific send channel. The directory name must match the channel.** **Import of Email, SMS and Push templates is available.** **Information about which files can be imported for each channel can be found here.**
images	**image_1.png**	**Directory for storing images**
template.json	—	**The file contains general data about the template (link to dynamic JSON content, additional settings, fragments and parts).** **You will find more detailed information below.**

Zip file example

TIP

Starting from version 68, images can be stored either at the same level as the HTML file or in a folder with one of the following names: images, image, img, imgs, pictures, pics. An archive containing such an email template should not include files like text.txt, subject.txt, or template.json, as their presence may cause import errors.

Basic structure for an email channel

```
template_123/
    email/
        html.html
        text.txt
        subject.txt
    images/
        image_1.jpeg
    template.json
```

Full structure for all channels

```
template_123/
    email/
        html.html
        text.txt
        subject.txt
        amp.txt
        is_amp.txt
        is_not_inline.txt
        from_email.txt
        from_name.txt
        replyto_email.txt
        replyto_name.txt
        attach/
            023f-ad8c-4412-9bba-16ac/
                attach_file.zip
    sms/
        sms.txt
        add_sms_url_scheme.txt
    push/
        title.txt
        body.txt
        click_url.txt
        icon.txt
        is_rich_push.txt
    images/
        image_1.jpeg
        image_2.jpeg
    template.json
```

Image paths in HTML

To ensure correct image rendering in an imported template, use relative paths that match the archive structure.

Recommended structure:

```
template_name
├── email
│   └── html.html
└── images
    └── banner.png
```

Example of a correct path in HTML:

```
<img class="img" src="../images/banner.png" alt="Banner">
```

Description of fields for each channel

Email

File	Example	Description
html.html	`<html>Hello, world!</html>`	**HTML version of the message**

File	Example	Description
text.txt	**Hello, world!**	**Text version of the message**
subject.txt	**Best books 2022**	**Subject line**
amp.txt	`<html>Hello, world!</html>`	**AMP HTML version of the message**
is_amp.txt	**false**	**Is there an amp version in the template or not**
is_not_inline.txt	**true**	**Do not convert CSS to inline**
from_email.txt	**awesome@example.com**	**Sender's email**
from_name.txt	**Awesome Books**	**Sender's name**
replyto_email.txt	**reply_awesome@example.com**	**Reply-to email**
replyto_name.txt	**Awesome Books**	**Reply-to name**
attach (directory)	**023f-ad8c-4412-9bba-16ac/ attach_file.pdf**	**Directory for storing attached files. It contains additional directories, each containing a separate file. Arbitrary names can be used to name additional directories.**

SMS

File	Example	Description
sms.txt	**Hello, world!**	**Message text**
add_sms_url_scheme.txt	**true**	**Add "http://" to links**

Push

File	Example	Description
title.txt	**Best books 2022**	**Notification title**
body.txt	**Hello, world!**	**Notification text**
click_url.txt	**https://example.com/**	**Click-to-click link**
icon.txt	**https://example.com/images/test.jpg**	**The URL where the notification icon is located**
is_rich_push.txt	**false**	**Use rich version for push message or not**
<platform>.json	`<html>Hello, world!</html>`	**Templated data for a specific browser or OS (when using rich push)**

template.json file

This file contains data that is displayed in the "Options" and "Content" blocks.

template.json example

```
{
  "json_url": "https://altkraft.com/example/content.json",
  "suppress_id": 1,
  "is_unsub_to_supp": true,
  "field_overrides": {
    "lead": {
      "_fname": "client"
    }
  },
  "parts": []
}
```

Description of fields in template.json

Field	Example	Required	Description
json_url	**"https://altkraft.com/example/content.json"**	**No**	**Dynamic JSON content URL**
suppress_id	**1**	**No**	**Suppression list ID**
is_unsub_to_supp	**true**	**No**	**Add unsubscribers to the attached suppression list**
field_overrides	`{ "lead": { _fname": "Client" } }`	**No**	**Customer data placeholders**
parts	`[ { "name": "part_1", "shortname": "part_1", "type": "text", "content": "" }, { "name": "part_2", "shortname": "part_2", "type": "text", "content": "" } ]`	**No**	**Template parts**

TIP

You can export any template from your account to get an example of a zip file to import.

Importing a template from a third-party service

Description

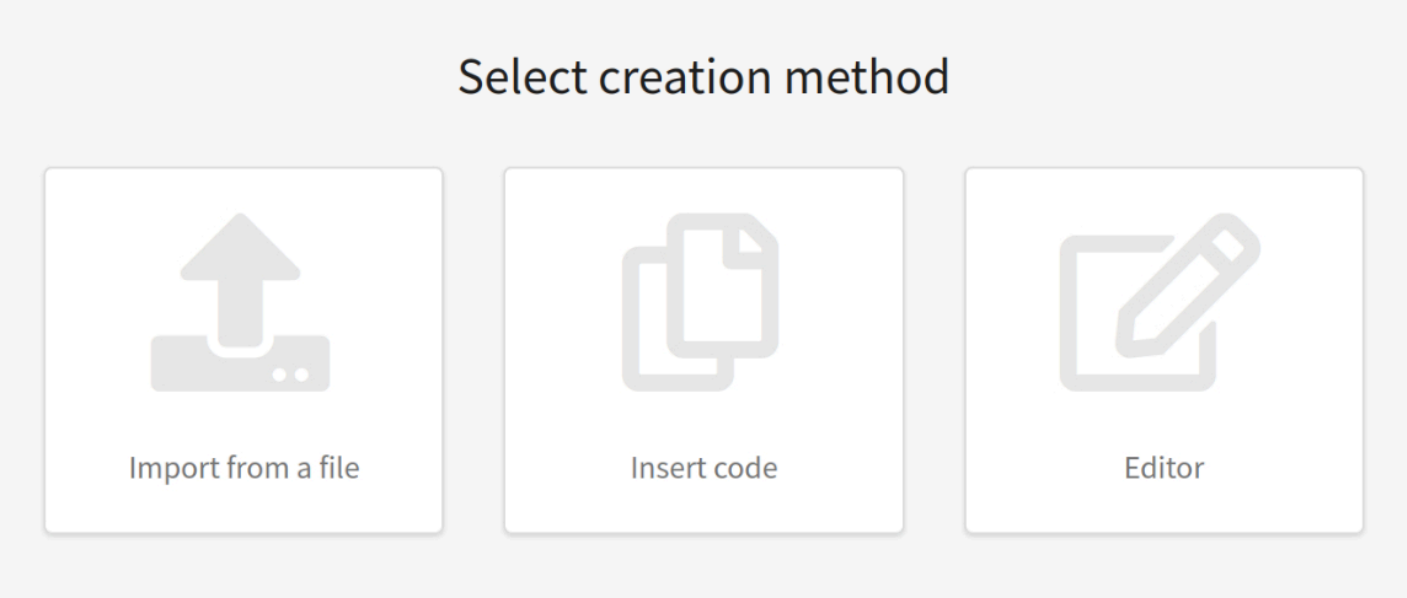
If you have ready-made message templates in other services, you can move them to Altcraft Platform. You can import both Email and SMS templates.

TIP

Import is available for the following services:

- **Unisender**
- **Mailchimp**
- **Sendsay**

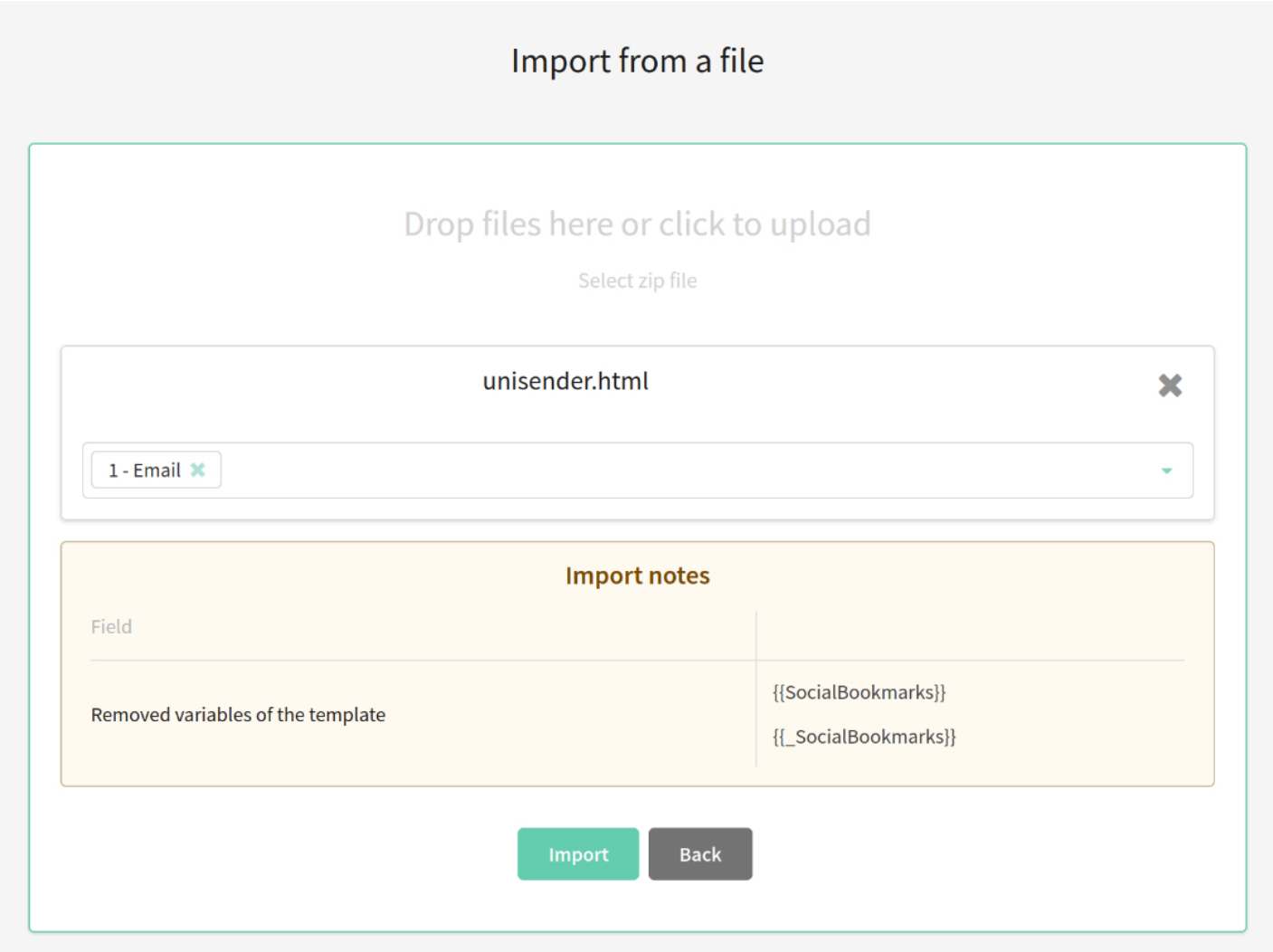
To get started, go to the "Templates section and click the Create button on the top panel. Next, select the creation method: import the template from a file or insert the code.



Import from file

If you want to move an Email template to the platform, you must first export the html file with the ready-made template from a third-party service. To move an SMS template, you need a file in txt format.

In the platform, click on the upload area and select the file on your device. After uploading, you will see a table with notes on importing. All variables removed from the template will be listed here.



TIP

Each service uses its own variables in message templates. In our platform, they look like this `{lead.email}`.
When importing a template from a third party service, Altcraft will automatically replace some variables. If there is no similar variable in the platform, it will be removed from the template.
See below for information on which variables will be saved.

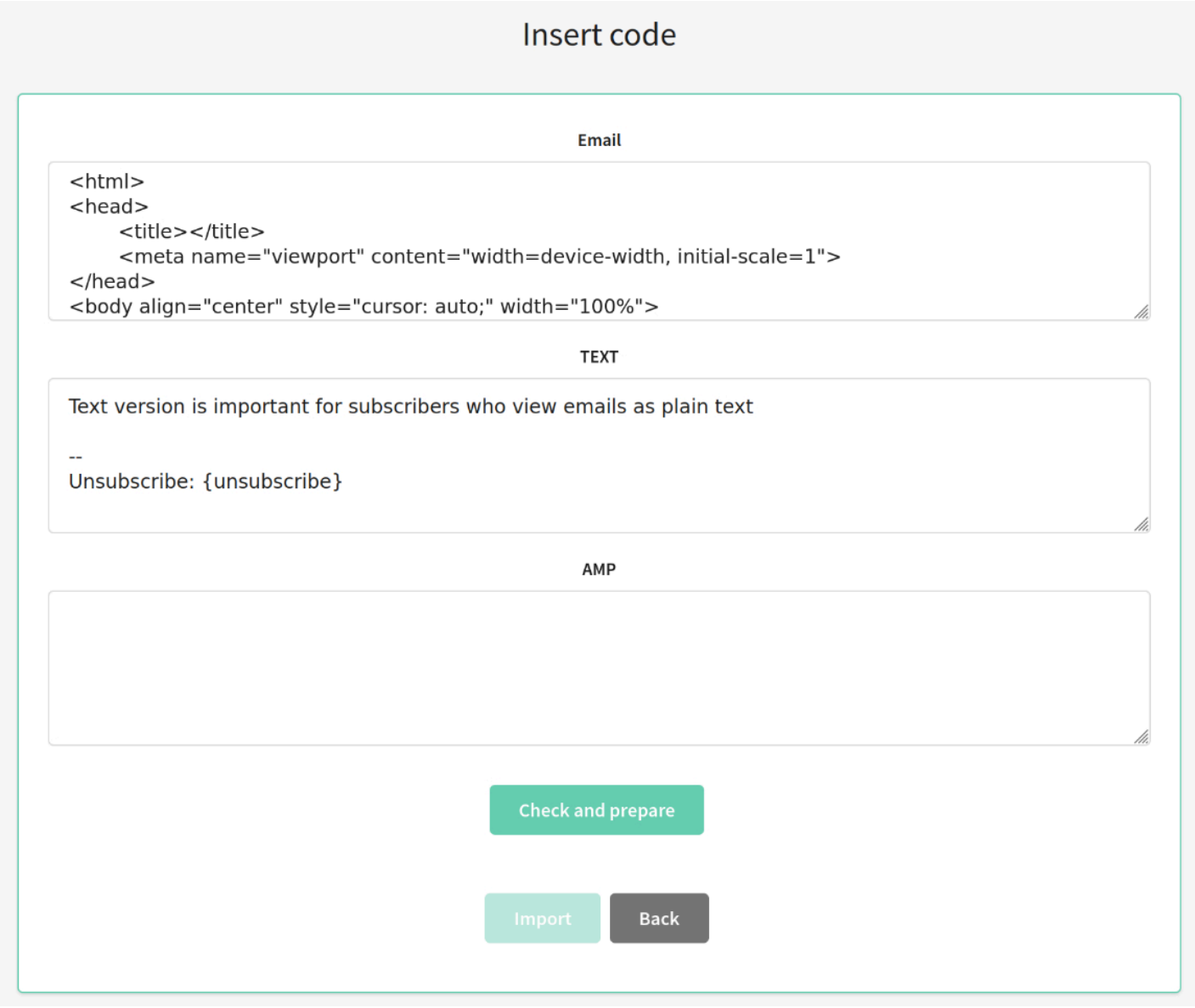
Insert code

When you create a template by inserting a code, the first thing you will be asked is to select a channel (Email or SMS).

If you choose Email, you will need to copy the template html code from the editor in a third party service. Next, paste the copied code to the Email text box. The text version of the letter will be generated automatically.

If necessary, you can manually add code for each version of the email:

- **Email — html version of email.**
- **Text — text version of email.**
- **AMP — amp version of email.**



Importing a template for an SMS channel is easier. You only need to fill in one text box.

After pasting the code, run its verification by clicking the Check and prepare button. The import notes table will also appear at the bottom, listing all the removed variables. Click Import".

Auto-replacement of variables in a template

Altcraft Platform automatically replaces third-party service variables with its own variables:

Unisender

Variable description	Altcraft	Unisender
First name	`{lead._fname}`	`{{Name}},` `{{_LocalPart}},` `{{LocalPart}}`
Subscription email address	`{subscription.email}`	`{{_Email}},` `{{Email}}`
Subscription phone	`{subscription.phone}`	`{{_Phone}}`
Email sender name	`{from_name}`	`{{_ListTitle}},` `{{ListTitle}}`
Custom profile field	`{lead.custom_field}`	`{{custom_field}}`
Confirmation link	`<a href="#" title="Confirm your subscription" confirm-link name="link1">Confirm your subscription</a>` **When importing the text version of the email, the confirmation link is replaced like this:** `[Confirm your subscription](#<confirm-link>)`**. You need to replace # with the URL.**	`{{ConfirmUrl}}`
Resource unsubscribe or global unsubscribe link	`{unsubscribe}`	`{{_UnsubscribeUrl}},` `{{UnsubscribeUrl}}`
Link to the email web version	`{webversion}`	`{{_WebLetterUrl}},` `{{WebLetterUrl}}`

Variable description	Altcraft	Unisender
Full date	`{datenow}`	`{{SendDate}}`

Sendsay

Variable description	Altcraft	Sendsay
First name	`{lead._fname}`	`[% anketa.base.firstName %]`
Last name	`{lead._lname}`	`[% anketa.base.lastName %]`
Subscription email address	`{subscription.email}`	`[% anketa.base.email %]`
Email subscription domain	`{subscription.email_domain}`	`[% anketa.member["confirm.host"] %]`
Gender	`{lead._sex}`	`[% anketa.base.gender %]`
City	`{lead._city}`	`[% anketa.base.city %]`
Date of birth	`{lead._bdate}`	`[% anketa.base.birthDate %]`
Custom profile field	`{lead.custom_field}`	`[% anketa.base.custom_field %]`
Condition	`{if lead.custom_field}{else}{end}`	`[% IF anketa.base.custom_field %][% ELSE %][% END %]`
Variable length	`{length(lead.custom_field)}`	`[% size(anketa.base.custom_field) %]`

Array iteration	`{for $index $i = items}`
	`  {$i}`
	`{end}`
		`[% FOREACH i IN items %]`
		`  [% i %]`
		`[% END %]`

Mailchimp

Variable description	Altcraft	Mailchimp
First name	`{lead._fname}`	***|FNAME|***
Last name	`{lead._lname}`	***|LNAME|***
Subscription email address	`{subscription.email}`	***|EMAIL|***
Subscription phone	`{subscription.phone}`	***|PHONE|***
Registration date	`{lead._regdate}`	***|OPTIN_DATE|*,** ***|OPTIN_DATETIME|***
Email sender name	`{from_name}`	***|LIST:DESCRIPTION|*,** ***|LIST:NAME|***
Sender's email	`{from_email}`	***|LIST:EMAIL|*,** ***|ABUSE_EMAIL|***
Custom profile field	`{lead.custom_field}`	***|custom_field|***
Confirmation link	`<a href="#" title="Confirm your subscriptiony" confirm-link name="link1">Confirm your subscription</a>` **When importing the text version of the email, the confirmation link is replaced like this:** `[Confirm your subscription](# <confirm-link>)` **. You need to replace # with the URL.**	***|LIST:SUBSCRIBE|***

Resource unsubscribe or global unsubscribe link	`{unsubscribe}`	***|UNSUB|*,**
		`{{unsub}}` **,**
		`{{unsub "http://example.com/unsub"}}`
		`{{unsub redirect_merge_var}}`

Link to the email web version	`{webversion}`	***|ARCHIVE|***
Full date	`{datenow}`	***|DATE:d/m/y|*** `{{date "Y-m-d"}}`
Full year	`{yearlong}`	***|CURRENT_YEAR|***
Resource ID	`{resid}`	***|LIST:UID|***
Mailing SUBID	`{subid}`	***|CAMPAIGN_UID|***
URL of a website, form, or application from resource settings	`{resurl}`	***|LIST:URL|***
Uppercase	`{uppercase("your text")}`	`{{upper "your text"}}`
Upper case for first character	`{uppercasefirst("your text is neat")}`	`{{title "your text is neat"}}`
Condition	`{if lead.custom_field}{else}{end}`	***|IF:custom_field|*...*|ELSE:|*...*|END:IF|***

TIP

If you would like to extend the variable substitution syntax, please send a request to support@altcraft.com.

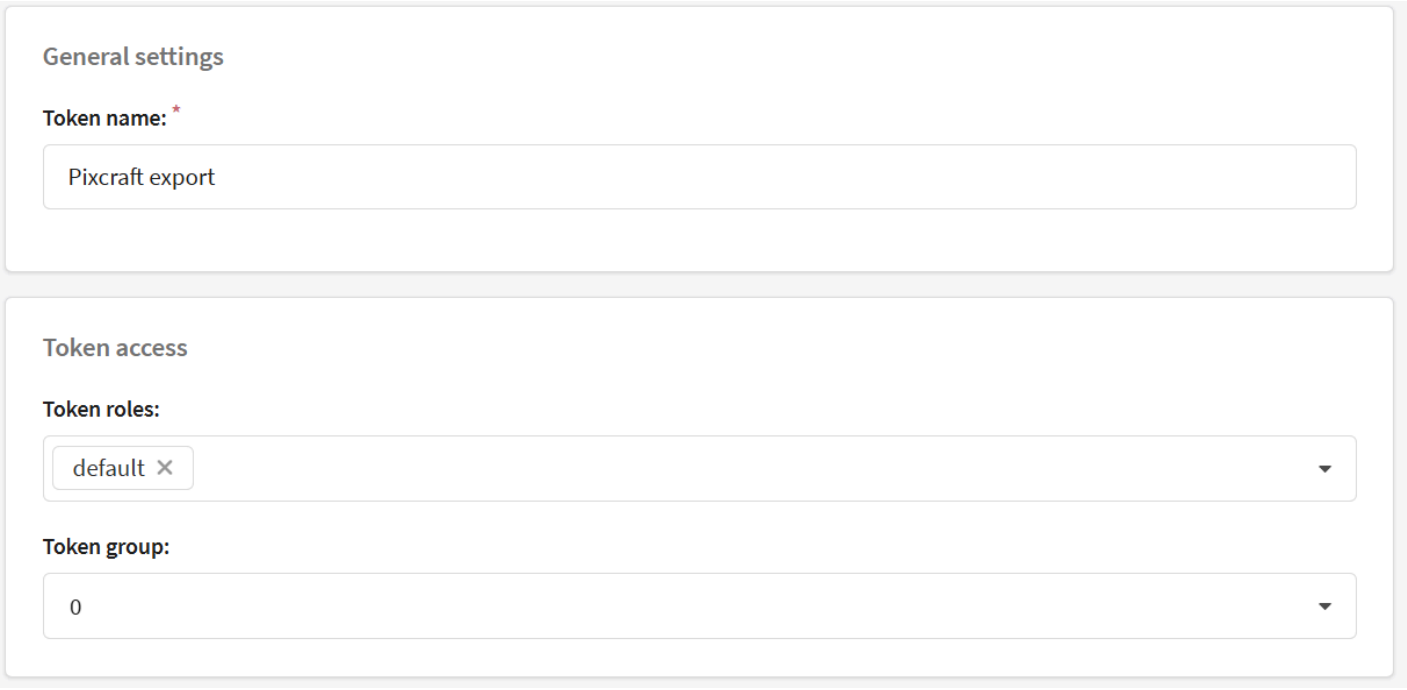
Exporting a template from Pixcraft

To export from Pixcraft, you need to create an API token.

Go to Settings/Tokens:

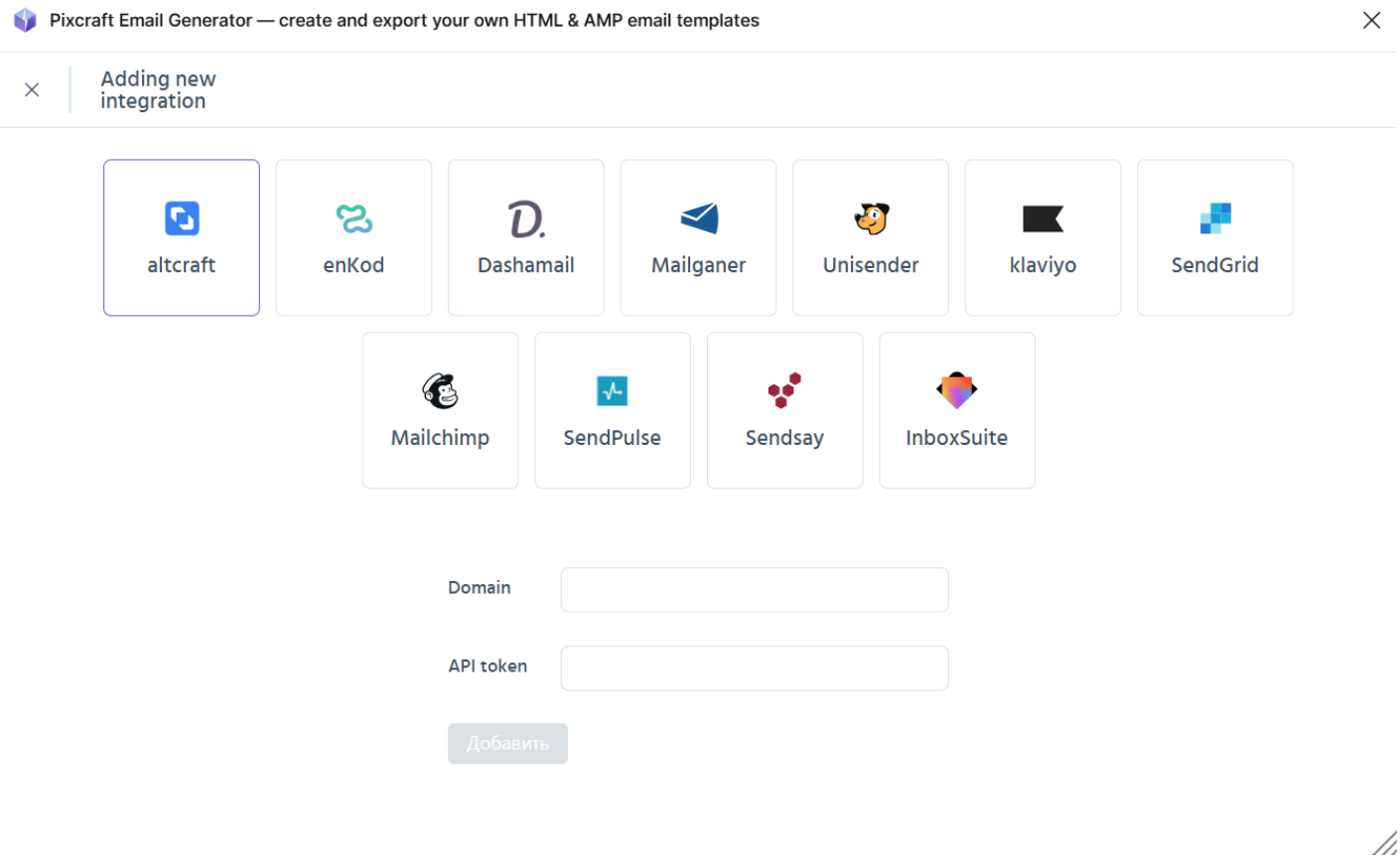
Click Create, specify a token name, for example, Pixcraft export.

Add a role that has permissions to create and edit templates.



Click Save.

Copy the API token and specify it when adding the integration to Pixcraft. To do this, in Pixcraft (Figma plugin) go to the Export to ESP tab and select Altcraft:

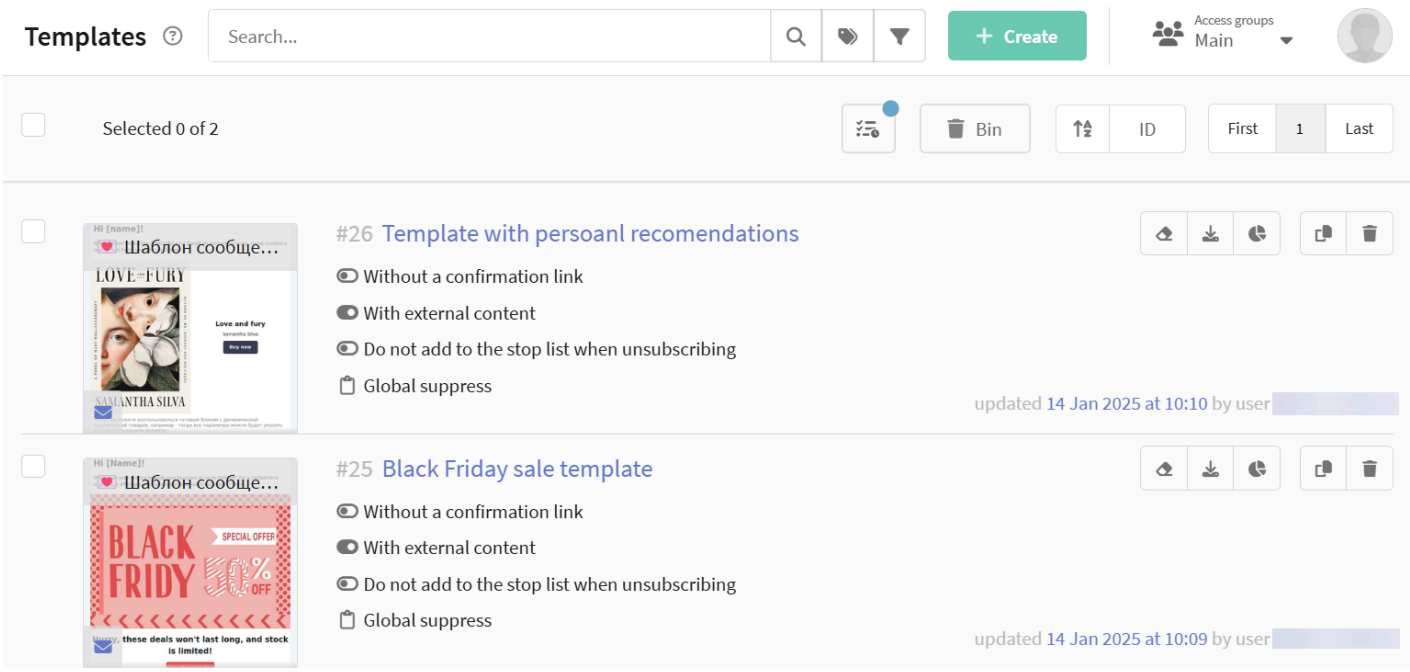


After adding the integration, select it from the list and click Export. The browser will open the exported email in the Altcraft interface.

Working with message templates

	To view existing templates and add new ones, open the **Templates section in the main menu, then go to the item of the same name:**

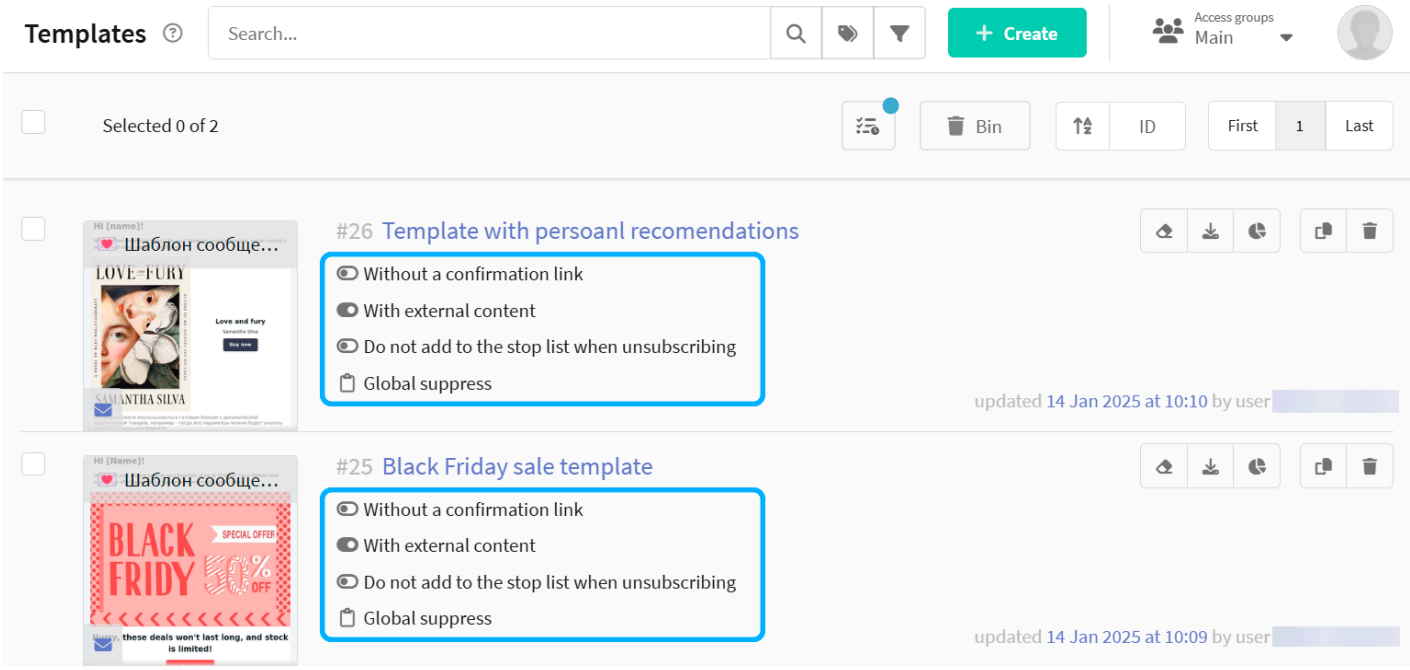
The list shows all of your message templates available in the account or in the access group. To find a template in the general list, enter the template name or its ID in the search bar. You can also filter templates by tags, sending (communication) channel. You can also display only drafts or ready-made templates.



Click on the template and it will open the message preview. Here you can check how the message will look in different channels and on different devices.

General information for each template is:

- **The presence of a confirmation link — a template with such a link is required when using Double opt-in subscription.**
- **The presence of external content — the use of links to dynamic JSON content in the template.**
- **Adding to the stop list when unsubscribing — if the subscriber clicks the unsubscribe link in the message template, he will be included in the selected stop list.**
- **Suppression list - a mailing using a template will block sending an email to a subscriber whose contacts are in the specified suppression list.**



A quick action menu is available:

— draft mode.

— export the template.

— go to the template report.

— clone the template.

— delete the template.

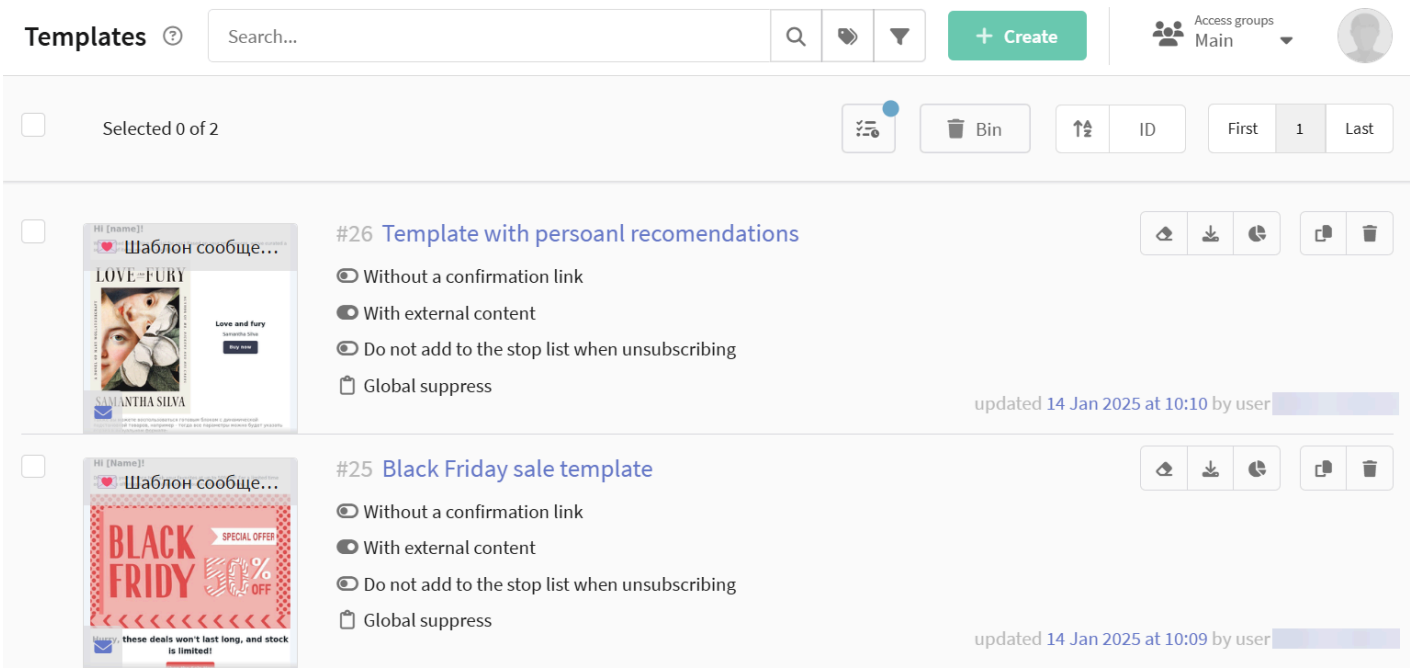
To add a new template, click the + Create button at the top of the page. If you need to edit an existing template, click on its title.

Read more about working with the template editor and about templates for each communication channel in the following articles:

Working in the editor

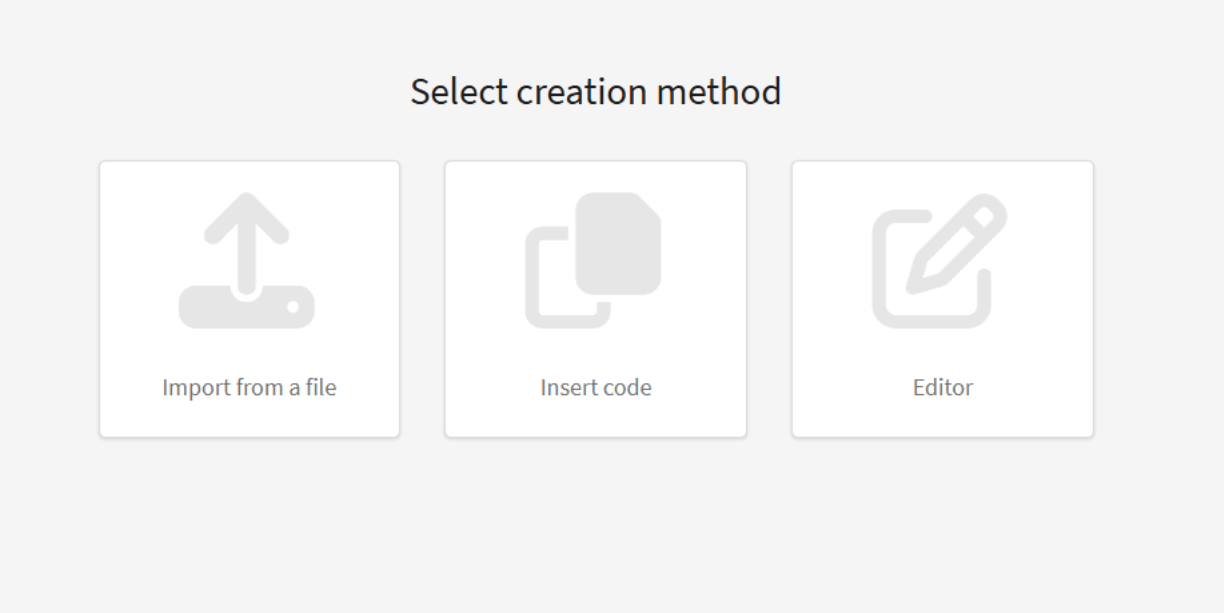
Creating a template

To add a new template, click on the + Create button at the top of the page:



Next you have to select the way to create a template:

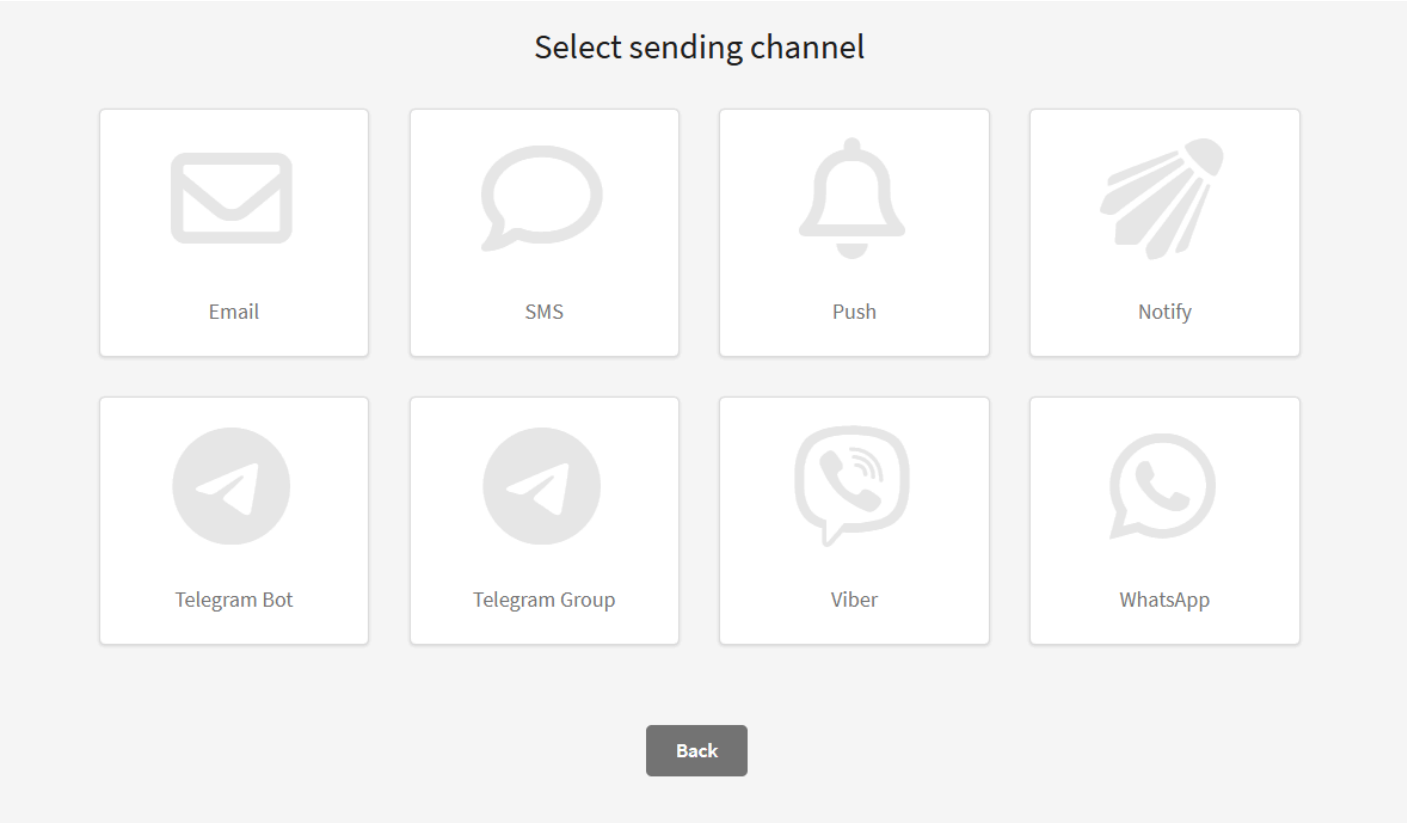
- **Import from a file — download the finished template as a zip archive or html file.**
- **Insert code — add the copied HTML code to the text window.**
- **Editor — create a template in the editor.**



TIP

If you need to transfer a template from a third-party service, use file import or code insertion.

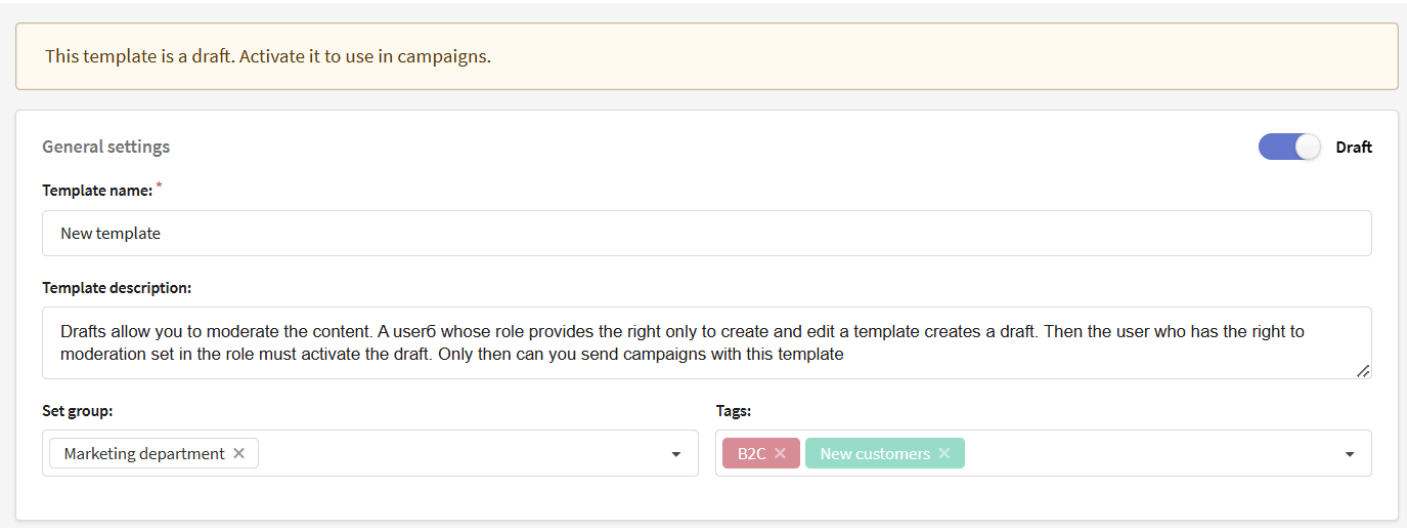
Select sending channel, or communication channel. In the future, the selected channel can be changed or a new one can be added.



General settings

In the general settings of the template, specify its name and description, add tags and if necessary set access groups.

If it is necessary to organize content moderation, the templates provide the function of creating Drafts. To do this, you need to create a user with a specific role. The role have to allow only create messages, without moderation. User with that role will be able to fill templates with content, but templates can not be used immediately in newsletters. The second user with moderation rights must open the template and deactivate the Draft option in the General settings.



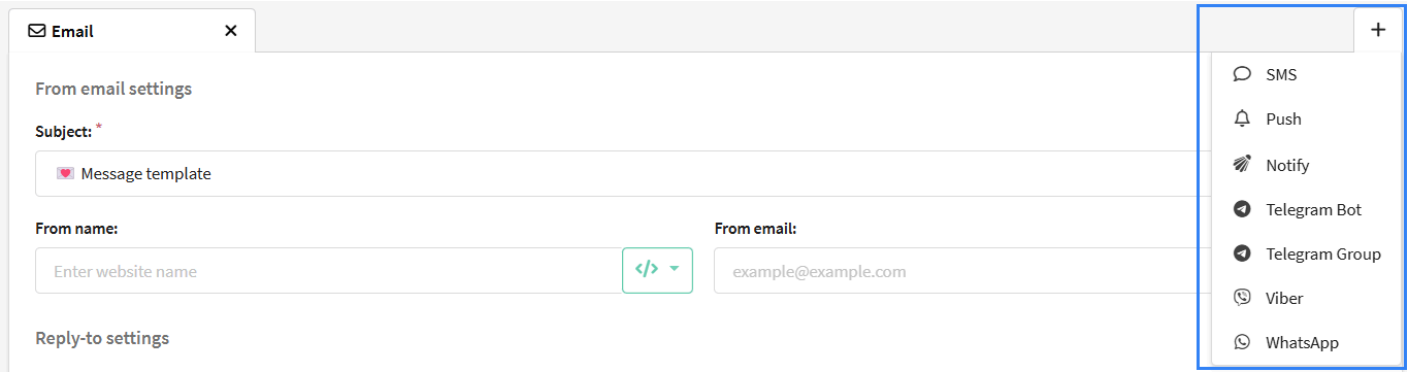
Content editor

The editor helps to create messages, quickly edit templates and also simplifies the personalization of content.

A single template can combine content for sending over all available communication channels. You define the communication channel at the creation's stage, but you can change it or add a new one. To add content to the template for a specific communication channel, click on the button

+

in the upper right corner:



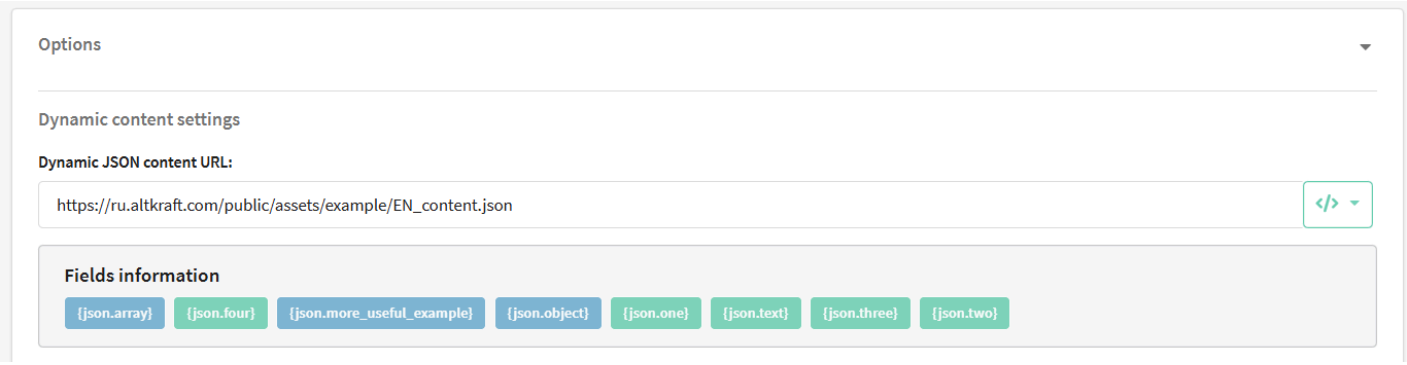
Working with each type of content is described in the articles:

- **Email;**
- **SMS;**
- **Push;**
- **Telegram;**
- **WhatsApp;**
- **Viber™;**
- **MAX;**
- **Notify.**

Options

In the Options section, you can continue customize the template.

In the Dynamic content settings block, specify the data source for JSON content in messages. If there is valid JSON at the entered address, a preview of the first-level objects will open:



CAUTION

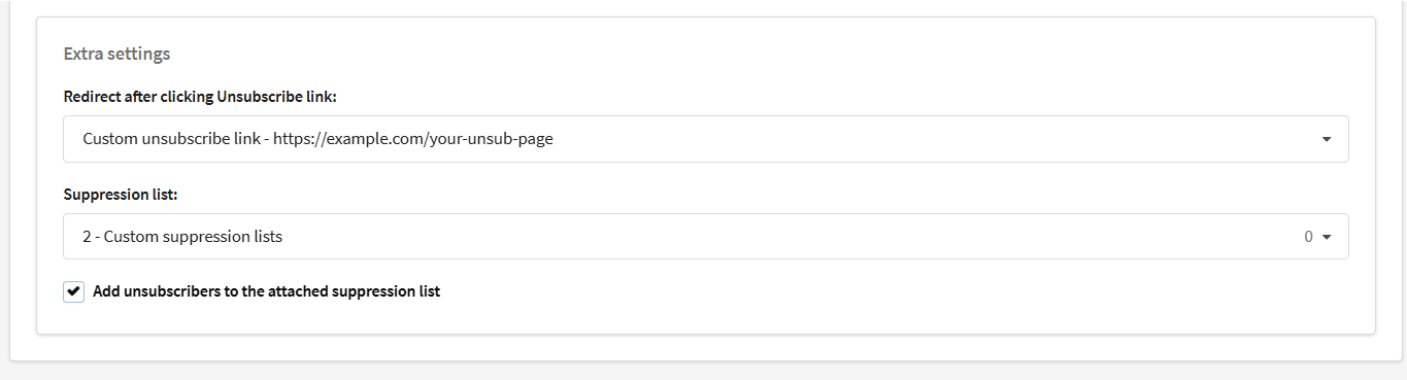
By default, the maximum size of dynamic JSON is 1 MB. In the on-premises version, this limit is set in the configuration file using the `MAX_REMOTE_CONTENT_BYTES` parameter.

In the extra settings you can select custom links for the message unsubscribes and subscription confirmations. For example, if you want to direct customers to the personal account of the site or to the page for subscriptions management.

TIP

User links unsubscribes and confirmations can be created by master account users. In the template options, it is only possible to select pre-created links. For more information, see the Settings section.

Also in the extra settings you can select a custom suspression list to use with the template. With this feature, it is easy to limit sending to customers who are not suited to the type of content being sent. Additionally, you can enable the option Add unsubscribers to the attached suspression list:

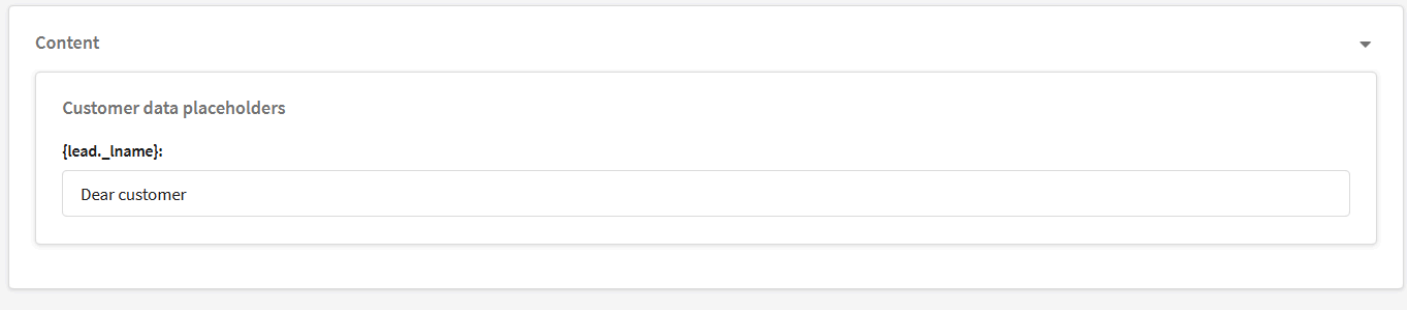


Default field values and parts

TIP

The Content tab appears in the editor when you use variables with [profile data] or parts of messages in the template.

In the tab Content you can define default values for variables that you use in the template. So, if you use customer name substitution, but for some profiles the name field is not filled in, the default value will be used, for example, “Dear customer”.

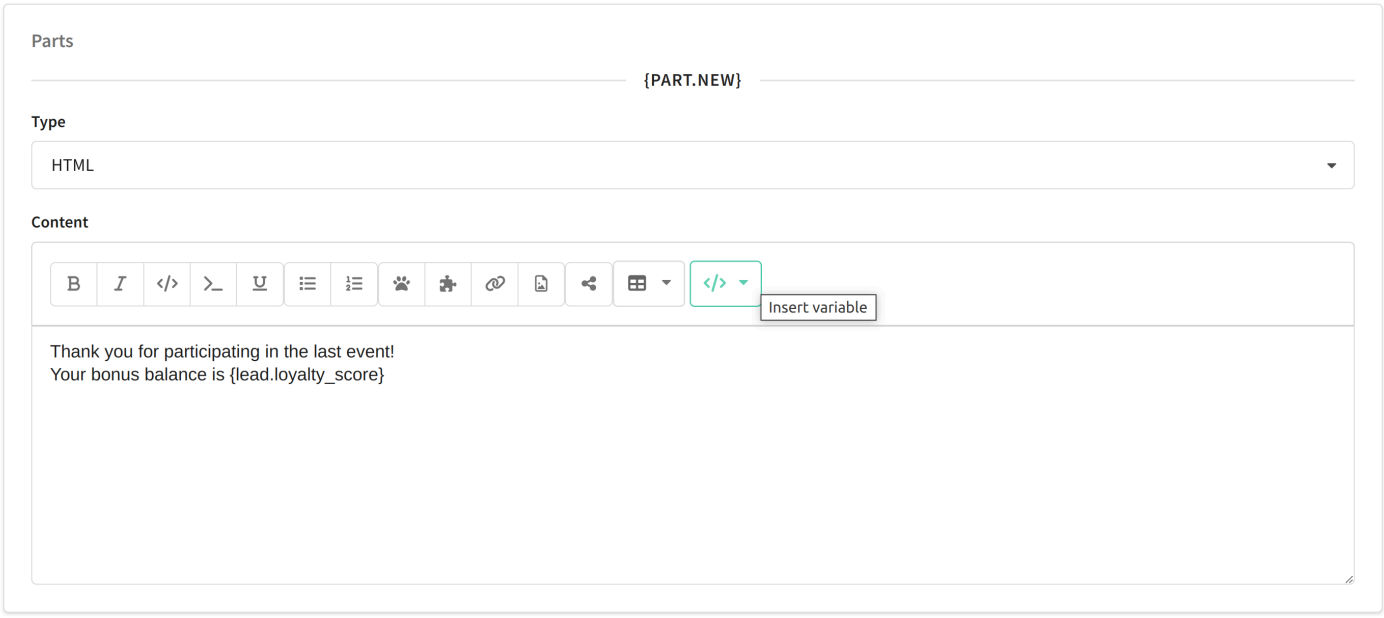


Parts of the message are similar to [fragments]. However, they can only be used in the template where they were created. Fragments can be used in several templates.

TIP

To start working with the parts, add the variable `{part.example_part}` to the template code. After that, the “Content” tab will appear at the bottom of the page, where you can customize parts of the template.

For each part, you can select the content type: HTML for email messages or text for the text version of email, as well as for SMS and Push content. Depending on the type of content, the corresponding content editor will be opened with all the necessary functions:



TIP

It is convenient to use parts together with logical expressions for a more compact display of the message code. For example, depending on the number of subscriber points (custom numeric field), one or the other part will be substituted:

```
{if lead.score gte 9000}
{part.part_1}
{else}
{part.part_2}
{end}
```

Message preview

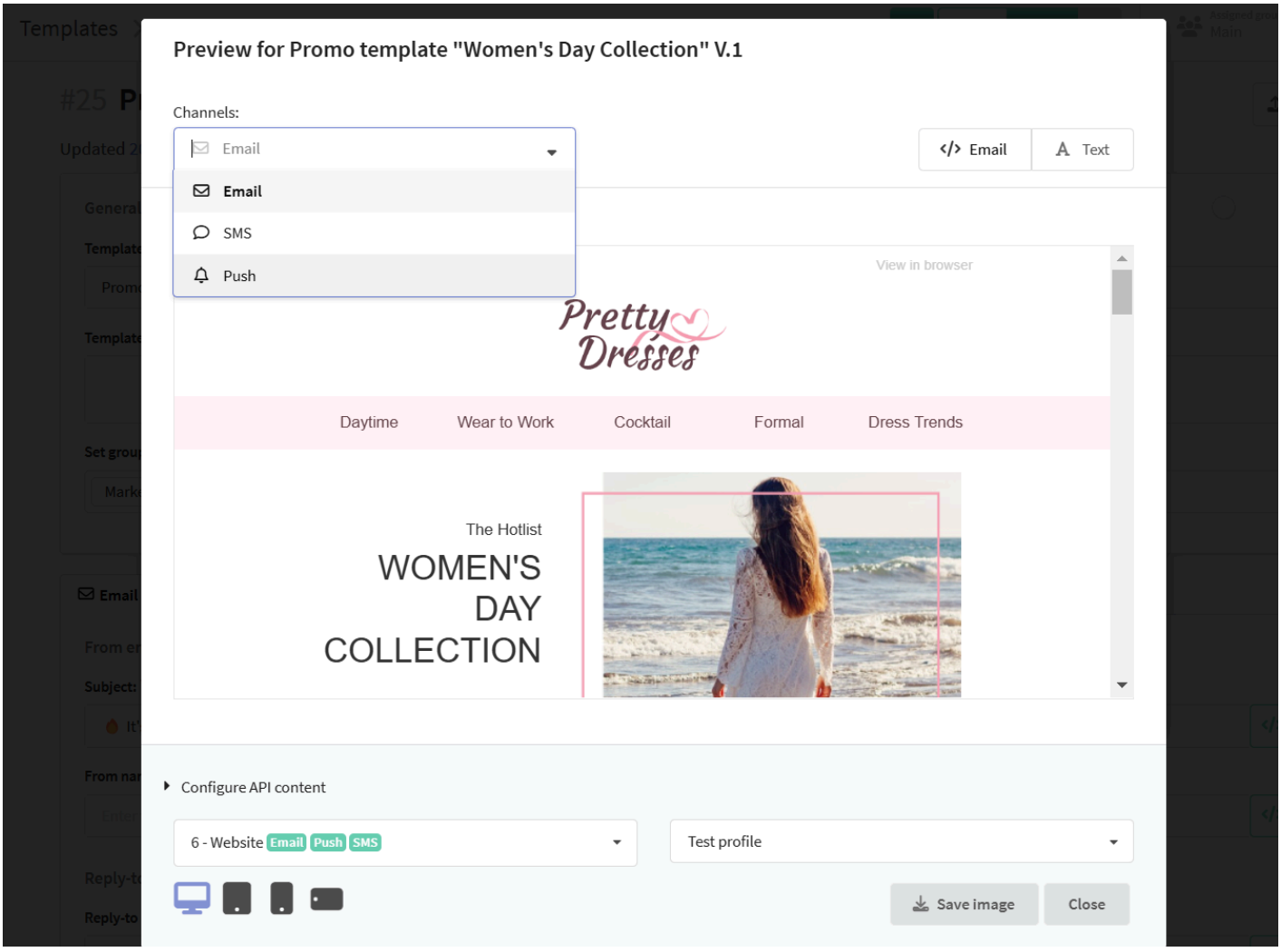
Previewing the message allows you not only to check the HTML layout, but also to see the results of content personalization, which is based on subscriber data. Click on this button at the top of the page to activate the preview:

In the preview window, you can select the desired channel for which you want to view the content. A preview of both the HTML and text versions is available for email.

At the bottom of the window there are buttons for switching desktop/mobile display modes, as well as personalization settings.

TIP

Select the resource from which you plan to send the template, and add a new profile in the drop-down menu. The search is similar to the database search. The segmentation rules are used there, so you can immediately select a profile with the necessary data.



A preview of the template in `png` image format is available for any channel. To download an image to your device, click on Save image.

Follow the rules to correctly upload the preview of the email template:

1. **CSS styles `text-decoration`, `text-decoration-line` and `text-decoration-color` are installed directly into the parent element of the text. For example:**

```
<div>
  <span style="text-decoration: line-through;">Text</span>span>
</div>
```

2. **If you use fonts imported from an external source (for example, Google Fonts), it is recommended to replace them — set the fonts installed in the system in the `font-family` property. For example:**

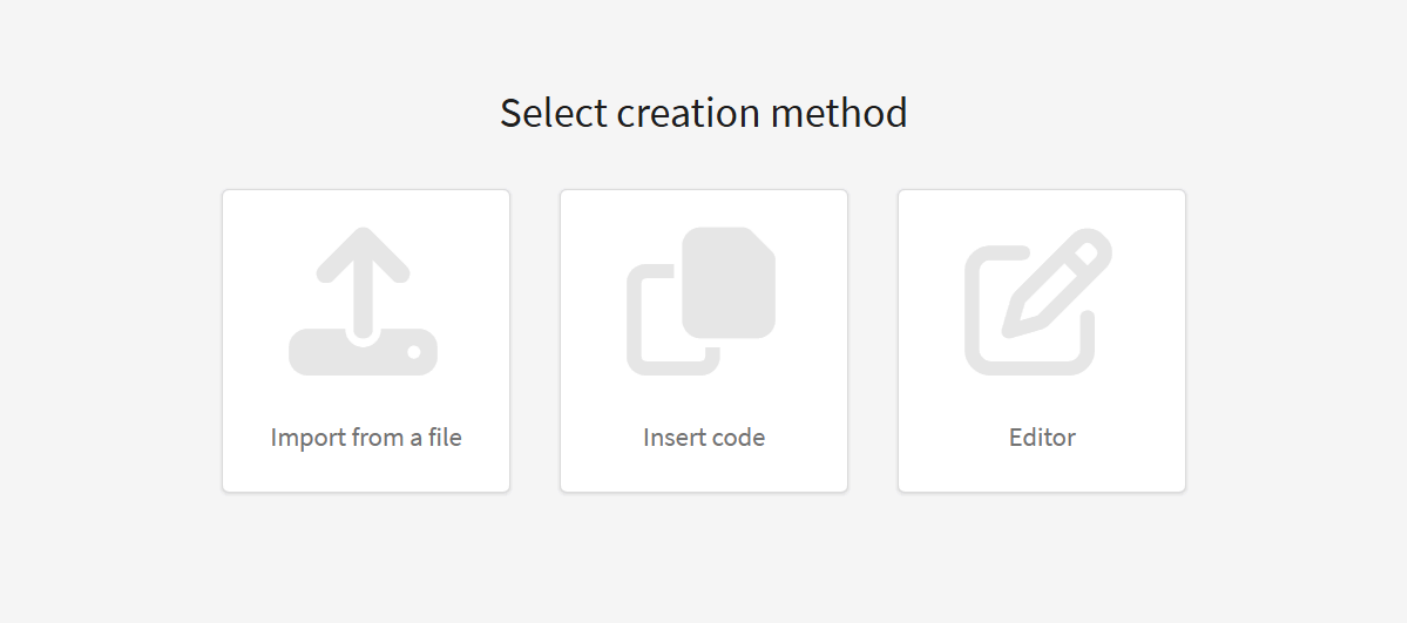
```
<!-- Correct display in preview -->
<span style="font-family: Roboto, Arial, Tahoma, sans-serif">Текст</span>

<!-- Incorrect display in preview -->
<link
  href="https://fonts.googleapis.com/css2?
family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap"
  rel="stylesheet"
/>
```

Email-template

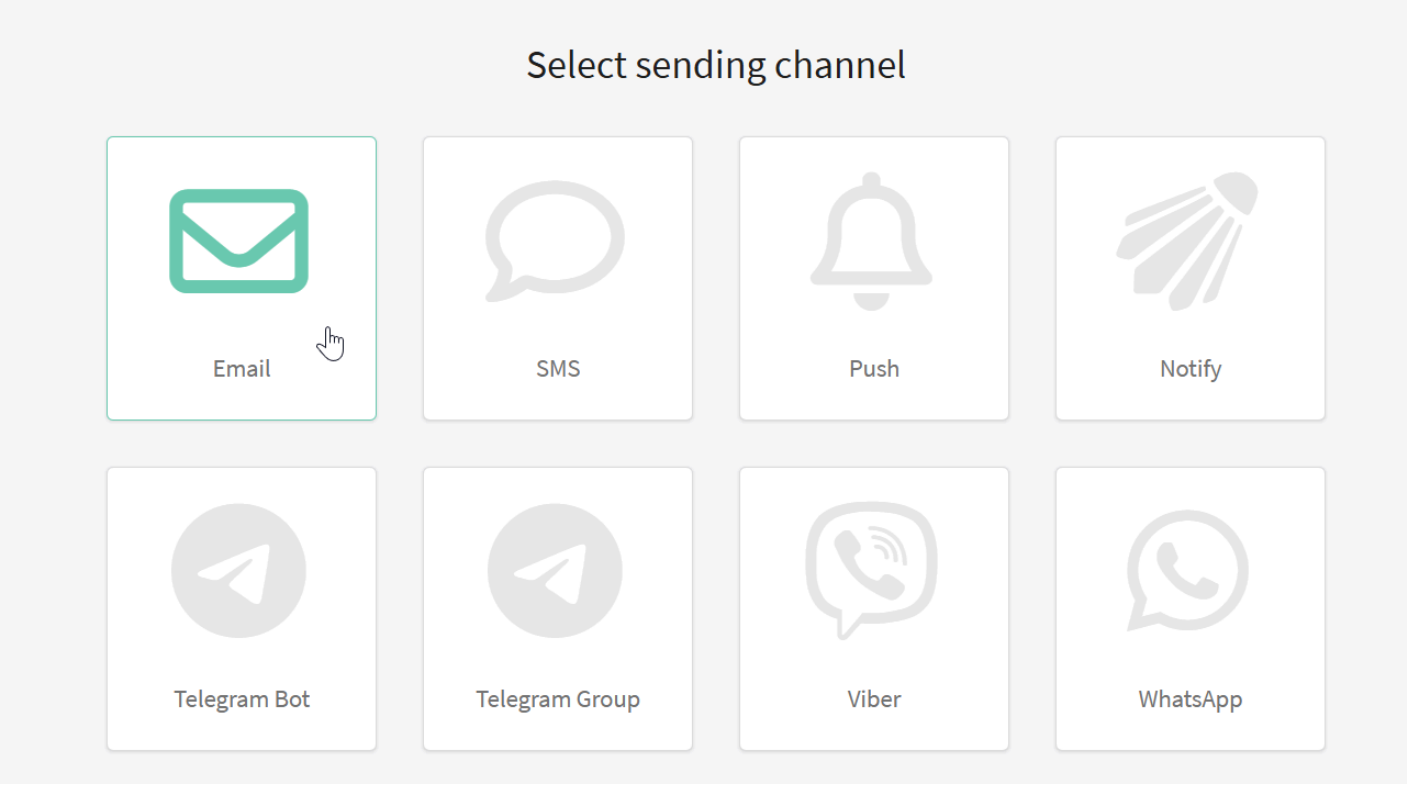
Getting started

In the main menu, select Templates and create a new message template. Next, select creation method:

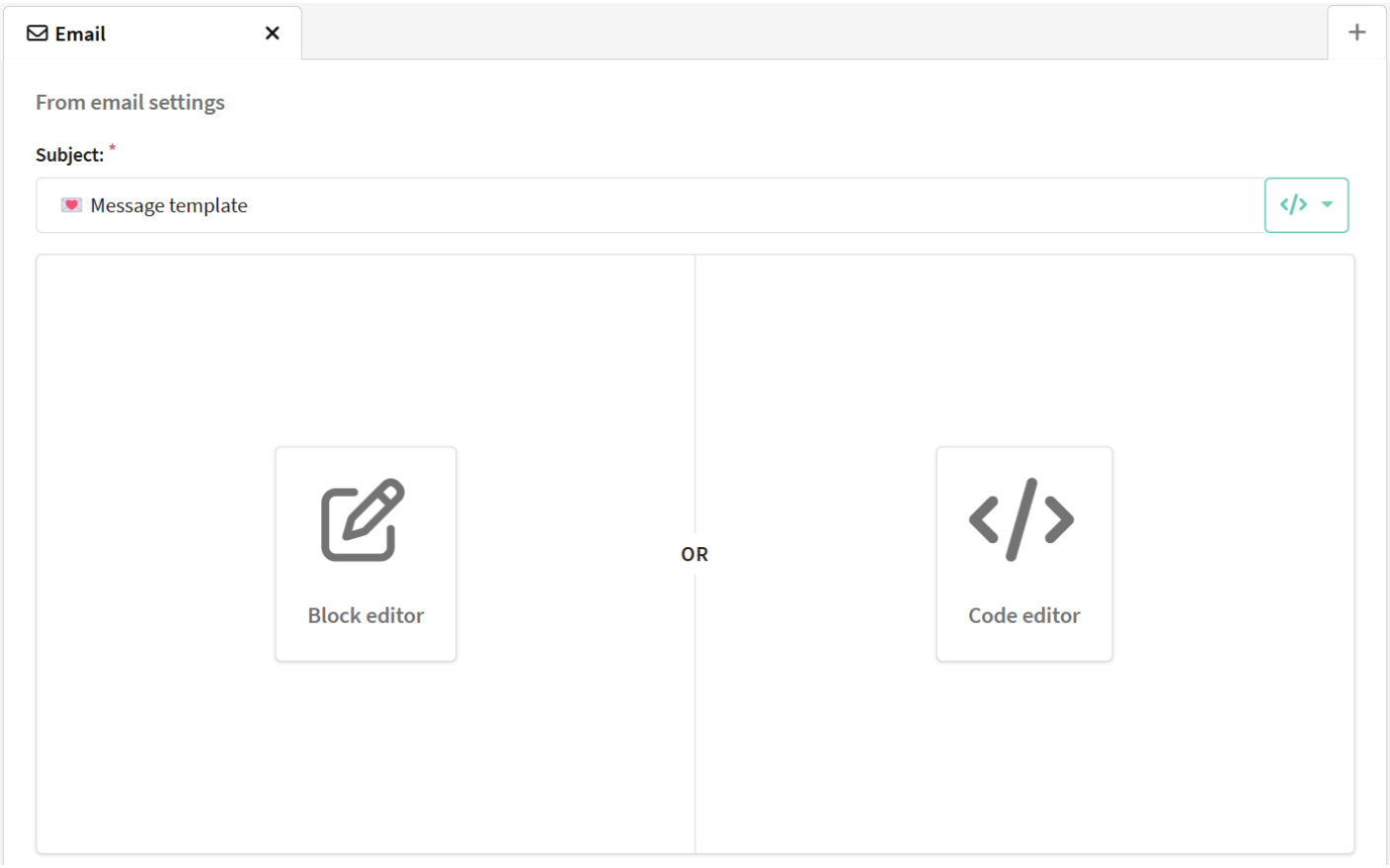


If you already have a ready template, you can import it into the platform from a file or insert HTML code. The structure of imported templates is described in this article. After loading the template you will automatically move to the editor.

To start working in the editor immediately, select Editor —> Email:

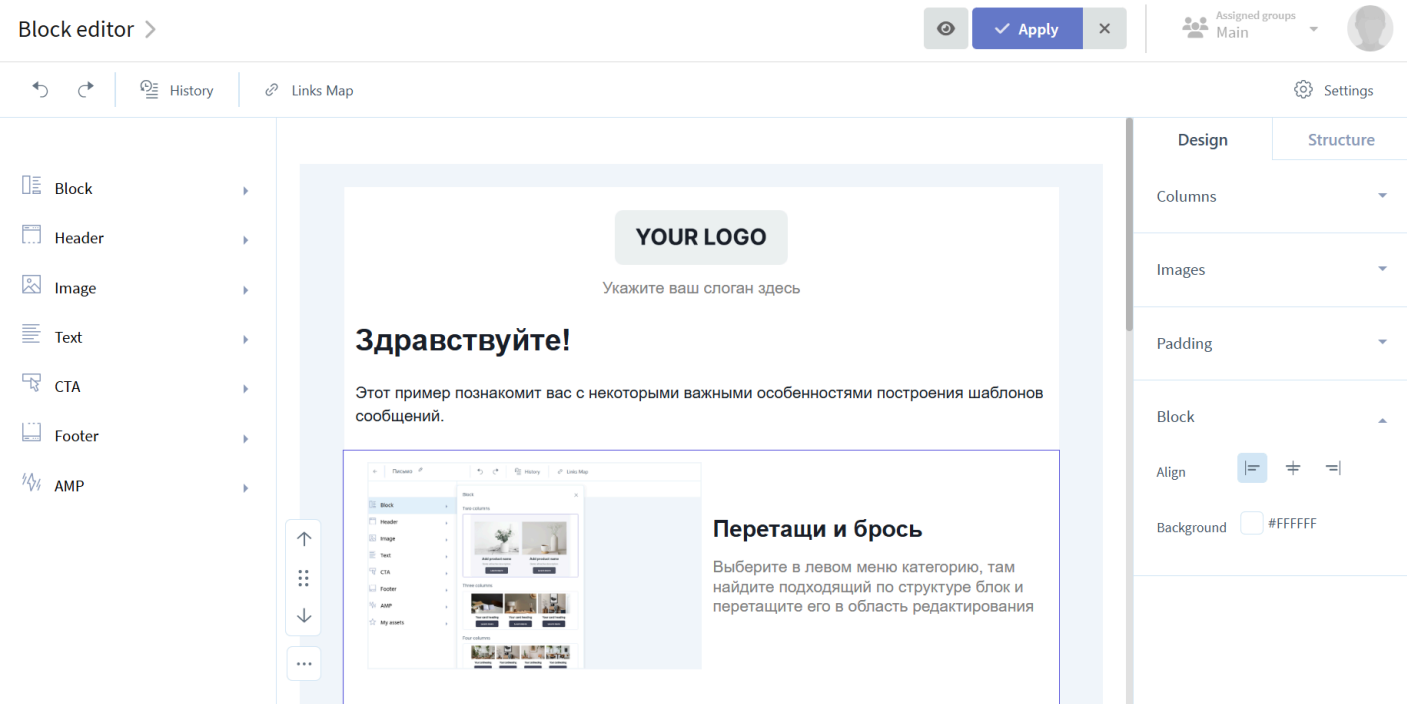


In the new email template customization window, you need to select the editor to continue working in.

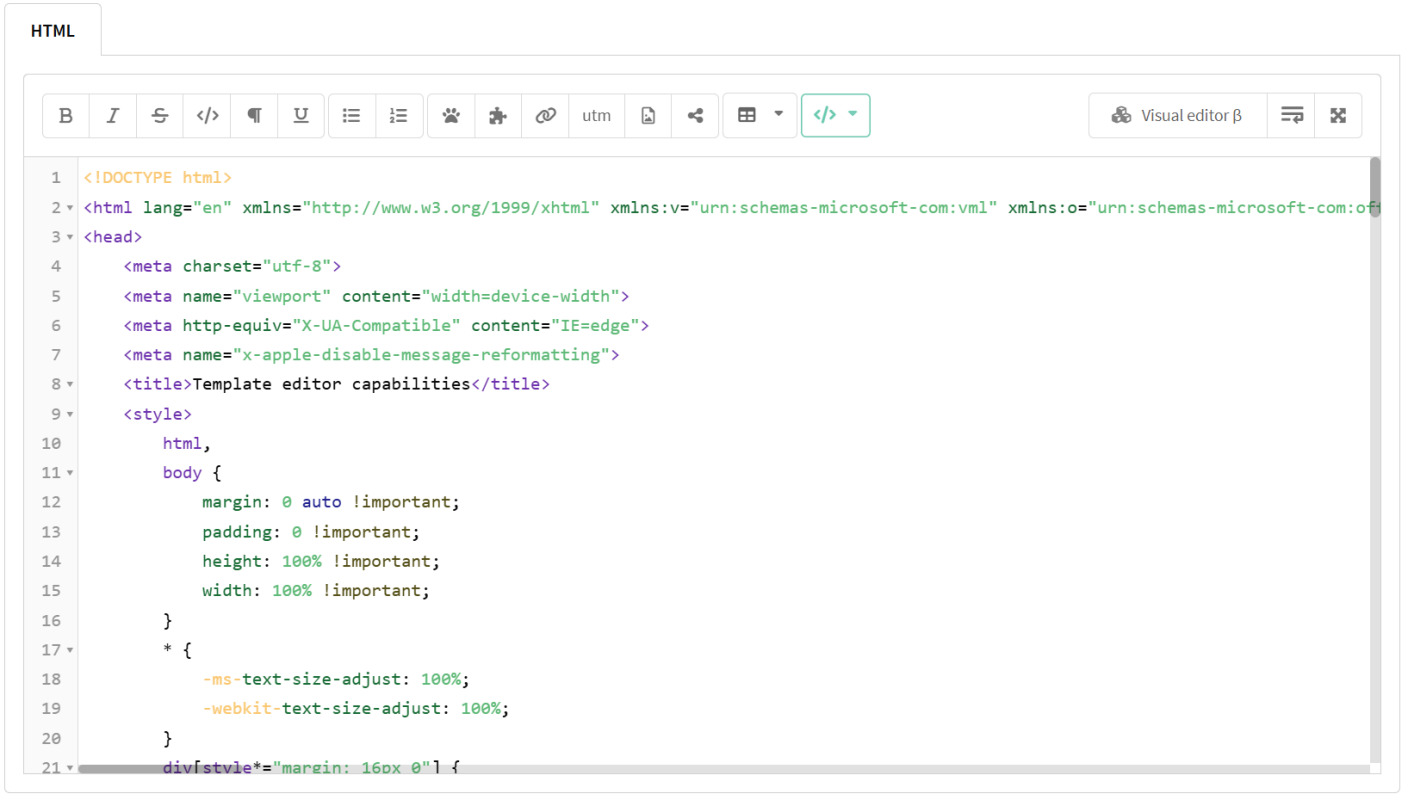


There are three editors built into the platform:

1. Block editor allows you to quickly create a template from scratch using ready—made blocks. It is suitable for those who are not familiar with HTML and CSS. AMP elements for creating interactive emails are also available in this editor. Read more about block editor features in [this article](#).



2. Code Editor is designed for working with HTML and CSS. Use it to make changes to an existing template. Read [below](#) for more information about working in the code editor.



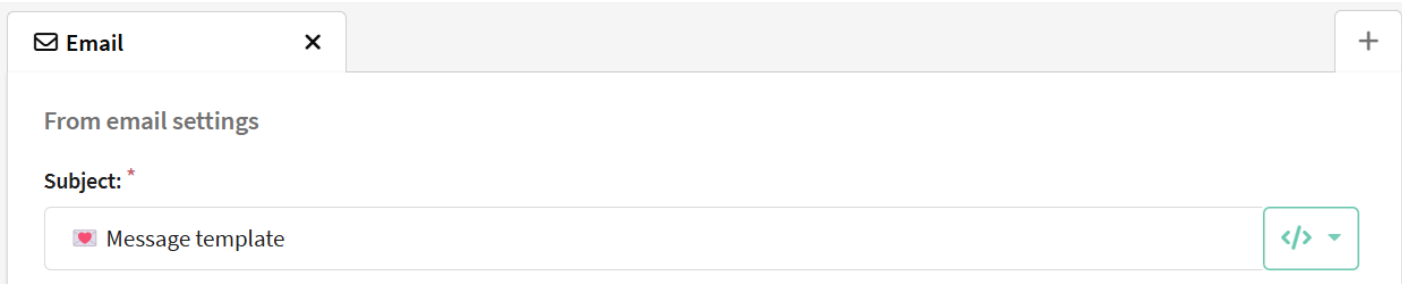
3. Visual Editor is suitable for editing a ready—made email template. With its help you can quickly change text color, add a link or replace an image without any knowledge of HTML and CSS. To access the visual editor, first open the code editor, then click the Visual Editor β button in the upper right corner. Working in the visual editor is described in [this section]().

CAUTION

Notice, you can switch from the block editor to the code editor, but you cannot switch back.

Email-template settings

In the main content settings for the email—channel, specify the subject that will be displayed in the recipient's email client. You can personalize the subject using the `</>` variable menu or manually using system variables. You can also add emoji to the subject.

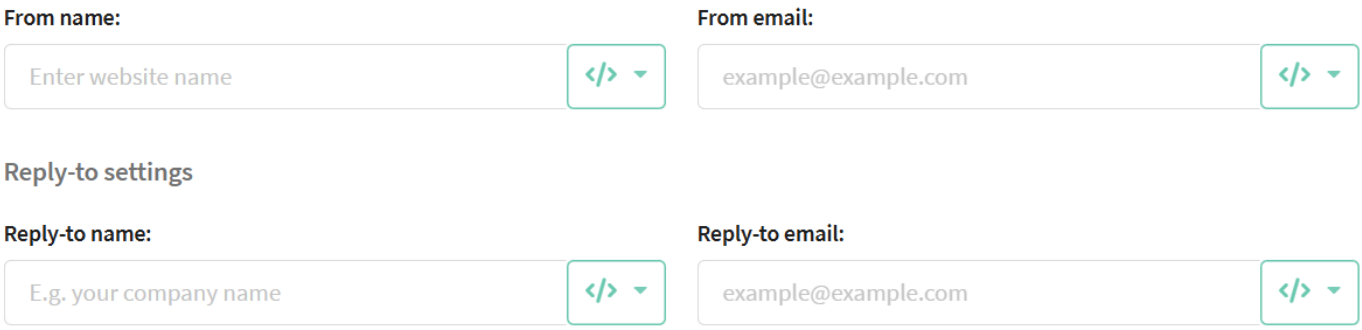


Fill in the following fields below:

- **From name — the name from which the email is sent to the subscriber. You can add personalization and emoji.**
- **From email — the address from which the email is sent to the customer.**

Fill in the following fields to Customize Feedback:

- **Reply-to name — name to reply to the message. Personalization is available: for example, you can specify here the name of the client's personal manager.**
- **Reply-to email — the e-mail address to reply to the message.**

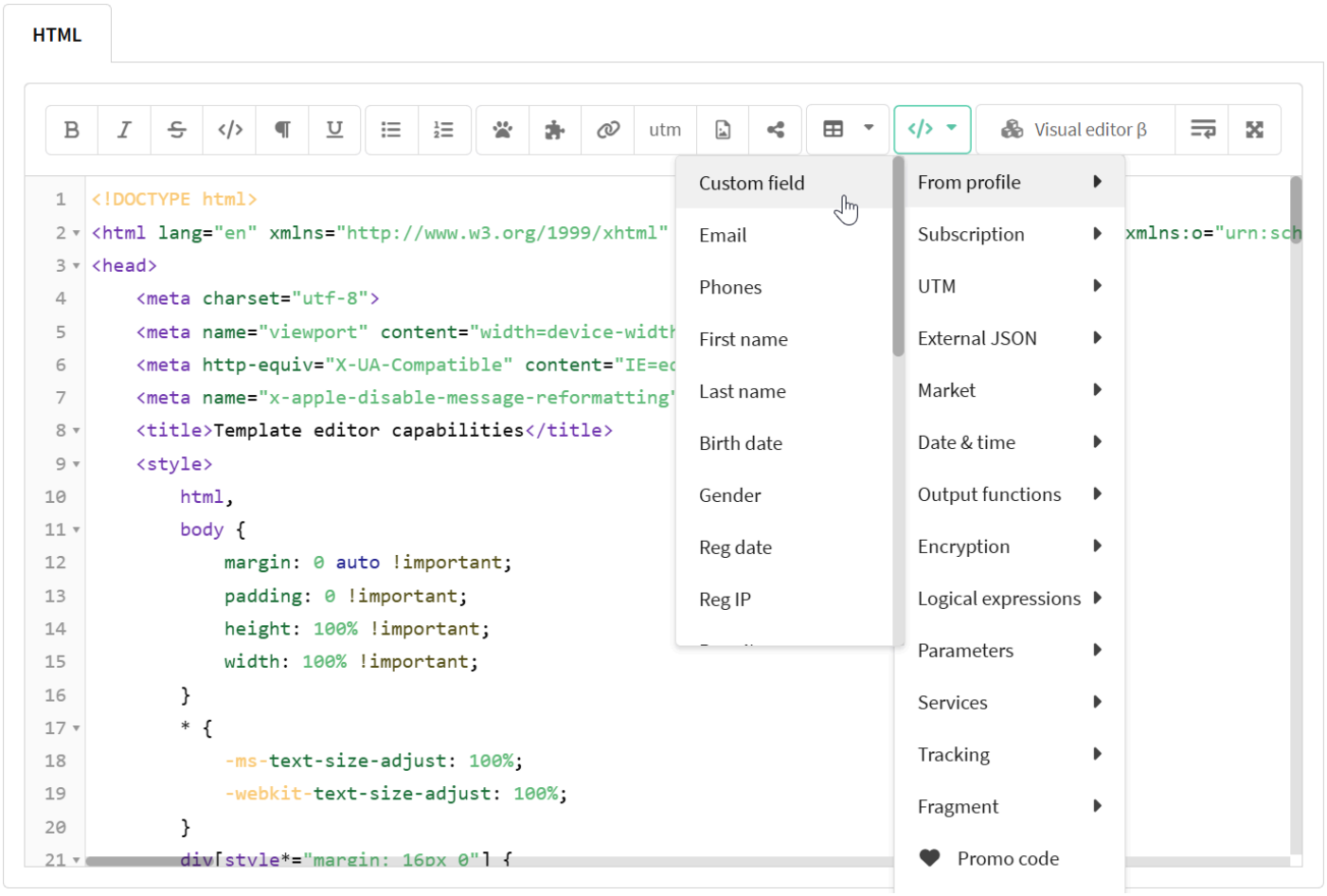


CAUTION

The From name and From email fields override the values configured for the Email channel in resource or, if no resources are used, the values of similar fields in profile database.

HTML-version of the email

Most subscribers see an HTML version of the message in their mailboxes. The code editor supports HTML and CSS, as well as personalization variables.



The following functions are available to facilitate formatting and working with objects:

Formatting

B

— bold text

I

— *italics*

S

— ~~strikethrough~~ text

¶

— formatted text

— underlined text

— bulleted list

— numbered list

— insert table

Service functions

— adding pixels to track the opening and reading of the email. If you don’t insert pixels while editing a message, the system will warn you about it when saving.

— adding external HTML content. You can also specify the source address of the variable manually `{[http://example.com ]}`.

— add or edit links.

utm

— setting UTM.

— add or edit images.

— adding buttons Social share links

— menu editor variables and functions. Using this button, you can add customer profile data, promo codes, loops and logical expressions, as well as dynamic content and system links to the message.

TIP

Using the row of buttons on the right side of the window, you can enable toggle word wrapping

or full screen mode

.

You can also use the visual editor to edit the finished template.

Text version of email

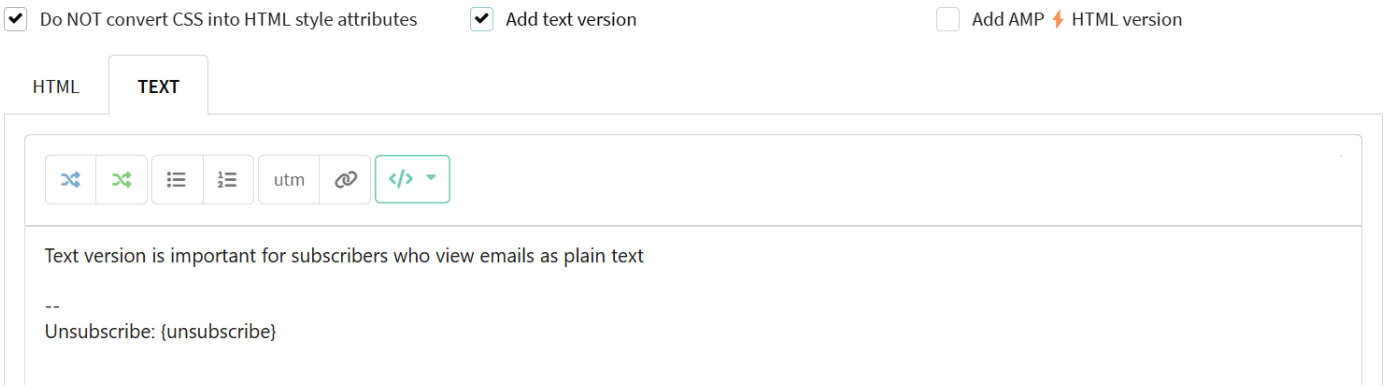
The text version of the email is text without formatting. It is displayed in the mail instead of HTML if the subscriber’s email provider or the client itself blocks the HTML version. This version is disabled by default. To include it in an email, check the box Add text version of the email.

You can generate a text version based on HTML using the buttons:

— generates a text version with hyperlinks.

— generates a text version using Markdown links, which can improve compatibility with various mail systems.

The text version also supports lists, UTM tags, links and variables.



CAUTION

If the HTML version of the email contains an unsubscribe link or a confirm link, it must also be present in the text version of the email. Otherwise, you will not be able to save the template.

AMP-version

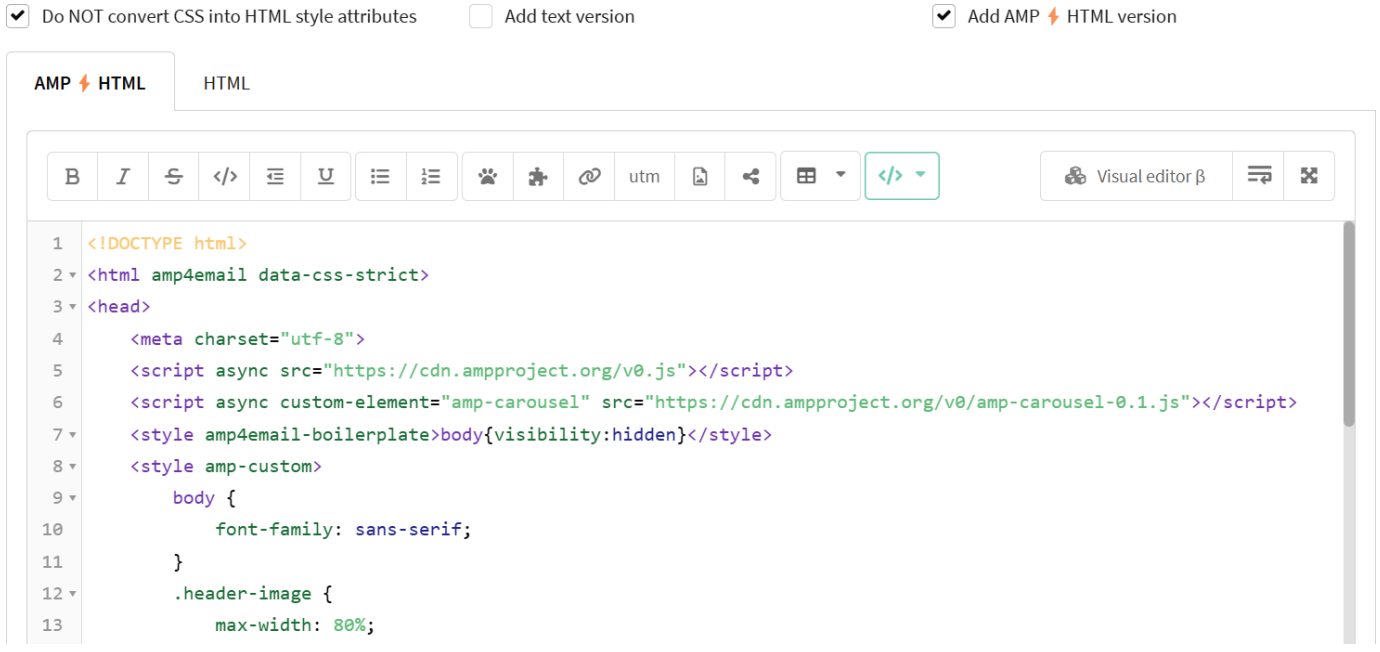
AMP allows you to add JavaScript components to emails and make them interactive. In an AMP email, the user can complete a survey, buy a product, or confirm registration without going to the site.

Email clients require special registration from the senders of AMP emails. For instructions on adding your sender's address to the allowed list, go to the official AMP documentation.

TIP

For those who are just starting to master AMP, this guide will be useful .

To send AMP emails, you need to check the box in the field Add AMP⚡HTML version. An additional tab AMP⚡HTML will appear below, where you can use AMP components.



TIP

At the moment, AMP emails support the following email clients:

- **Gmail;**
- **Yahoo;**
- **Mail.ru.**

If an email arrives on an email client that does not support AMP technology, the recipient will see the HTML version.

Links in the AMP version

In the AMP version of the email, only the `data-name` attribute is supported for naming links. When saving the template, the platform will automatically insert the attribute into all links.

```
<a href="https://example.com " title="Link text" data-name="link1">Link text</a>
```

Also note that if you add an AMP parameter to a link, you do not need to wrap the link with tracking, otherwise it will not work. To disable tracking, add the `no-track` attribute. Clicks on such a link will not be tracked.

```
<a href="${{exampleUrl}}" title="Link text" data-name="link1" no-track>Link text</a>
```

Escaping template expressions

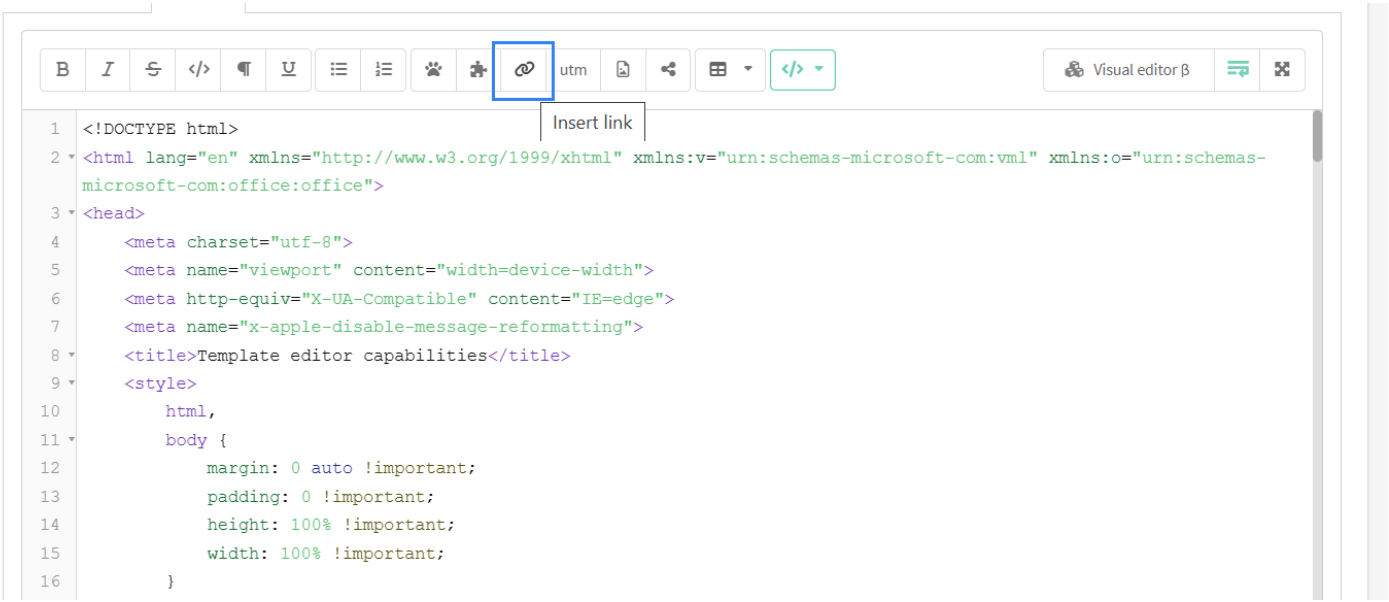
The AMP templating syntax is the same as the Altcraft templating syntax. To avoid uncorrected code processing, it is necessary to escape template AMP expressions using the $ symbol.

For example, if you are using the `<amp-list>` component, then replace the variable for substituting the list item `{{title}}` with `${{title}}`:

```
<amp-list id="myList" src="https://foo.com/list.json">
  <template type="amp-mustache">
    <div>${{title}}</div>
  </template>
</amp-list>
```

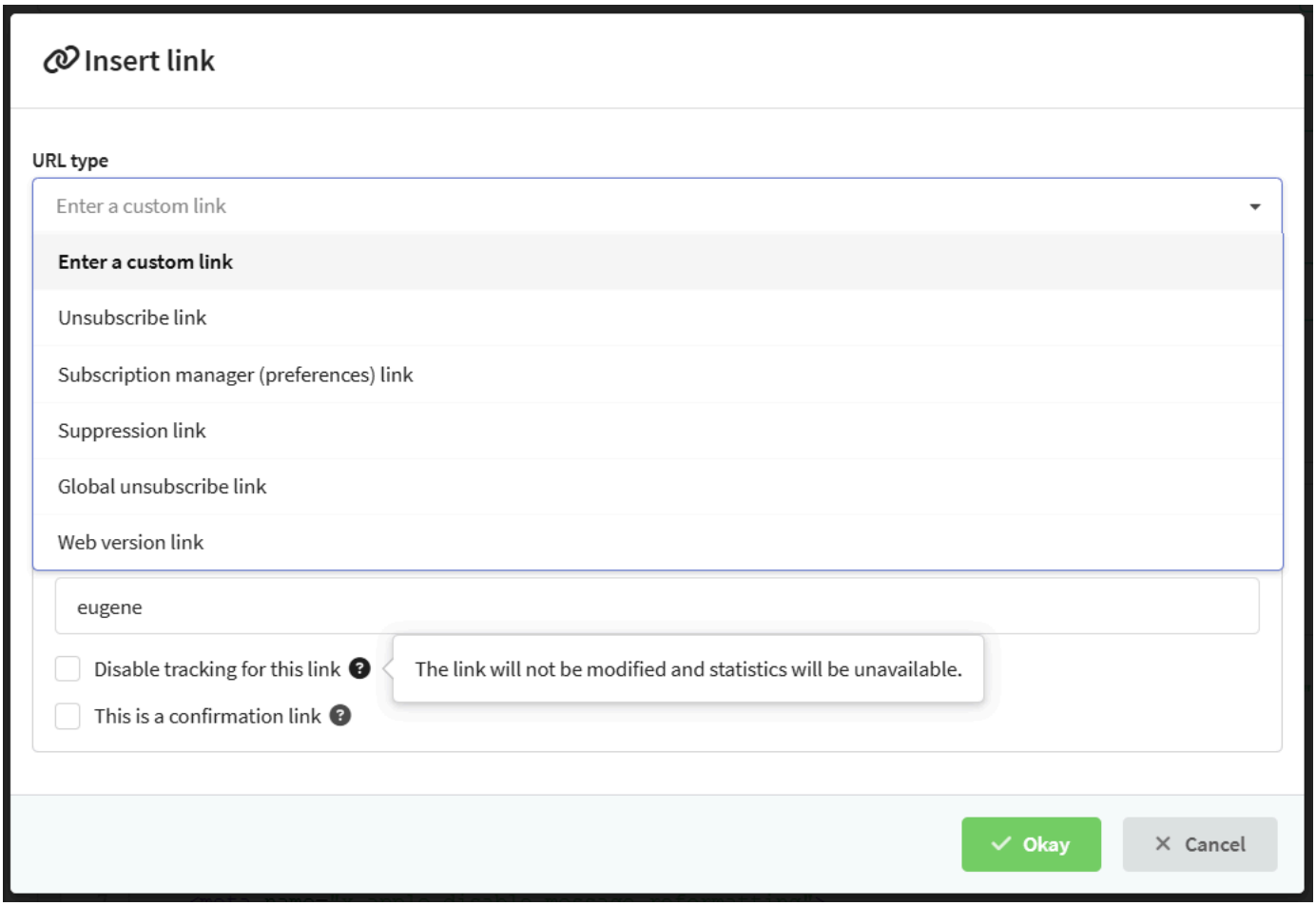
Link builder

A convenient link builder is built into the email editor. To add a link, place the cursor in the appropriate line and click on the button Insert a link. If you want to change a link that you have already added to the template, select it in the editor and then open the constructor:



In the constructor, you can select the link type depending on the purpose:

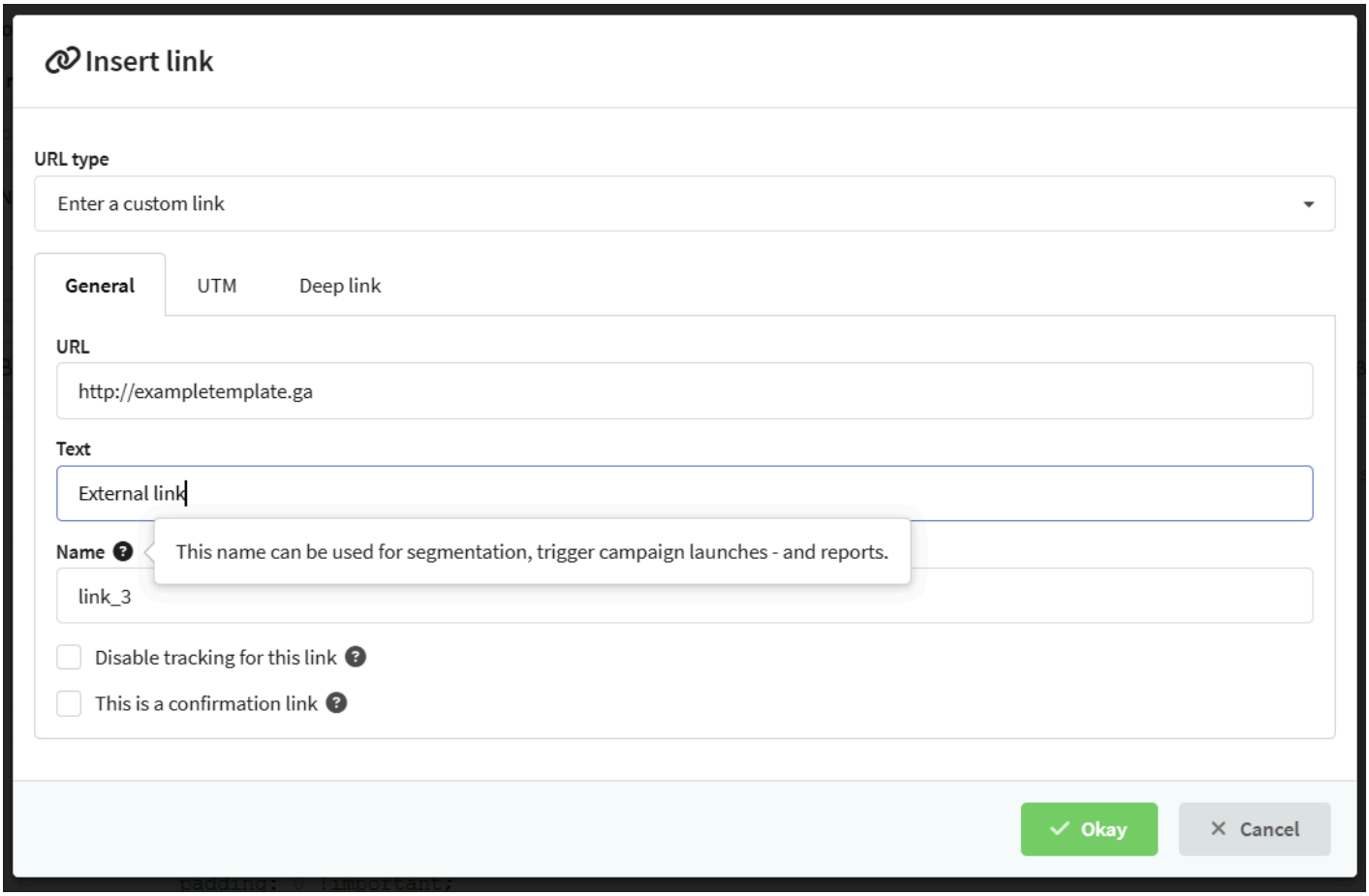
- **Enter a custom link — link to any web page except user links unsubscribing and confirming.**
- **Unsubscribe link — a link to the system or user's subscription termination page for the resource. The subscription status changes to Unsubscribed. Does not affect the global profile status and other subscriptions.**
- **Subscription manager (preferences) links — link to the subscription management system page. The client will be able to suspend their subscriptions, resume or unsubscribe from newsemails.**
- **Suppression link - link to add to the stop list, by which the client will be able to block the receipt of certain content.**
- **Global unsubscription link — changes the global profile status of the client to Unsubscribed, completely ending the possibility of communication.**
- **Web version link - link to an exact copy of the letter sent to the subscriber. If the client has problems displaying the email, it will be available via a link from any browser.**



General

In the general settings enter the URL to go to if you are adding your link. For system links, the field will be pre-filled. Also enter text to display in the email and name. Name is used in reports, segmentation and when launching triggers and scenarios.

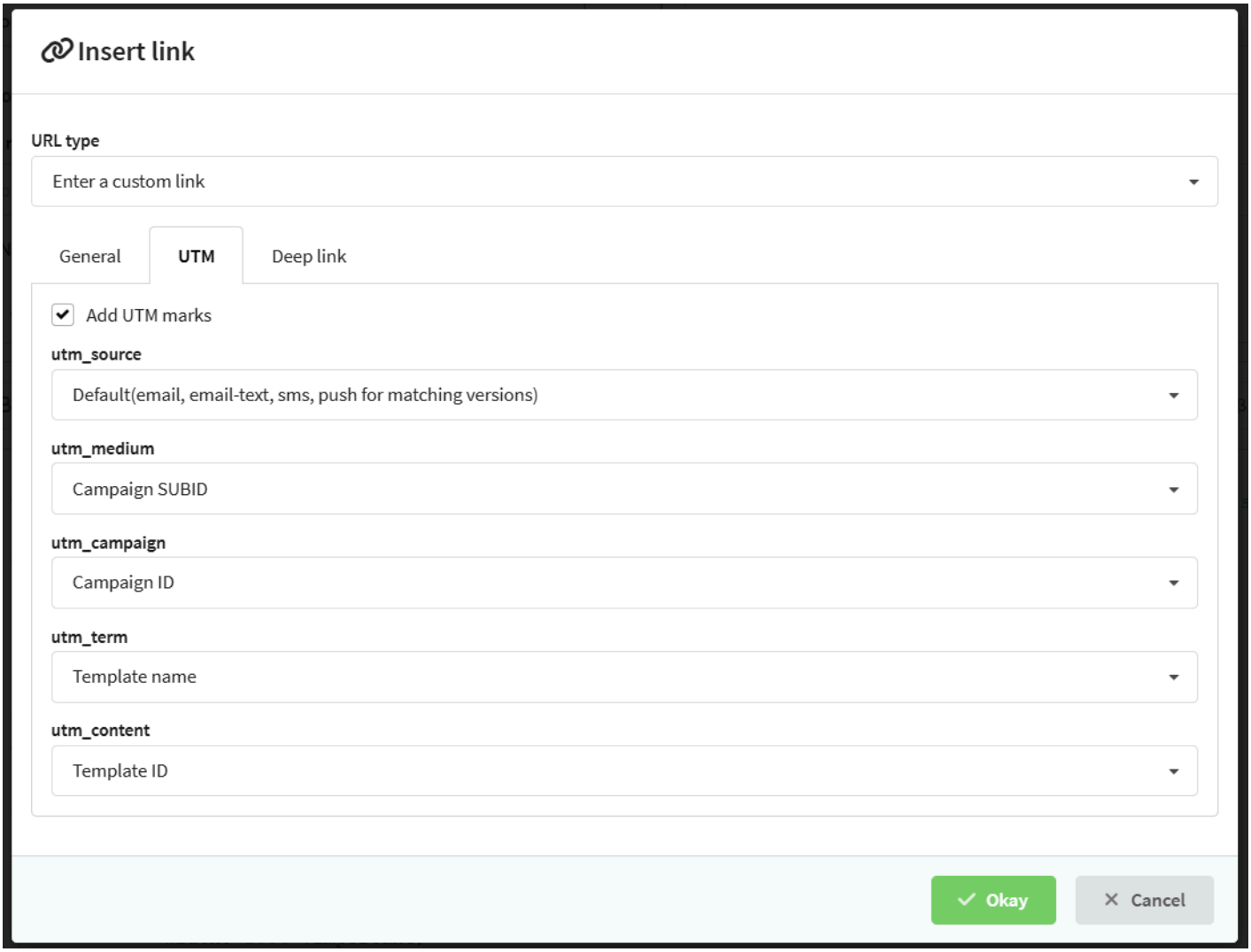
Additionally, you can disable redirection to tracking when clicking on the link, as well as make any link a subscription confirmation link.



UTM marks

For each individual link in the email, you can override the UTM tags. To do this, go to the UTM tab and activate the Add UTM marks.

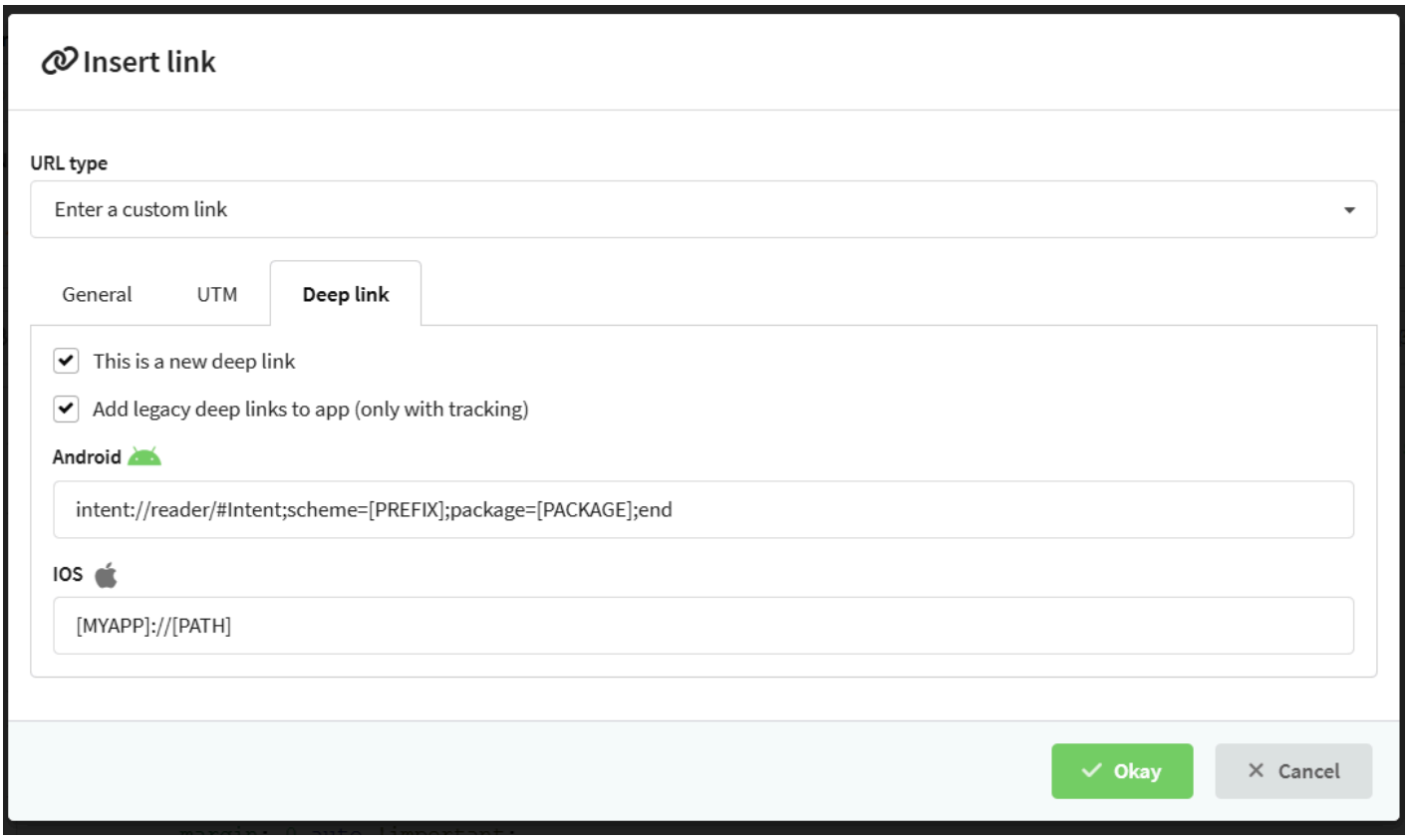
Variables can be used as labels, such as the name of the mailing or databases.



Deep link

If you want to use a link in the email to go to the mobile application, activate the option This is a new deep link on the corresponding tab.

To specify different links for Android and iOS platforms, add legacy deep links. Please note that such links can only be added using tracking.



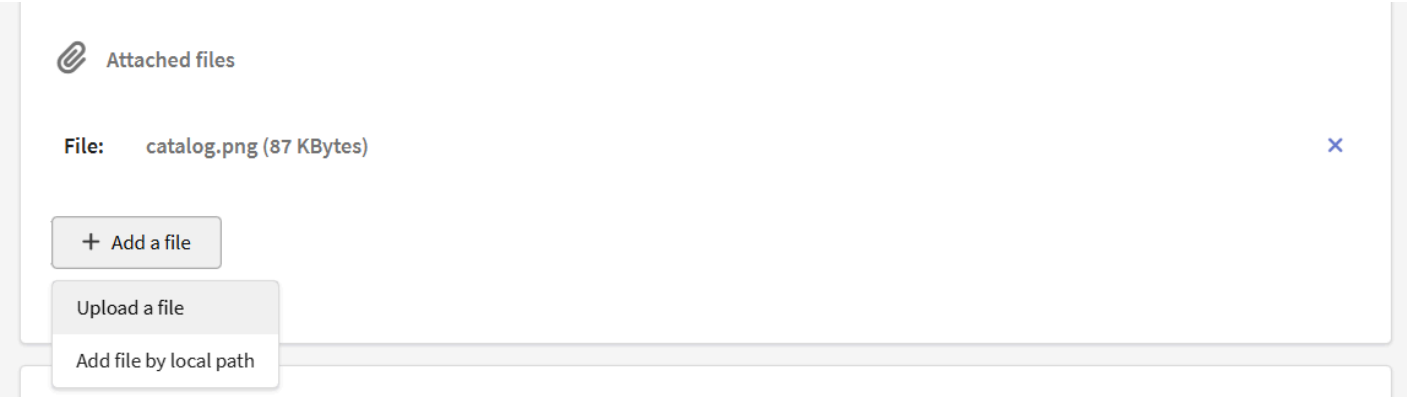
Email attachments

INFO

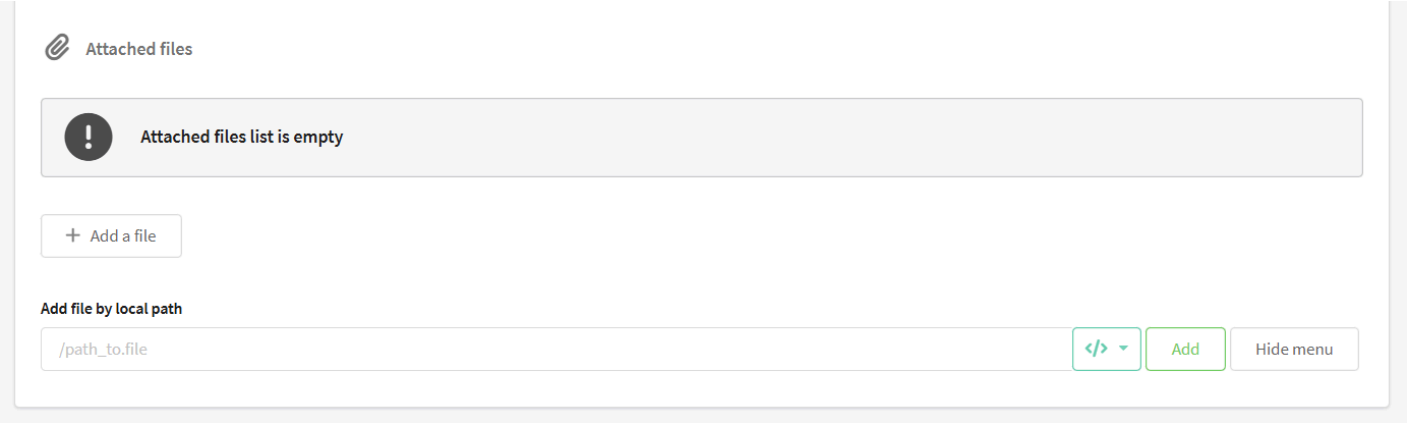
You can attach a file of any format to the email, but the maximum file size is limited to 12 MB. This value can be changed via the `MTA_ATTACH_MAX_SIZE` configuration parameter.

To attach a file to an email, use the form at the bottom of the editor. There are two options:

1. **Upload a file to the platform from your computer:**



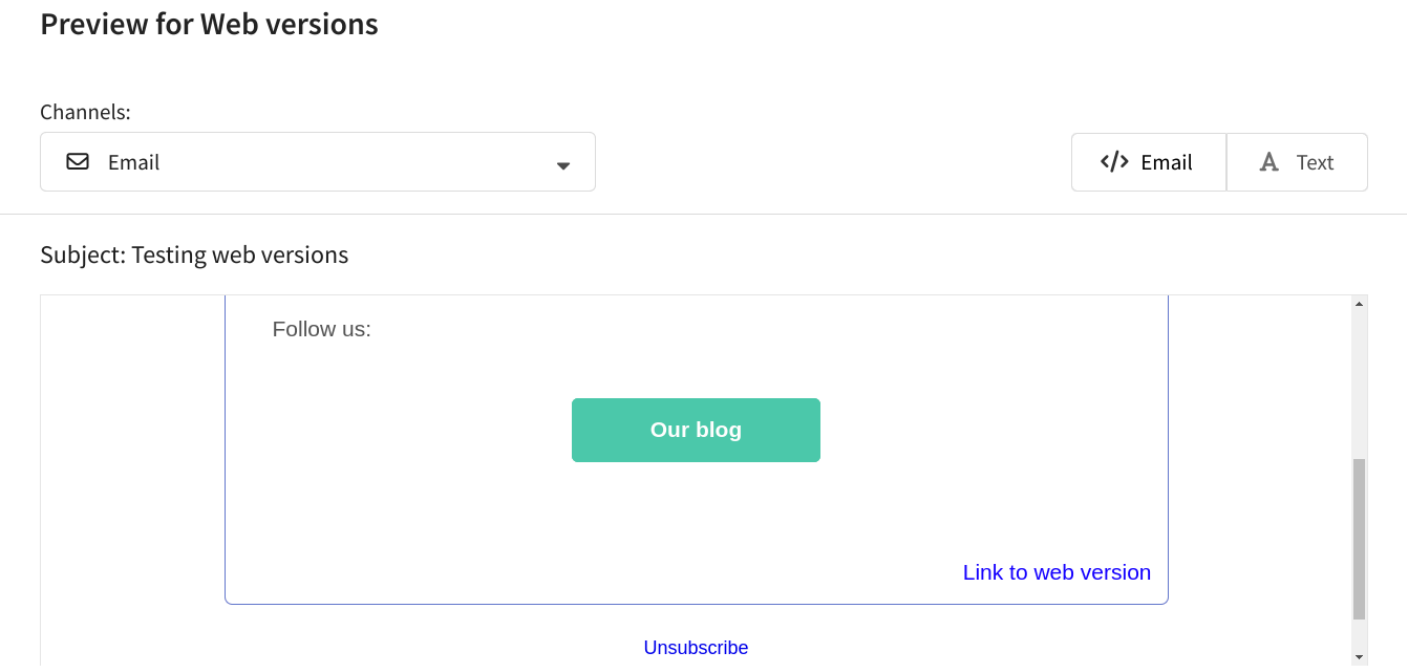
2. **Select the storage directory of the file on the server on which the platform is installed. You can use client profile variables in the directory path. Enter the file location and click Add:**



You can delete downloaded files and saved file paths by clicking on the cross on the right side of the file:

Web-version of email

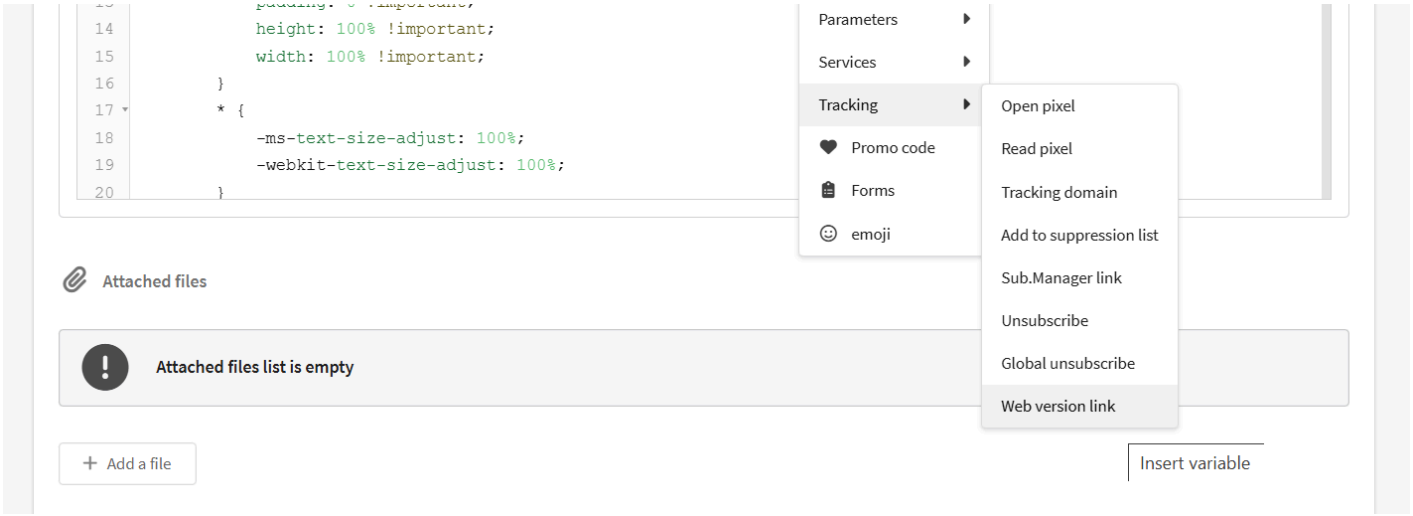
The web version of email is the version of the email message that the user can view as a web page in the browser in case of incorrect display in mail services. Web versions increase the chances that your message will be viewed even with unexpected mail client errors. Also, the presence of a link to the web version allows you to quickly and conveniently share a letter if necessary.



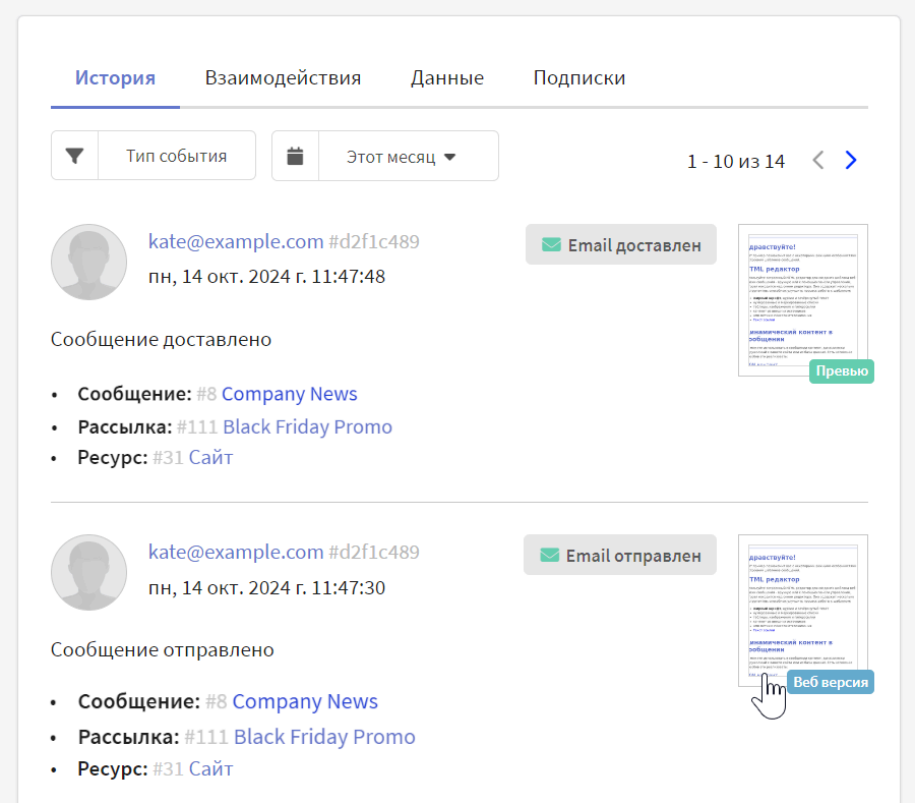
You can add a link to the web version through the link builder or manually:

```
<a href="{webversion}" title="Web version of the email" name="web_version"> Open an email in a browser</a>
```

It is also possible to add a link through the variables menu. Open it by clicking </> on the toolbar. Select Tracking — Web version link:



You can view the web version of the email sent to the client in [profile history](). By default, the web version is stored for 30 days. The platform administrator can change the retention period in the account settings.



TIP

The link to the web version does not open in preview mode and when sending a test email. The web version is generated only for sent messages. To check the link, send an email to the test subscriber via the broadcast.

INFO

By default, saving web versions is disabled in the account. To activate this feature, the platform administrator must configure the [web version policy](). After that, links will appear in the profile history to view the web versions of the sent messages.

If a link to the web version is not included in the email template, the web version of the email will not be saved. However, the administrator can modify this setting by enabling the Always save email web versions option in the account settings.

SMS template

This article describes how to create a message template to send via SMS channel.

Message body

Enter the text of the short message in the input field. Links, UTM tags and variables can be used in the message body:

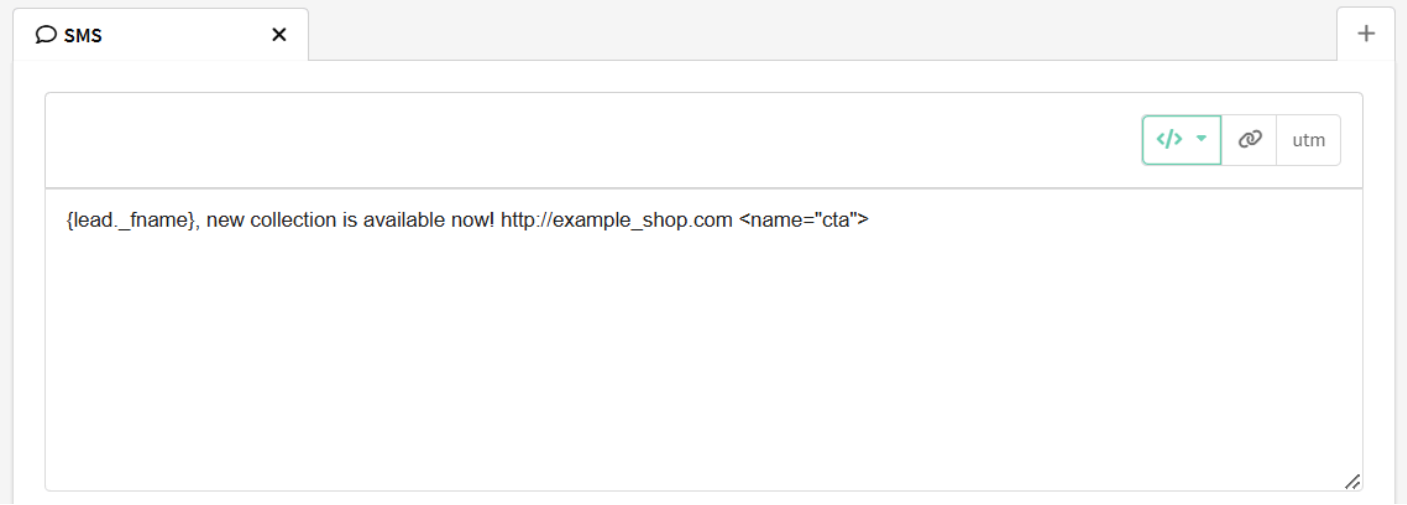
</>

— menu of variables and functions the editor. Using this button, you can add customer profile data, promo codes to the message, loops and boolean expressions, as well as dynamic content and system links.

— link editor.

utm

— setting UTM labels.

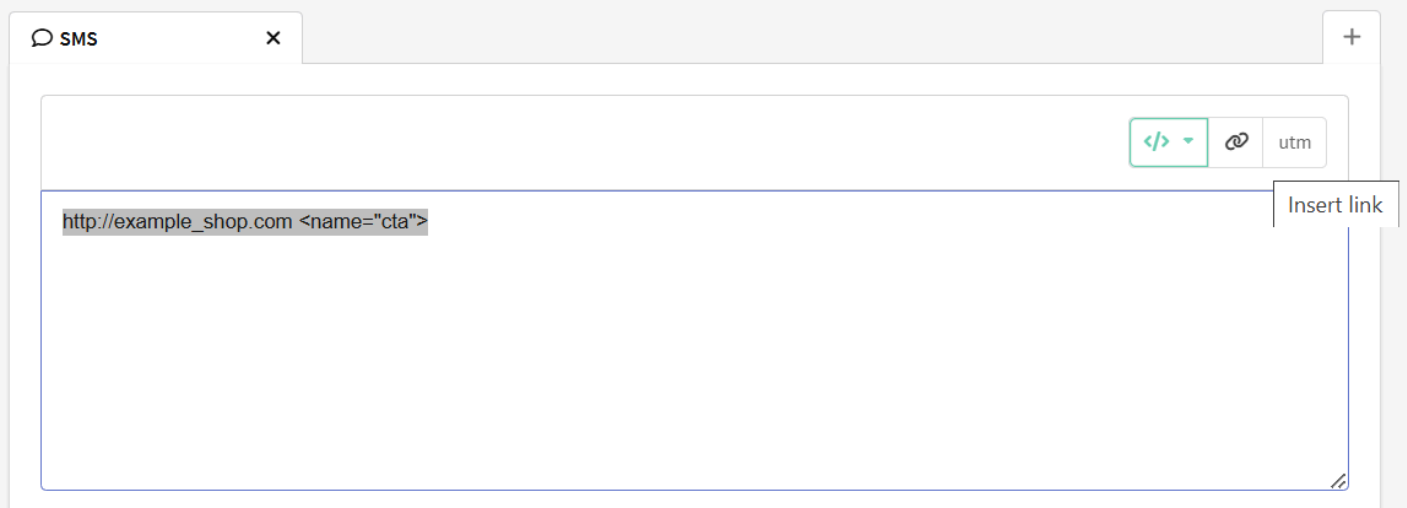


SMS link editor

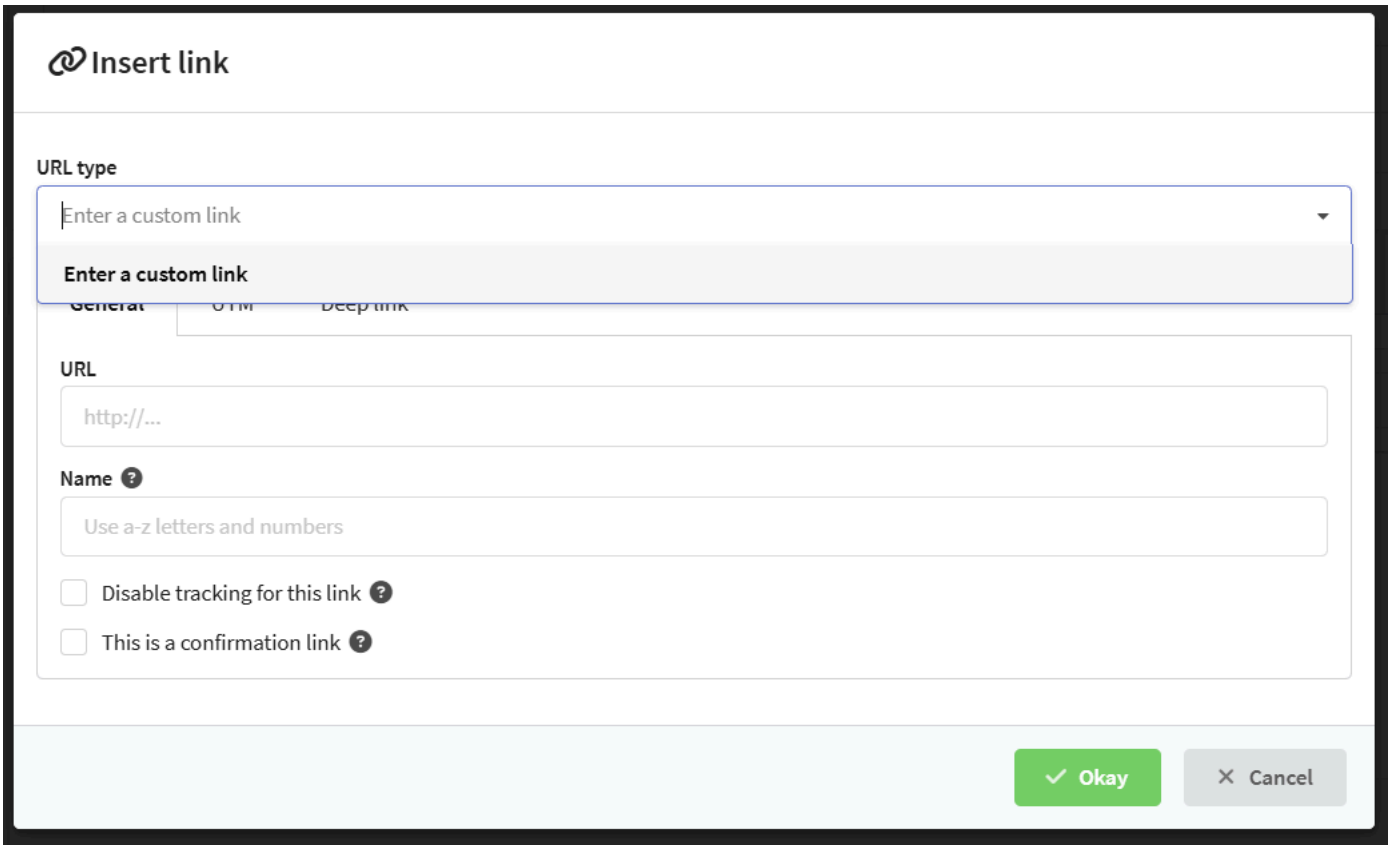
TIP

Links inserted through the platform are automatically shortened. The final link path (excluding tracking domain) length is 12-14 characters, including the `/`.

The link editor is available by clicking on the button Insert link. If you want to edit a link that has already been added to the template, select it in the editor and then open the form:



One type of links is available for SMS messages. It is a custom link. It allows you to add a link to any web page.

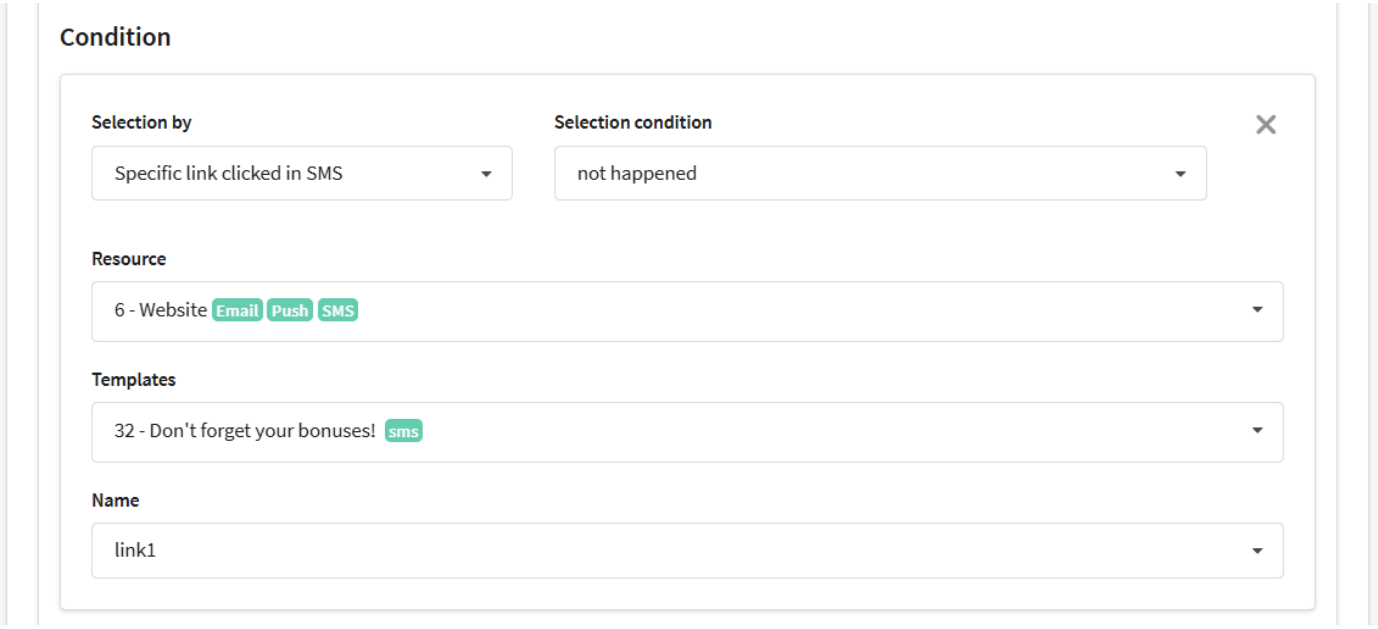


General

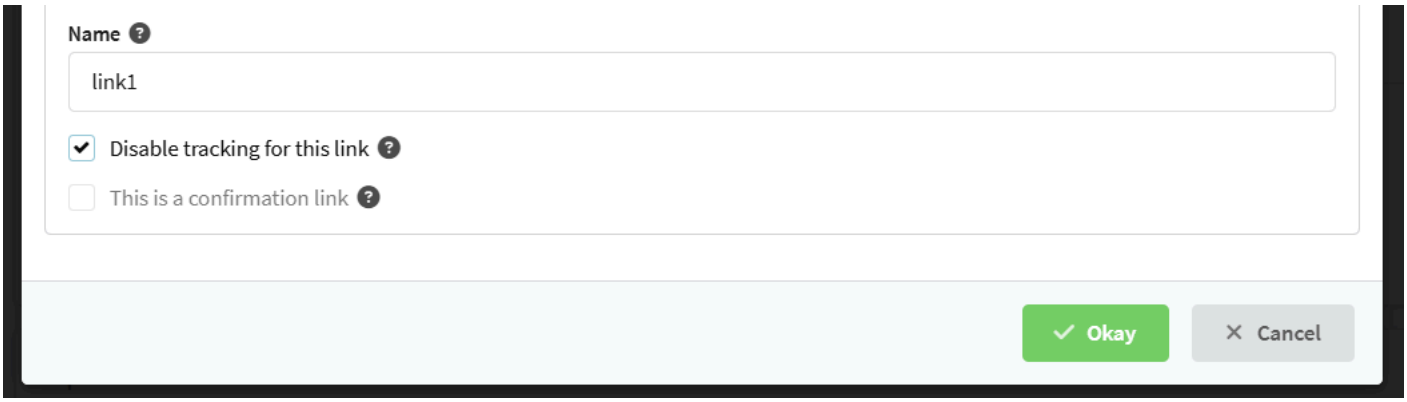
In the general settings enter the URL to go to. Also specify the name, which is used in reports, segmentation and when launching triggers and scenarios. The link name can only consist of numbers and Latin letters.

CAUTION

Please note that when you select profiles by clicking on a link in an SMS, you must specify the link name, not the URL, in the selection conditions.



Additionally, you can disable redirection to tracking when clicking on the link, as well as make any link a subscription confirmation link.



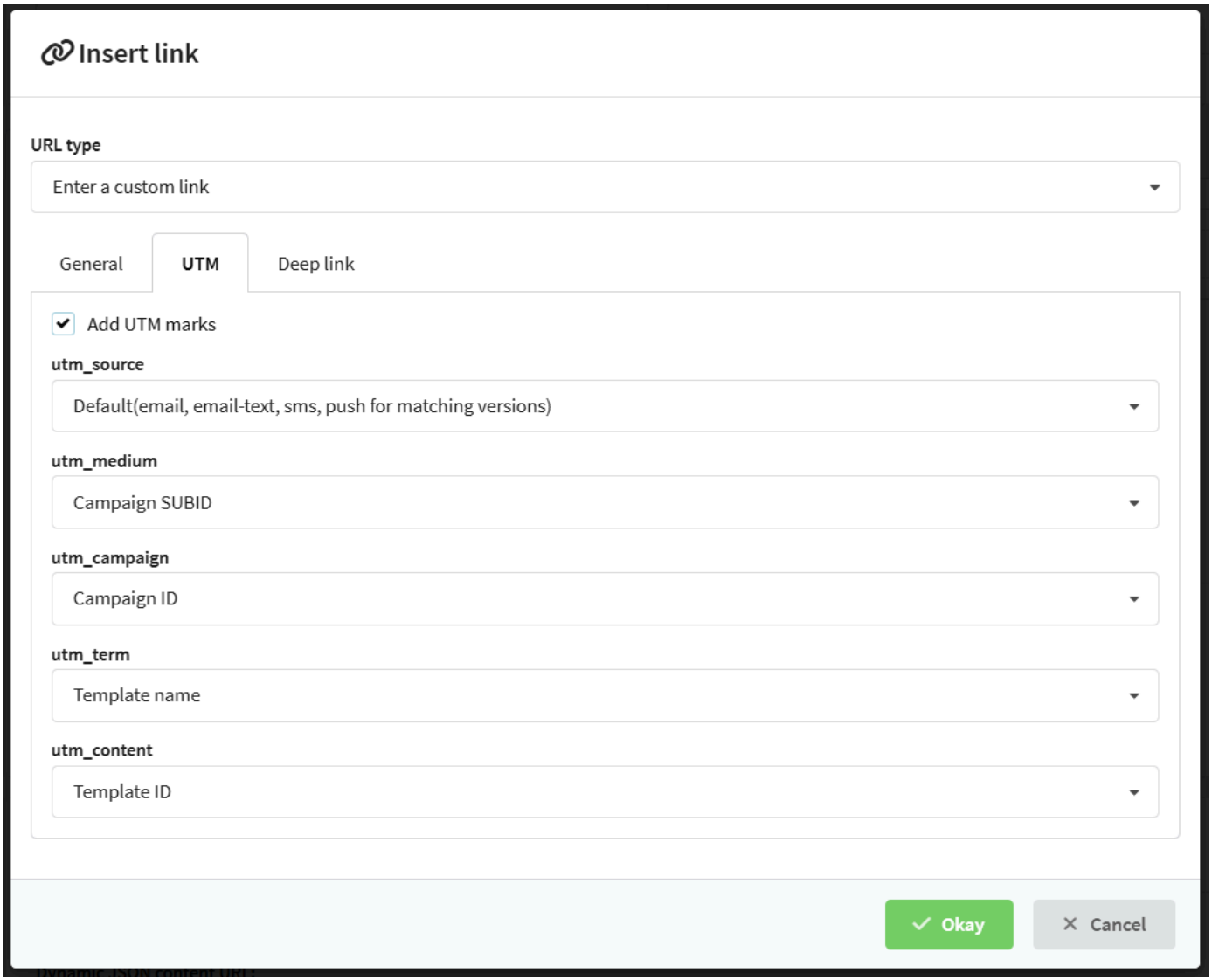
In SMS messages, angle brackets are used to indicate parameters (for example, `<name="">` will be used for the name parameter). Everything you specify in angle brackets will not be included in the final message. Usage examples:

- `<no-track>` **— disables redirection to tracking when clicking on a link.**
- `<confirm-link>` **— makes a confirmation link.**
- `<name="example_name">` **— sets the name of the link.**

UTM marks

For each individual link in the message, you can override the UTM tags. To do this, go to the UTM tab and activate the Add UTM marks.

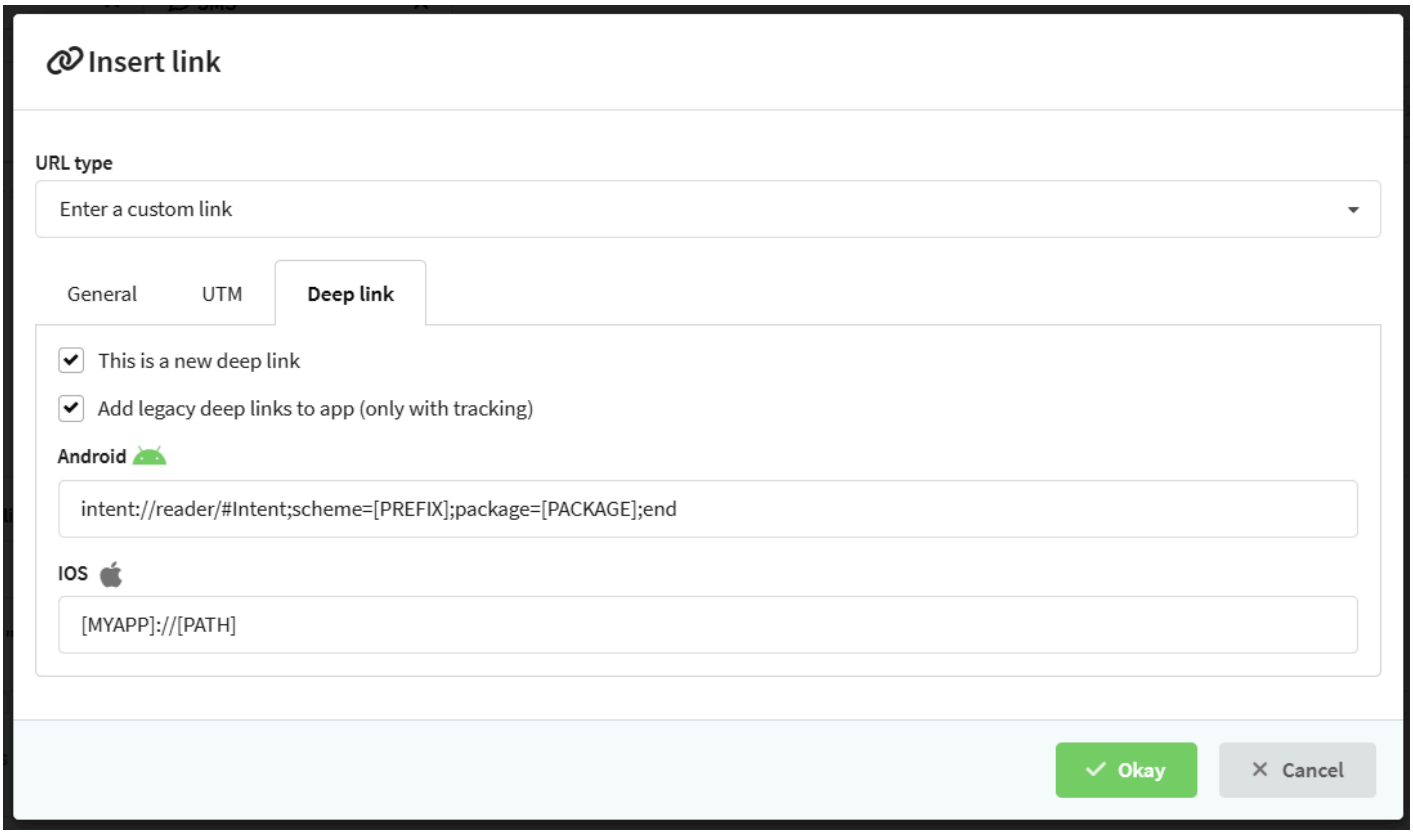
Variables can be used as labels, such as the name of the mailing or databases.



Deep link

If you want to use the SMS link to go to the mobile application, activate the option This is a new deep link on the corresponding tab.

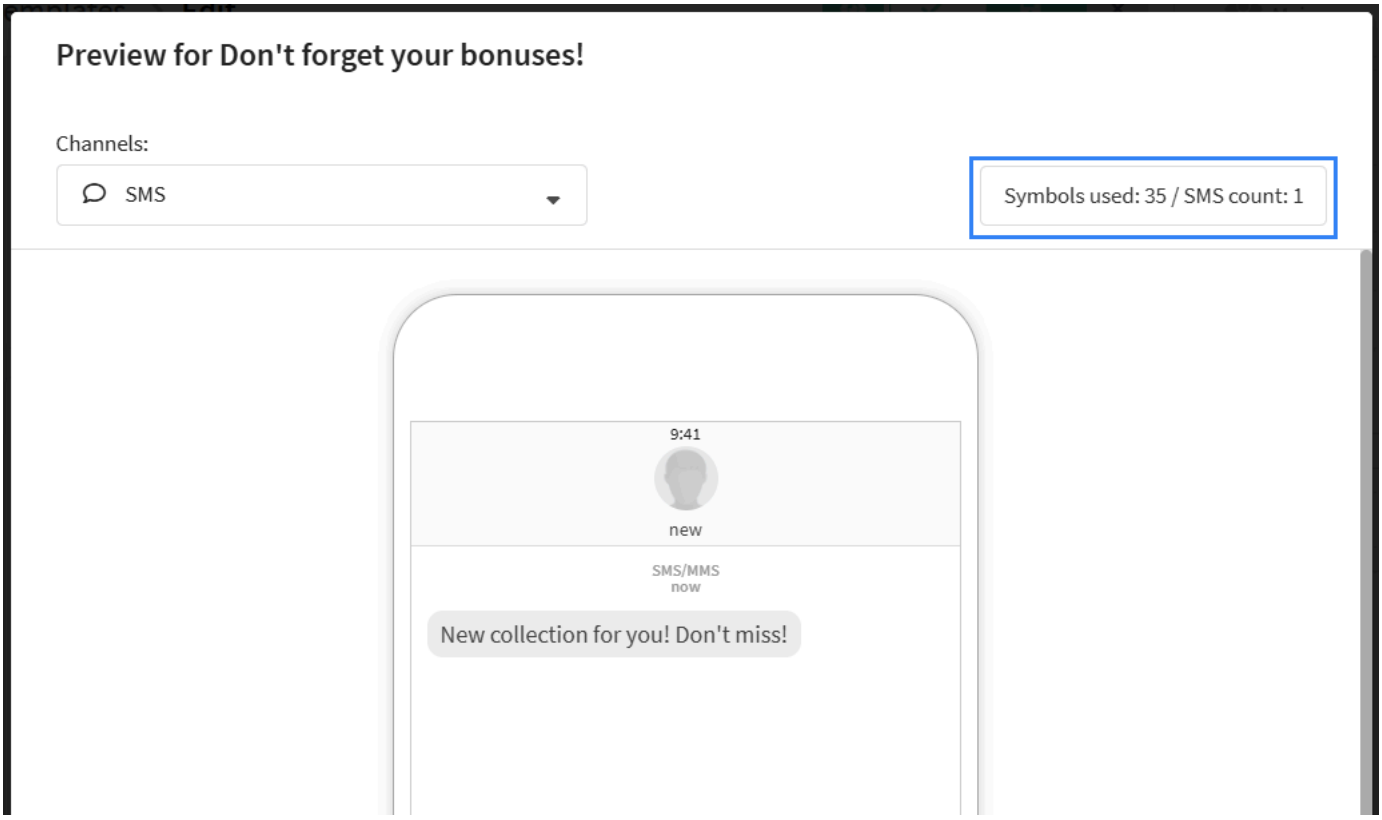
To specify different links for Android and iOS platforms, add legacy deep links. Please note that such links can only be added using tracking.



Limit on the number of characters

When sending SMS messages, it is important to take into account the length of the text, since the number of characters affects the cost of sending. One SMS message in Cyrillic can contain up to 70 characters and in Latin the are 160 characters, including spaces and punctuation marks. If you exceed these limits, your message will be charged as two SMS. The presence of at least one Cyrillic character makes the entire message Cyrillic.

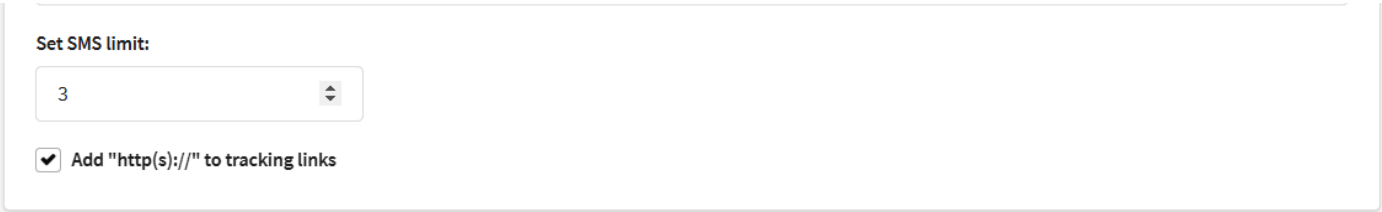
The editor automatically counts the number of characters and messages. To see the counter, save the template and switch to preview mode:



TIP

Please note that the data that is pulled from dynamic variables may exceed the expected number of characters.

The number of sent SMS messages can be limited by specifying a value in the “Set SMS limit” field. If the message text exceeds the specified SMS parts limit after splitting, the message will not be sent at all. For example, if a user sets the limit to 3 SMS and the text length is 250 characters, such a message will not be sent during the mailing.



In addition, the `http(s)://` protocol is not added to the template to shorten links, but you can enable auto-substitution. This only applies to tracking links.

INFO: Notice

When a template includes logical expressions or loops, the platform counts every formatting character — spaces and line breaks — as part of the final message. Each space or line break adds one character to the SMS and may affect the cost. To save characters and control message length, keep logical constructions on one line.

For example, instead of:

```
{if lead._city equal "Moscow"}
We invite you to a conference in Moscow on June 30!
{else}
We invite you to a conference in your city!
{end}
```

Use:

```
{if lead._city equal "Moscow"} We invite you in Moscow on June 30!{else}We invite you in your city!{end}
```

This approach keeps the logic intact and avoids unexpected SMS length growth.

Push template

The platform offers two editors for push templates: a standard editor and a dedicated Rich Push editor. The Rich Push editor enables adding banners and buttons, as well as configuring push settings tailored to specific browsers or operating systems.

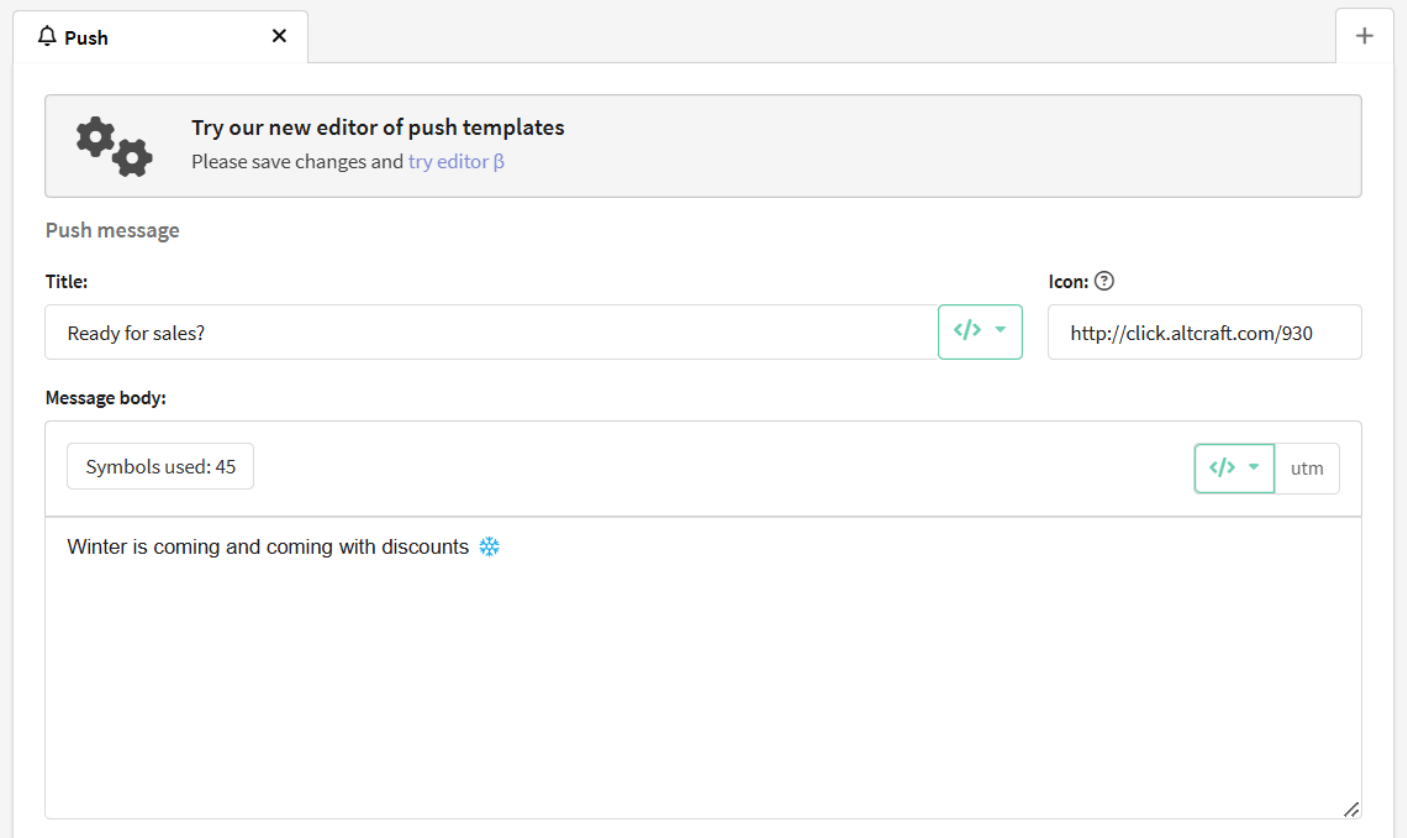
Standard editor

For push notifications, you can configure:

- **Title — It is recommended to use no more than 25 characters so that the text is not cut off on devices. Emojis are supported. The header can be personalized using the variable menu**

  .
- **Icon — an image to display in the notification. In the browser Apple Safari, only one image is supported for all notifications, its configuration is performed in [setting up the push channel in resourse](#).**
- **Message body — the main text of the notification. Emojis are supported. The maximum text length is 256 characters, it is recommended to use up to 150 characters. You can also use variables in the message body to personalize**

  . The editor automatically counts the number of characters:

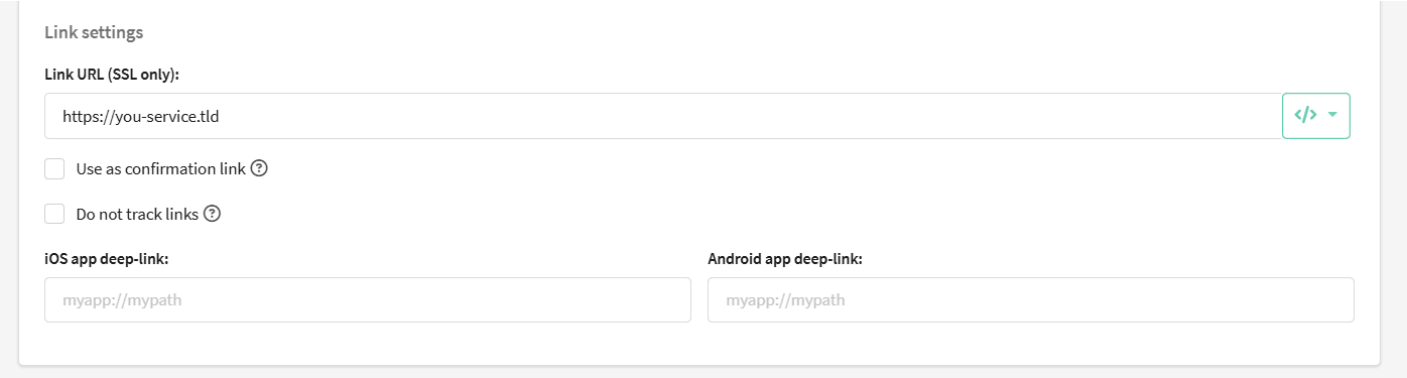


TIP

Links inserted through the platform are automatically shortened. The final link path (excluding tracking domain) length is 12-14 characters, including the `/`.

In the block Link settings are available:

- **Link URL (SSL only) — The URL that opens when you click on the notification. Only the secure protocol (SSL) is used. A menu of variables is available for inserting, for example, promo codes, so that the user immediately goes to the page with the offer and the calculated discount.**
- **iOS app deep-link — link to go to iOS apps.**
- **Android app deep-link — link to go to Android apps.**



To add utm tags to the link click on the

utm

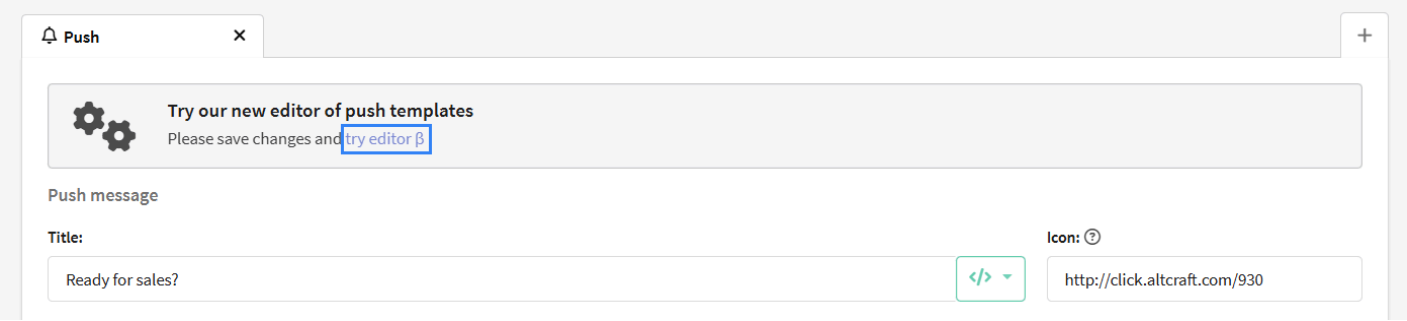
button in the upper right corner of the editor or set the labels manually.

Additionally, the link in the push can be a confirmation link. You can create a double opt-in subscription mechanism. You also can disable tracking. If you decide not to wrap links for tracking, keep in mind that in this case the platform will not be able to capture clicks on the link, and deep-link links will become unavailable.

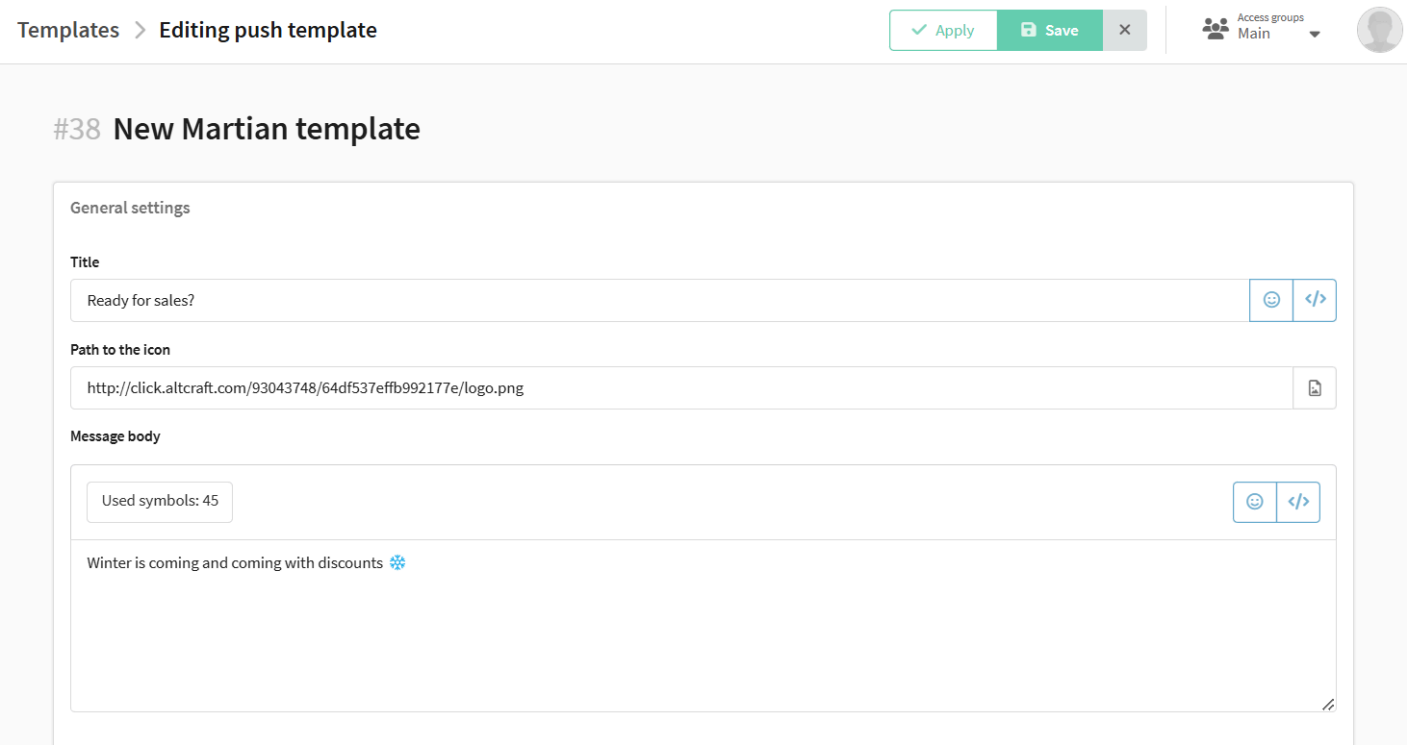
Rich Push editor

Get started

To switch to the editor for Rich push notifications, you need to prepare the initial version of the message in a standard editor (you can do it without a link to go to) and save the template using the "Apply" button. After that, the try the β editor button will be available in the editor:



The title, the path to the icon and the body of the message will be filled with the data that you specified in the standard push editor. You can edit these fields if necessary.



Click-through link

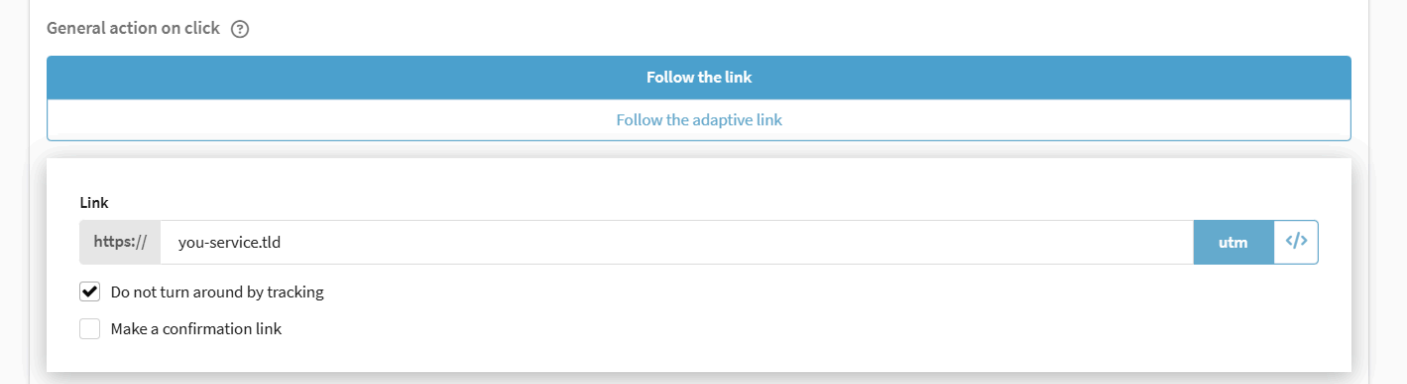
The general action on click is the link that the user will click when they click on the push notification. In this block, you set up a link. Users will click on this link unless a unique link is set for different browsers and OS. The unique link is set below in the advanced settings section.

There are 2 types of links that can be used in the Rich push template editor:

- **regular link;**
- **adaptive link.**

Regular link

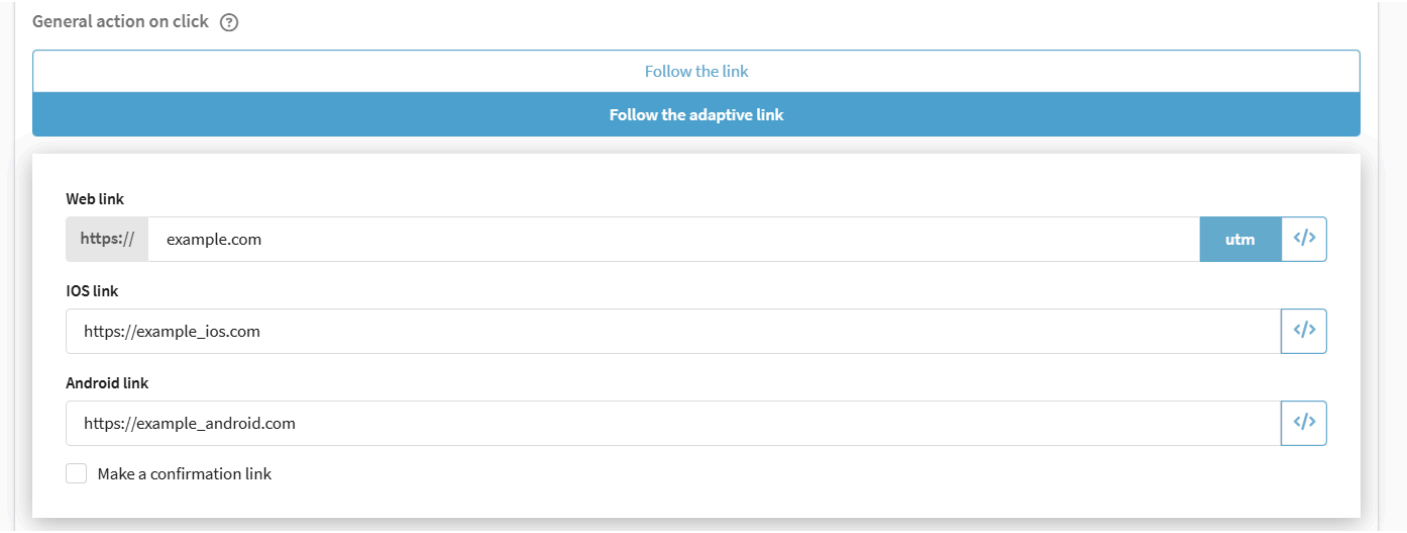
- **Specify the URL for following. You can only use https link, that means that a link must lead to a site that runs on a secure SSL connection.**
- **Insert variables using the button `</>`.**
- **Add UTM tags to track subscriber traffic. Click utm and in the window that opens, configure the labels, and then click Customize UTM-tags.**
- **Activate the field if necessary Do not turn around by tracking — this way you will disable the redirection to tracking when clicking on the link, and click statistics will not be collected.**
- **If you activate the field Make a confirmation link, then a template with such a link can be used to confirm the subscription. When clicking on the link, the profile status will be updated: `Unconfirmed` —> `Subscribed`.**



Adaptive link

Adaptive link is a link to redirect the user to different URLs, depending on which device he opens the push notification from (desktop, Android or iOS).

- **Web link — the user will click on this link if he opens a push notification from a desktop device**
- **IOS link — The user will click on this link if they open a push notification from an iOS device.**
- **Android link — The user will follow this link if he opens a push notification from an Android device.**



Additional settings:

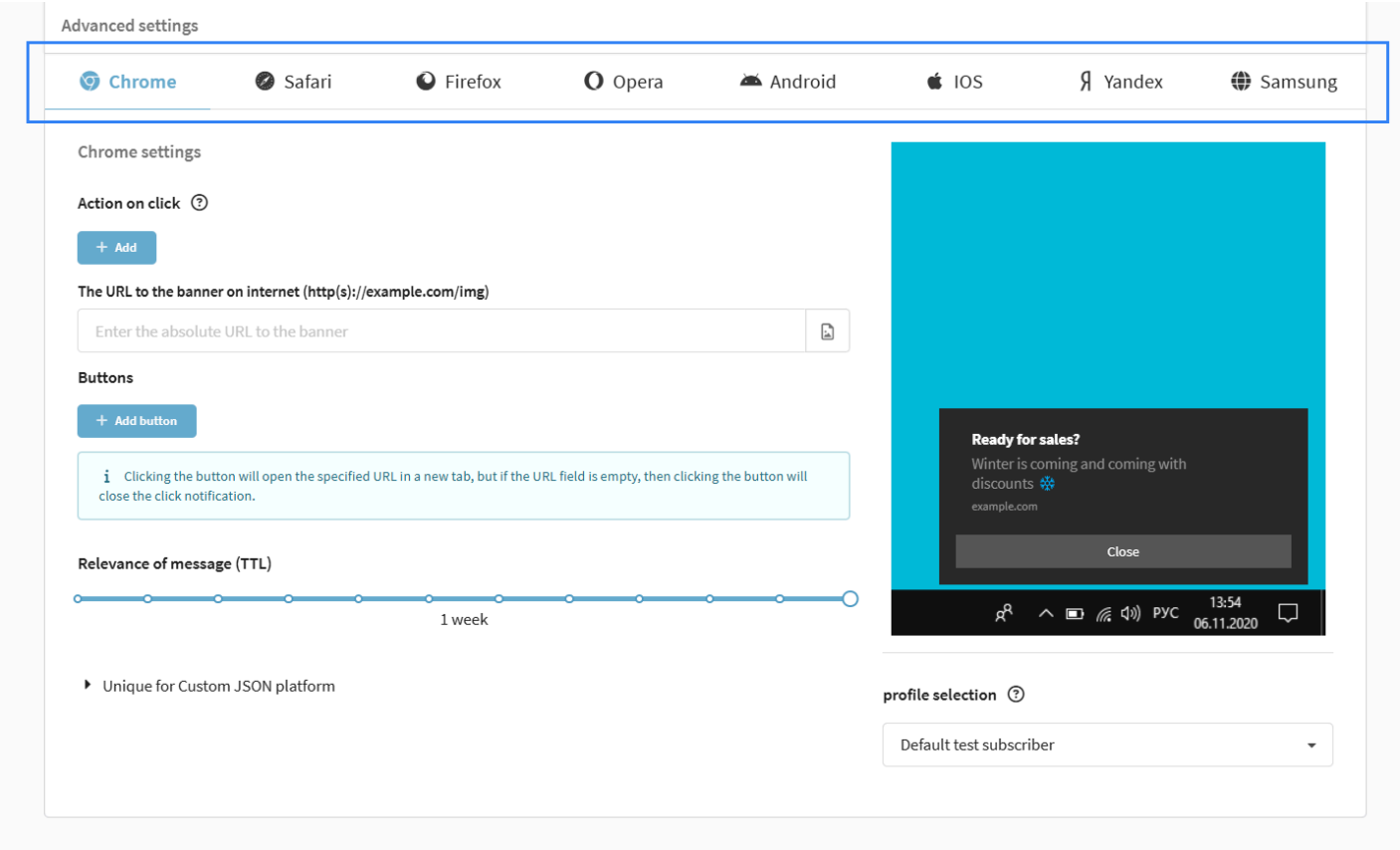
- **Make a confirmation link — a template with such a link is used to confirm the subscription. When clicking on the link, the subscription status of the profile will change from `Unconfirmed` —> `Subscribed`.**
- **Add variables to the link using the button `</>`.**
- **Add UTM tags. To do this, click the “utm” button and in the window that opens, configure the labels, and then click Customize UTM-tags.**

Advanced settings

The extended version of the push notification editor allows you to create push messages for different OS variants and browsers. Switch between OS and browsers using the panel and configure a push message for each of them.

In addition, a preview of push messages is available to you on the left.

You can also specify a test subscriber to see exactly how the created push message will be displayed on his device.



Individual push settings are available for 6 browsers and 2 operating systems. Most of the settings are the same.

Action on click

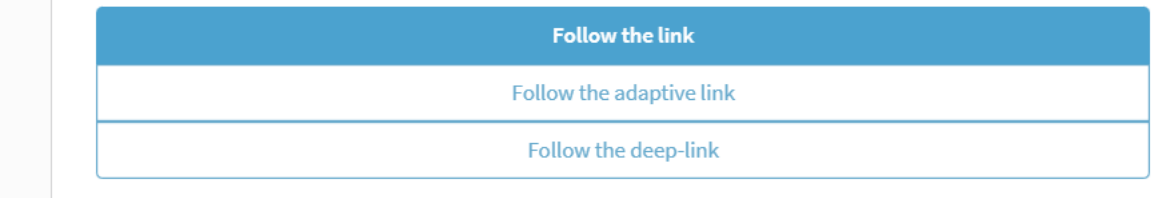
For each browser and OS, you can add click-through link. We remind you that it is a link for a user to follow.

How do I set up a click-through link?

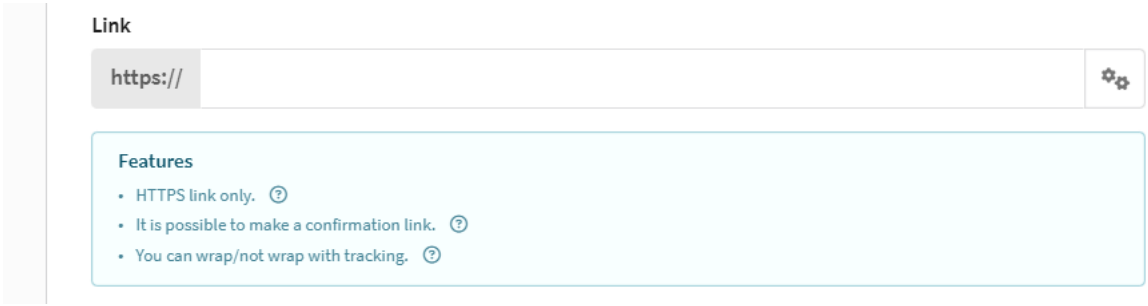
1. Click + Add:

2. Select the type of the click-through link:

- **Regular link;**
- **Adaptive link;**
- **Deep-link link.**



3. Enter the URL of the link and configure it:

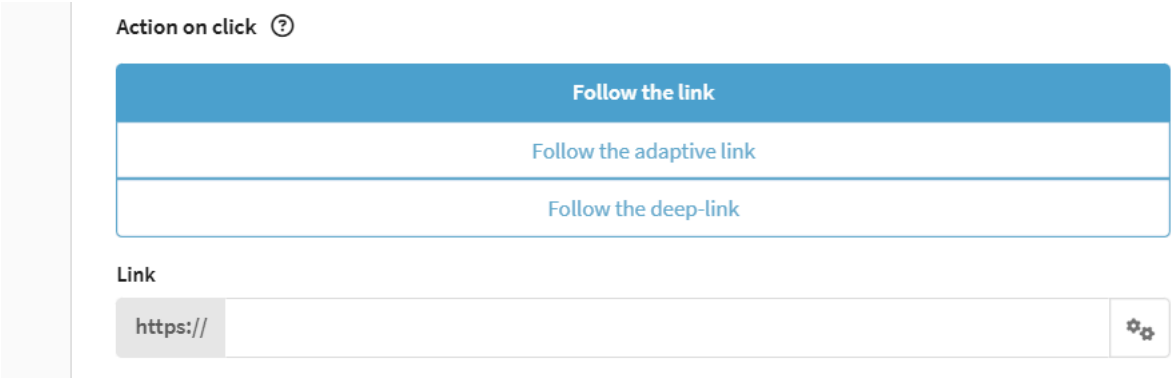


CAUTION

Please note that the link that you specify for each platform “overrides” general action on click.

General action on click

Action on click

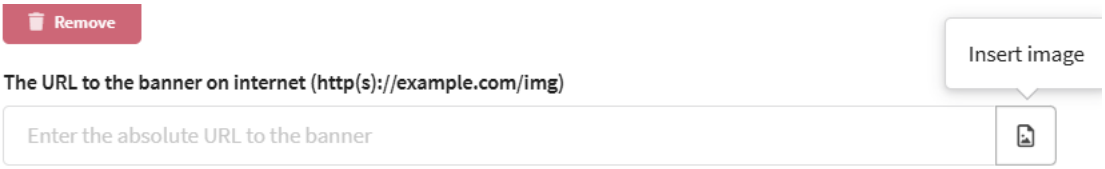


If you leave “action on click” to the platform empty, then even in this case it will “overwrite” general action on click.

When you leave the field with the click link empty, then when you click on it, the user will simply close the push notification.

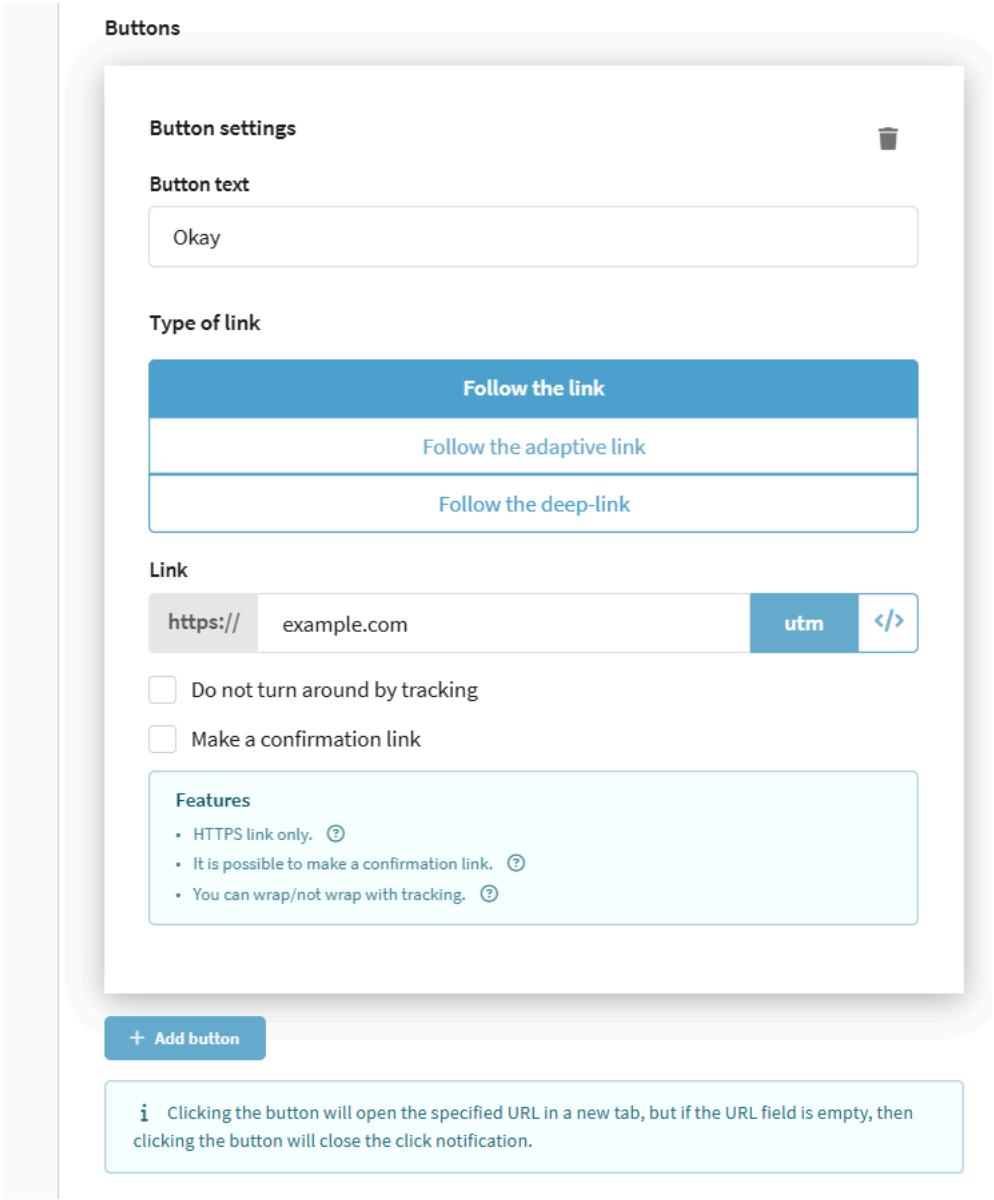
Banner

Specify the URL of the image. If the URL is correct, the banner image will be displayed in the preview. In addition, you can add an image from the gallery.



Buttons

Click Add button. In the window that opens, enter the button text of the button and select the action that the user will perform when he clicks on the button. Read about setting up a click-through link right there.

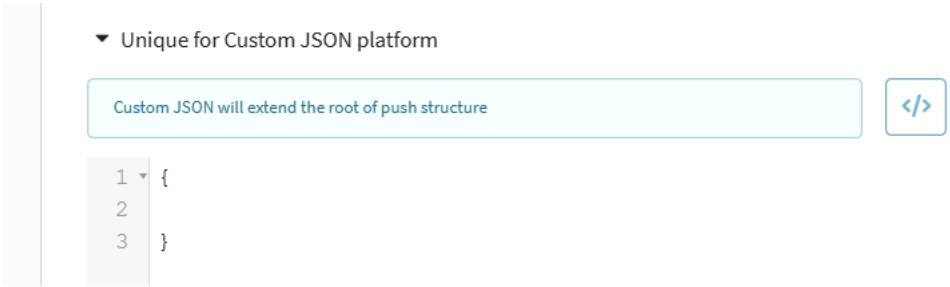


CAUTION

If the field Link is left empty, then when clicking on the button, the user will close the push.

Unique for Custom JSON platform

You can pass additional fields in JSON format to the message to expand the root structure of the push message.



Relevance of message (TTL)

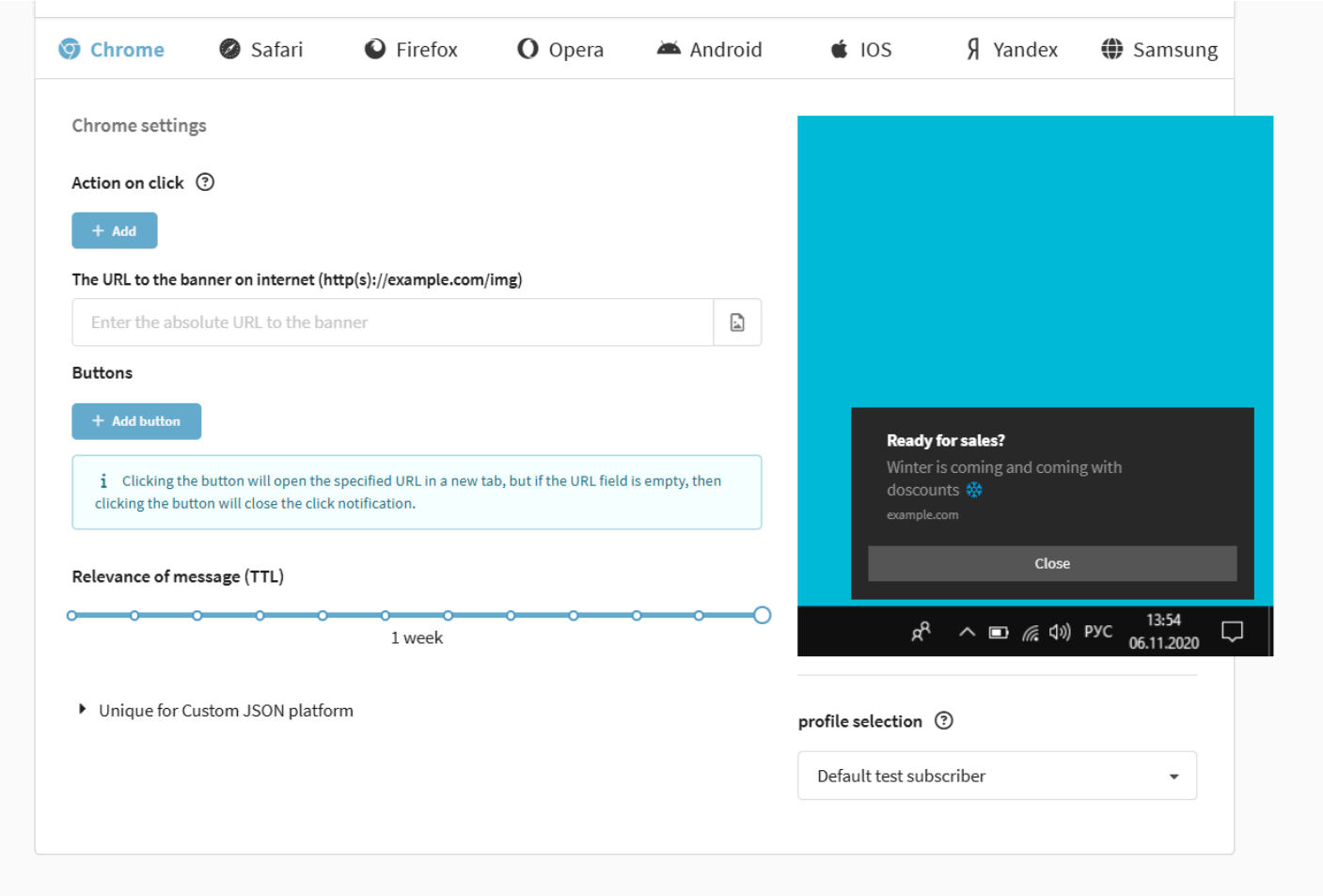
TTL (“time to live”) is the period of time during which the push service monitors the status of a recipient’s device for its connection to the Internet. As soon as a recipient appears online within the specified time, a push message will be delivered to him. If the time has expired and a recipient has not appeared on the network, the system stops tracking its status, and, accordingly, the push notification is considered not delivered.

Move the slider to set the “time to live” of the push message:

Chrome

The following settings are available for this browser:

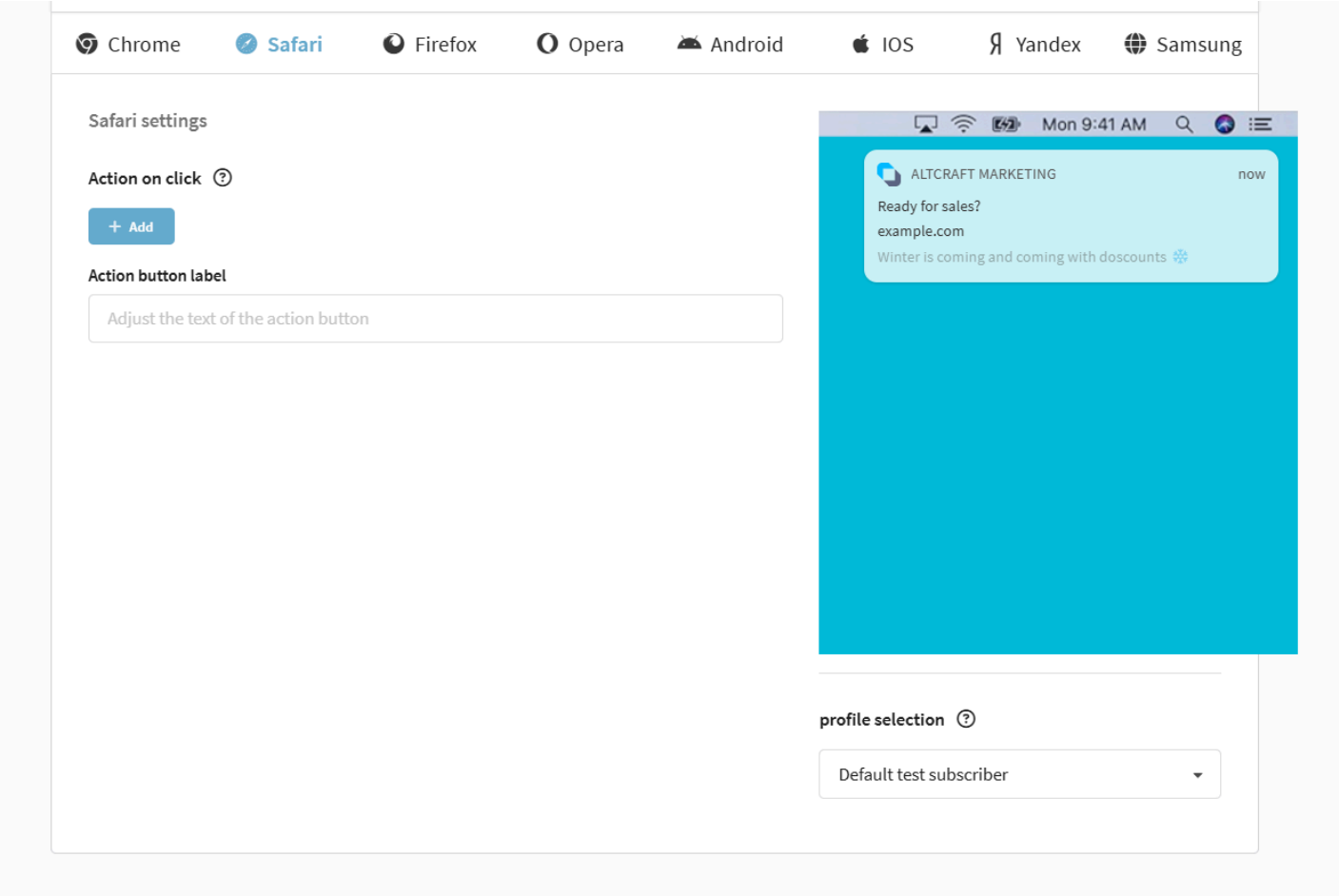
Field	Value
Action on click	**The link that a user will follow when they click on a push message**
Banner	**Link to a banner image**
Buttons	**When the button is clicked, a user will perform a certain action**
TTL	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happens**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Safari

The following settings are available for this browser:

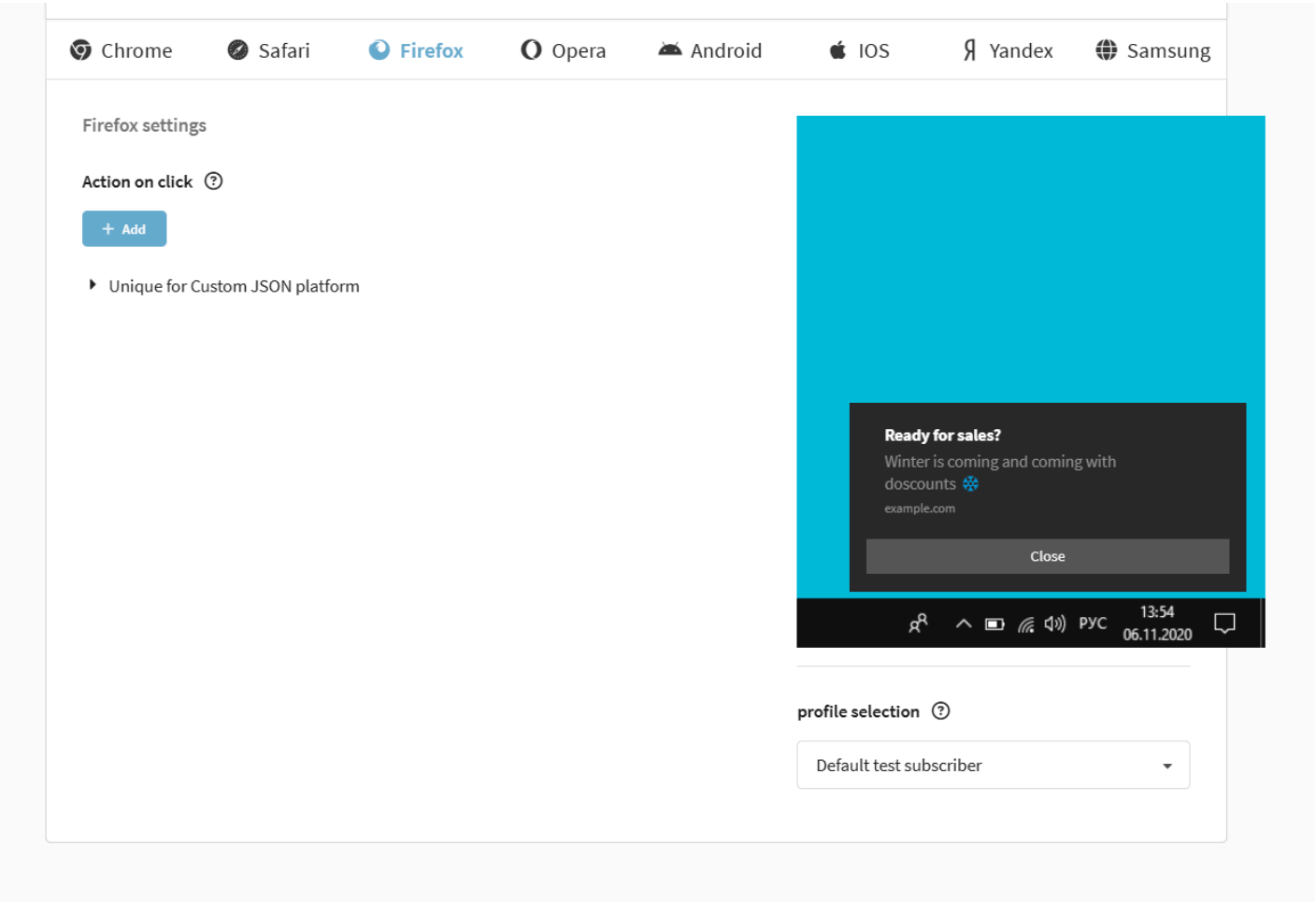
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Action button label	**When the button is clicked, a user will perform a certain action**



Firefox

The following settings are available for this browser:

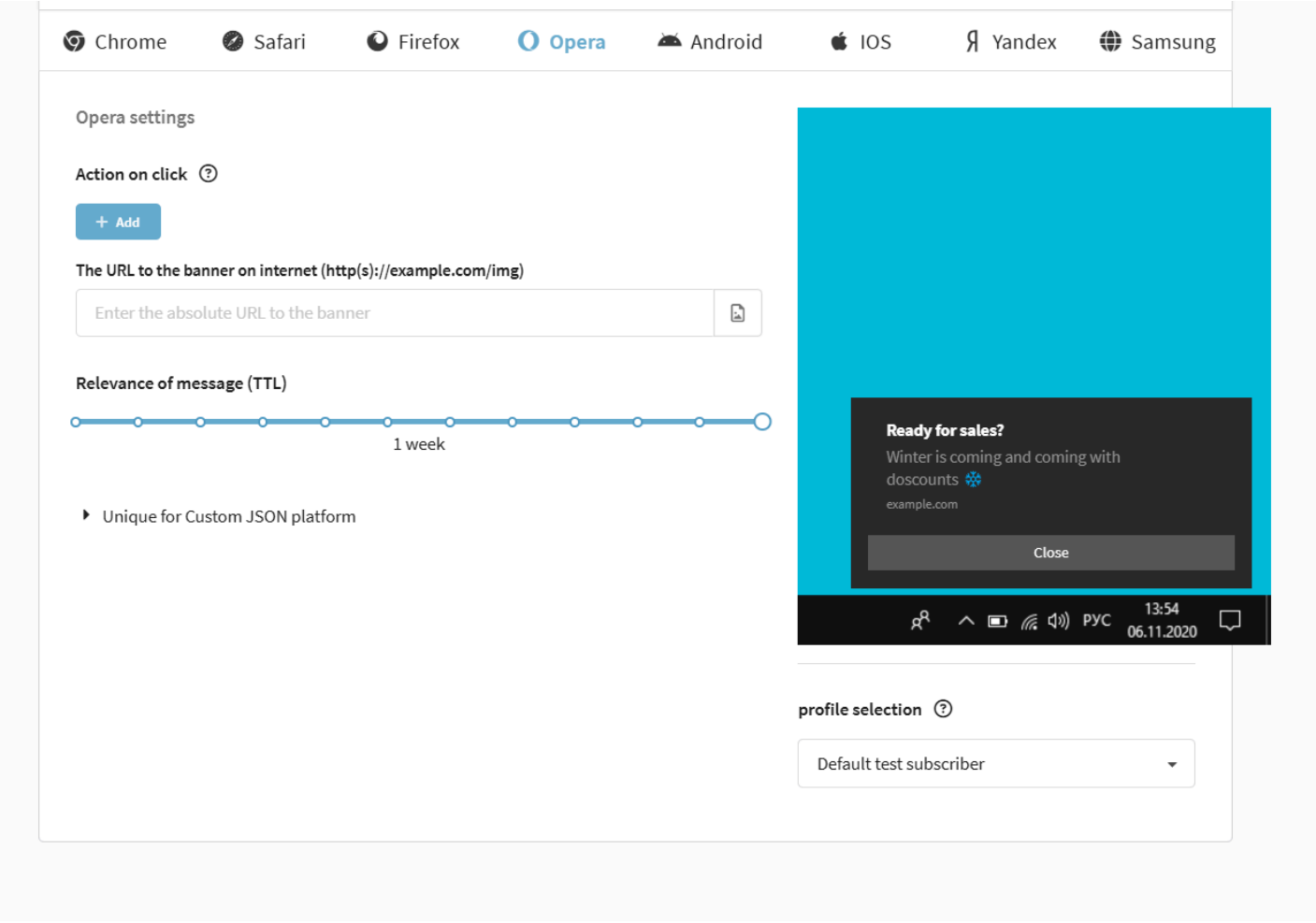
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Opera

The following settings are available for this browser:

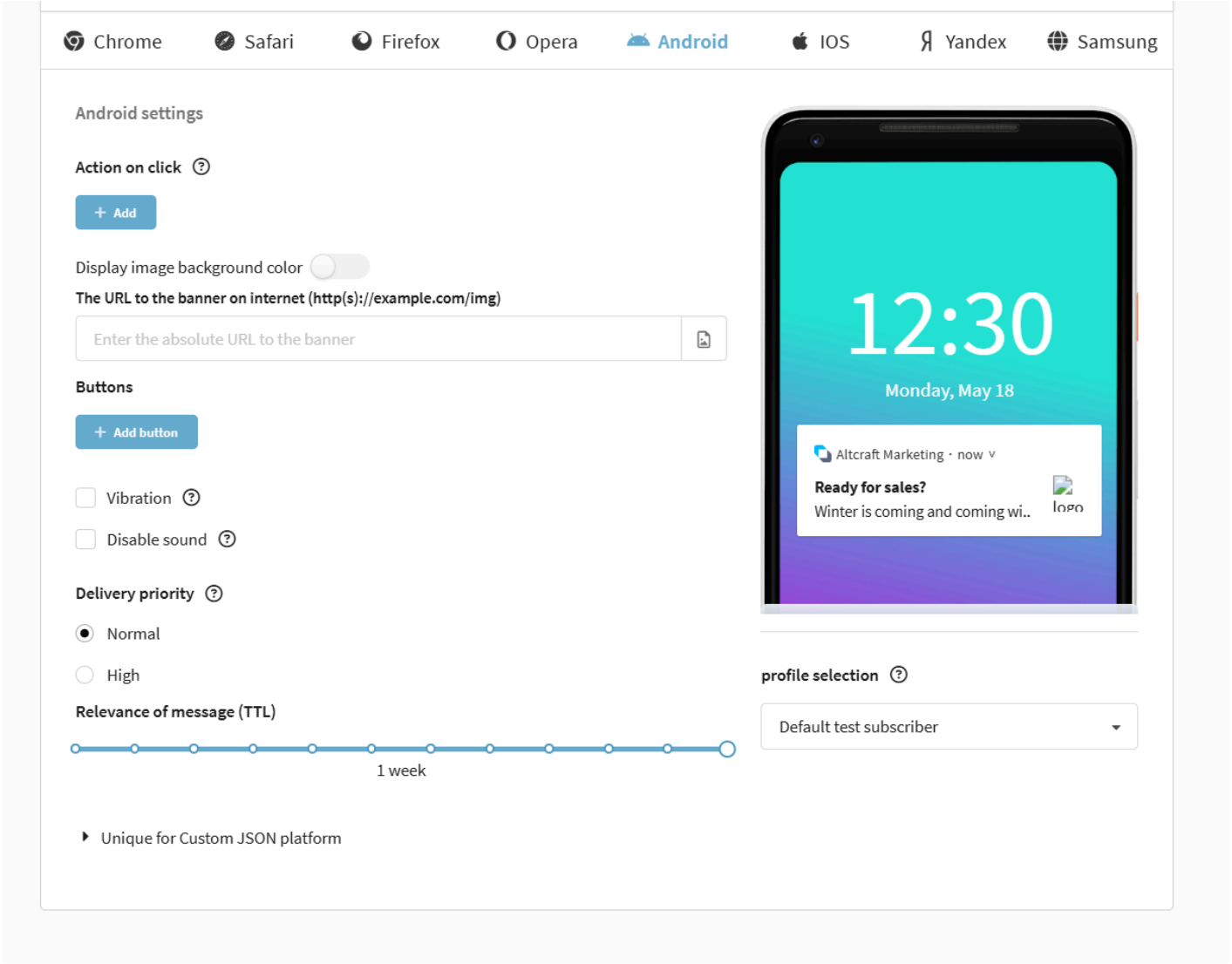
Field	Value
Action on click	**A link that the users will follow when they click on a push message**
Banner	**Link to the banner image**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happens**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Android

The following settings are available for this OS:

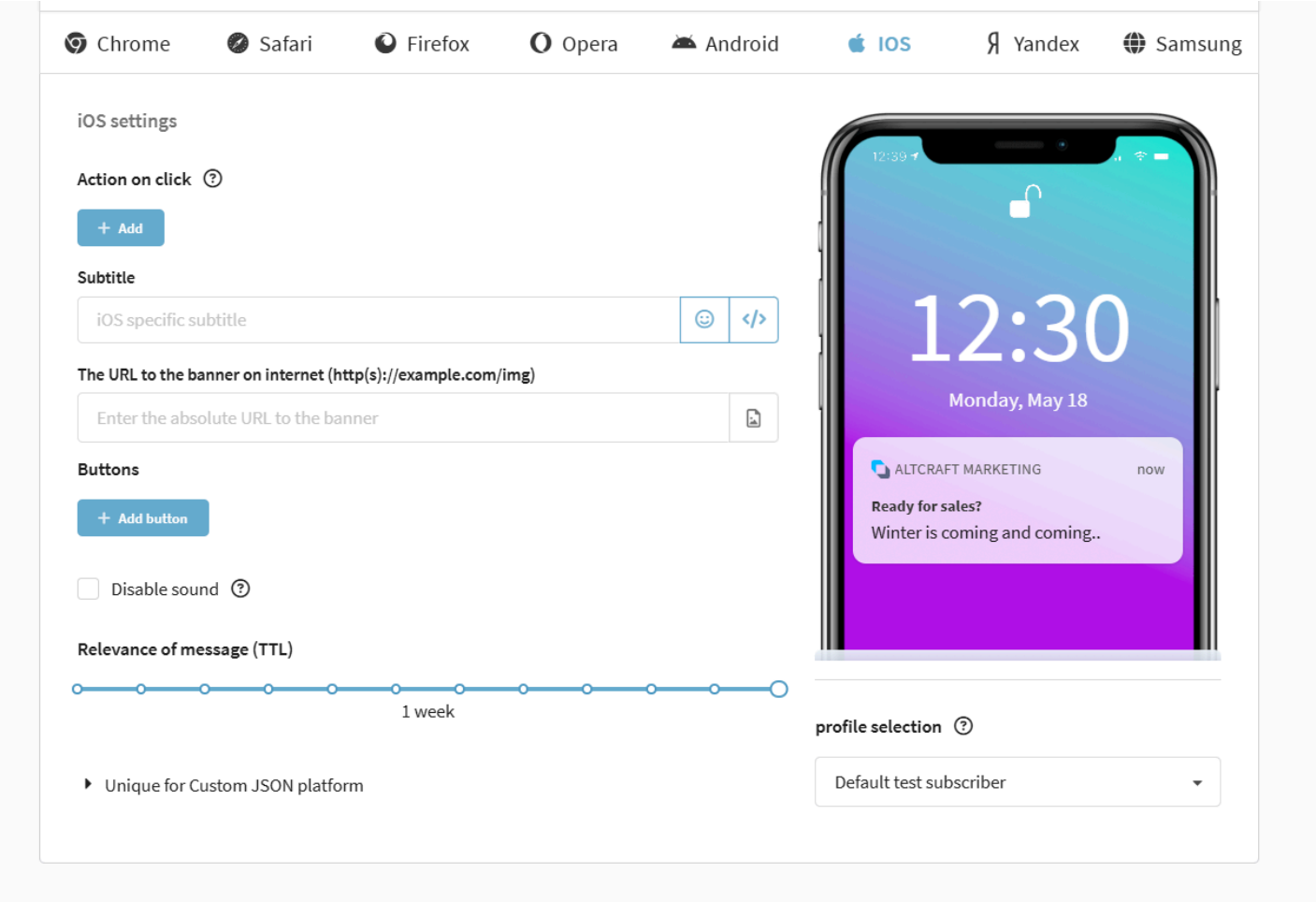
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Display image background color	**Background of the push notification**
Banner	**Link to the banner image**
Buttons	**When the button is clicked, a user will perform a certain action**
Vibration	**When receiving a push notification, user's device will reproduce vibration**
Disable sound	**When delivered to a device, a push notification will not play sound or vibration**
Delivery priority	**Normal: the delivery of messages with normal priority can be postponed indefinitely to save battery power.** **High: push notification launches the app in the background and immediately gets internet access.**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happens**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



iOS

The following settings are available for this OS:

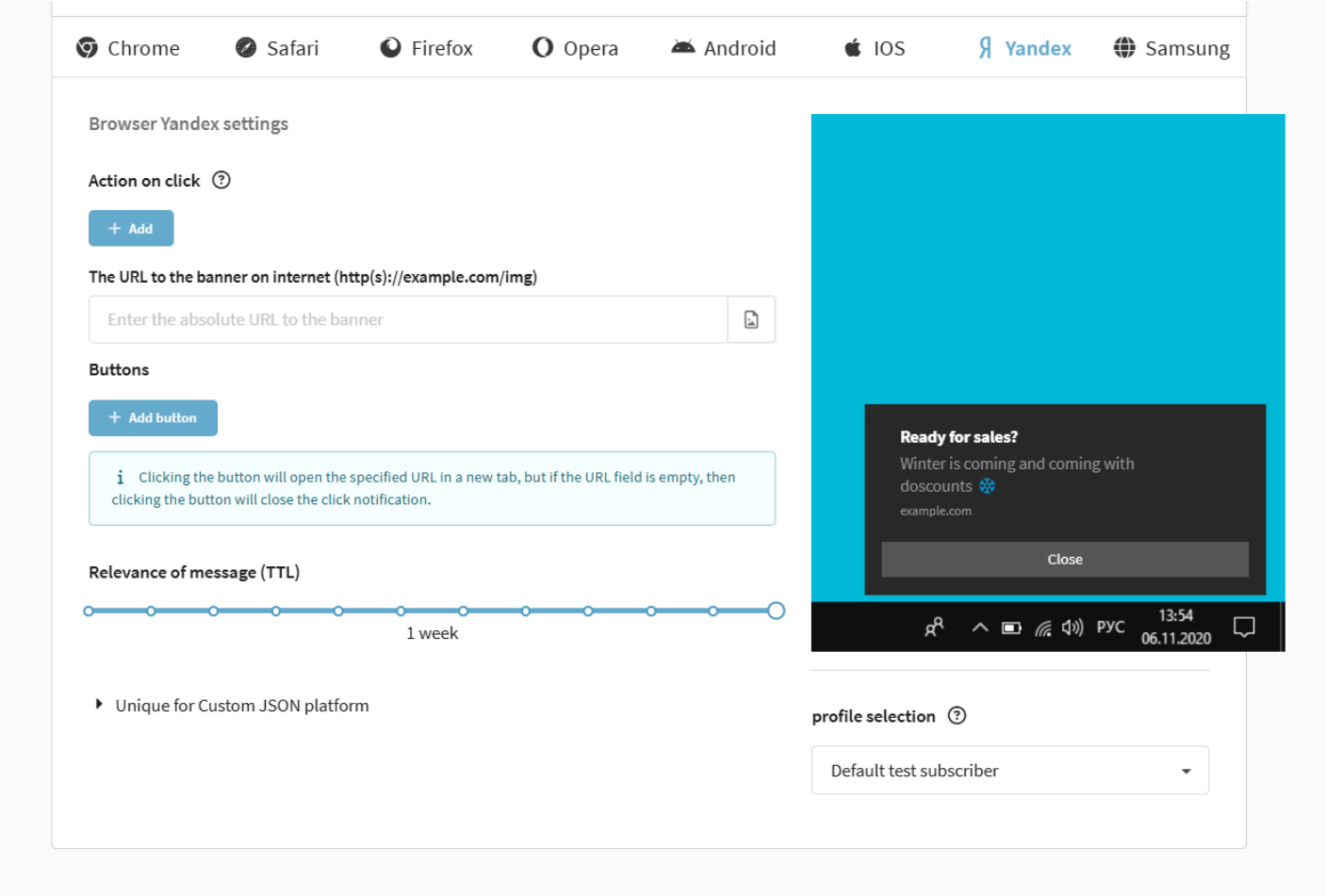
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Subtitle	**IOS-specific subtitle**
Banner	**Link to the banner image**
Buttons	**When the button is clicked, a user will perform a certain action**
Disable sound	**When delivered to the device, the push notification will not play sound and vibration**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happens**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Yandex

The following settings are available for this browser:

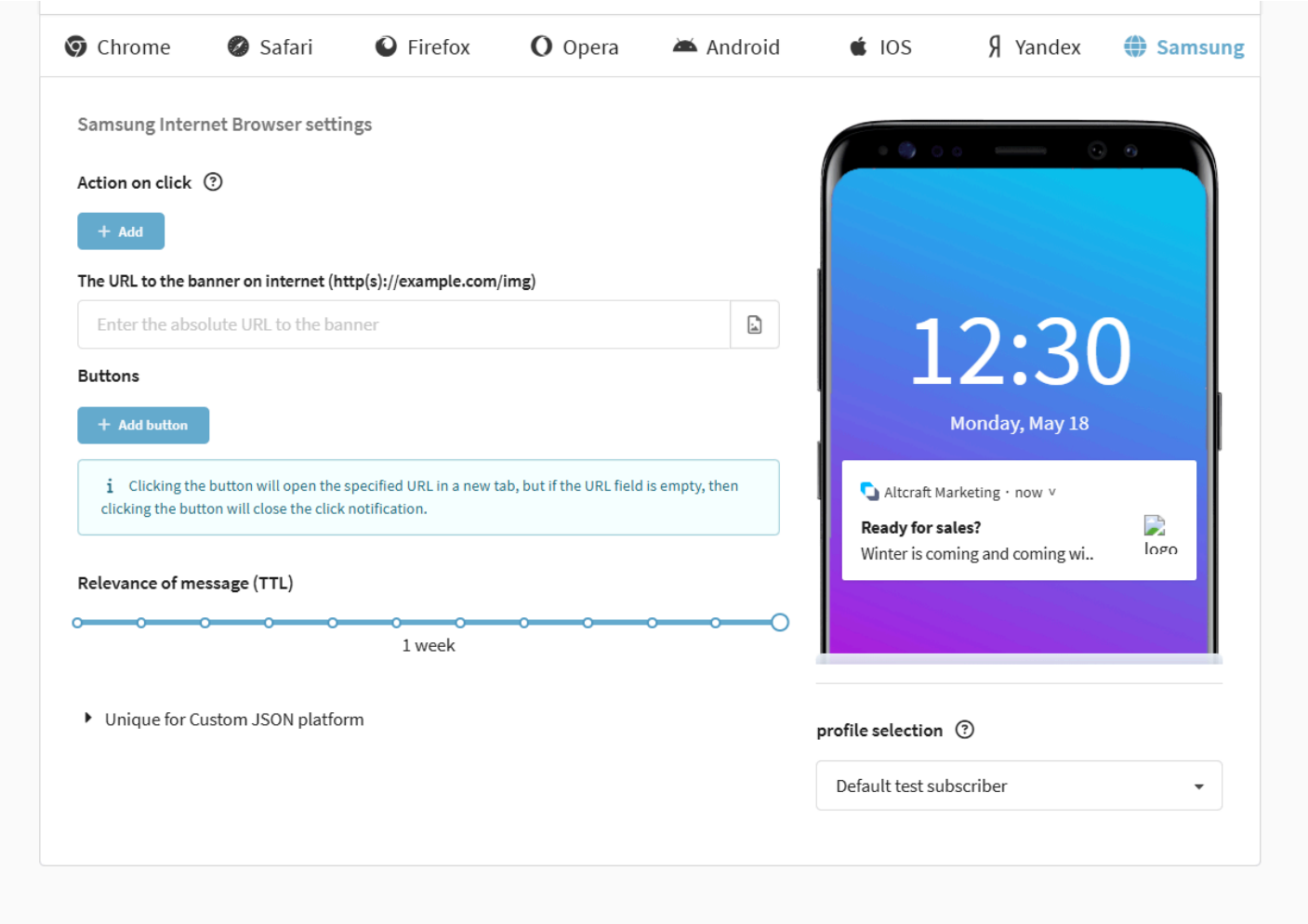
Field	Value
Action on click	**A link that users will follow when they click on a push message**
Banner	**Link to the banner image**
Buttons	**When the button is clicked, the user will perform a certain action**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happens**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Samsung

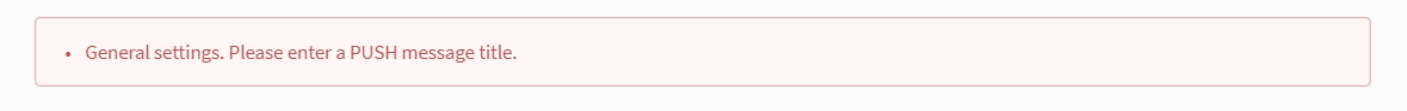
The following settings are available for this browser:

Field	Value
Action on click	**A link that users will follow when they click on a push message**
Banner	**Link to a banner image**
Buttons (no more than two at once)	**When the button is clicked, the user will perform a certain action**
Relevance of message (TTL)	**The time during which the service monitors a device's connection for delivery of a message. Then an expiration happens**
Unique for Custom JSON platform	**Additional fields in JSON format to expand the root structure of the push message**



Once you have set up all the necessary parameters for your template, save it by clicking on the “Save” button:

If you made some mistake when filling template fields, it will be displayed at the top of the screen. Correct your error and save the template again.



Now you can use this template to send Rich Push notifications. Read more about how to do this here: [Web push: setup and connection](#).

MAX template

TIP: MAX Bot and MAX Group

Templates for sending via MAX Bot and MAX Group are configured in the same way, except that profile data variables are not available for MAX Group because it is an indirect communication channel.

This article describes how to create templates for sending messages via MAX Group and MAX Bot.

Message elements

A MAX template consists of elements. To start editing a template, click + Element:

The following elements are available:

- **Text — sends a text message**
- **Image — attaches an image to the message**
- **Video — attaches a video file to the message**
- **Audio — attaches an audio file to the message**
- **File — attaches a file to the message**

To interact with the recipient, you can add inline buttons to the message:

- **Text button — displays a button with text**
- **Link button — opens the specified link**
- **Callback button — after being clicked, sends the specified text or command back to the bot**
- **Open app — opens a mini app**
- **Request location — requests the user's location**
- **Request contact details — requests the user's contact and phone number**
- **Button group — a combination of buttons of several types**

These buttons cannot be used without other elements.

The following limits apply to MAX templates:

- **Up to 32 elements in one template — up to 30 buttons, text, and one media file**
- **You can attach only one media type to a message (images, video, audio, or file)**
- **Up to 12 images in one template, while all other media support only one attachment**
- **A single button group can contain up to 7 buttons if only text and callback buttons are used, or up to 3 buttons if at least one button of another type is used (link, open app, request location, or request contact details)**

TIP: Link previews

If the message text contains links to web pages, MAX displays a preview of that page by default:

If you do not want the preview to be displayed, enable the Disable link previews in MAX switch.

Text

A text message that will be sent by the bot. For easier formatting, the top panel includes the following functions:

B

— bold text

I

— italic

~~S~~

— strikethrough text

— link editor (see more here). Links inserted through the platform are shortened automatically. The final link path length excluding the tracking domain is 12–14 characters, including `/`.

utm

— add UTM tags

</>

— list of editor variables and functions for message personalization

☺

— insert emoji. Note that each emoji counts as two characters.

On the right, there is a character counter. The maximum length of a single message is 4000 characters. Note that data inserted from dynamic variables may exceed this limit. To ensure reliable delivery, we recommend monitoring the message length and shortening the text when necessary.

Image

Sends a message with images. You can attach up to 12 images. Files are selected from the gallery or uploaded from a device.

The maximum image size for insertion into a template is 10Mb. Supported formats: .jpg, .png, .gif

If this element is used in the template, other media elements (video, audio, or file attachments) cannot be added.

Preview in MAX

Video

Sends a video. You can attach only one video file. The maximum video size for insertion into a template is 5Mb. Supported formats: .mp4, .mov .webm .mkv .avi

If this element is used in the template, other media elements (image, audio, or file attachments) cannot be added.

Preview in MAX

Audio

Sends an audio message. You can attach only one audio file. The maximum audio size for insertion into a template is 5Mb. Supported formats: .mp3, .ogg, .wav, .m4a, .m4p, .flac, .wma, .aa

If this element is used in the template, other media elements (image, video, or file attachments) cannot be added.

Preview in MAX

File

Sends a file up to 5Mb in .zip, .tar, .gz, .xls, .xlsx, .csv, .txt, or .pdf format

If this element is used in the template, other media elements (image, video, or audio) cannot be added.

Preview in MAX

Text button

Displays a button with text. You can add emoji and platform variables to the button text. The maximum number of characters in button text is 128.

Preview in MAX

Link button

Opens the specified link. For this button, you need to specify the text displayed on it and the destination link. You can add emoji and platform variables to the button text. The maximum number of characters in button text is 128.

Preview in MAX

Callback button

When a callback button is clicked, the bot receives a notification and accepts the message payload. You can add emoji and platform variables to the button text. The maximum number of characters in button text is 128. The payload can contain plain text or a command that the bot should execute.

Preview in MAX

Open app

Opens the specified mini app when clicked. You need to specify the button text and the app identifier. It can be passed either as `Web_app` (the username of the bot whose mini app will be launched, or a link to it, passed as a string value) or as `Contact_id` (an integer value, the bot ID). You must pass only one parameter.

INFO

We recommend passing `Web_app`, since using `Contact_id` may cause errors on the messenger side.

You can add emoji and platform variables to the button text. The maximum number of characters in button text is 128. The payload can contain plain text or a command that the bot should execute.

Preview in MAX

Request location

When clicked, sends the user's location. You can add emoji and platform variables to the button text. The maximum number of characters in button text is 128.

Preview in MAX

Request contact details

When clicked, sends the user's contact details. You can add emoji and platform variables to the button text. The maximum number of characters in button text is 128.

Preview in MAX

Button group

A combination of several buttons. A single button group can contain up to 7 buttons if only text and callback buttons are used, or up to 3 buttons if at least one button of another type is used (link, open app, request location, or request contact details).

Preview in MAX

Preview and save

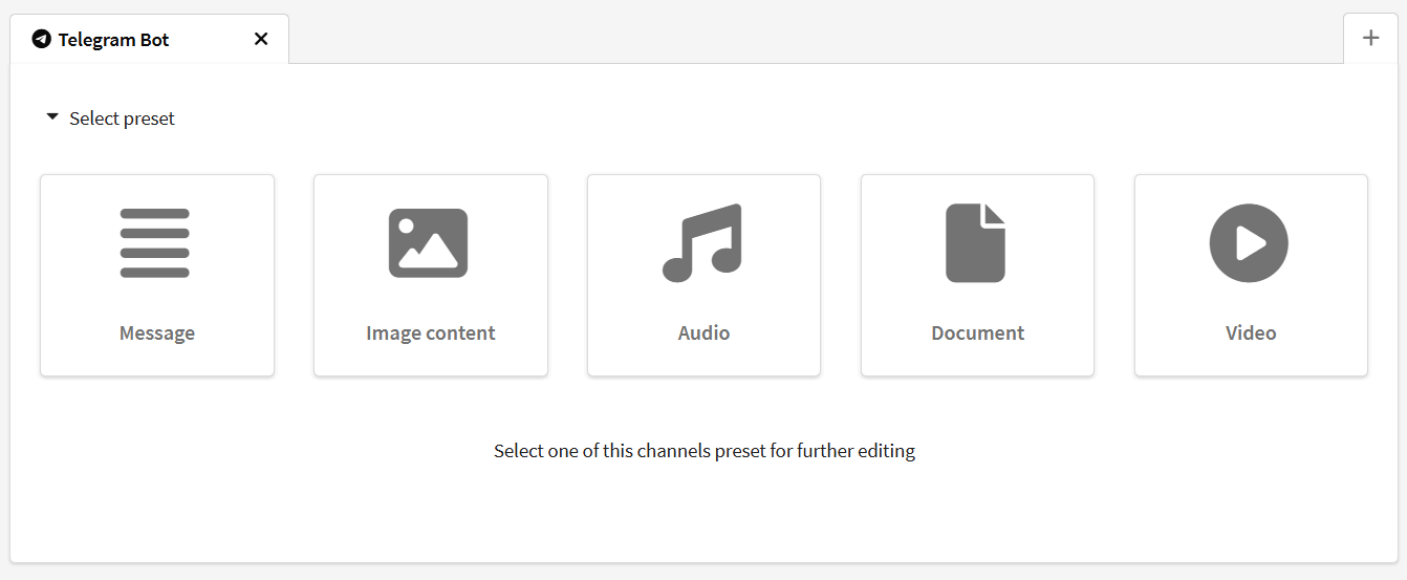
At the top of the template editor, there is a button for switching to preview mode. It can be used to check how the message will appear in the recipient's messenger:

After all required elements have been added, save the template. You can then use it in Campaigns via MAX Group or MAX Bot.

Telegram template

This article describes how to create templates for sending messages through channels such as Telegram Group and Telegram Bot.

First, select the type of content you want to send to subscribers:

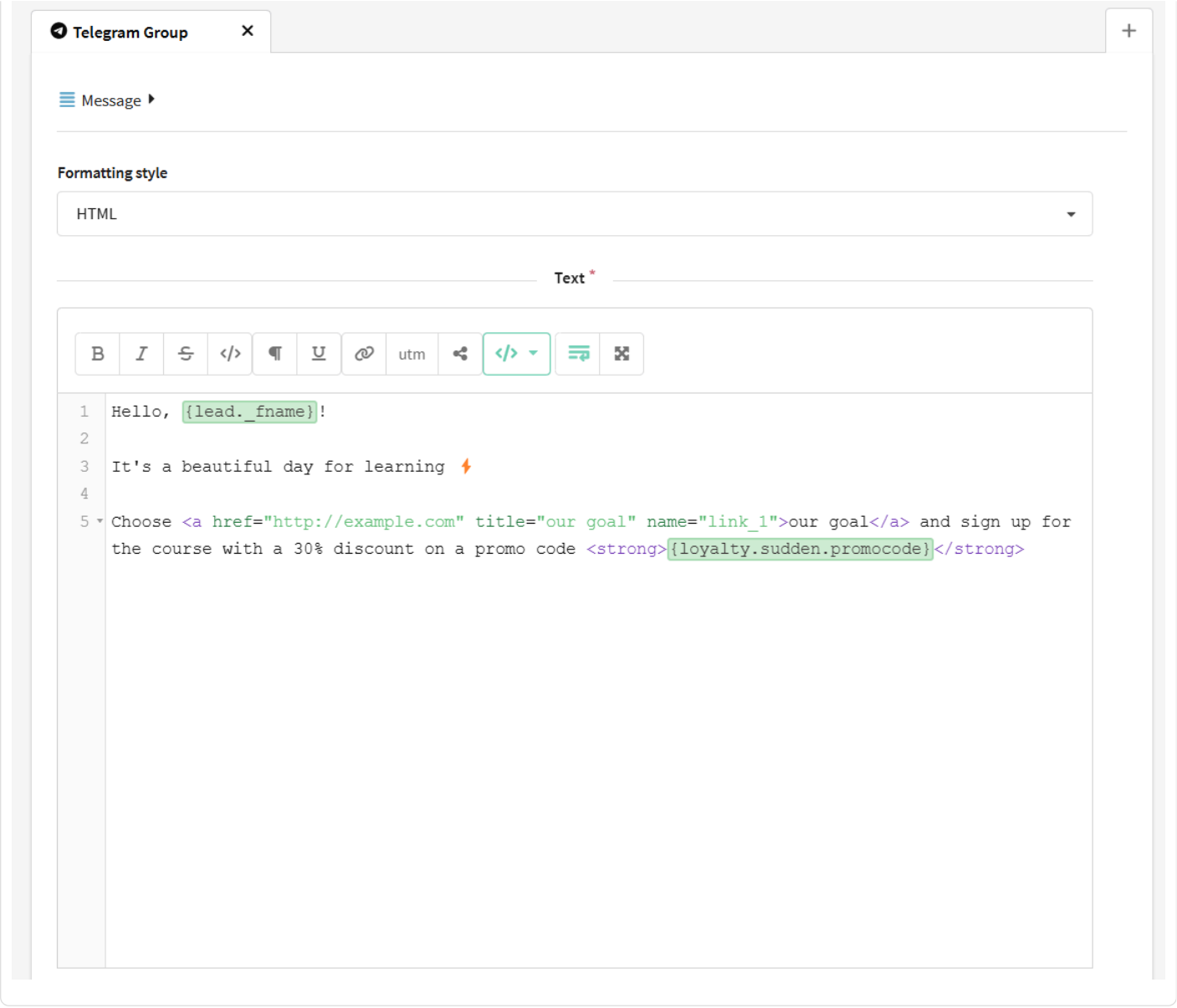


Message

Select the formatting style. We recommend choosing HTML, since the rest of the formatting styles are outdated. Then enter the text of the message.

TIP

The maximum length of a text message in Telegram is 4096 characters. Please note that data that is pulled from dynamic variables may exceed this limit.



The toolbar in the Telegram editor

The following functions are available to facilitate formatting and working with objects:

B

— bold text.

I

— italics.

~~S~~

— ~~strikethrough text~~.

</>

— code block.

¶

— preformatted text.

U

— underlined text.

🔗

— link editor, read more right there. Links inserted through the platform are automatically shortened. The final link path (excluding tracking domain) length is 12-14 characters, including the `/`.

utm

— adding UTM tags.

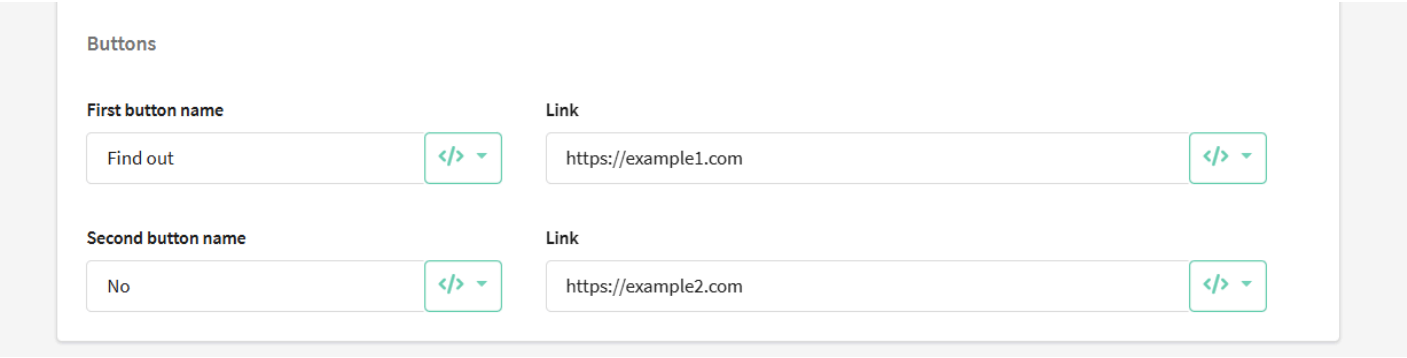
— adding buttons Social share links.

— menu of variables and functions of the editor. Using this button, you can add customer profile data, promo codes to the message, loops and boolean expressions, as well as dynamic content and system links.

— toggle word wrapping.

— full-screen mode.

To add a button to a message, enter a name and a link that the user will follow by clicking on the button. It is possible to add no more than 2 buttons.

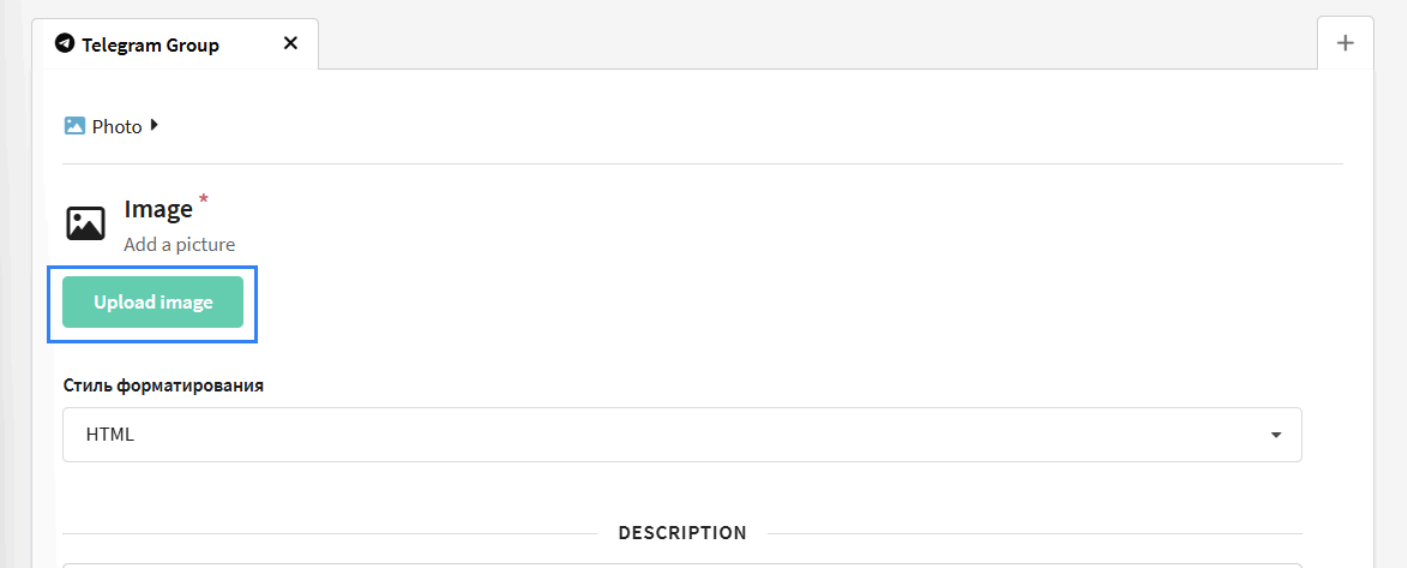


TIP

The links in the templates for Telegram Bot and Telegram Group are shortened.

Media files

In order to add am image, click on the button Upload image and select an image in the gallery.



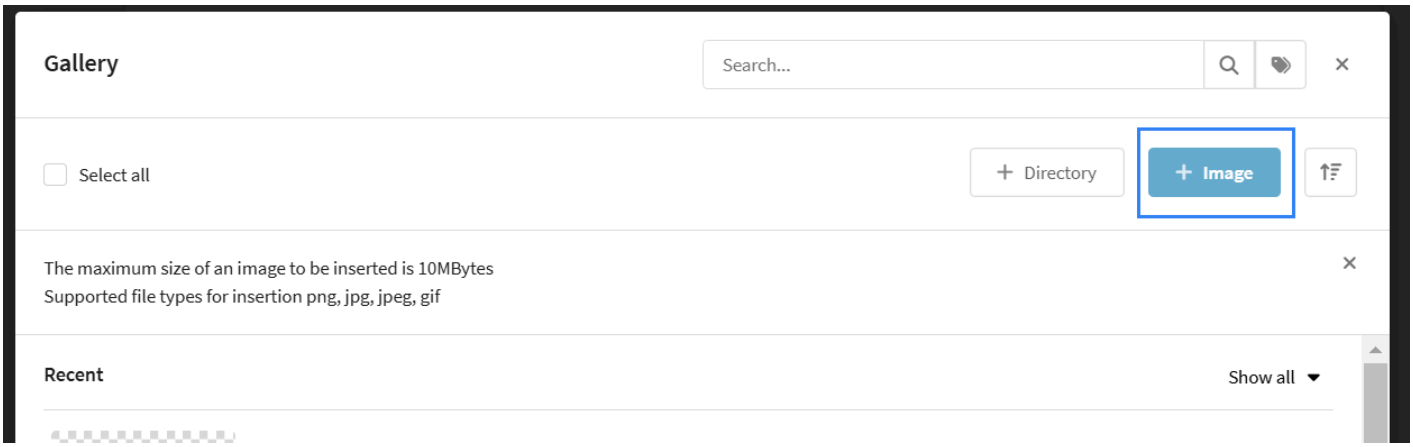


Image requirements:

- **Formats: PNG, JPG, JPEG, GIF.**
- **Maximum size: 10 MB.**
- **The sum of the width and height of the photo should not exceed 10000.**
- **The ratio of width and height should not exceed 20.**

In order to add an audio, video or document, use one of the options:

1. **Download the file from your device.**
2. **Add the file to the local path. Select the file storage directory on the server where the platform is installed. You can use client profile variables in the directory path. Enter the location of the file and click “Add”.**

Audio file requirements:

- **Formats: MP3 and M4A.**
- **Maximum size: 5 MB.**

Document requirements:

- **Formats: any document format.**
- **Maximum size: 5 MB.**

Video requirements:

- **Formats: MP4.**
- **Maximum size: 5 MB.**

TIP

You can also add a description (up to 1024 characters) to files of any type.

WhatsApp template

Description

At the moment, only template messages (HSM) can be sent via Altcraft Platform.

A template message is an HSM message that must be pre-approved by Facebook. Use this type of template when initiating a conversation with a user. As of July 1, 2025, each sent template is billed separately based on its category.

If a customer contacts you first on WhatsApp, a 24-hour window opens during which you can send service and non-template messages free of charge.

Message templates for WhatsApp are created in the provider's personal account.

CAUTION

Currently, only typical templates (HSM) can be sent via the Altcraft Platform.

Message templates for sending to WhatsApp are created in the provider's personal account.

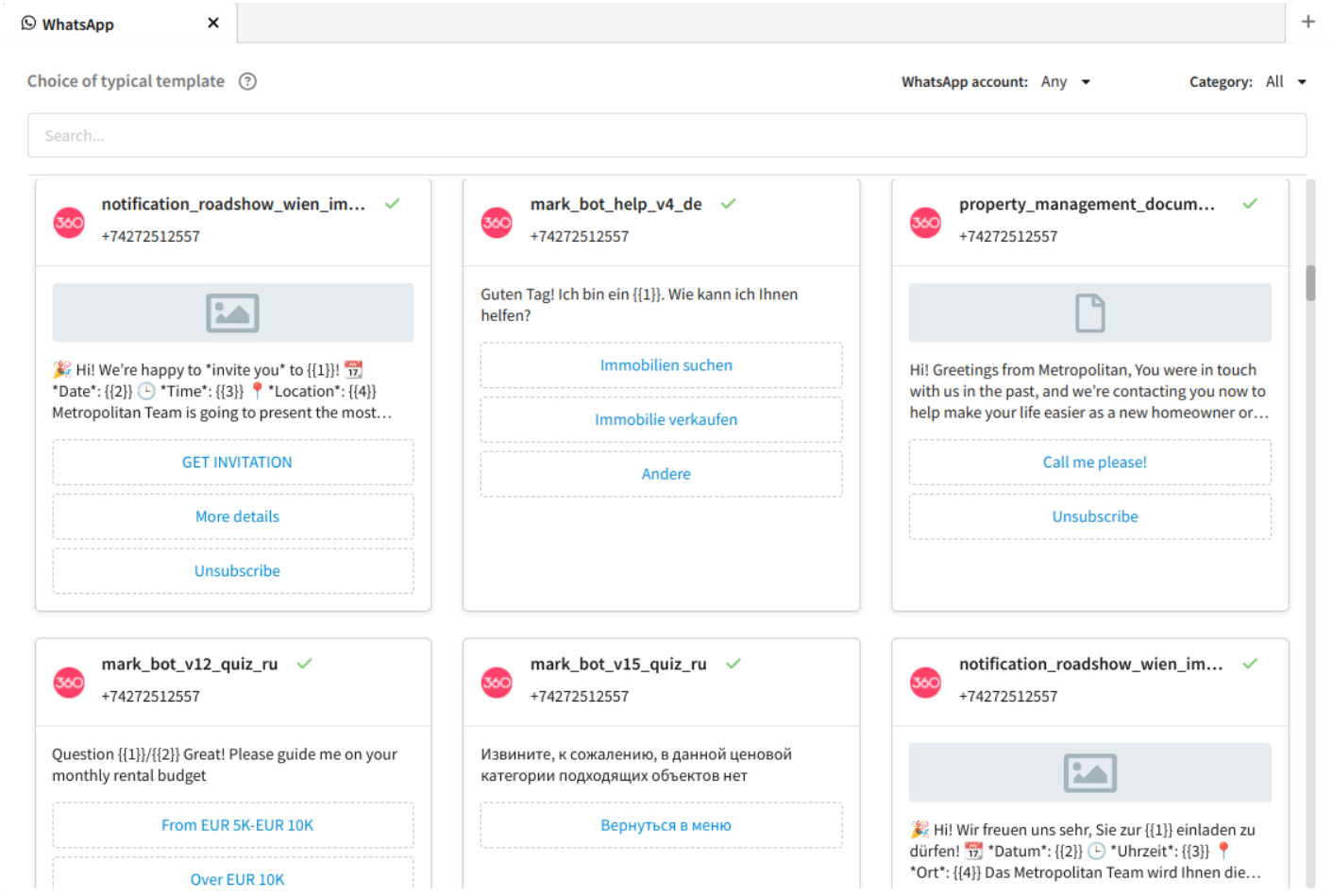
TIP

You can find more information about creating templates in your provider's documentation.:

- **Edna: https://pulse-alpha-six.vercel.app/docs/template/whatsapp/operator-template**
- **360dialog: https://docs.360dialog.com/docs/360-client-hub/template-management-ui**

Creating a message

All templates available in WhatsApp accounts connected via Edna or via 360dialog are automatically available in Altcraft Platform and displayed in the editor. To search for the desired template in the platform, you can enter a query in the search bar or use the filter by account/category in the upper right corner.



CAUTION

Mailings can only use templates agreed with Facebook.

The status of the template is displayed to the right of its name:

— the template has successfully passed moderation;

— template rejected;

— the template is under moderation;

— the template has not been submitted for moderation.

Select the template you want to use in the mailing. To create a message based on a WhatsApp template, click on it. The template structure will be displayed in the editor:

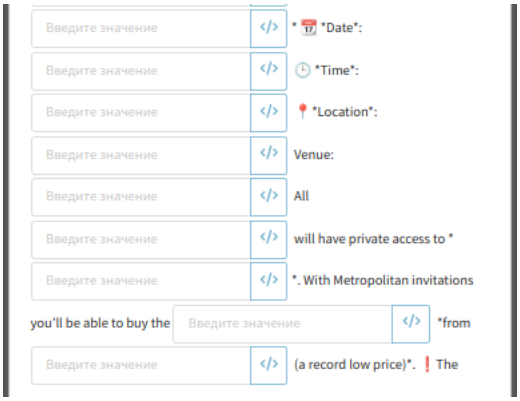
- **Header — template header with a welcome text or media file;**
- **Body — the main text of the message with variables (required block);**
- **Footer — additional text;**
- **Button — a block with buttons to add interactivity.**

You can change template elements such as variables, media files and buttons.

Variables

A text block in WhatsApp consists of fixed text and variables. Variables are the template text that changes for each new message (name, address, event name, date and time, link, etc.).

You can manually set variable values or add Altcraft variables. To open the list of variables, click `</>`. When sending a message, the entered text is automatically inserted into the template.



TIP

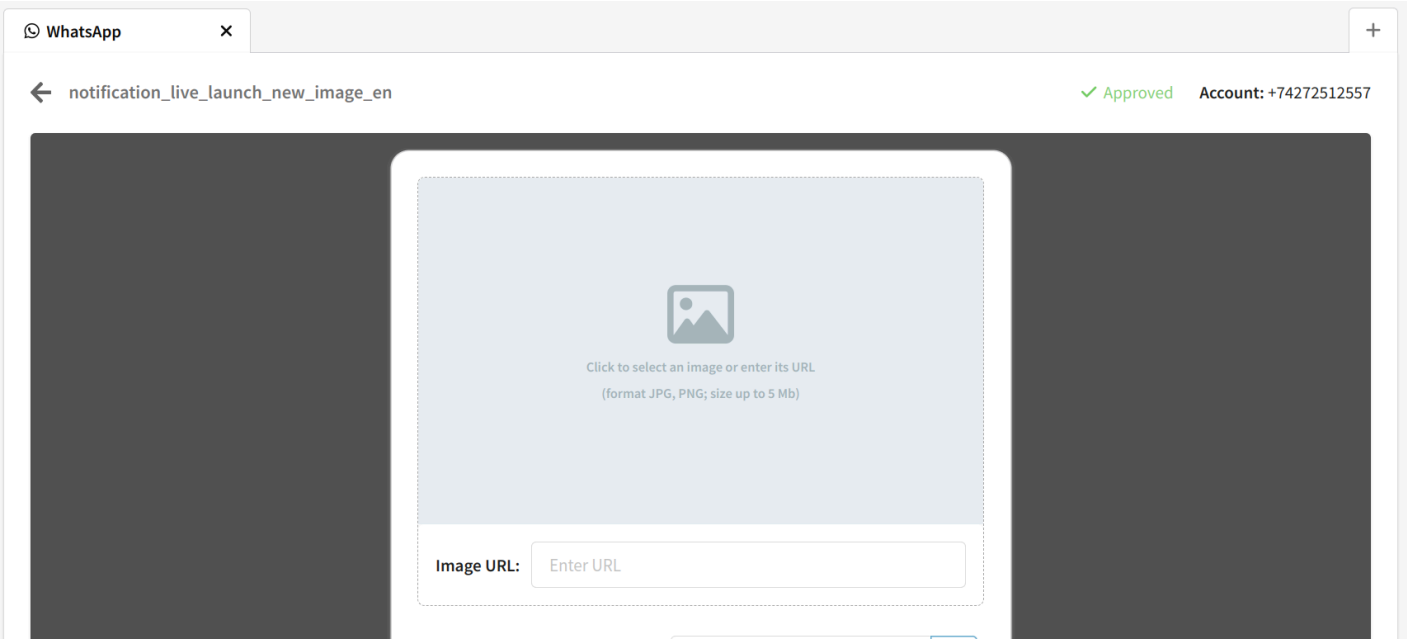
Links inserted through the platform are automatically shortened. The final link path (excluding tracking domain) length is 12-14 characters, including the `/`.

Media Files

Available media files:

- **image (JPG/PNG, up to 5 MB, recommended image size is 640x640.)**
- **file (any correct MIME type, up to 16 MB)**
- **video (MP4, 3GP, 3GPP, up to 100 MB)**

To insert an image, click on the download area and select the image in the platform gallery or enter the URL:



To add a file or video, enter the URL where the file is located.

Buttons

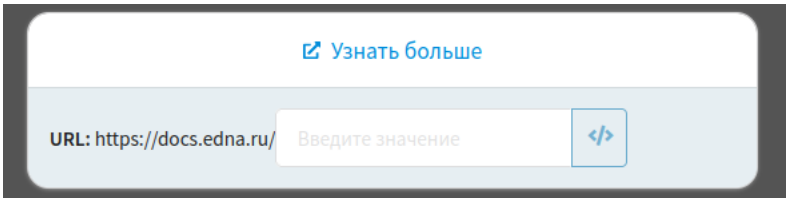
WhatsApp offers two types of buttons for HSM templates:

- **Call-to-action — call-to-action for users;**
- **Quick-reply — for user to reply quickly.**

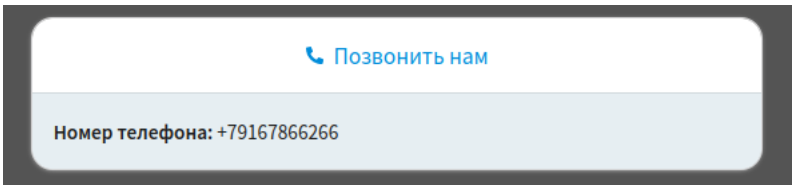
Call-to-action buttons

The CTA is a link button (URL) and a call button (Phone). There can be only 2 buttons of this type in the template.

When clicking on the link button, a user goes to the specified page. If a parameterizable URL is specified in the WhatsApp template, you need to fill it in.

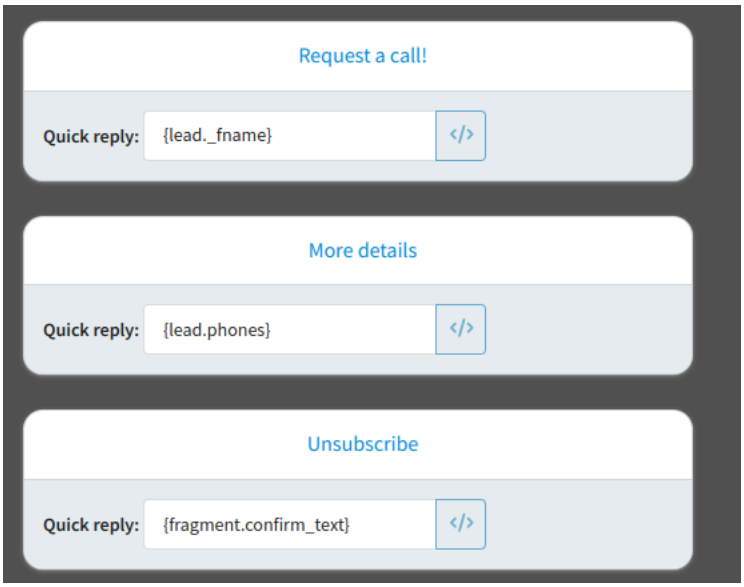


When you click on the call button, the number is dialed. The phone number cannot be edited.



Quick-reply buttons

When clicking on the button, a user receives a quick response written in the template. The template can contain no more than 3 buttons of this type.



TIP

You can save the template after filling in all the variables, buttons and adding a media file.

Preview the message

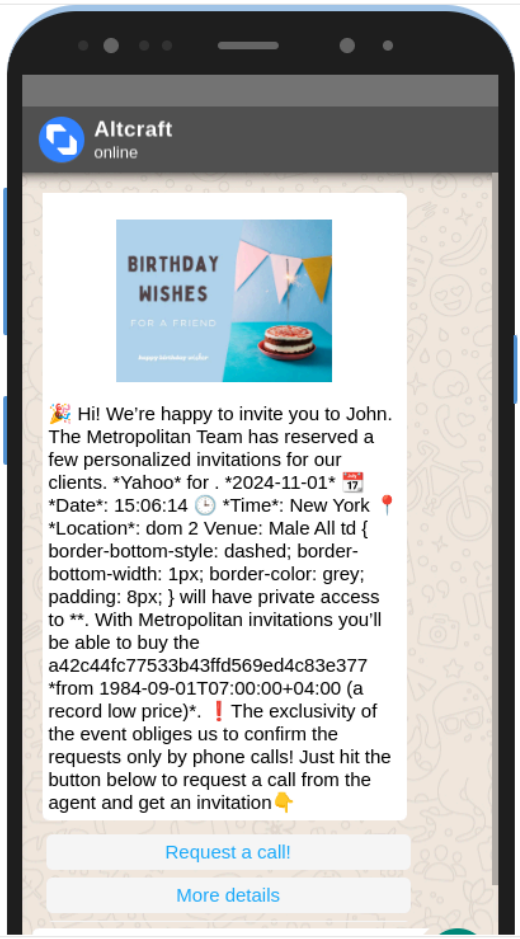
If you want to check how the message will look, go to the template preview mode by clicking on the button

:

Preview for New planetary template

Channels:

WhatsApp



Configure API content

Do not use resources

Test profile

Save image

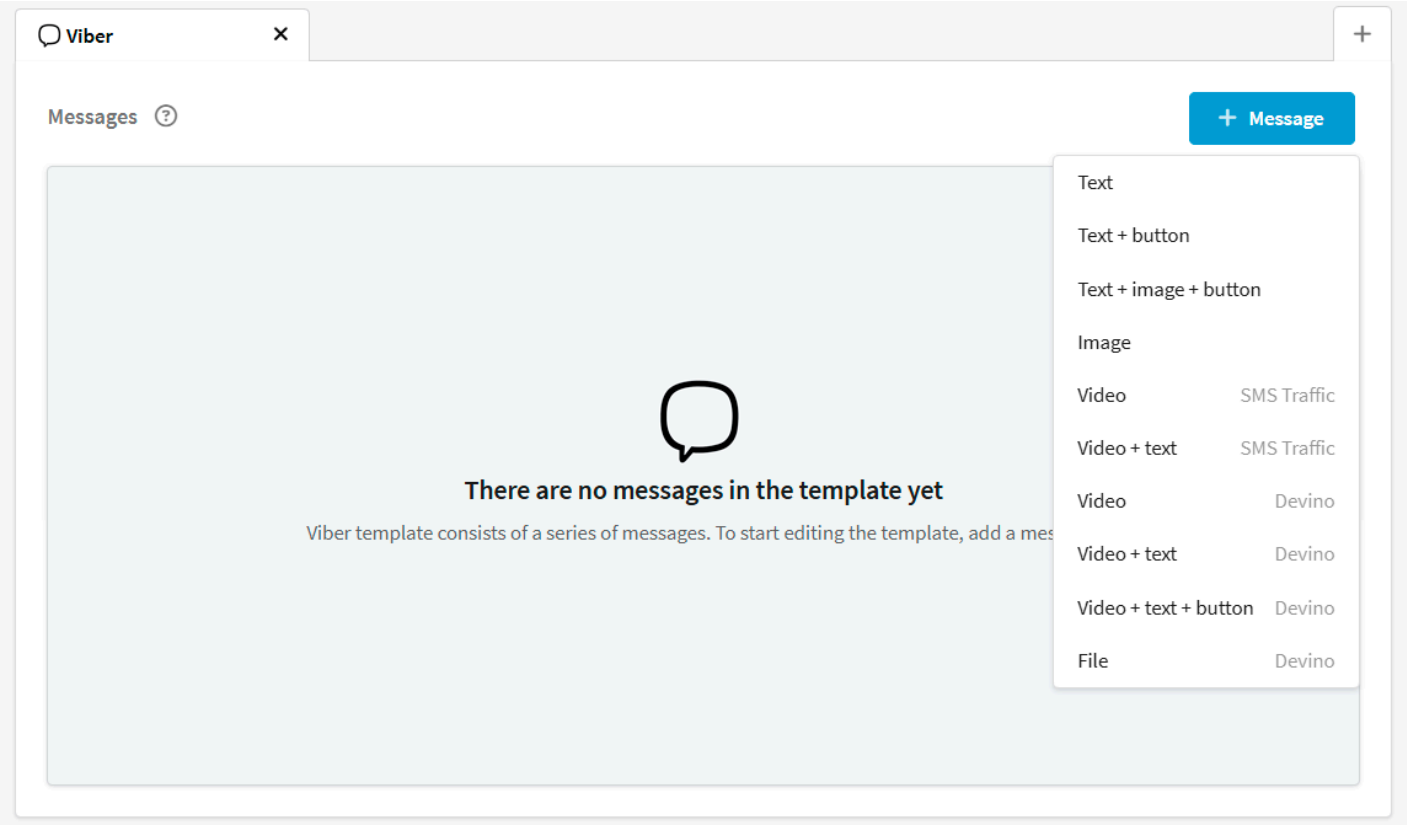
Close

Quick reply: {fragment.confirm_text}

Viber™ template

Creating a template

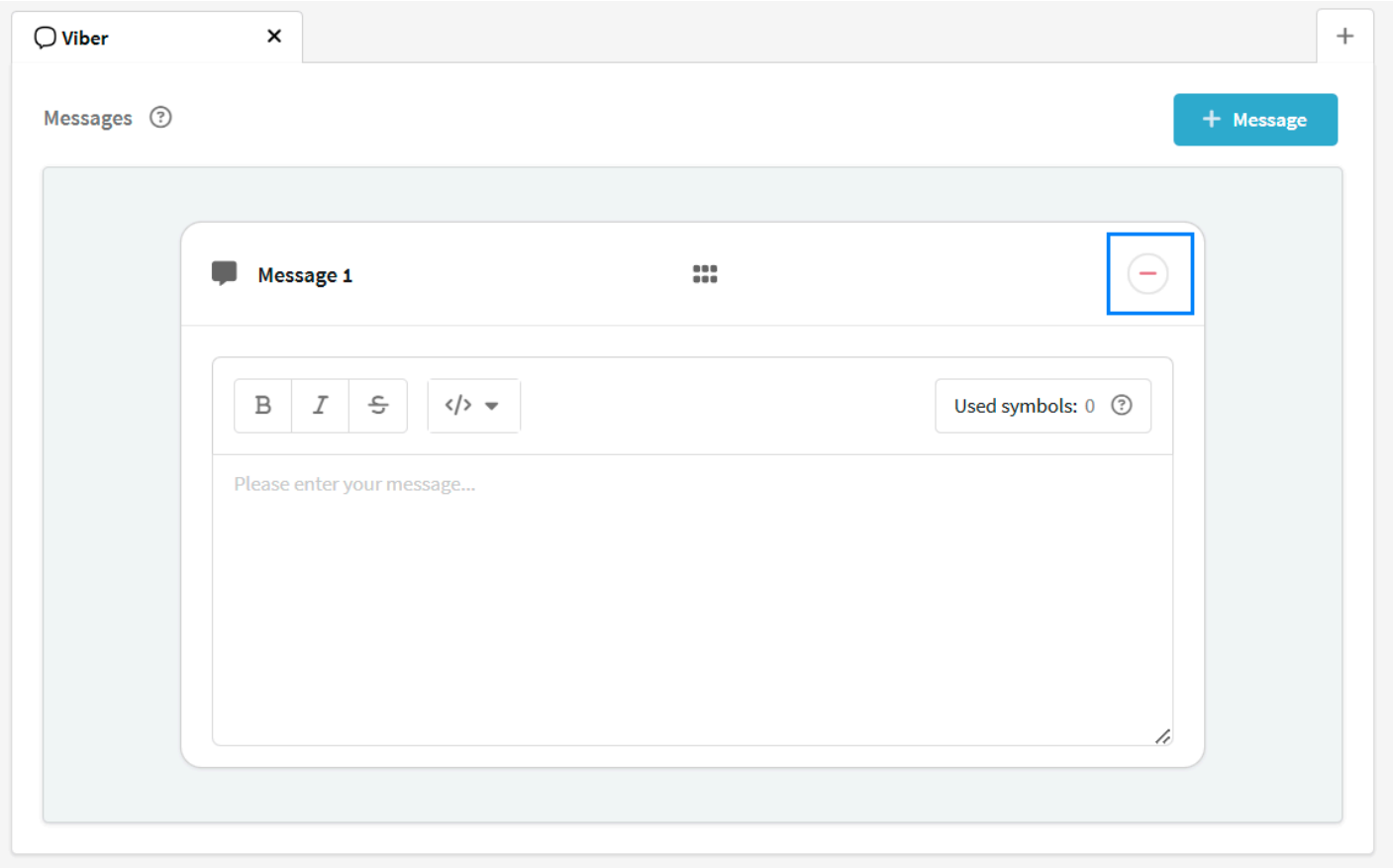
Viber*™ template consists of a series of messages. To start editing a template, add a message using the + Message button:



The maximum number of messages in a series is 5. If multiple messages are used in the template, each of them will be sent with an interval of 2 seconds.

To remove an unnecessary message, click

.



Types of messages

The following types of messages are available:

Message type	Devino	SMSTraffic	MTS OmniChannel	Rapporto
Text	+	+	+	+
Text + button	+	+	+	+
Text + image + button	+	+	+	+
Image	+	+	+	+
Video	+	+	-	-
Video + text	+	+	-	-
Video + text + button	+	-	-	-
File	+	-	-	-

Text

Enter the text of your message in the editor using the Markdown format. For ease of formatting, the following functions are available on the top panel:

B

— bold text

I

— *italics*

~~S~~

— ~~strikethrough~~ text

</>

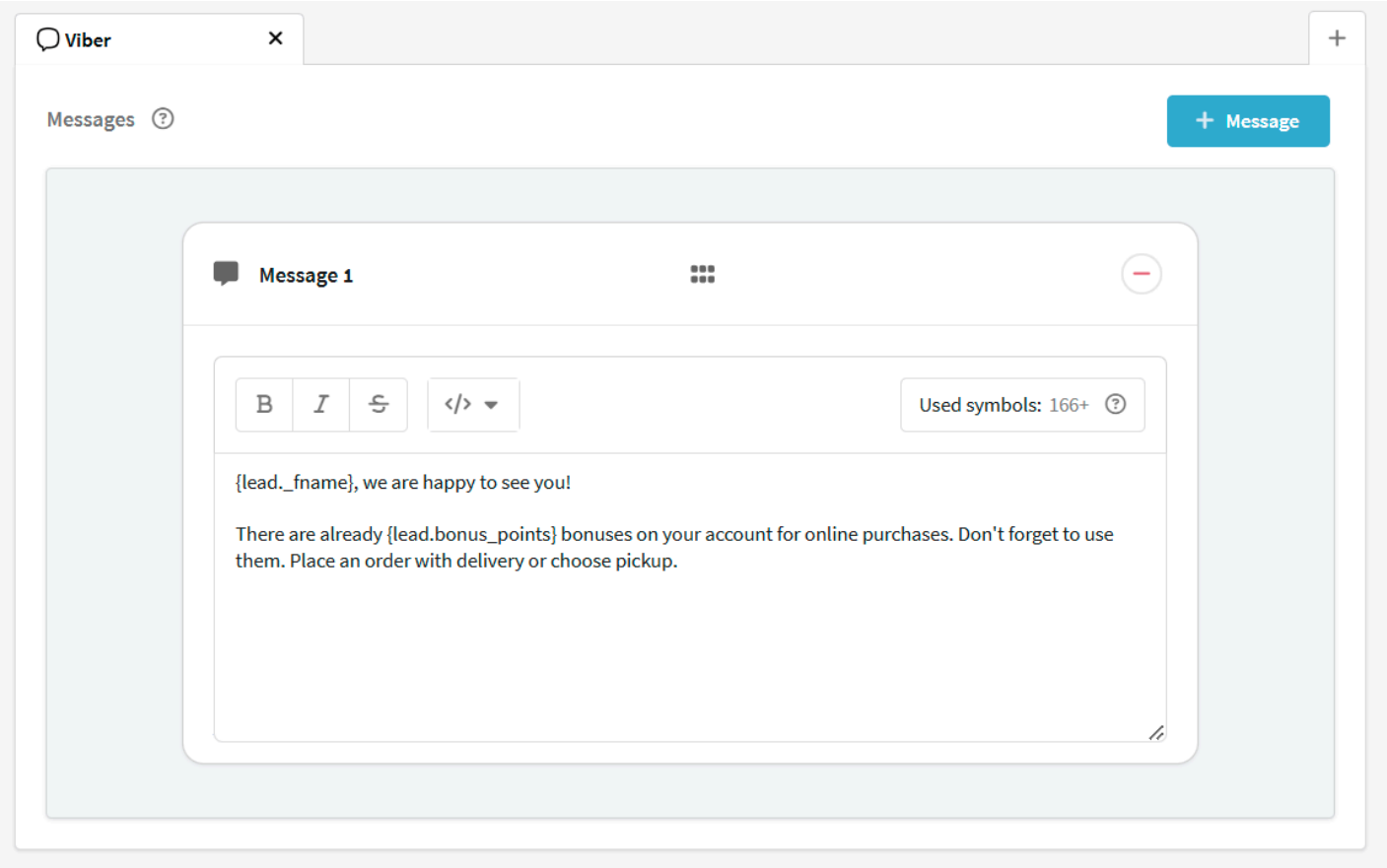
— list of editor variables and functions to personalize the message.

CAUTION

Please note that Markdown markup is displayed in preview mode (for example, `"text"`), but the recipient of the message will see the correct formatting (“text”).

Also, you cannot add links in Markdown format to the “Text” block.

Links inserted through the platform are automatically shortened. The final link path (excluding tracking domain) length is 12-14 characters, including the `/`.



TIP

You can add emojis to Viber*™ messages. To insert an emoji, you need to use its code. For example, to add 😊, type `(smiley)`. Codes for all available emojis can be found at this link.

Please note that when calculating the number of characters in a message, each character in the emoji code is taken into account.

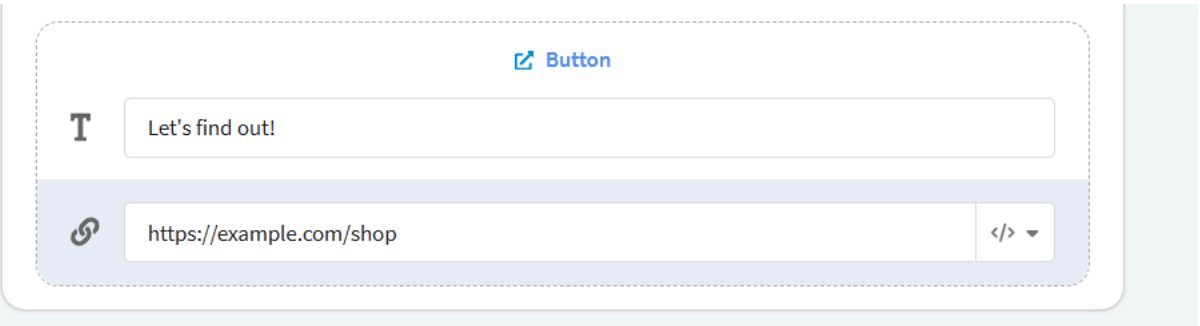
Emojis are not displayed in preview mode!

INFO

There is a character counter on the right. The maximum length of a single message is 1000 characters. Please note that data that is pulled from dynamic variables may exceed this limit. To ensure reliable delivery, we recommend that you monitor the length of the message.

Button

Specify the text of the button that will be displayed on the button and the link to click through. You can add variables to the URL.

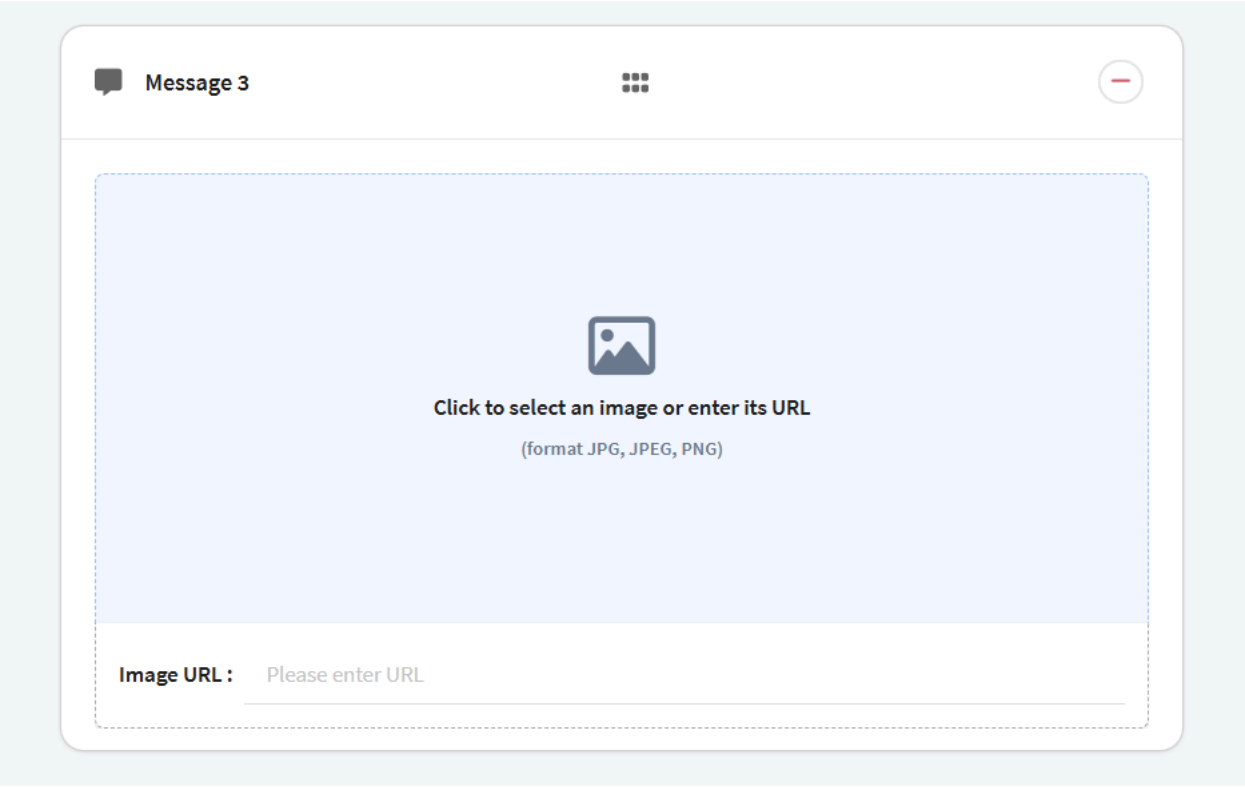


Image

Supported formats are jpg, jpeg, png. The maximum size is 5 MB.

There are two ways to add images:

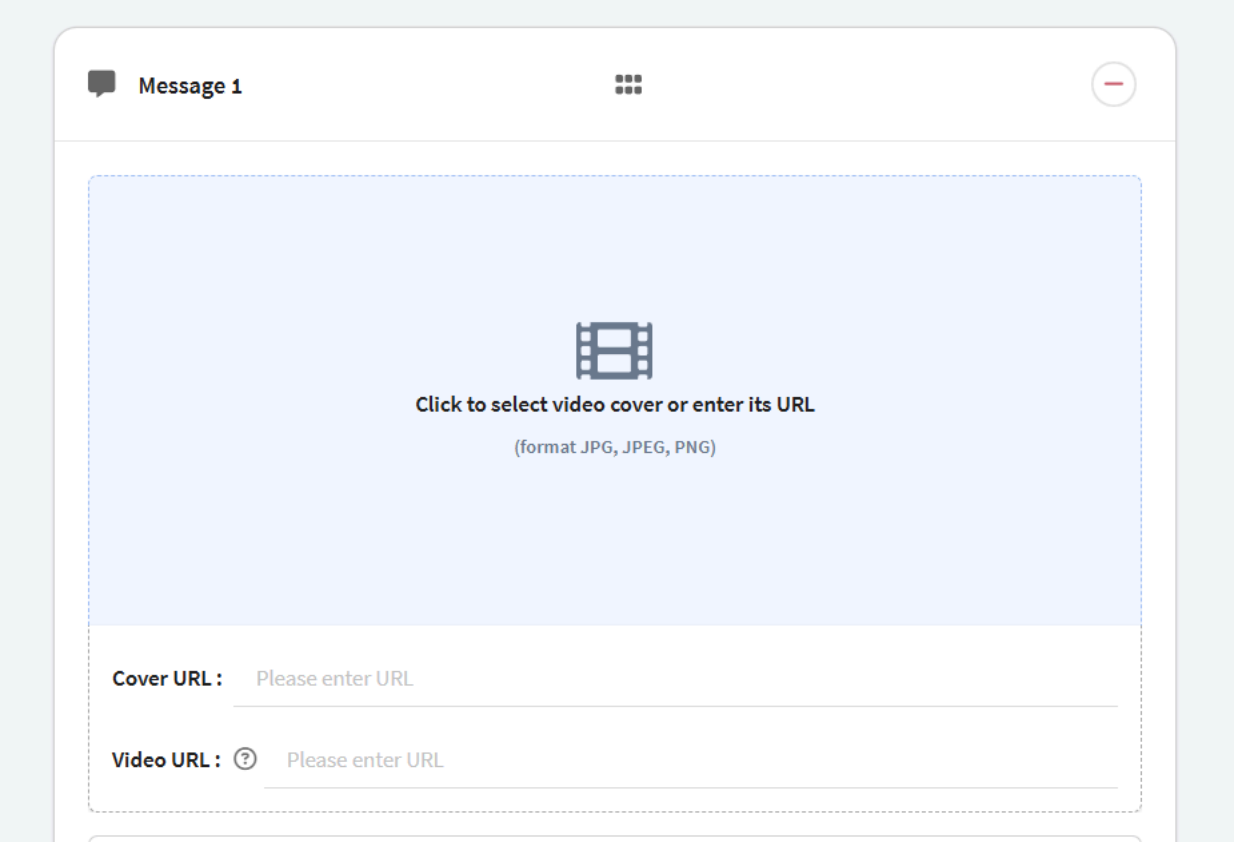
- **From the platform gallery. To open the gallery, click on the download area. Select the desired image or upload a new one from your device.**
- **From an external source. Specify the link to the image.**



Video

Supported formats are mp4, 3gpp. The codec is H.264. The maximum size is 5 MB.

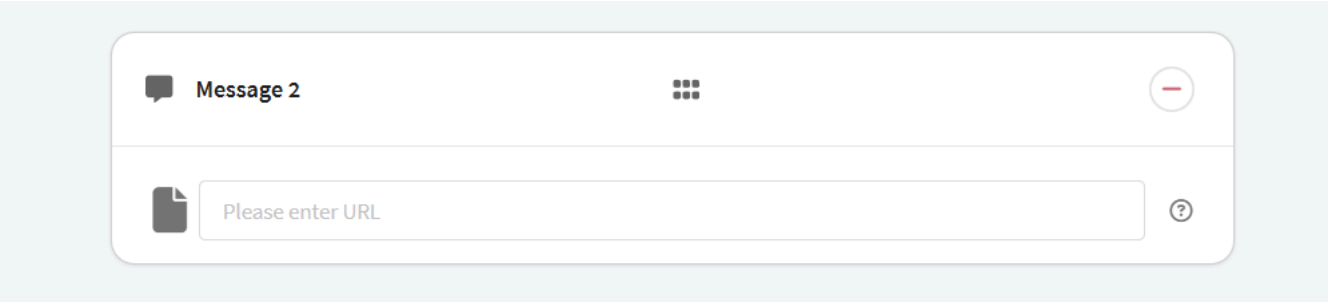
To add a video, specify its URL. You also need to install a cover for the video. You can select an image from the platform's gallery or insert a link to the image.



File

Supported formats are doc, docs, rtf, odt, odf, txt, pdf, eps, xls, xlsx, ods, csv. The maximum file size is 200 MB. The limit on the number of characters in the file name is 25 (including the extension).

To send the file, specify the link to it.



**** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

Notify template

INFO

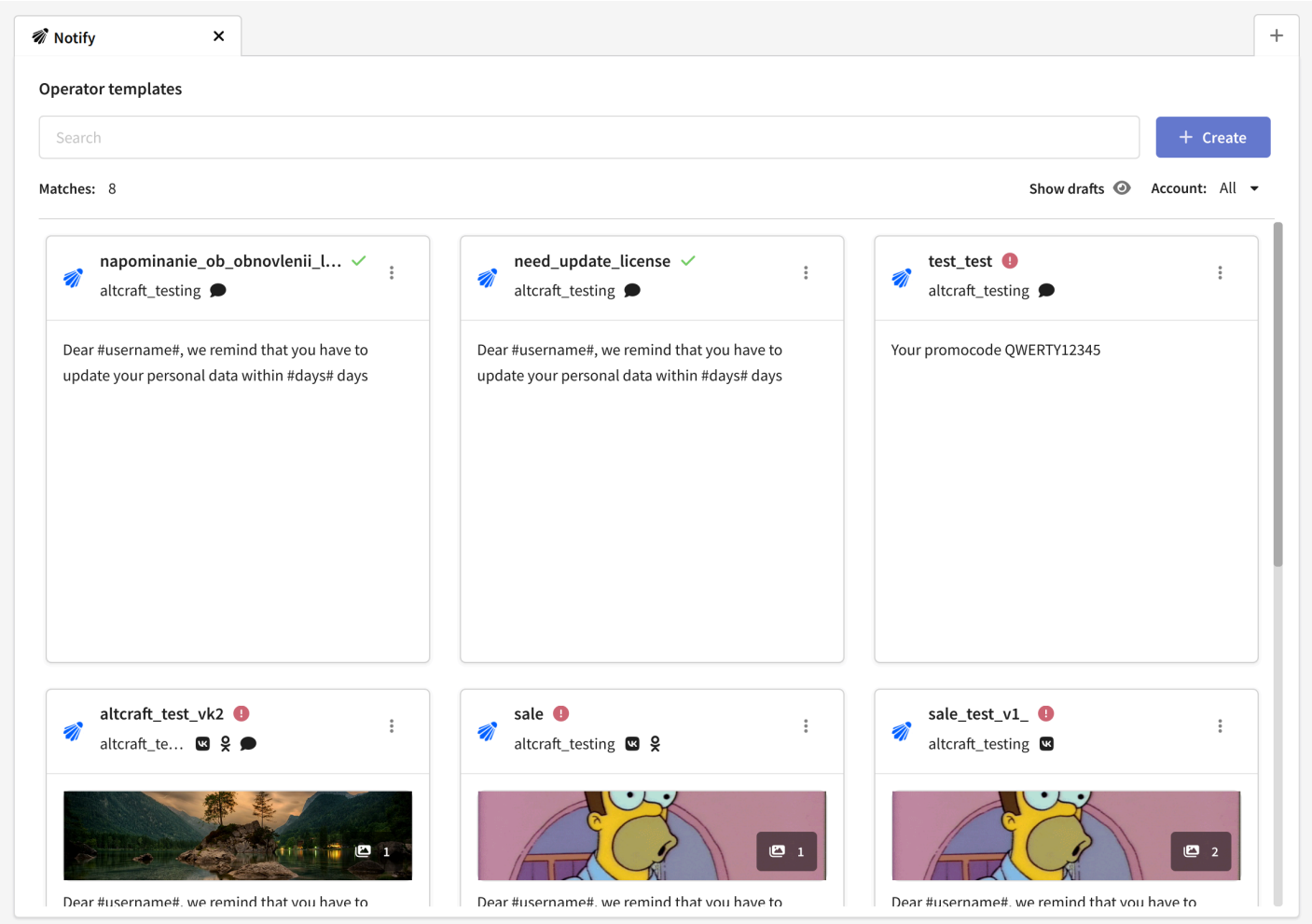
According to the service policy, new access to the Notify service personal account is currently not provided.

Support for current users is provided by [MAX Business] tech support.

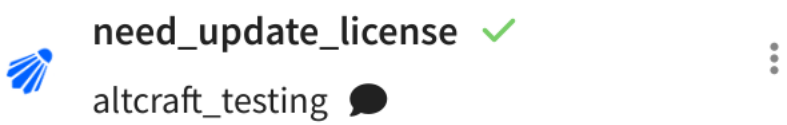
Using the Notify channel you can send business-messages via VK, OK and also send SMS notifications. Social media messages and SMS messages are customized in two different sends that follow each other.

Existing operator templates

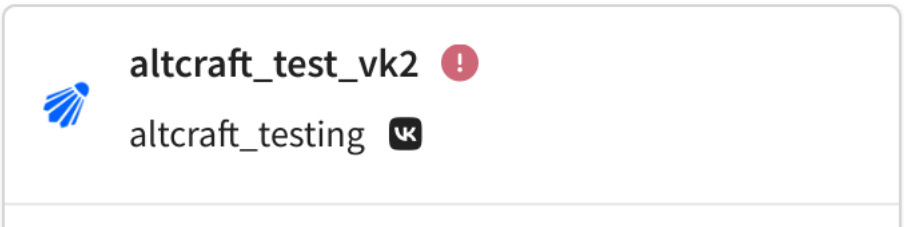
When you open the template editor for Notify, you will be shown the operator templates created in your personal Notify account:



All templates for sending are moderated by the service. Templates approved by the service are marked with a check sign:



Templates that have been archived or have not been moderated will carry a warning. They cannot be used until you edit them and they are approved:

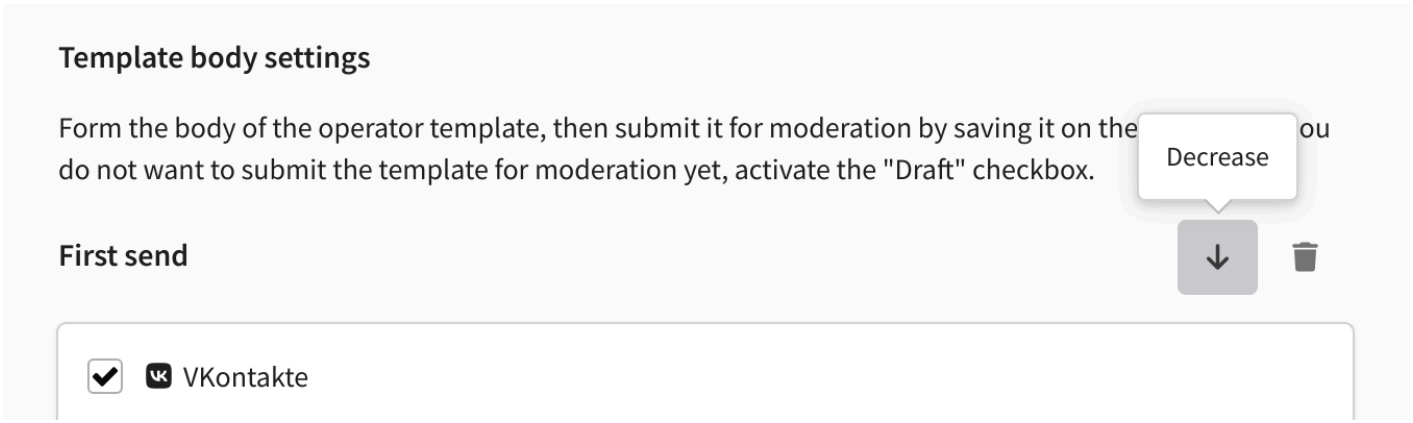


Creating an operator template

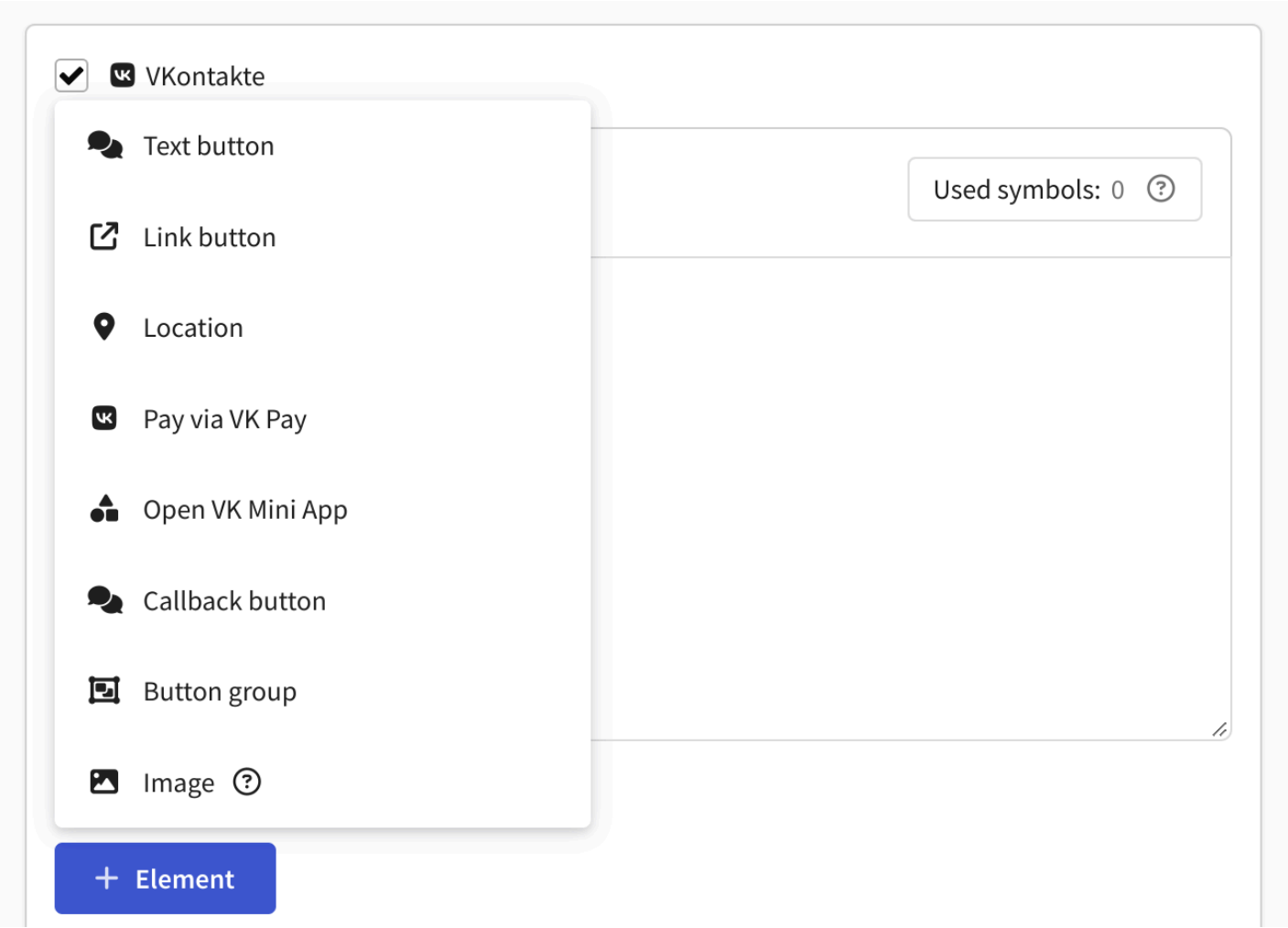
You can also create a new Notify template while in the platform. To create a template, click the corresponding button. The window of template creation will open, in this window you should specify name of the template and service configured in your personal cabinet:

There is an option in the template to set time to wait for the message to be delivered and read. The delivery wait time is the time during which Notify will attempt to resend the message if any error occurs. When this time expires, a second send will be triggered. Read event waiting time is the amount of time Notify will wait for the message to be read on VK or OK. When this time expires, a send with an SMS message will be triggered.

The Notify template consists of sends. One send is a message to VK or OK, the other sending is an SMS to the user's phone number. You can add or delete a sending, as well as change the order of their sending by clicking the corresponding button:



The message body defines the content of your notification. An OK message can be text-only. You can attach an element to a VK message:



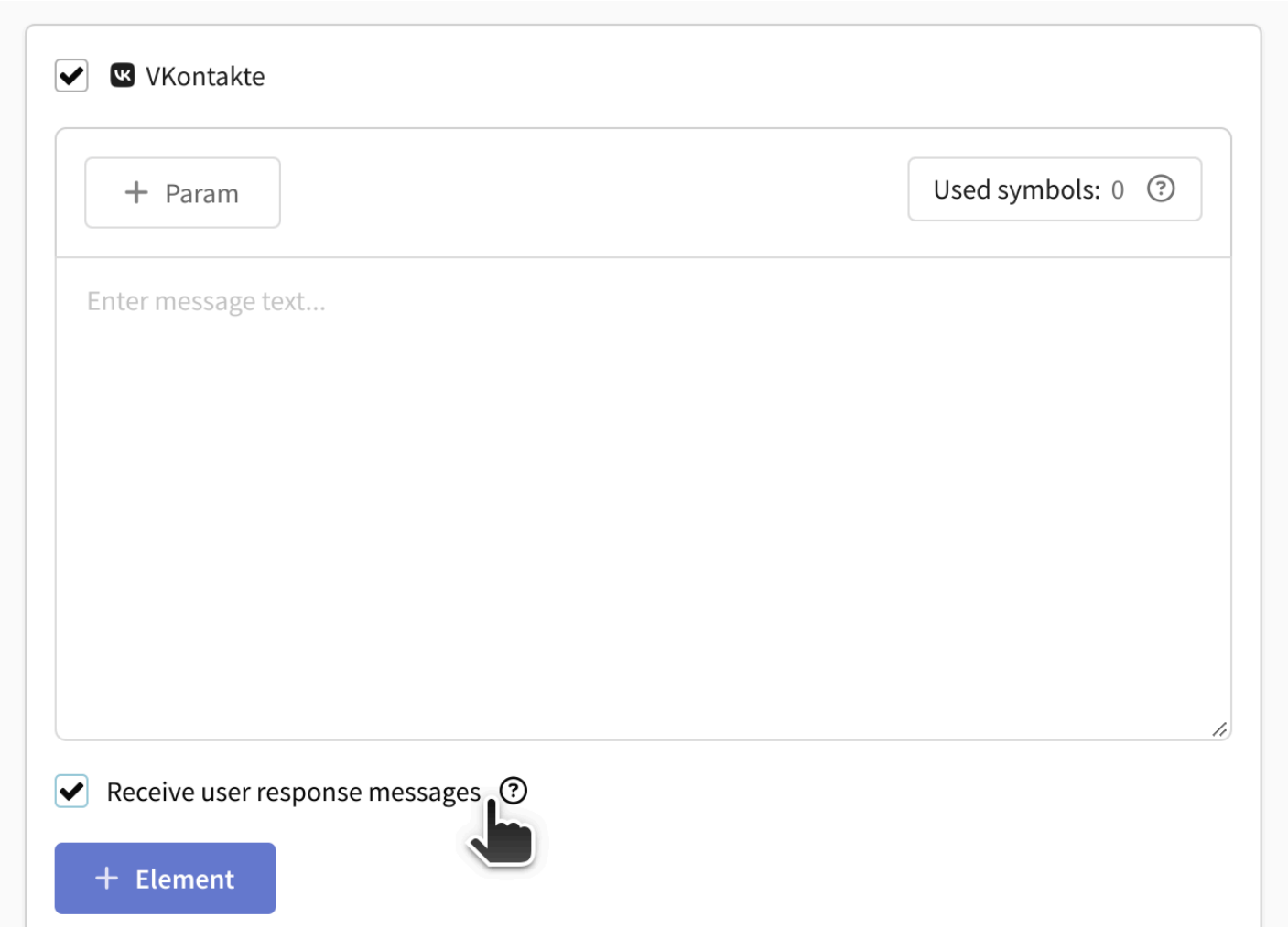
- **Text Button - sends a message with text. For a text button you can choose a color, as well as attach additional information in JSON format.**
- **Link Button - opens the specified link. In the button you can specify its text, a link to go to, as well as additional JSON information.**
- **Location - when clicked, opens a dialog box with location information. The user can select location in this dialog box and send it to the chat. Community administrators will receive the map image in chat.**
- **Pay via VK Pay - opens VK Pay payment window with predefined parameters. For the button it is necessary to specify a hash - a string containing payment parameters and application ID.**
- **Open VK Mini App - opens the specified VK Mini Apps application. To customize the button you need to specify the application ID, Owner ID, as well as a hash for navigation in the application.**
- **Callback Button - allows you to receive a notification of a button press and perform the required action without sending a message from the user. You can specify a caption and color for the button.**
- **Button group - cobmination of the above buttons.**
- **Image - up to 5 images with total weight not more than 50 MB. Supported formats - JPG, PNG, GIF.**

TIP

Read more about elements in Notify documentation.

You can insert parameters into the message body. Later you can substitute field values from the platform's databases into them.

You can also allow the client to send reply messages in the chat with the company. To do this, activate the Receive user response messages box in the settings:



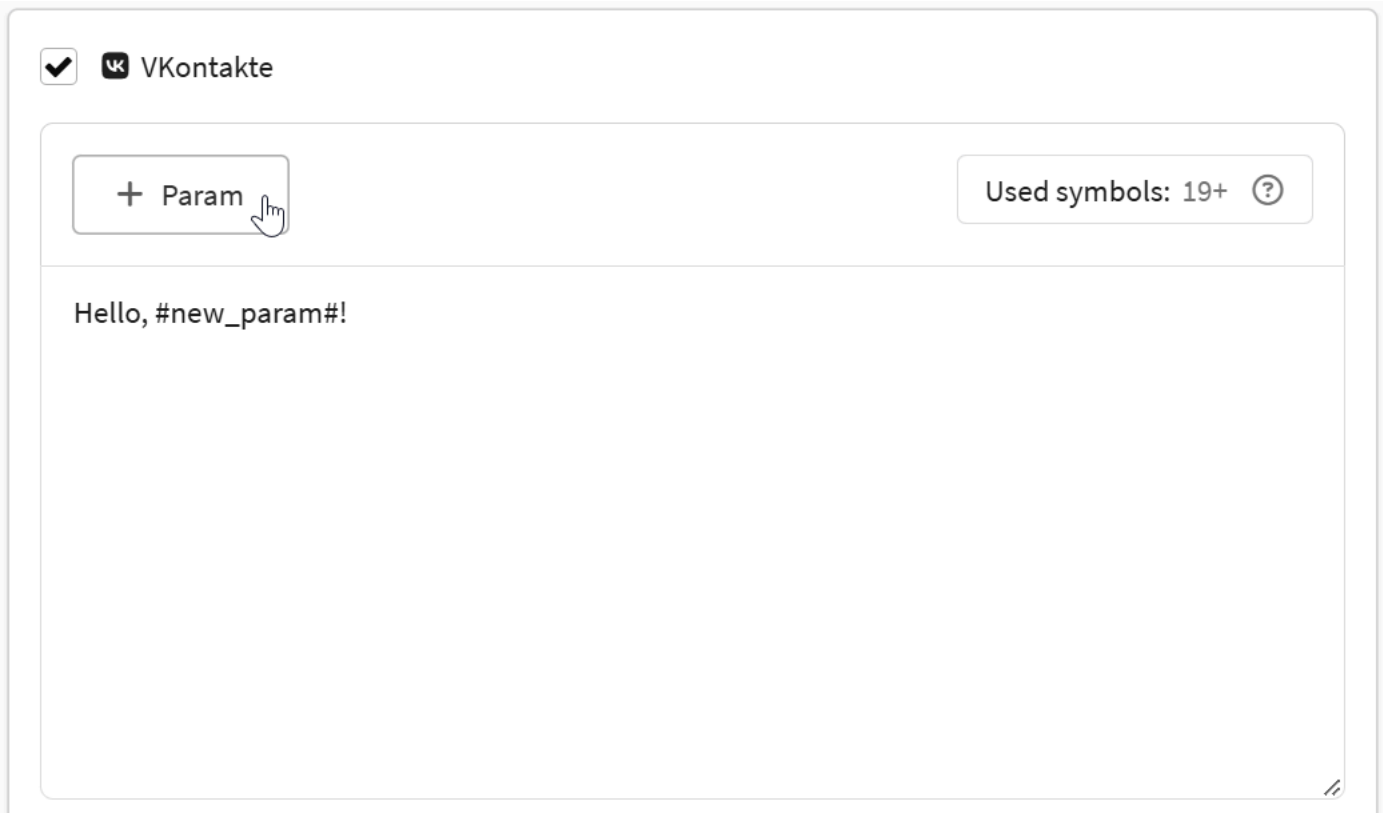
There are several types of SMS messages - advertising, identification, service and transactional. Message type affects its charging in Notify, you can get detailed information about it from the service support.

Once you save the template, it will be sent to moderation in the service. If you don't want to send it for moderation, activate the Draft switch.

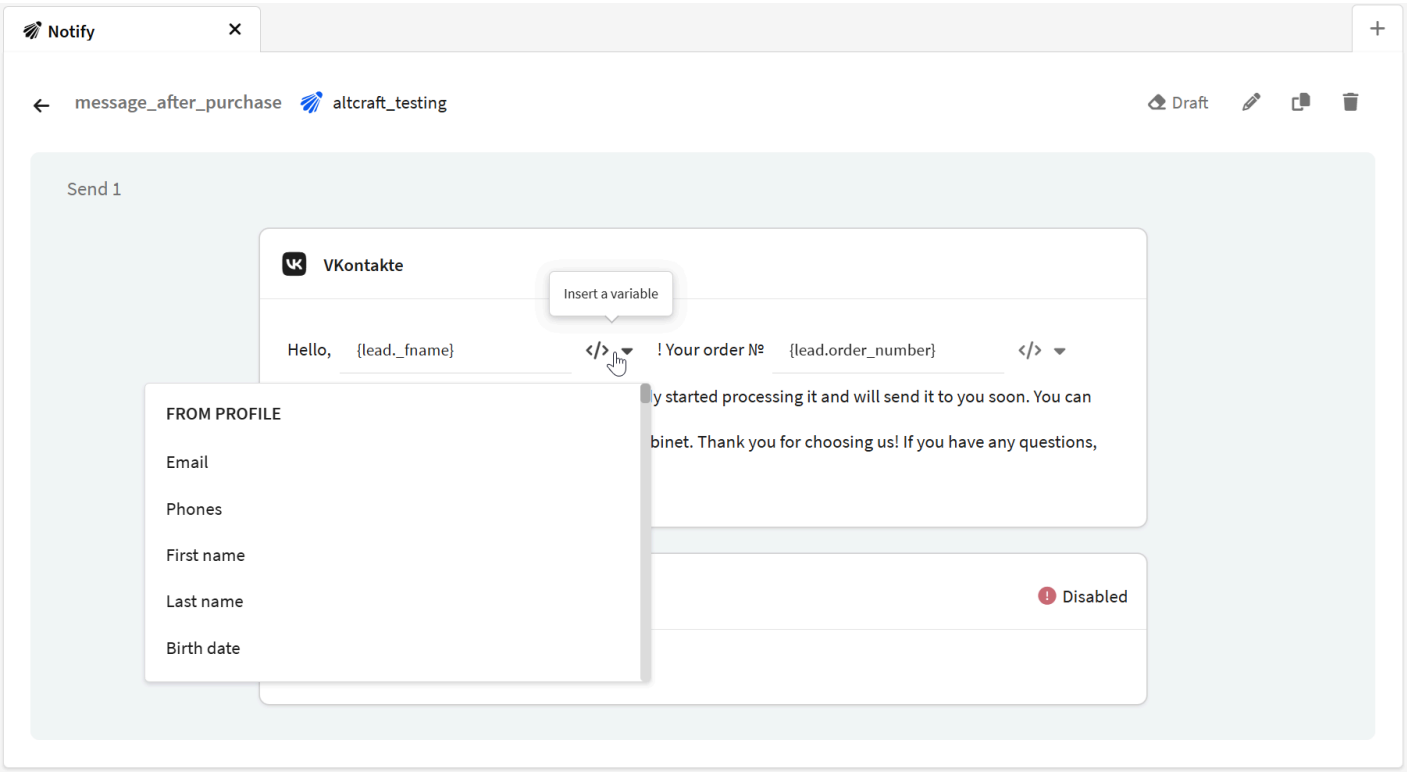
Managing parameters

The text block in Notify consists of fixed text and parameters. Parameters are template text that changes for each new message (name, address, event name, date and time, link, etc.).

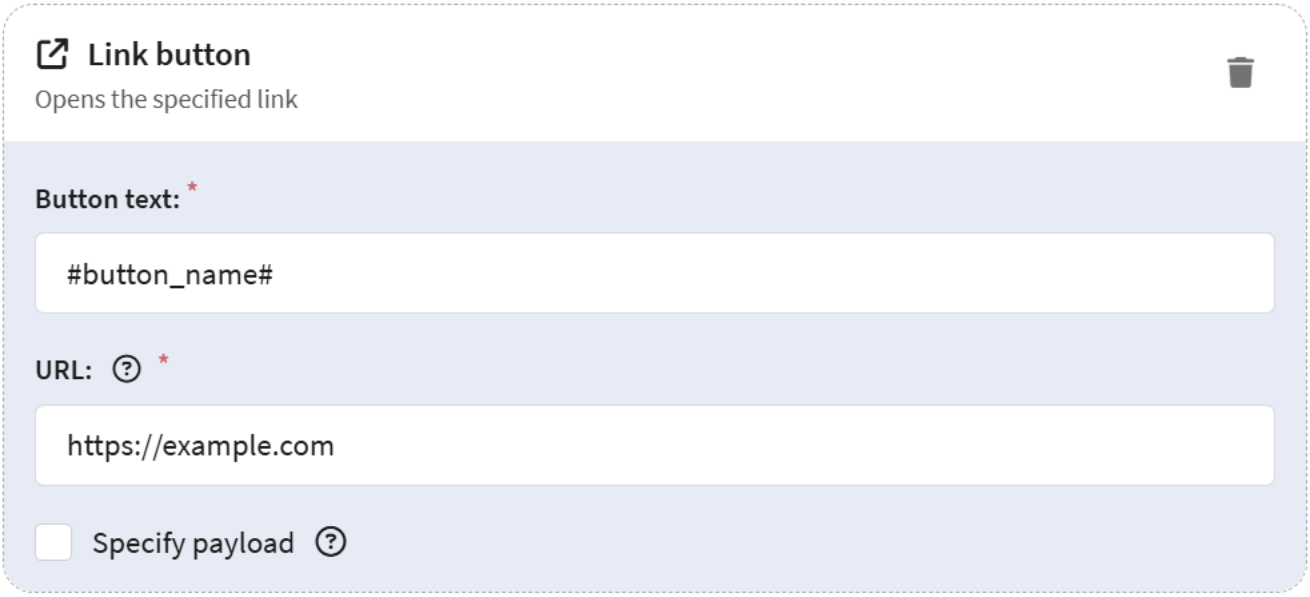
A parameter can be added to the text block in two ways: manually by typing `#parameter_name#`, where `parameter_name` is an arbitrary parameter name, or by using the +Parameter button, which will automatically insert it into the text:



You can manually prescribe parameter values or add Altcraft variables. To open the list of variables, press `</>`. When you send a message, the entered text will be automatically substituted into the template.

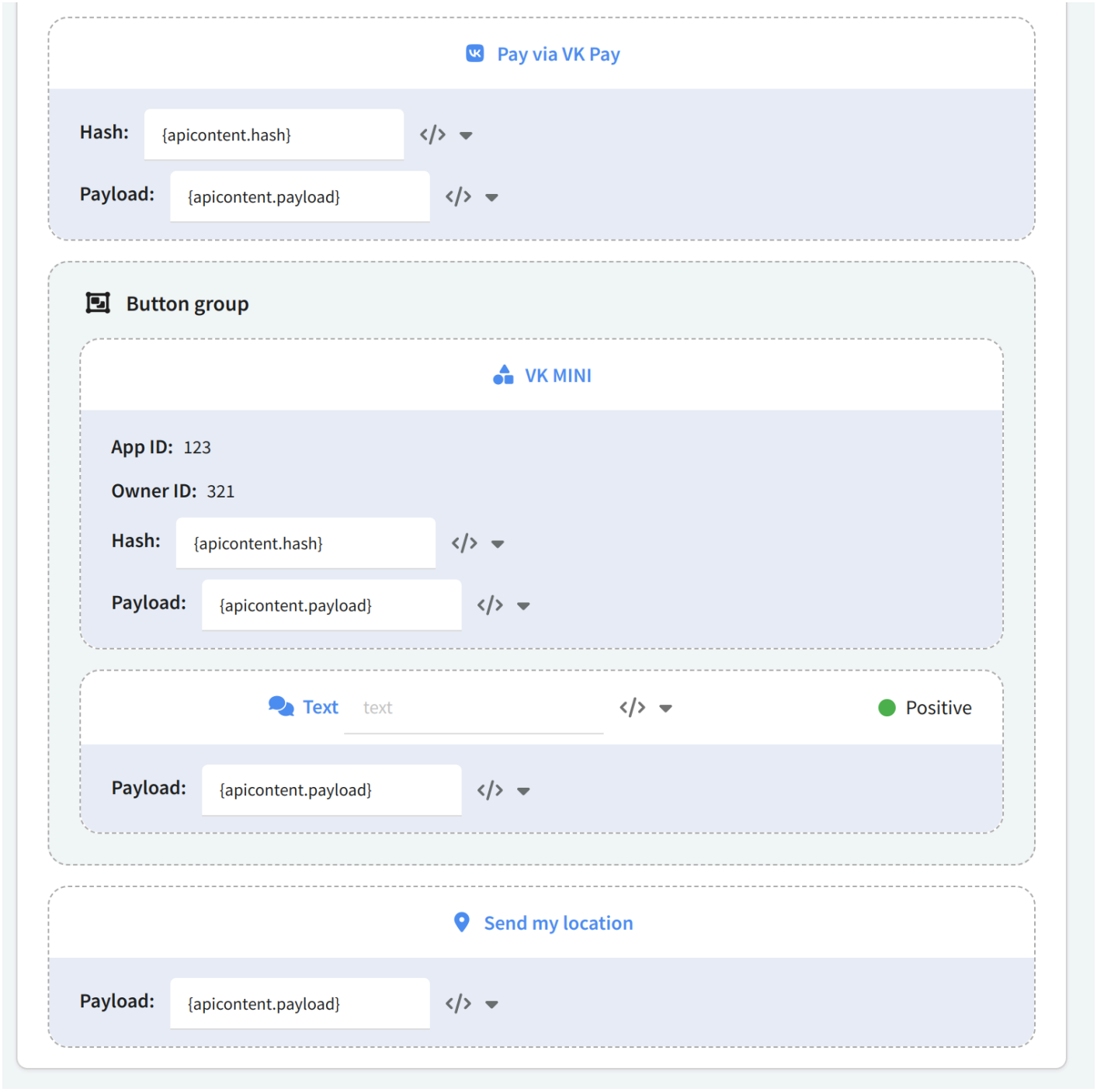


Parameters can be used not only in the Notify text block, but also in elements such as buttons:



Template Preview

After selecting an approved template, you can specify the required values or variables from the platform for the parameters:



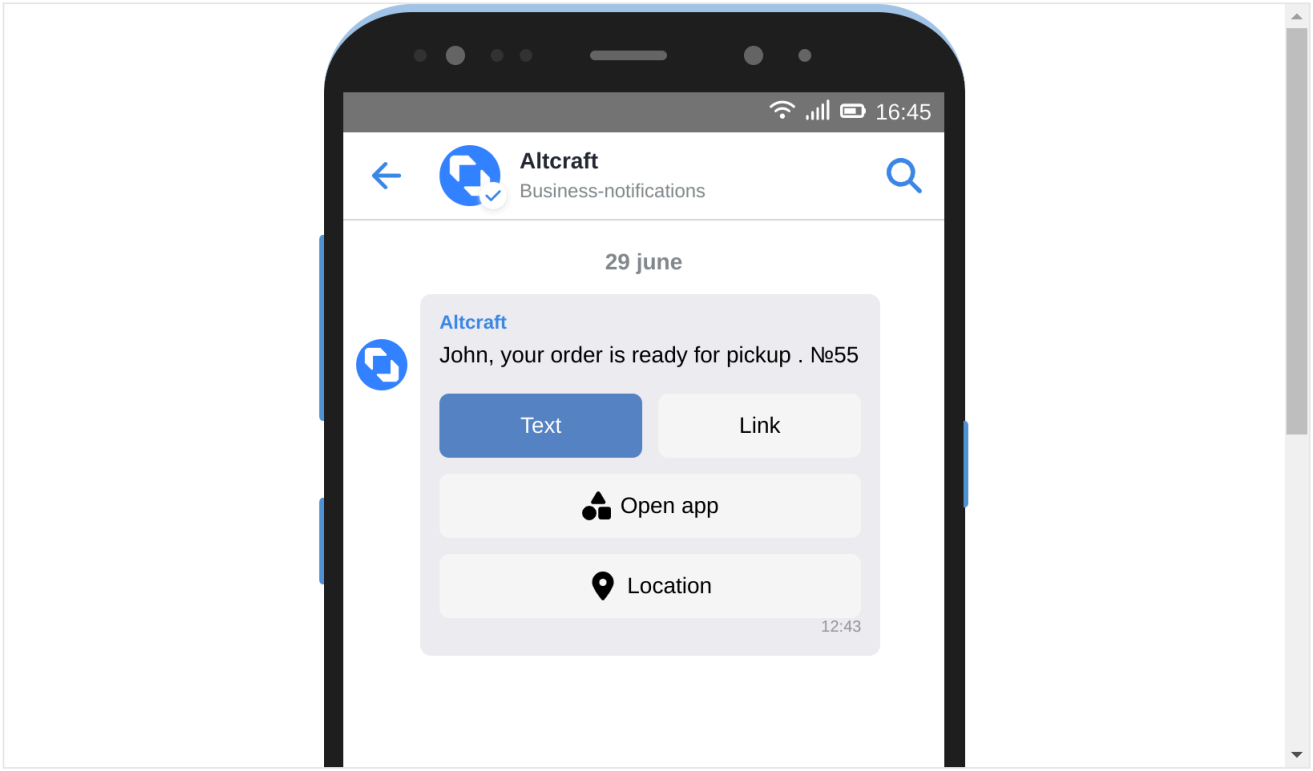
To see how the user will see the received message, click the preview button at the top of the screen:

Preview for Notify

Channels:

Notify

VKontakte



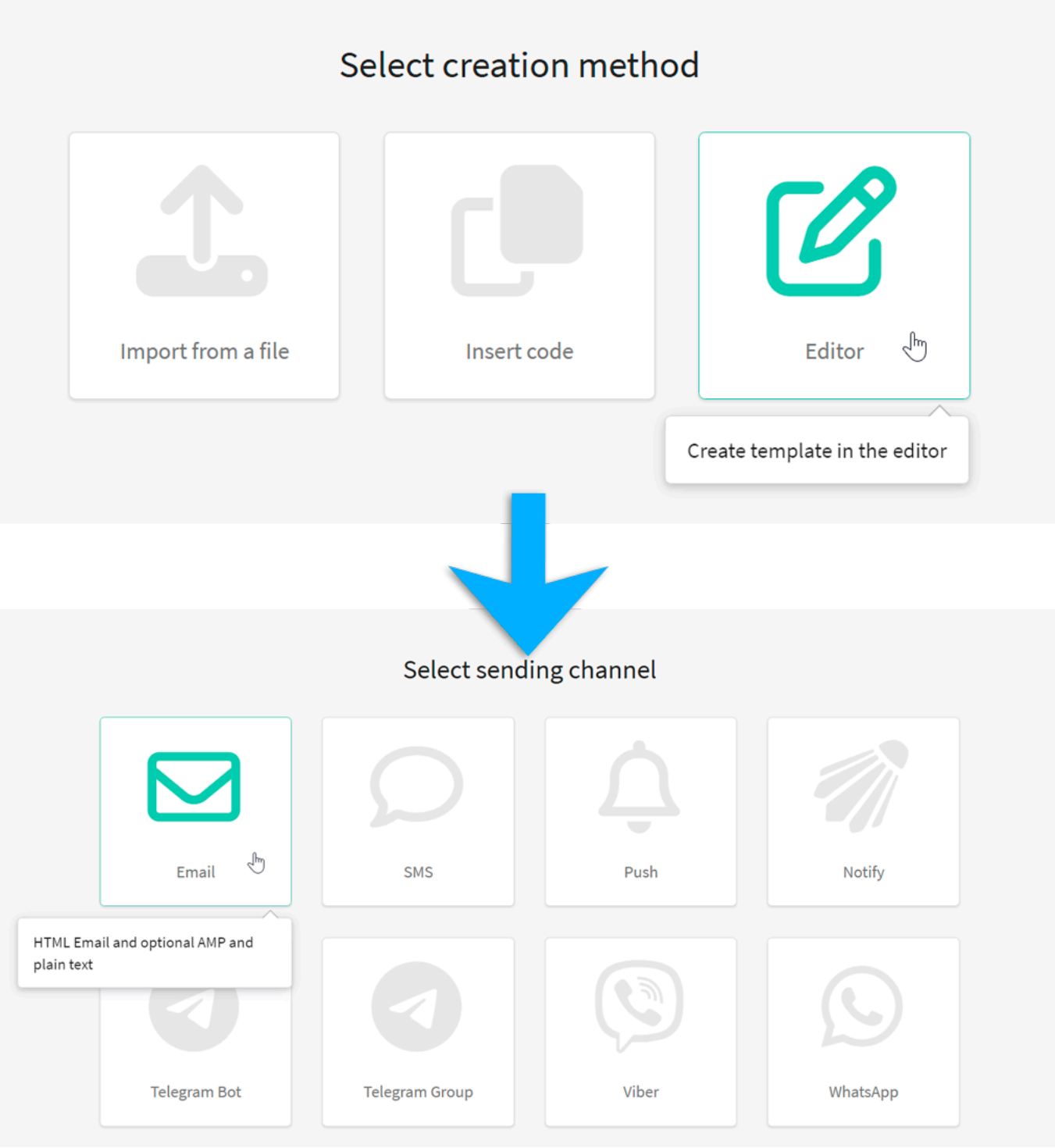
Don’t forget to save the template in the platform after making all the changes. You can now use this template to send out Notify feeds.

Visual editor for email-template

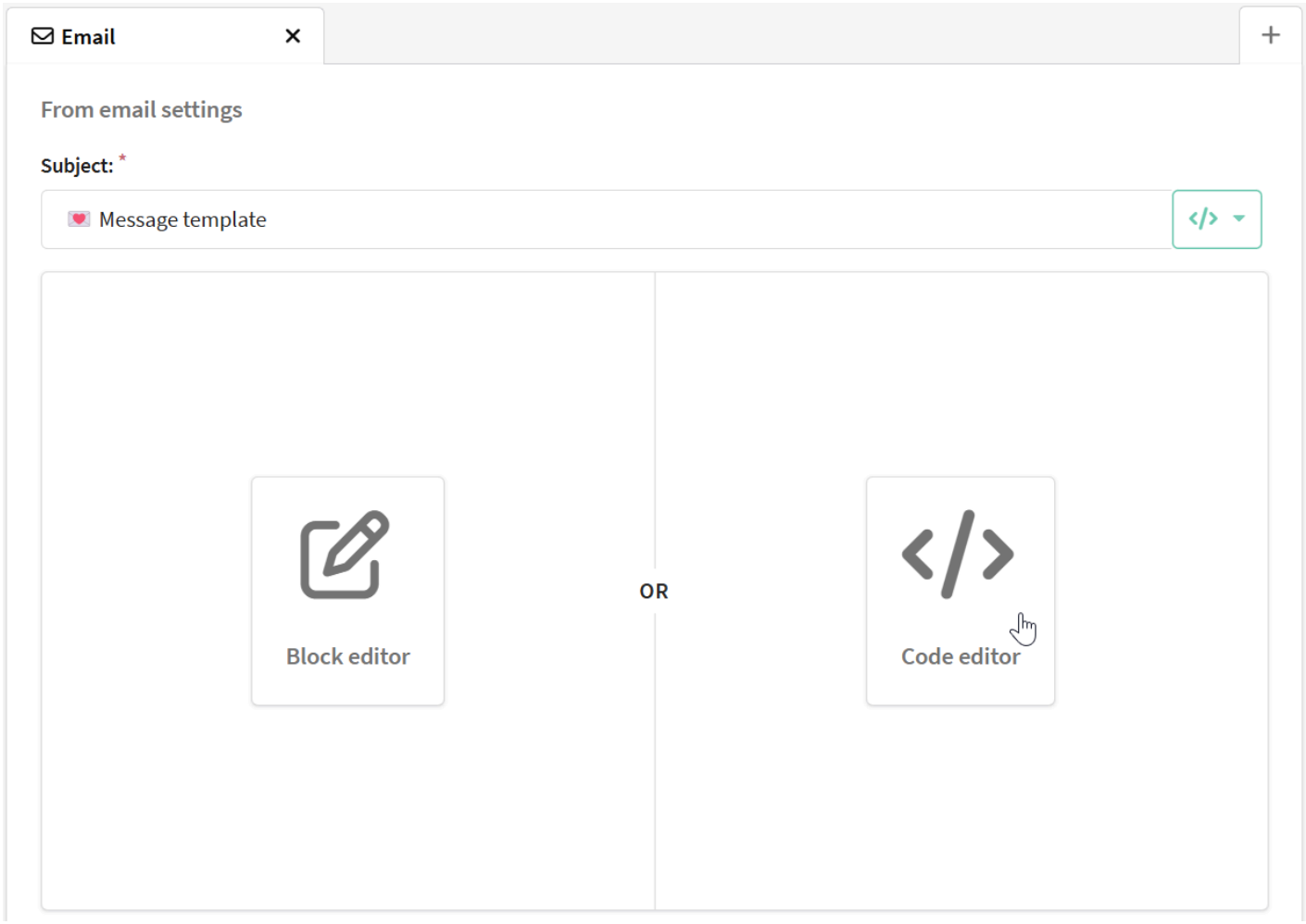
Visual editor for email-template is a tool for creating, editing and customizing marketing email templates that greatly simplifies the process of designing attractive and functional emails. Thanks to its user-friendly interface, users can easily customize the design of templates to their needs, even without programming knowledge. Among the key advantages of the editor are the ability to preview instantly, adaptive design for different devices and a wide range of ready-made elements that can be easily combined.

How to get started in the editor?

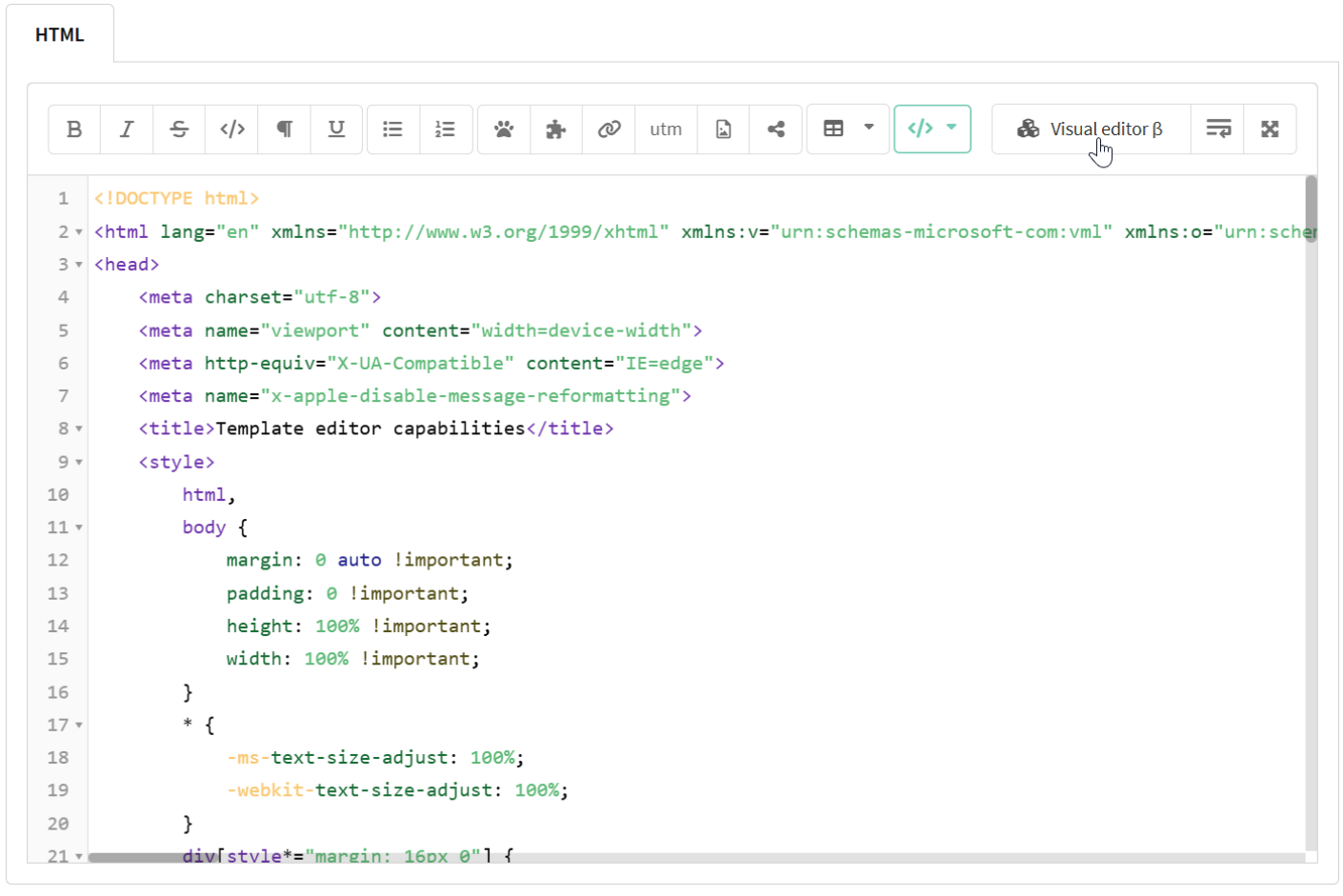
In the main menu, open Templates and create a new message template. Select Editor —> Email.



You will see an email template customization window, select Code Editor.



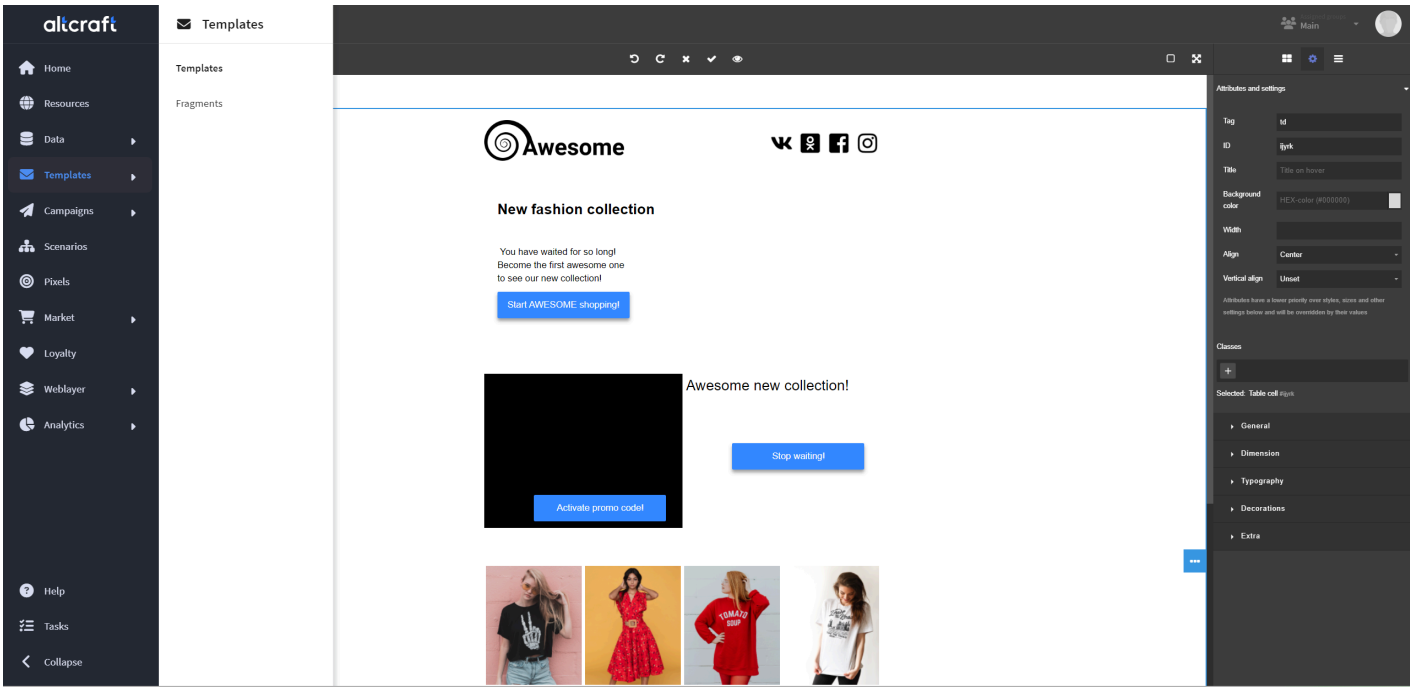
You will find yourself in the HTML code editing window, in it click on the Visual Editor button.



Read more about working with the visual template editor in the following articles:

Visual editor interface

The editor divides the page into two parts. In the central part, you see your template. Here you can select, move and edit blocks. On the right side, there is a panel where you can work with template elements: select the required elements and change their parameters.



Above the central part of the editor is the toolbar.

Using buttons in its left part, you can change the preview mode (desktop, tablet, mobile) to check your template responsiveness without leaving the editor:

Buttons in the central part of the toolbar can be used to:

- **undo or redo the last action;**
- **discard or save changes in the template;**
- **open the message preview (you can select a test profile to know how exactly this profile will see your template).**

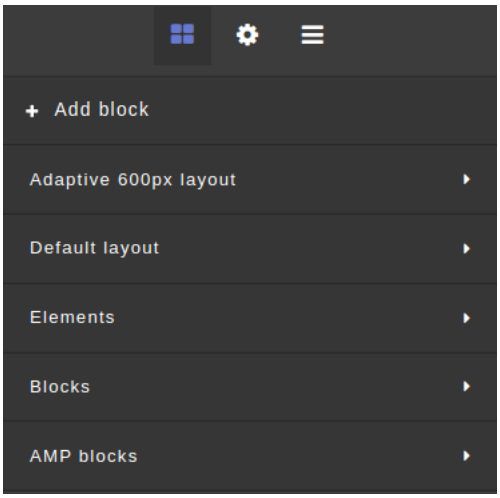
The buttons in the right part of the toolbar:

— view components: use it to see element borders;

— full screen.

The right panel of the editor contains three tabs:

1. **Blocks**
2. **Style manager**
3. **Layer manager**



💡

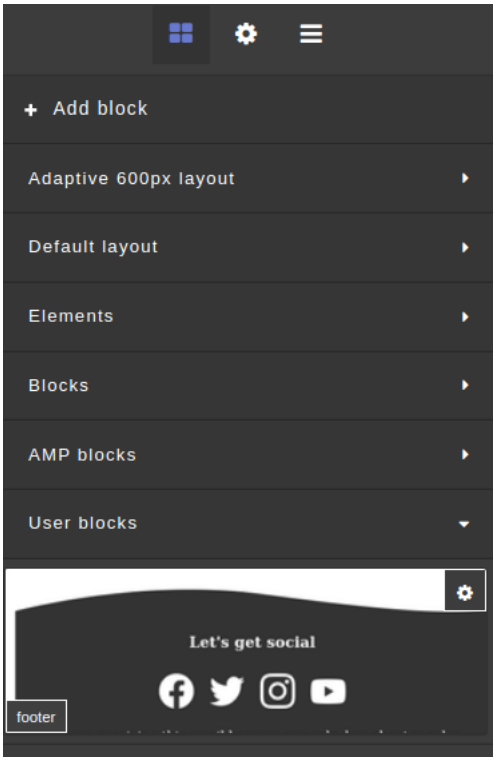
Our next step is to learn how to add elements or blocks to the template.

💡

Adding blocks

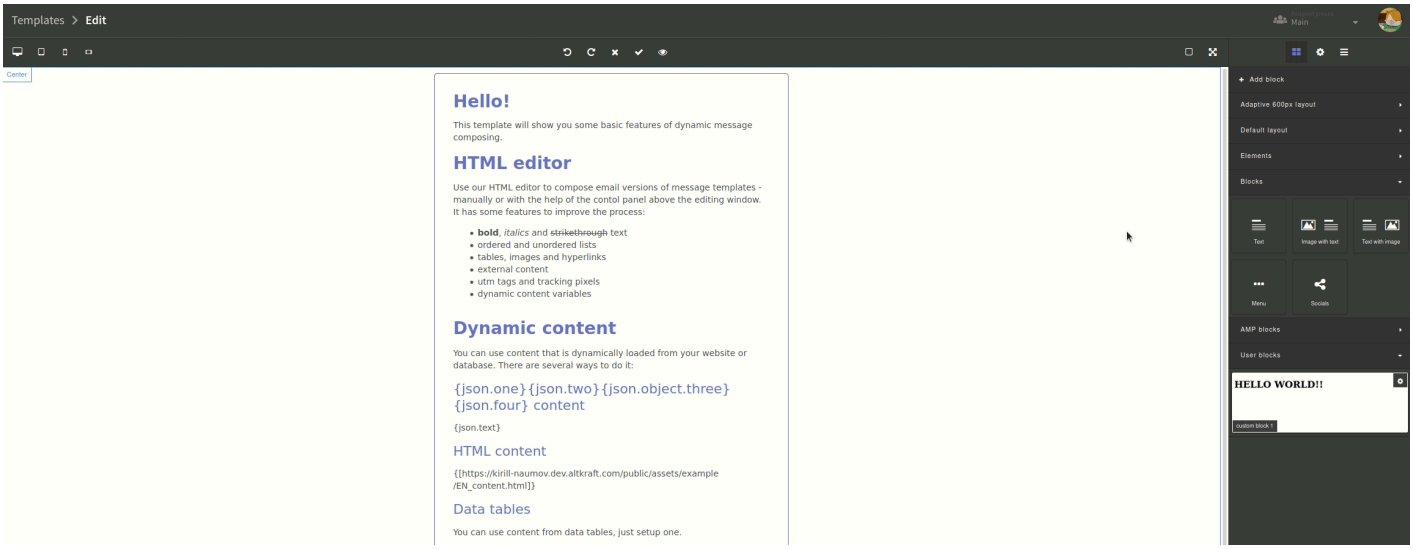
Available elements

Element	How to use
Adaptive 600px layout	**Several table rows, divided to cells with fixed maximum width. It can be used for main Email template sections with standard 600px width.**
Default layout	**Several table rows, divided to cells in percent ratio without width limit. It can be used in blocks limited by template width and sections for full screen.**
Elements	**Separate layout elements, like heading, divider, banner, link or button.**
Blocks	**Sets of elements for quickly creating template parts. These elements can be text with heading and image, menu, or social media links.**
AMP blocks	**Blocks for interactive AMP messages. “Carousel” and “Accordion” blocks are supported.**
User blocks	**You can save template elements and sections as custom blocks. They will be available in the visual editor for each template, like message fragments. To transfer message section to custom blocks, click “+ Add block” and copy the fragment content to the pop-up window.**



How to add blocks to templates

To add block to the template, just drag and drop it to the place you want. The parental block will be highlighted in yellow and the new element will be highlighted in green:



How to manage elements

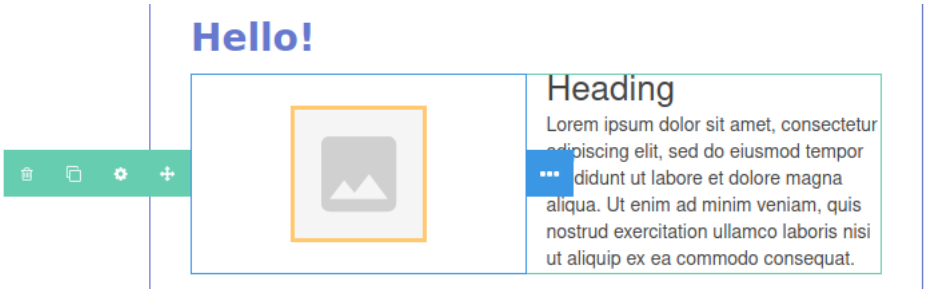
The content of the message in the visual editor consists of elements (text, headers, images, video) and the HTML layout (table cells in which elements are placed, as in containers).

General operations are divided in 2 menus:

- **Container operations — can be displayed by clicking on the green icon at the left part of layout container**

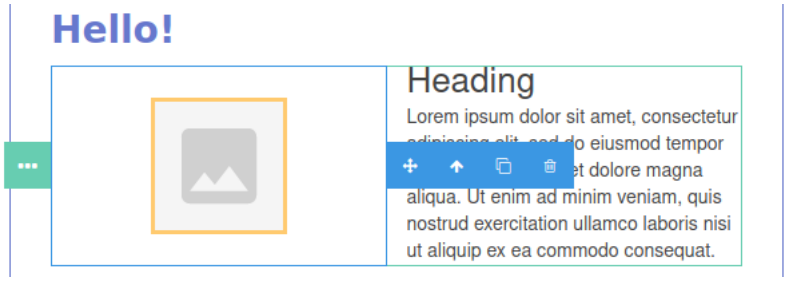
- **Elements operations menu — can be displayed by clicking on the blue icon at the left part of layout container**

When you add a new table cell or a new container, operations menu will be available. After tab is selected, you can move, configure, duplicate or remove it using green buttons.

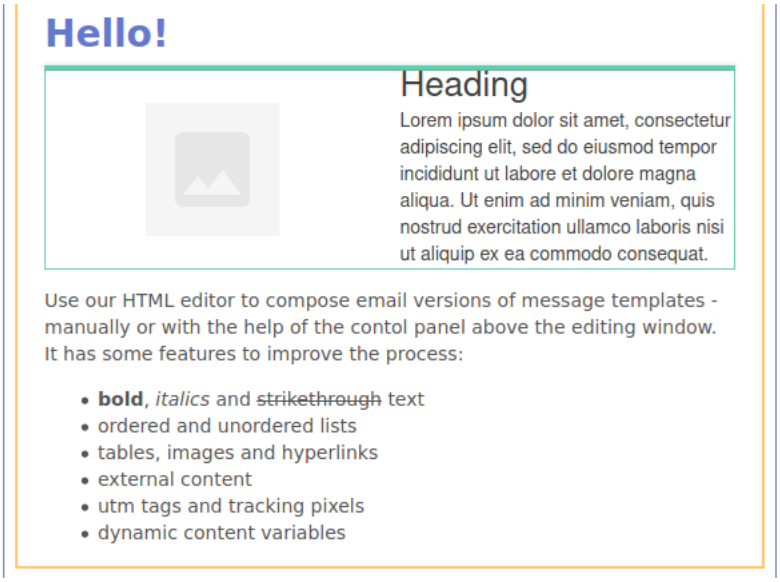


Operations menu is also available for each element inside the container. You can:

- **move an element inside a container,**
- **move it to the upper level container,**
- **duplicate,**
- **remove it, using blue buttons.**



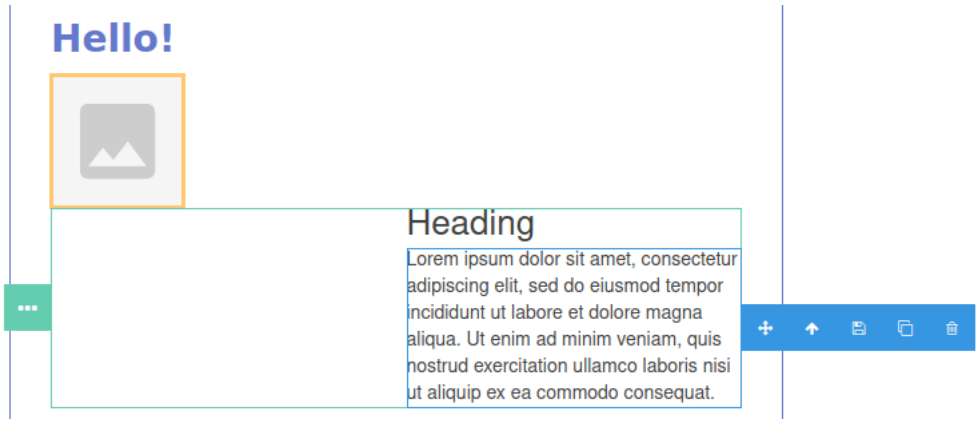
When you are moving the element, message sections is highlighted in green, parental container and other elements of the section are highlighted yellow. Green line will mark a new place for an element or a container.



Elements and element groups can be saved as user blocks. To do this, click on diskette icon in the operations menu.

TIP

To save entire sections content, select it by pressing container operations menu icon at the left part. After this, you can use element operations.



TIP

On the right side of the editor, for each selected element, a style menu is available. Here you can:

- **Configure the attributes and settings of the element. Attributes are described for each type of element;**
- **Configure display options for an element. CSS styles are added directly to the HTML tags to improve the readability of the email for email clients. Styles are described in a separate article.**

💡

Now let's take a closer look at each element.

Elements and their settings

Adaptive layout

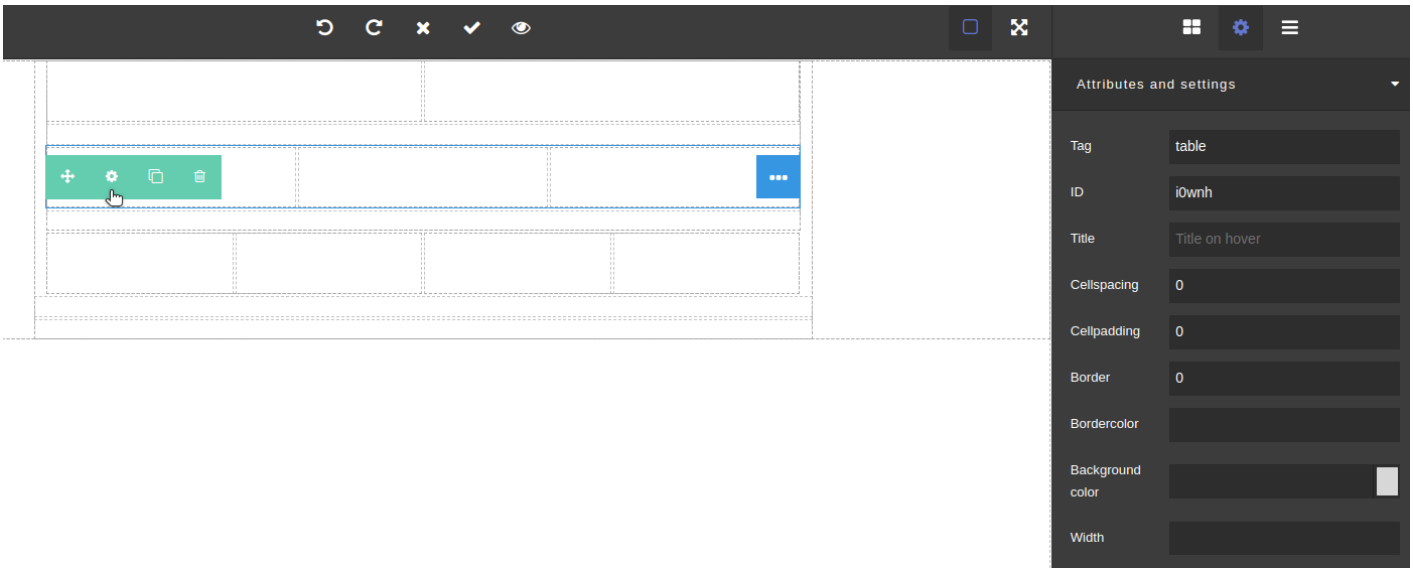
The following attributes and settings are available for adaptive layout strings:

- **HTML tag**
- **Element ID**
- **Title on hover**
- **Cellspacing**
- **Cellpadding**
- **Border**
- **Border color**
- **Background color**
- **Width of the section**

TIP

To see the Attributes and Settings tab, select the element of the adaptive layout, hover over the green icon on the left side of the container and click

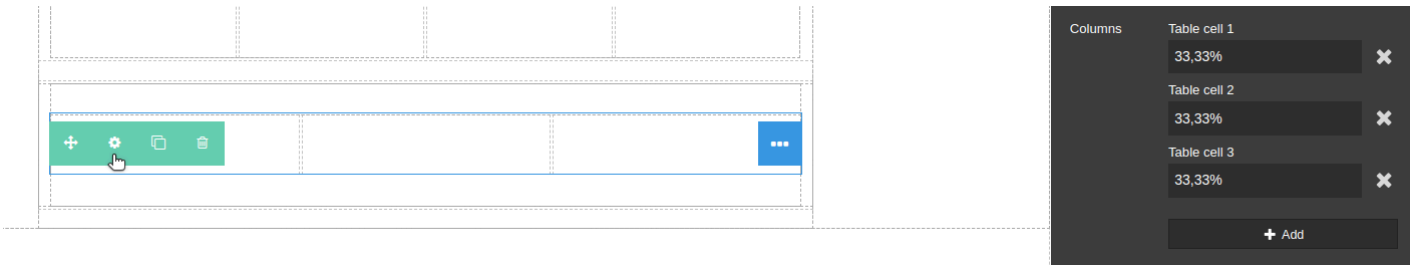
in the pop-up window.



Adaptive layout cells are displayed as containers, for which you can set only title and identifier.

Default value

When it comes to the default layout, you can also add cells and change width percent ratio.



The following options are available fordefault layout cells:

- **ID**
- **Title**
- **Background color**
- **Width**
- **Horizontal and vertical align**

Elements

You can use WYSIWYG editor to edit the text in such elements as Heading, Text, Link and Button.

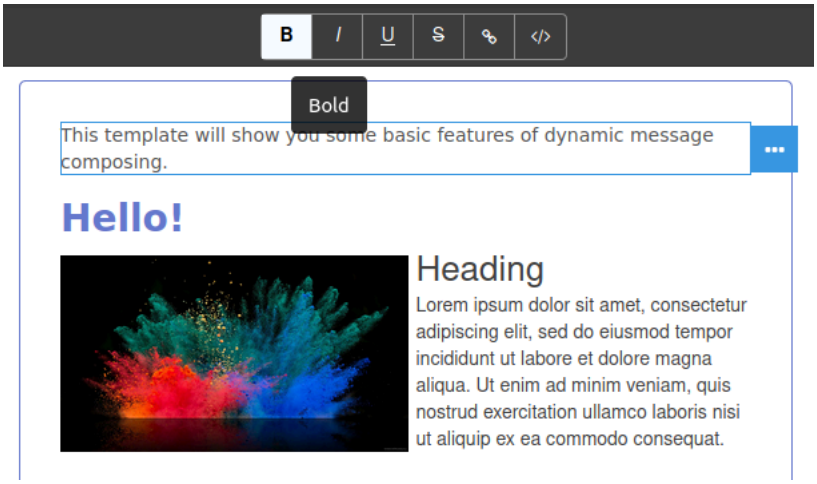
WYSIWYG editor for text elements

The WYSIWYG editor is available for text elements, like Header or Text. Click twice on the text element to edit its content in the central part of the editor.

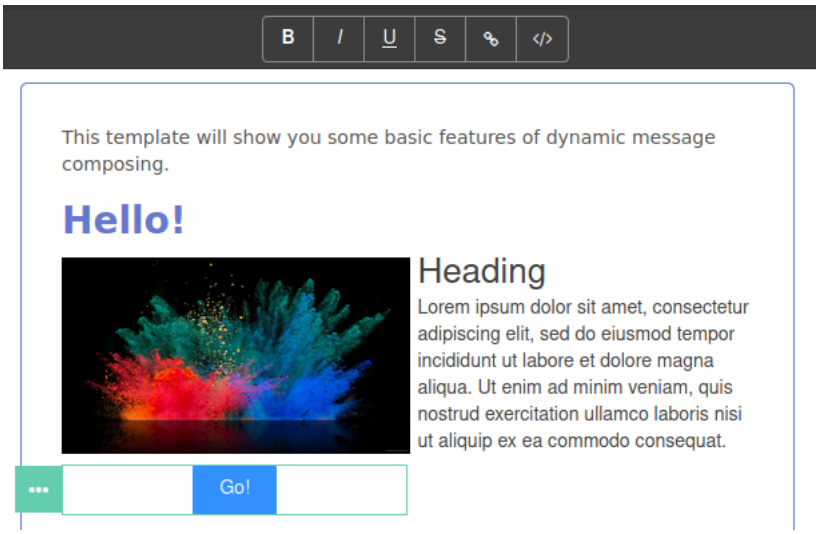
Double-click on the text element and the toolbar at the top of the editor will switch to text editor mode. You will be able to edit the text of the element in the central part of the editor. By the way, you'll immediately see the result.

Toolbar let you:

- **format text (bold, *italic*, underline, ~~strikethrough~~),**
- **add links,**
- **add variables.**



You can edit text on the Button element in the same way:

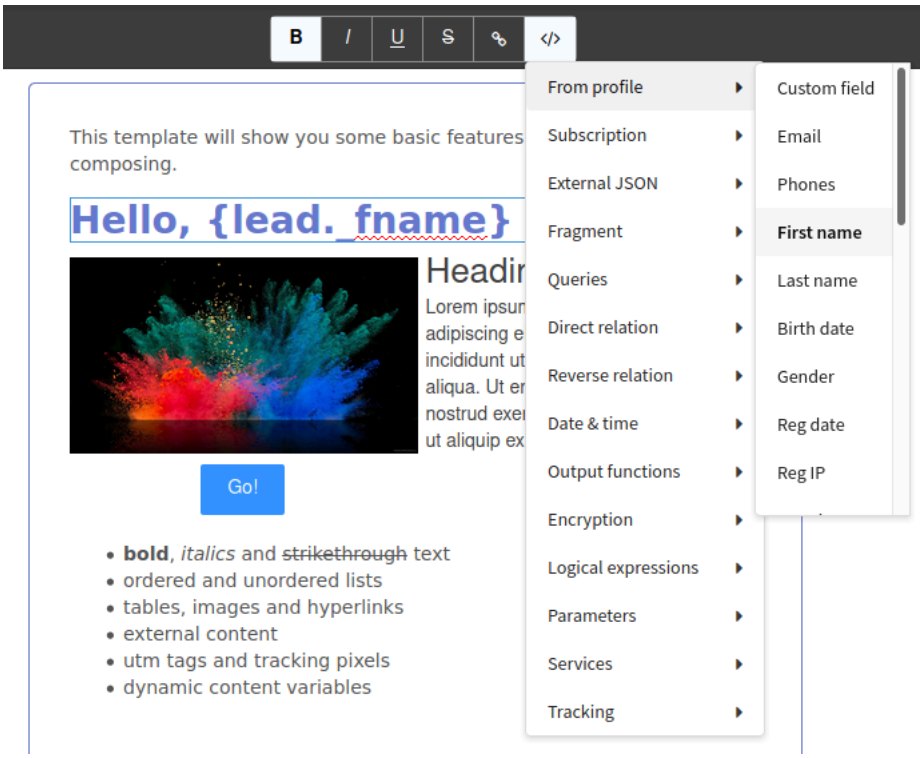


TIP

Text display options, as well as the behavior of links and buttons, are configured on the right side of the editor and are described later in the next article.

You can use the variables menu to add profile data and logical expressions to the template.

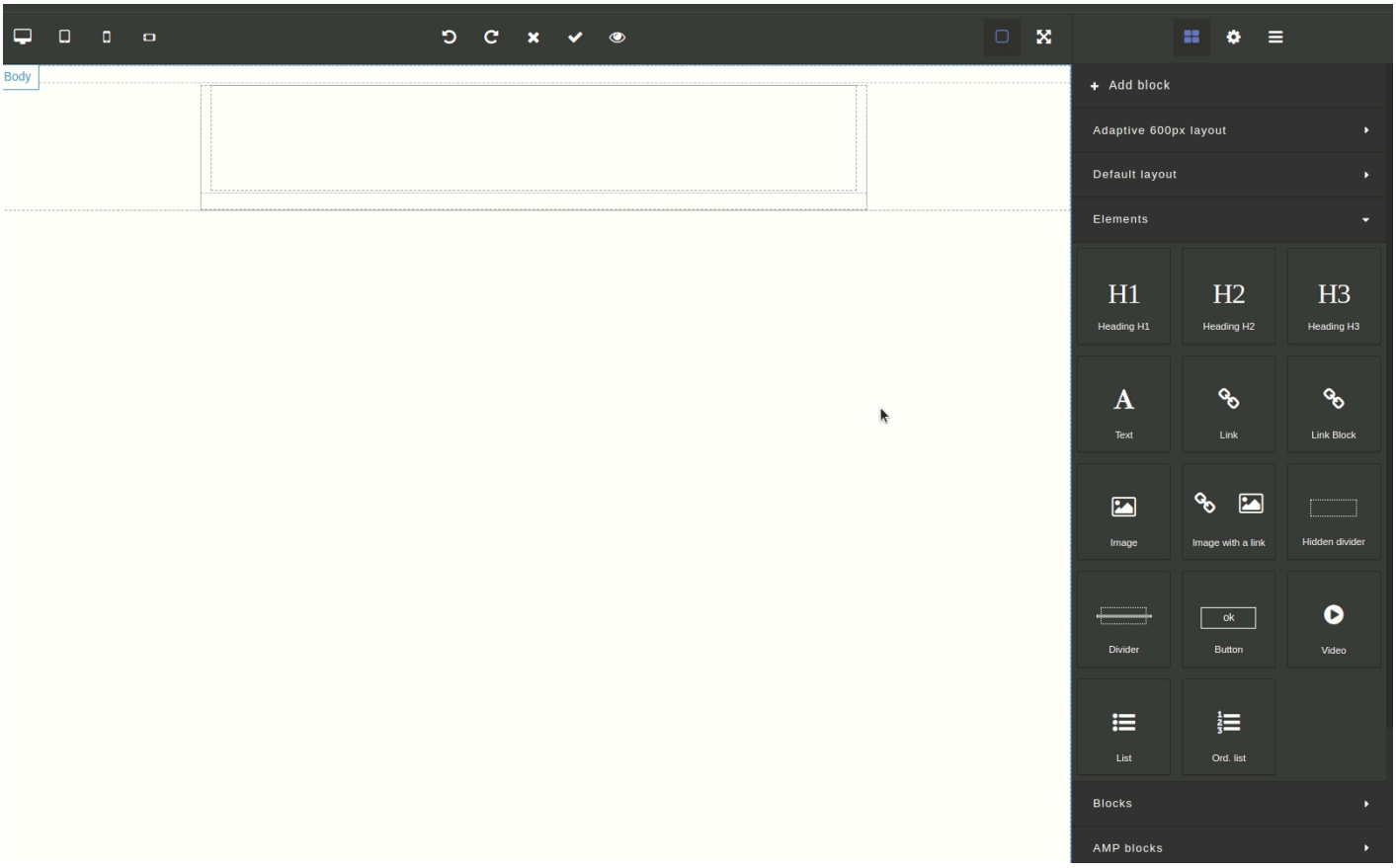
To insert custom fields value into the template, use `{lead.custom_field}` expression (`custom_field` - full name of custom field).



Let's take a look at each element:

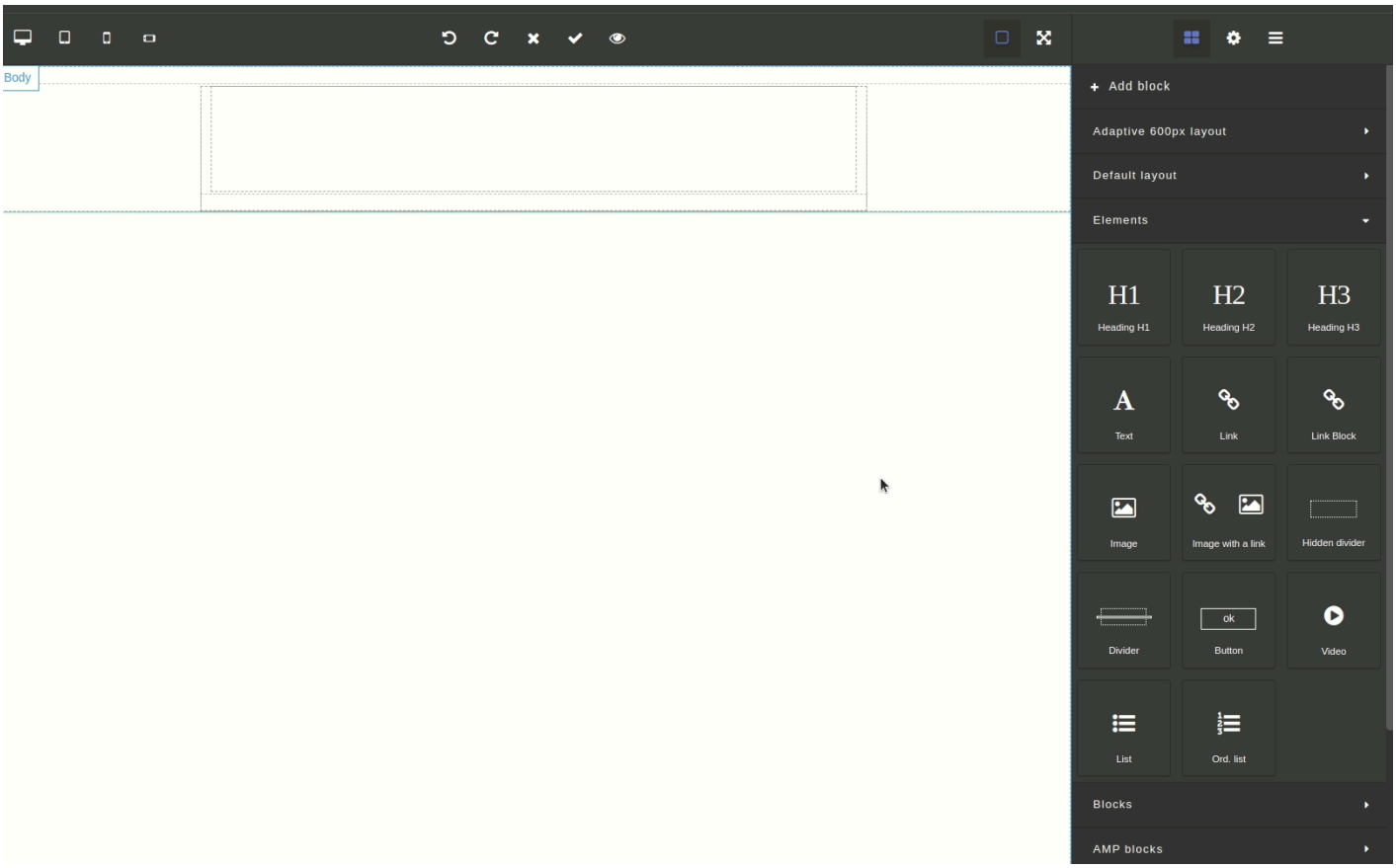
Heading

Heading attributes and settings: • **ID** • **Title**	



Text

Text attributes and settings: • **ID** • **Title**	



Link and Link block

Link attributes and settings: • **ID** • **Title** • **Link** • **Open link in (new window or current window)**	

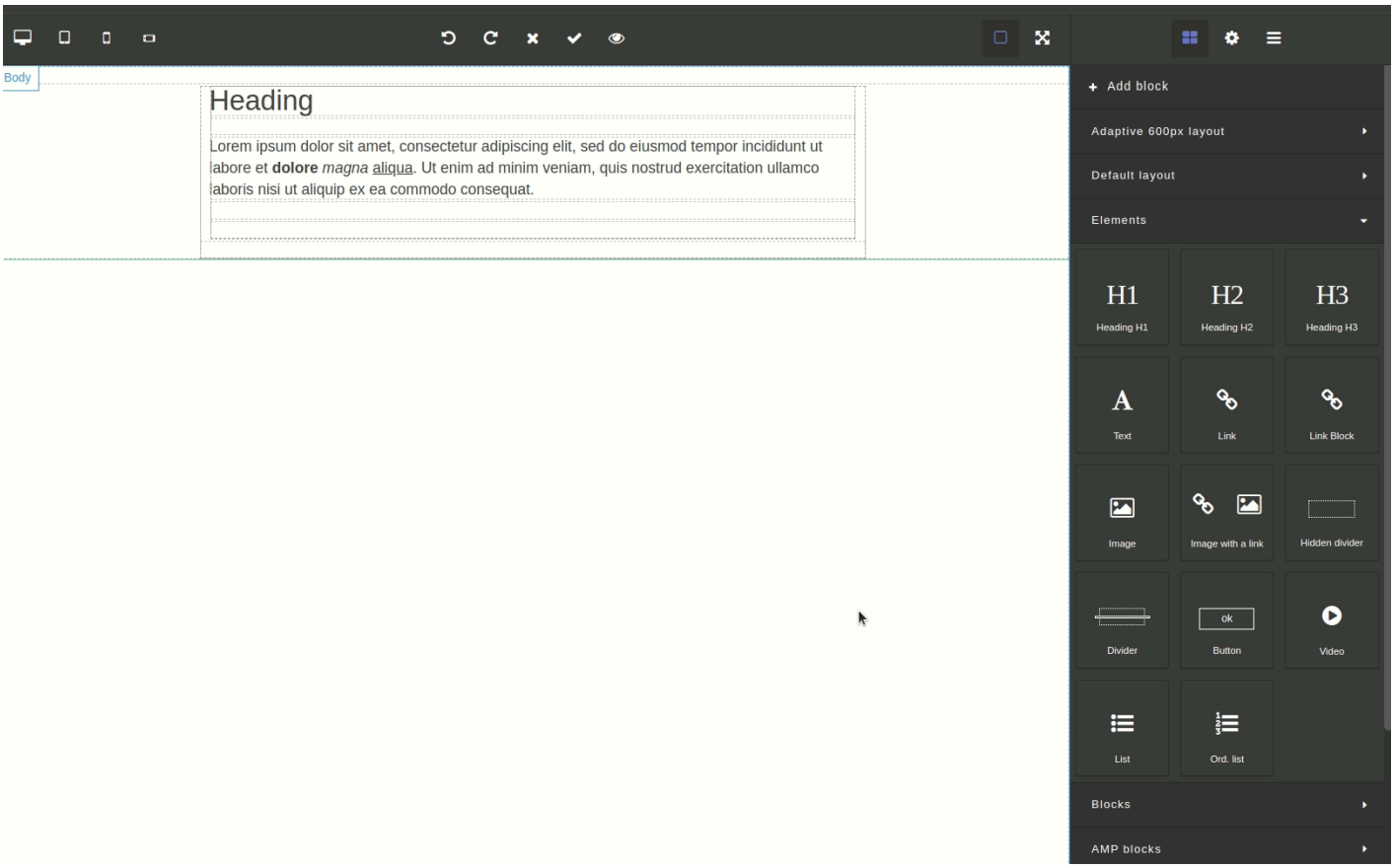


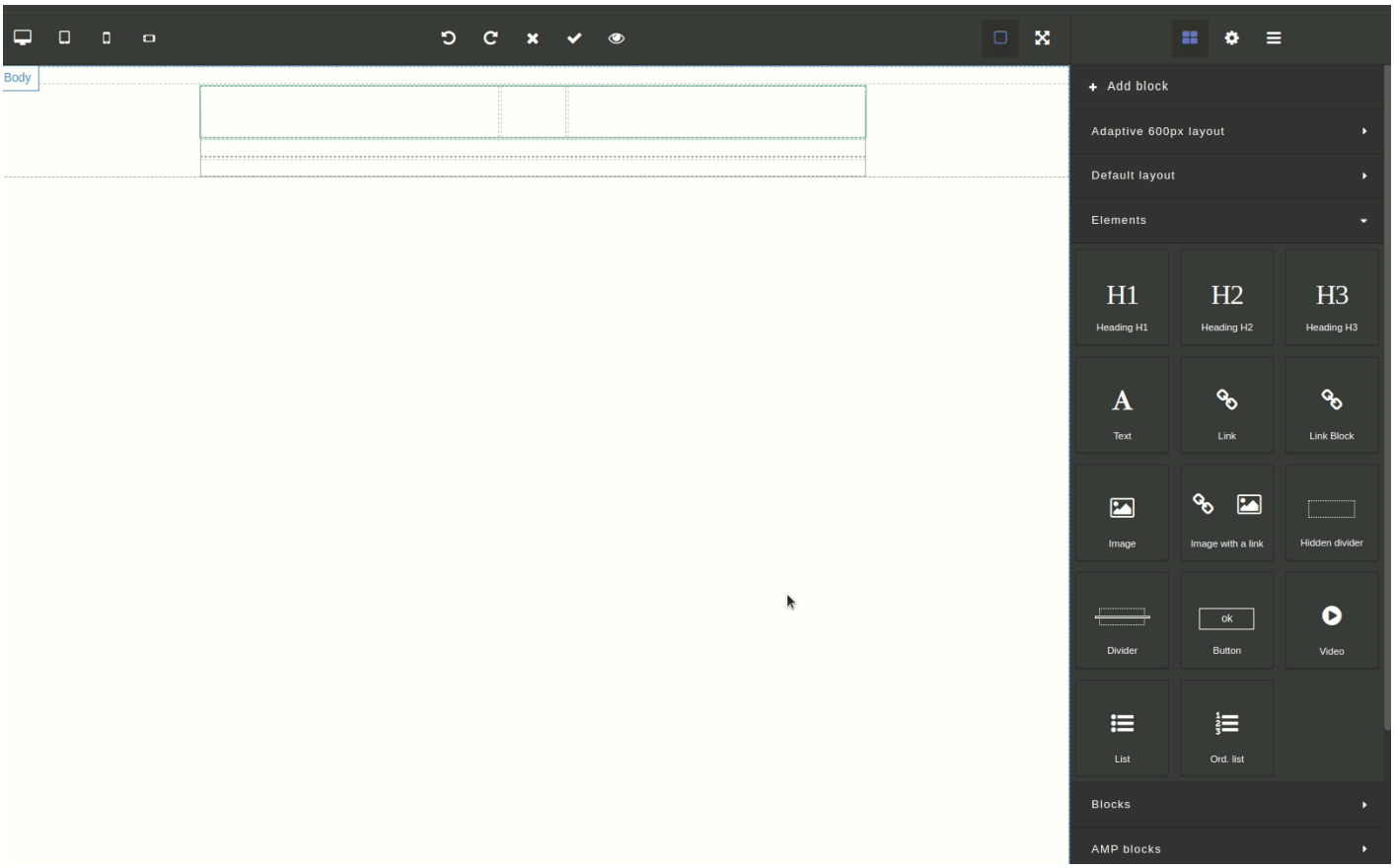
Image and Image with a link

Click twice on the element to open platform image gallery. You can use the search by file name or by tags.

Using form at the top part of the gallery, you can upload your images. Just drag ad drop the file you want, or click on the Browse button and select image.

After you select image, open it and click Insert button. You can also add or remove tags for images.

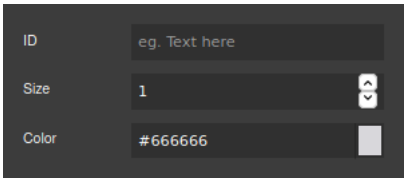
Image attributes and settings: • **ID** • **Title** • **Alternative text** • **Width and height (in pixels)** • **Image URL (in platform or network)**	



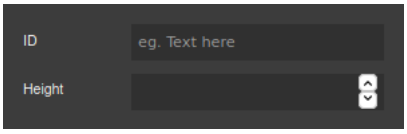
Divider and Hidden divider

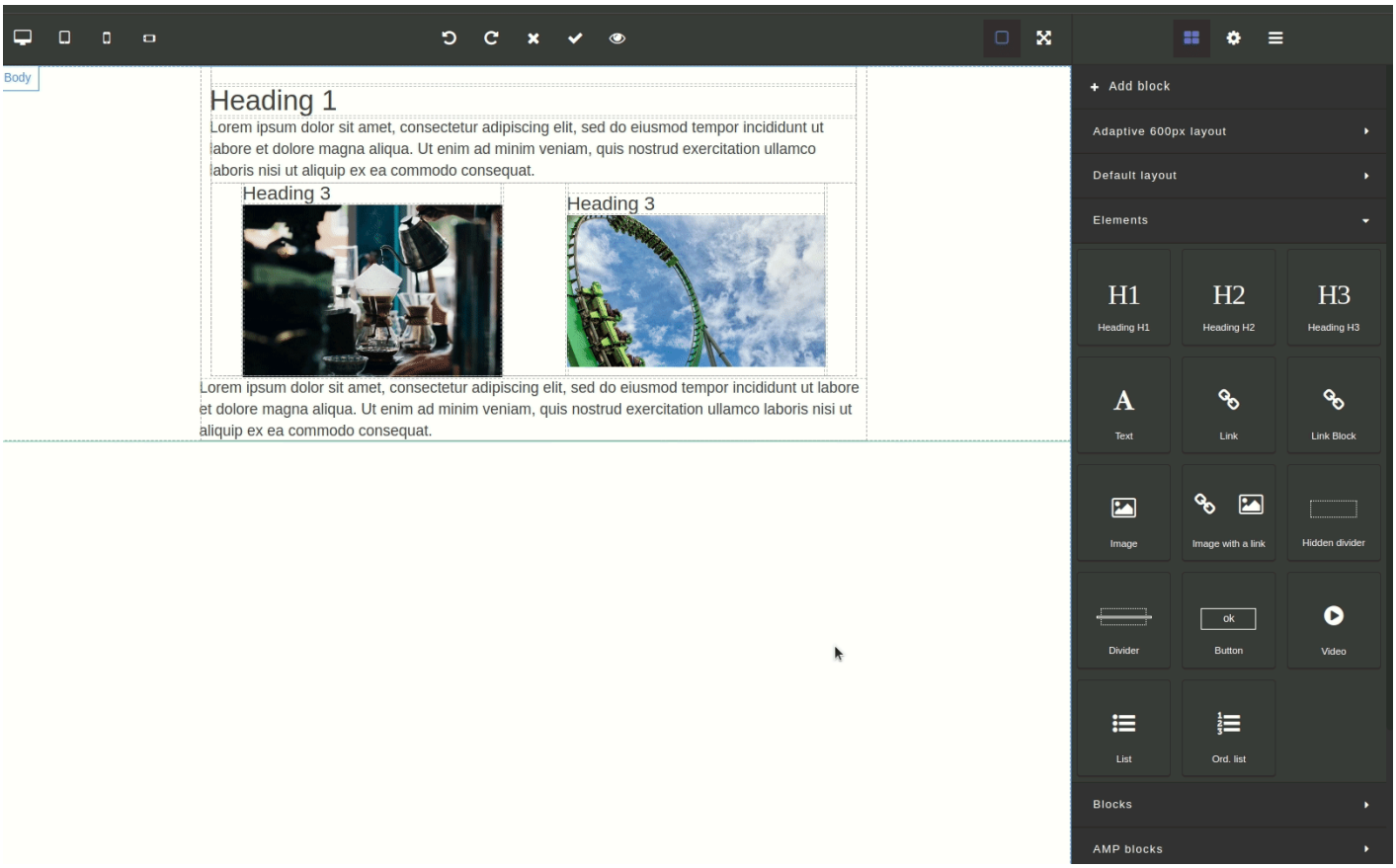
Dividers allow you to visually separate items from each other.

For the divider, you can set the size in pixels and choose a color:



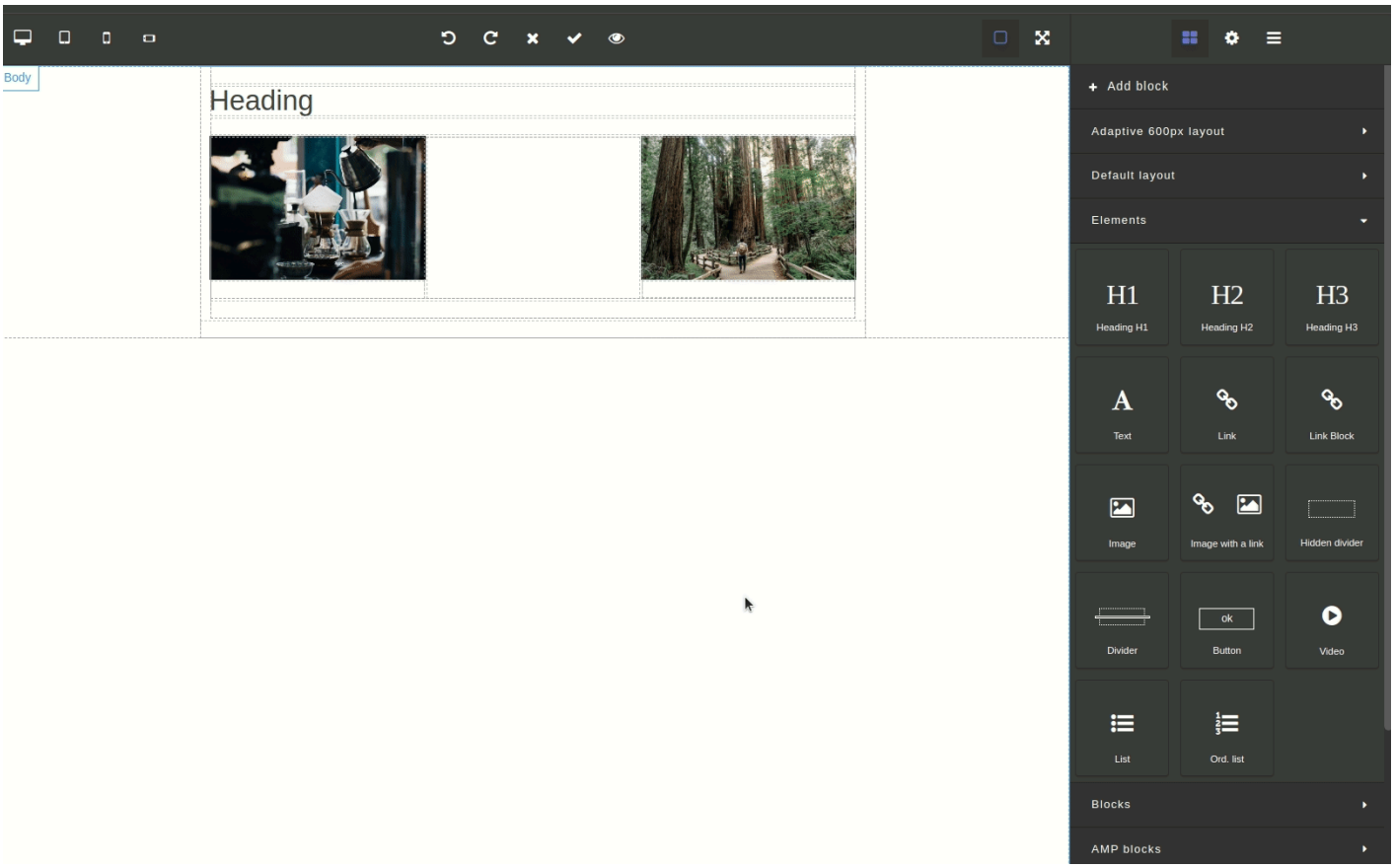
For the hidden divider, you can set height (in pixels):





Button

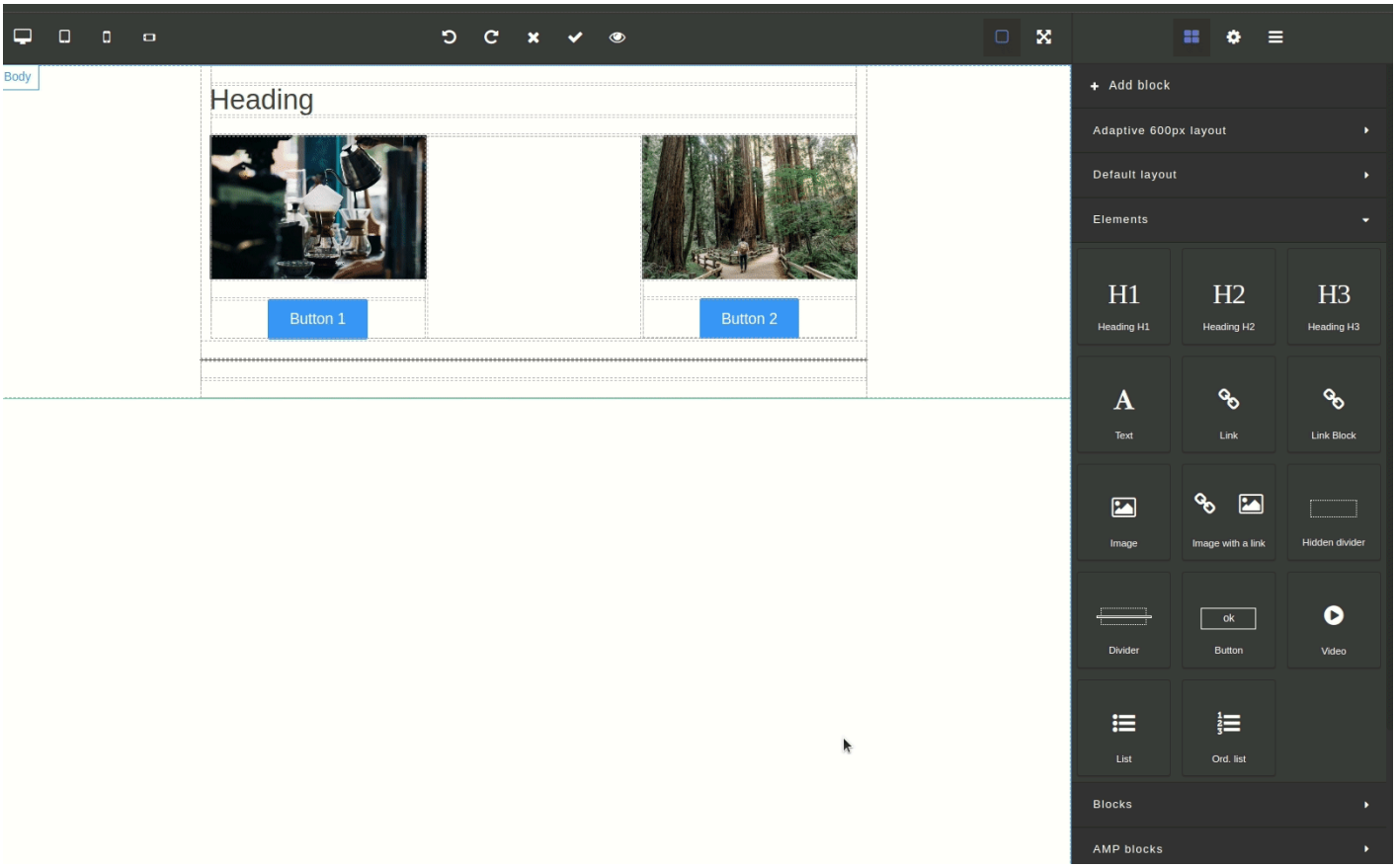
Button attributes and settings: • **ID** • **Title** • **Link** • **Open link in (new window or current window)**	



Video

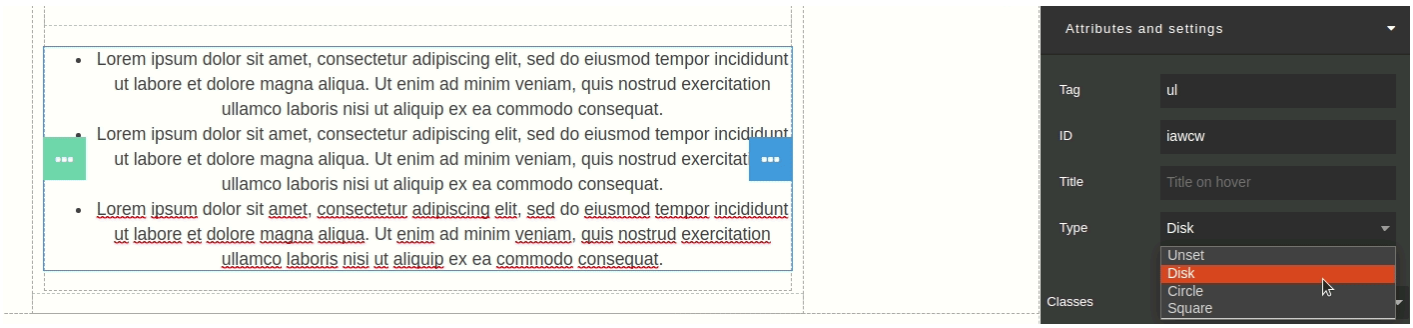
Video attributes and settings:

- **ID**
- **Title**
- **Video URL**
- **Preview image**
- **Link**
- **Player width and height**

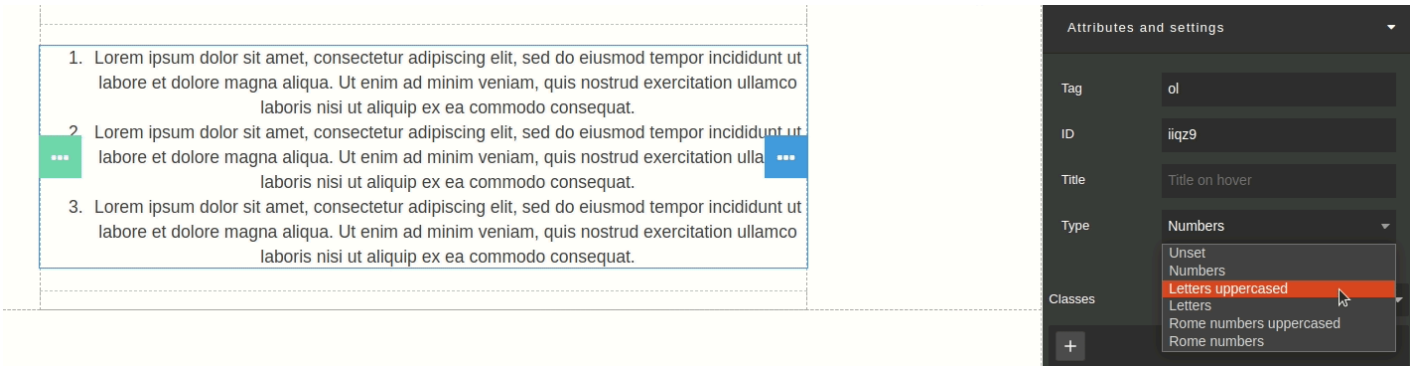


List

For an un ordered list, you can set ID and title, and select marks type (disk, circle or square):



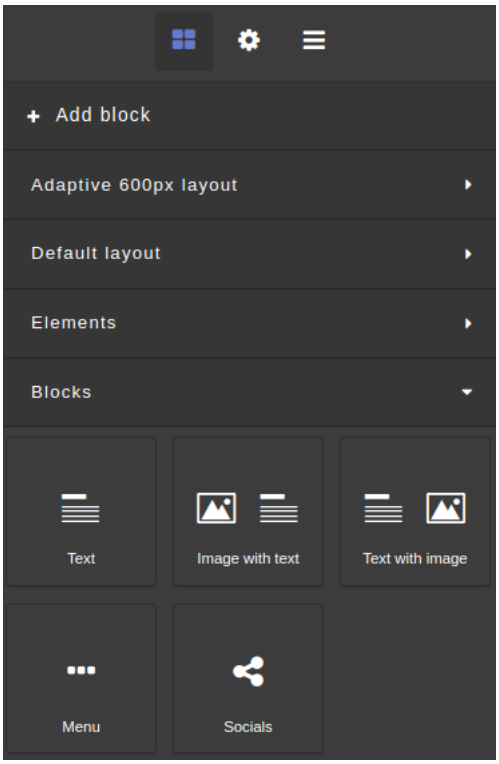
For an ordered list, you can select numeration type (numbers, Roman numbers or letters):



Blocks

Blocks are containers that already contain certain elements.

The blocks make it easier to work with the template, since they are some kind of templates for the platform user, which he, with a high degree of probability, uses in almost every email message. If you are tired of repeating the same monotonous action over and over again, then it's time to use blocks.



Text

The Text block is a container that includes an h1 heading and a paragraph.

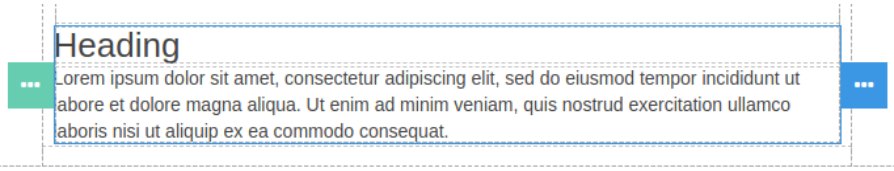
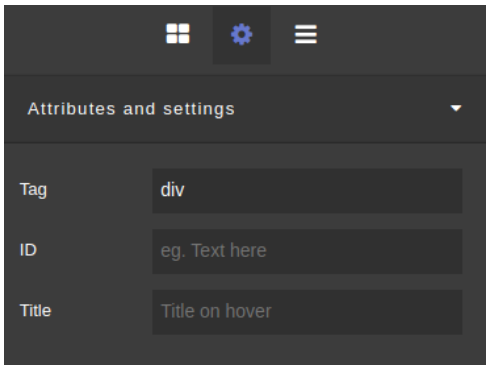
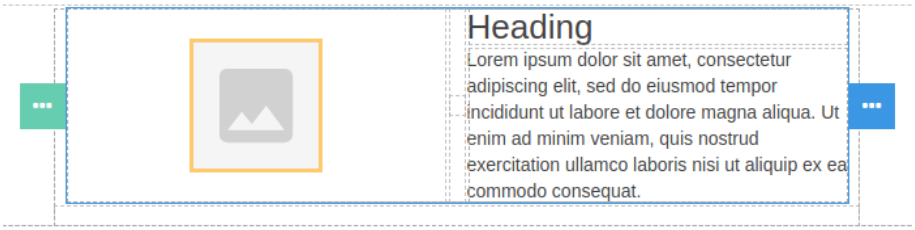
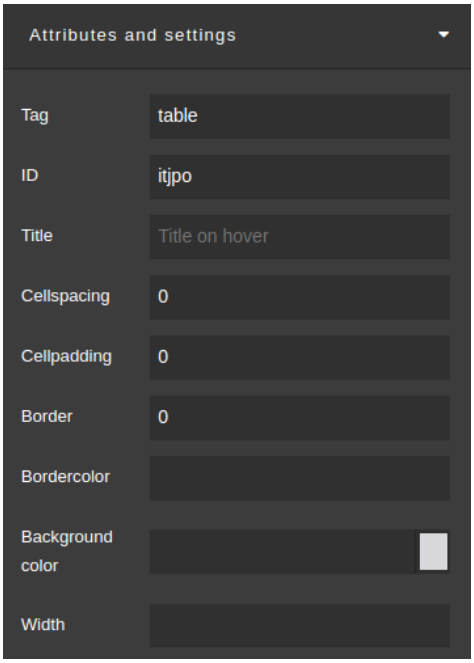
How it looks	Attributes and settings
	

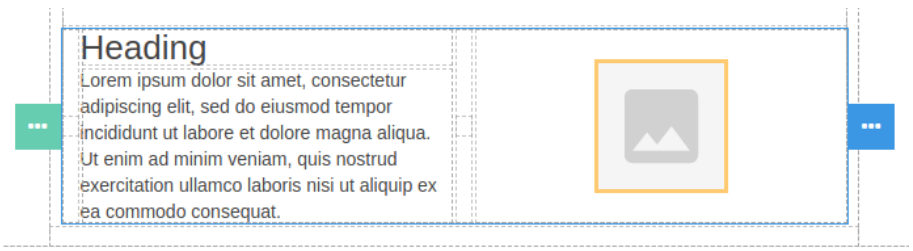
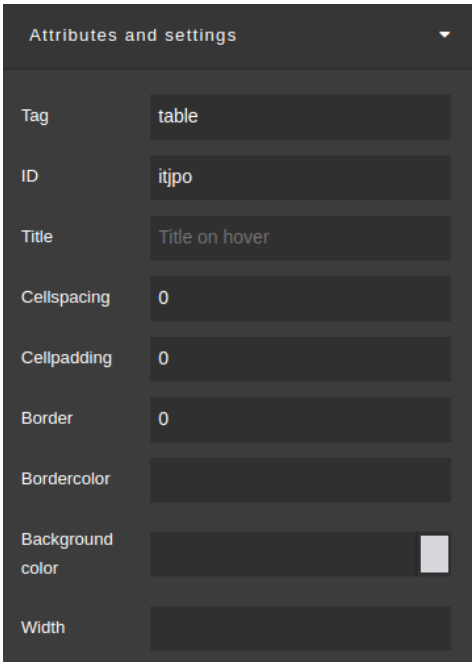
Image with text

The Image with text block is a table consisting of two cells. The cell on the left contains the image, and the cell on the right contains the h1 heading and paragraph.

How it looks	Attributes and settings
	

Text with image

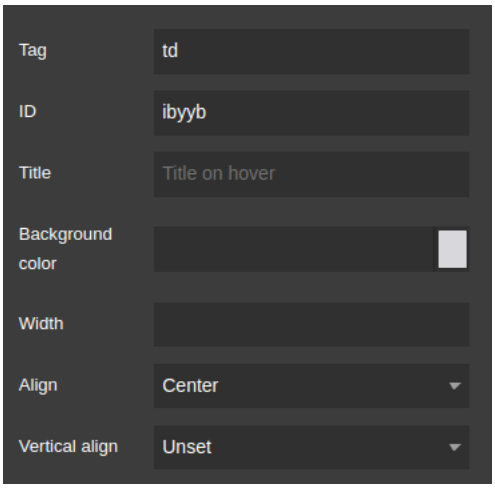
The Text with Image block is a table consisting of two cells. The cell on the left contains the h1 heading and paragraph, while the cell on the right contains the image.

How it looks	Attributes and settings
	

Menu

The Menu block is a table consisting of 4 cells of the same width.

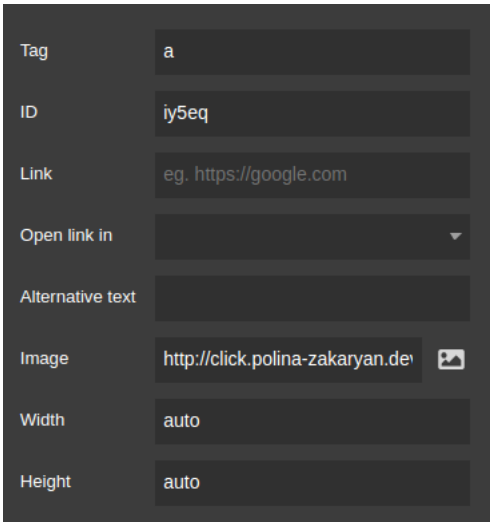
Each cell includes a link that you can customize in the Attributes & Settings panel. There you also have the ability to change the outer and inner margins between cells, adjust the border width and the total width of the element, set the background color and vertical and horizontal align.



Socials

The “Socials” block is a container that includes 8 images with links to social networks.

For each item, you can set your own link to click, add alternative text, customize the image, and change the width and height of the element.



AMP blocks

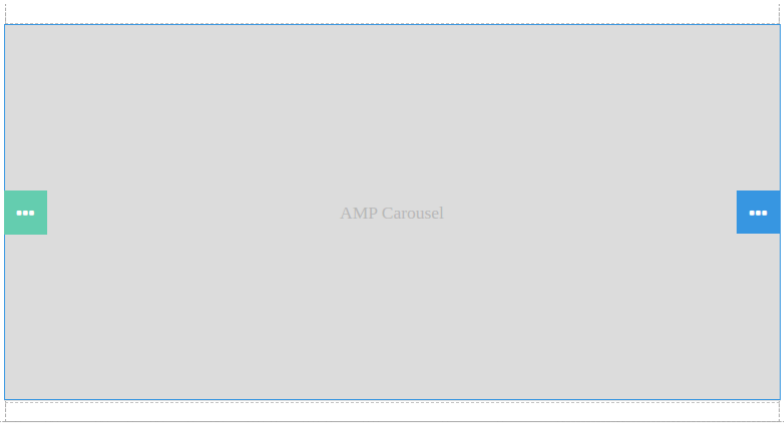
AMP emails is a technology that allows marketers to embed interactive elements such as carousel and accordion into their emails. The user, in turn, can view the data without even opening a new tab to visit the website.

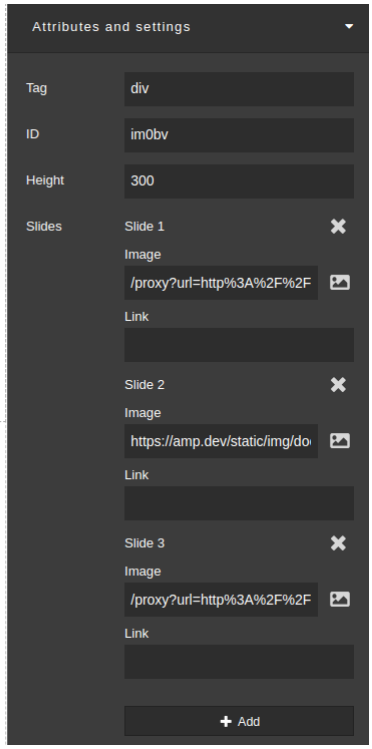
How to use AMP blocks in the Altcraft Platform:

Carousel

AMP carousel is a block that includes several slides. A carousel allows multiple pieces of content to occupy the same area of space on a page or screen.

AMP carousel settings are set in the “Attributes and settings” section. You can set element ID and height and add slides, for which you can set images and links. You can upload your image, select it from the gallery, or insert image URL:

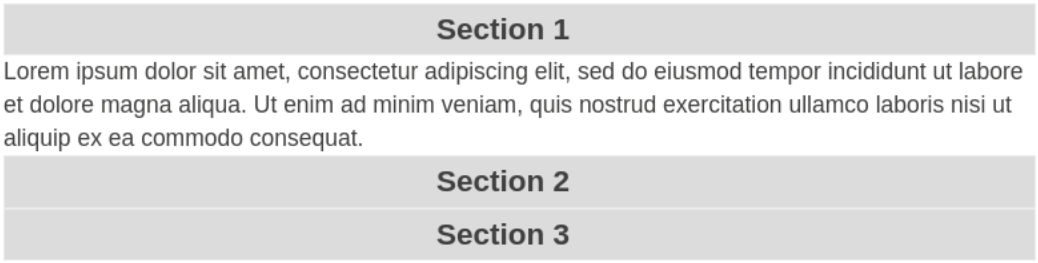




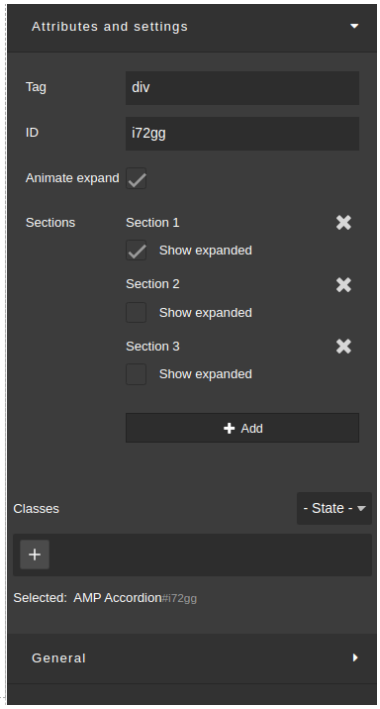
Accordion

AMP accordion is a block consisting of several tabs with elements. Each element can be “expanded” with a specific action from the user side (click, mouseover). You can add any elements to the sections.

An interface element such as the accordion not only helps users navigate quickly through content, but also allows the UI designer to include large amounts of information in a limited space.



AMP accordion settings are set in the “Attributes and settings” section. Select the block and add the desired amount of tabs. Sections can be opened or closed by default.



If you’ve already figured out how to add and configure elements in the template, the last thing left is to [style](#) the elements

Custom blocks

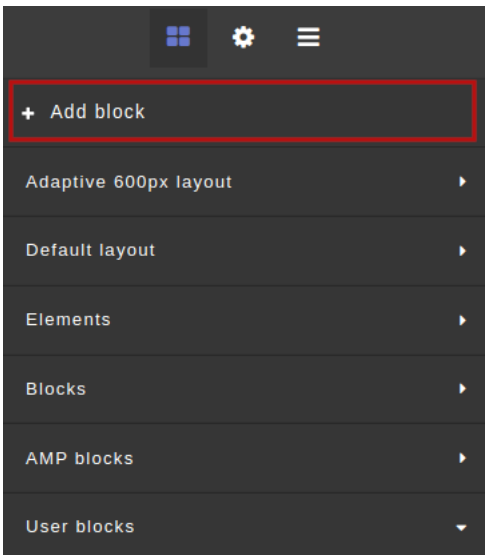
Save an entire section of the template as a custom block and then use that block over and over again. By the way, you can use it not only in the template where it was created, but also in any other.

What can be custom blocks? Any content that repeats from template to template. For example, the header or footer of a message.

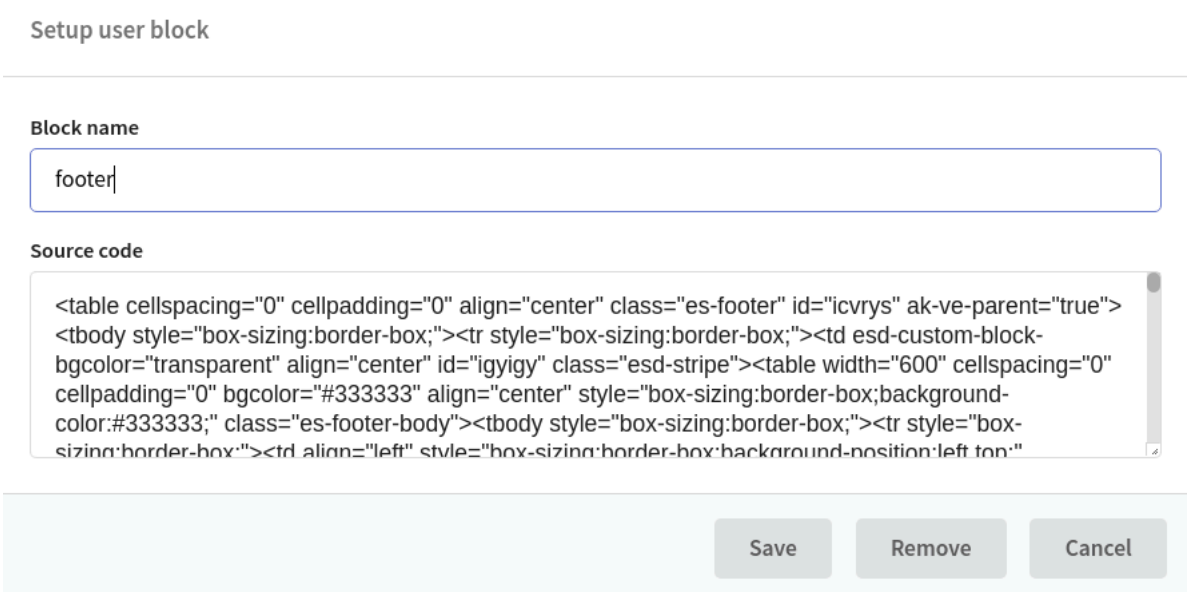
How to create a custom block

Option 1: Add source code

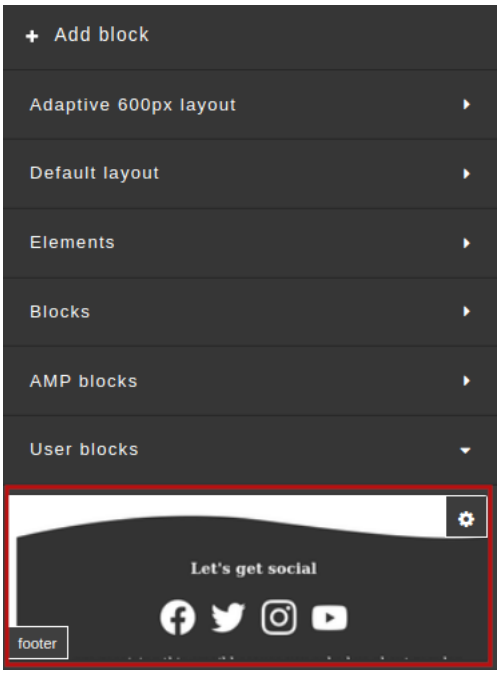
In the right panel of the visual editor, go to the “Add blocks” section. Click “+ Add Block”.



Enter source code of the fragment in the pop-up window and save the block.



The new custom block will be displayed after all available elements of the visual editor.



Option 2: Save the template container

Select the group of elements you want to save. You can simply hover the cursor over the desired container in the template, or you can use the section "Layer Manager".

After you select the desired section, you'll see the button for calling the menu of container operations —

. Click the floppy disk icon

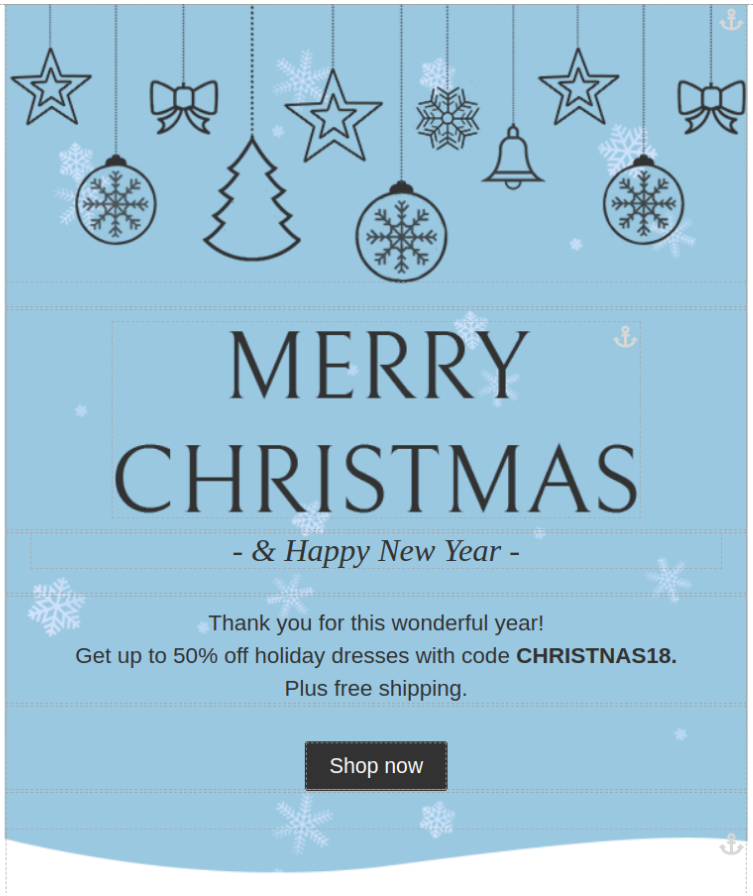
to save the block.

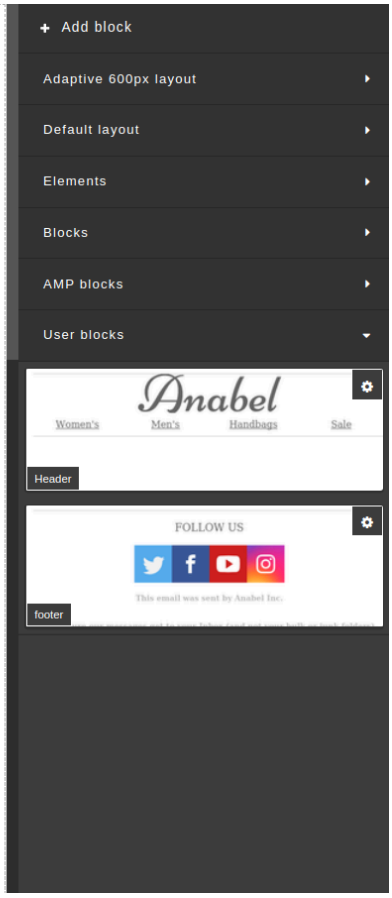
The new custom block will be displayed after all available elements of the visual editor.

How to add a custom block

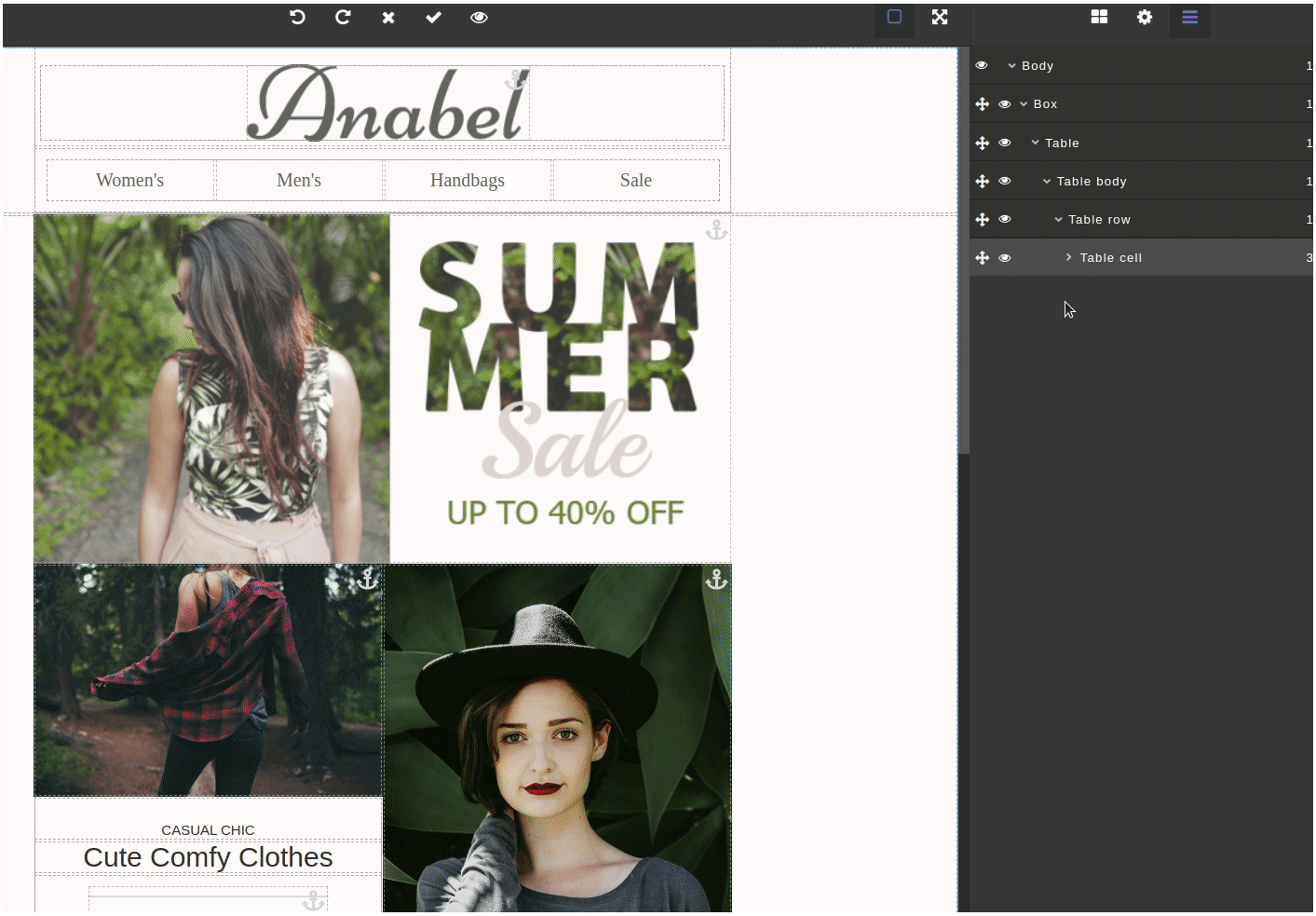
Select a custom block by clicking it, and then drag it to the part of the template you need. Let's get into practice.

Let's say we send emails to customers of the clothing store. Christmas is just around the corner, and we decide to give our subscribers a promotional code for a discount. This is how our template looks like:

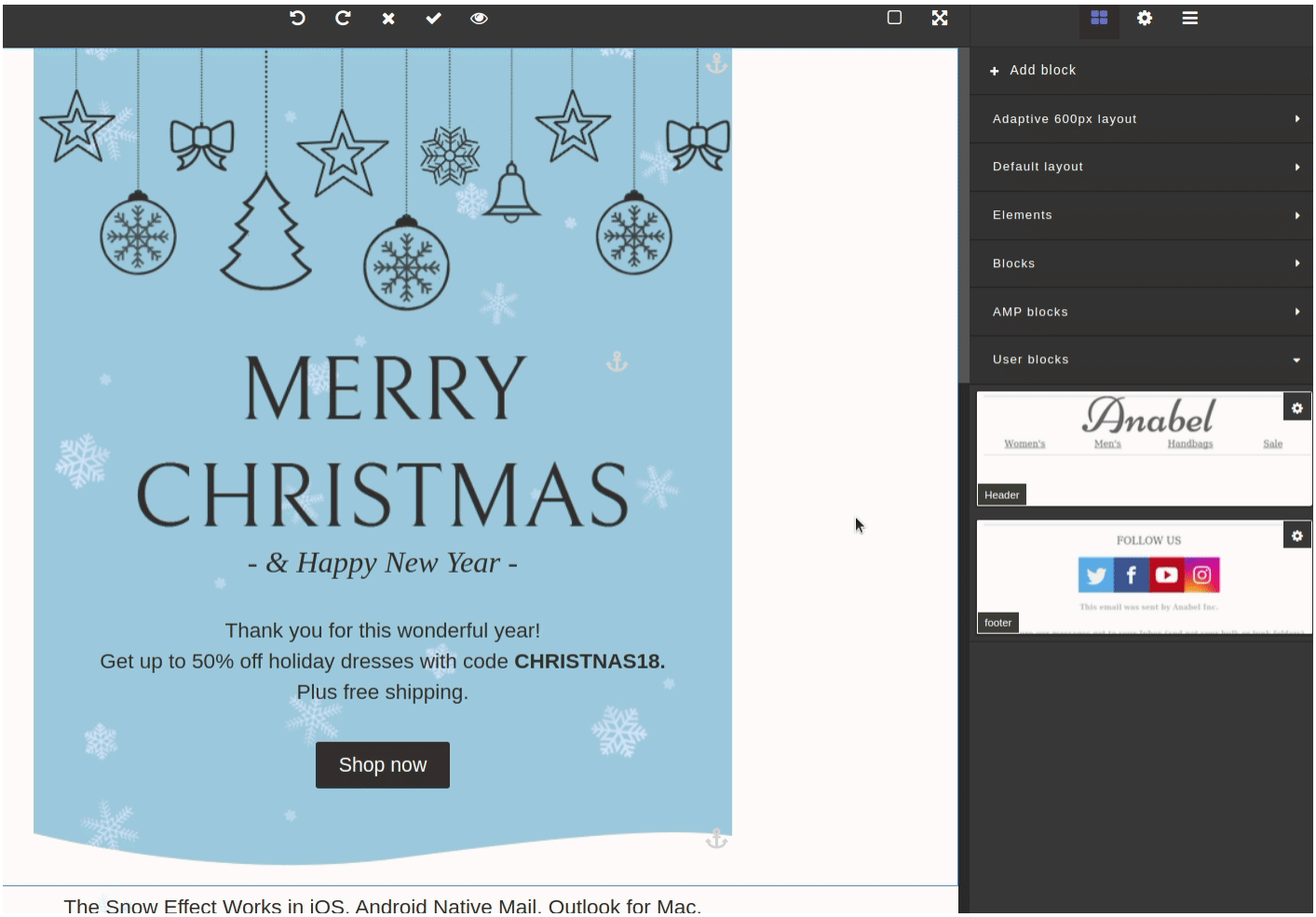




Of course, the main part of the message has changed in compare with usual messages, but the header with the store logo and the footer with our contacts will remain the same as in the daily mailings. Fortunately, we have long since saved these elements as user blocks.



Let's add our custom blocks to the template.



Good job, the template is ready, and we saved a lot of time!

👍

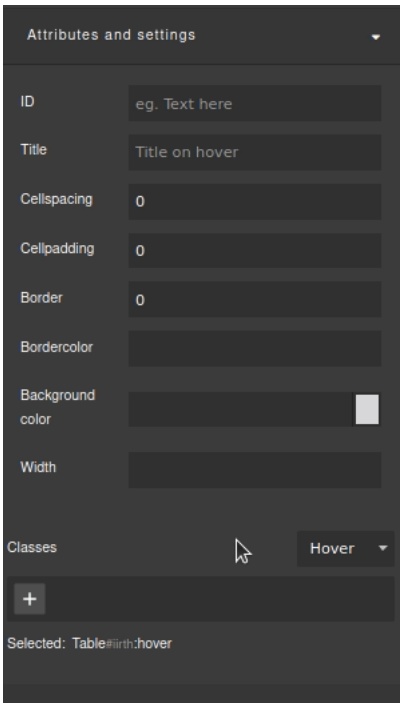
Style manager

Click on the central button in the right panel to open style manager for the selected element.

Each element has its own settings. Basic attributes are available for all elements. They are stored in the “style” attribute of HTML layout, so you can easily configure your template for perfect display in all mail clients.

Classes and state

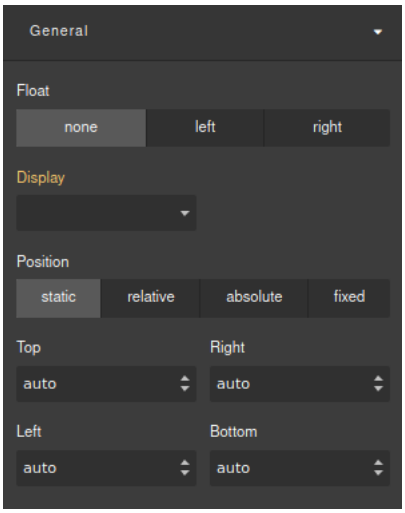
You can set class for element or configure state for different effect during click or moving cursor



General

You can set element position in the template:

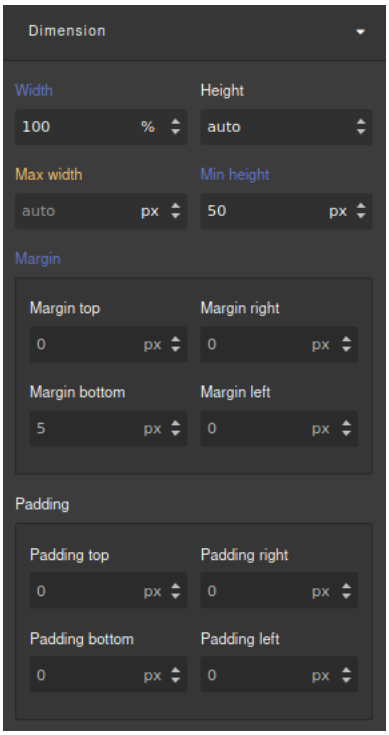
- **Float — element align by left or right side**
- **Display — inline, block, inline-block *или* flex — element display mode**
- **Position, Top, Left, Bottom — element position and spacing**



Dimension

In this section, you can set element display parameters:

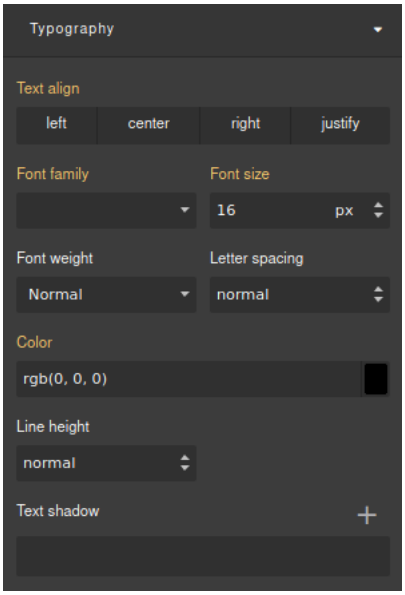
- **Width, Height**
- **Max width, Max height**
- **Margin**
- **Padding**



Typography

Text elements display parameters are situated in Typography section:

- **Text align**
- **Font family**
- **Font size**
- **Font weight**
- **Letter spacing**
- **Color (RGB or HEX)**
- **Line height**
- **Text shadow**



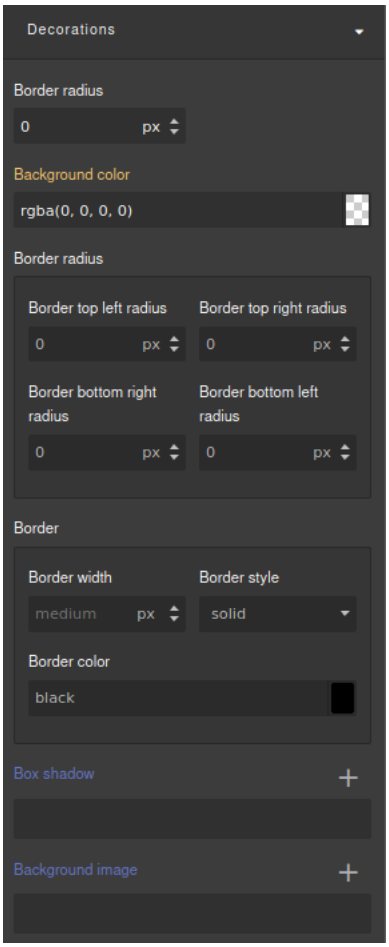
Decorations

You can set element border, background and shadows in this section:

- **Border radius**
- **Background color**
- **Border radius**
- **Border**

Box shadow — adds shadow for the element. You can configure shadow type, position, blur, spread, color

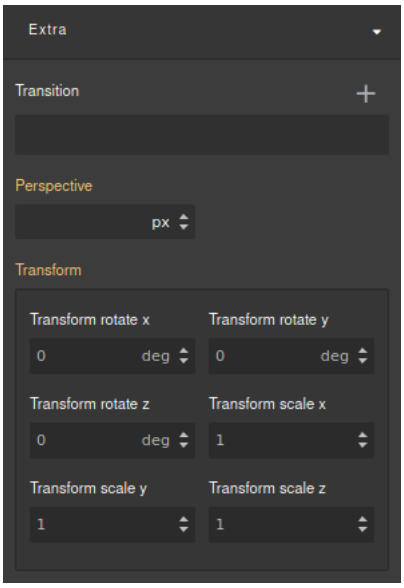
Background image — adds image as element background. You can upload your image or select it from the gallery, set image size, position and repeat:



Extra

In this section, you can configure element animation during state changes. The following parameters are available:

- **Transition**
- **Perspective**
- **Transform rotate, Transform scale**



Layer manager

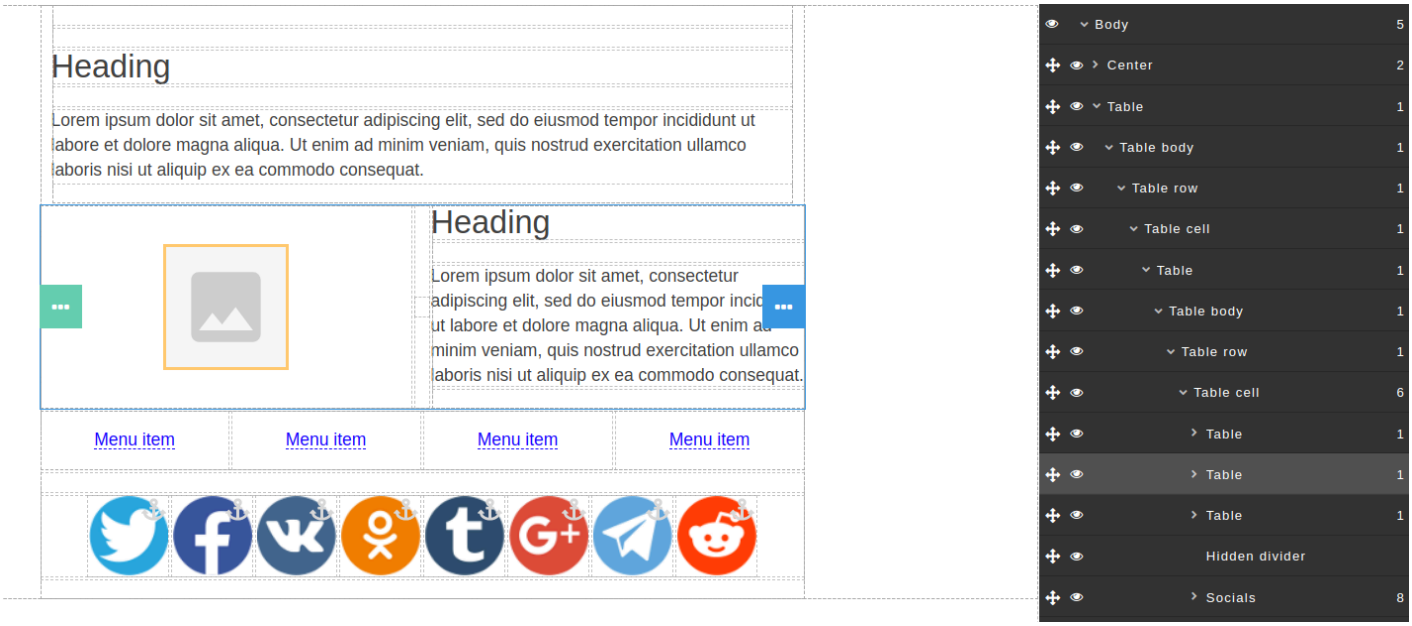
The last section in the right pane of the editor is Element Structure.

Here you will find a tree of elements (HTML DOM). Use it to get to any element in the template in a couple of clicks. Such a tool can be very useful, especially if you have to work with a template that you did not create. One look at the structure of the elements will allow you to understand the general structure of the template and adjust it without going into details.

When you open the element structure, a list of all the template elements appears on the right side. Here you can clearly see how the elements are nested within each other.

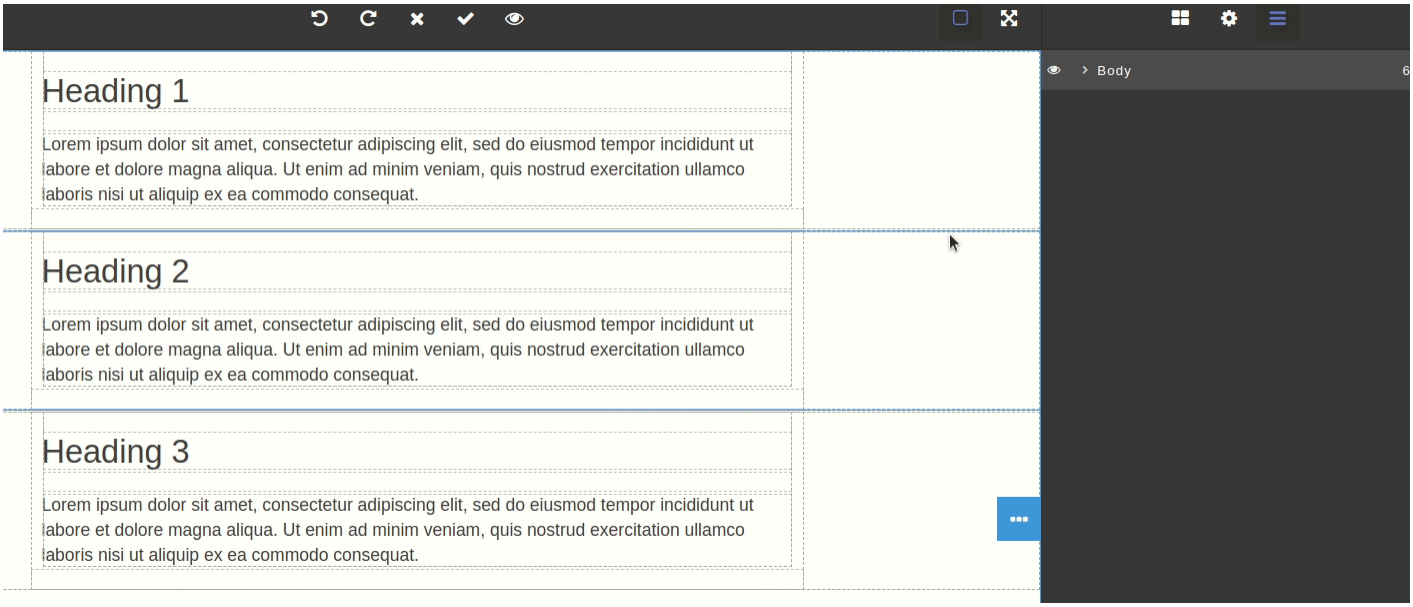
The number opposite each item shows how many children the parent has.

If you hover over or click on an element, the area occupied by it in the template will be highlighted in blue.



So how can we use the element structure?

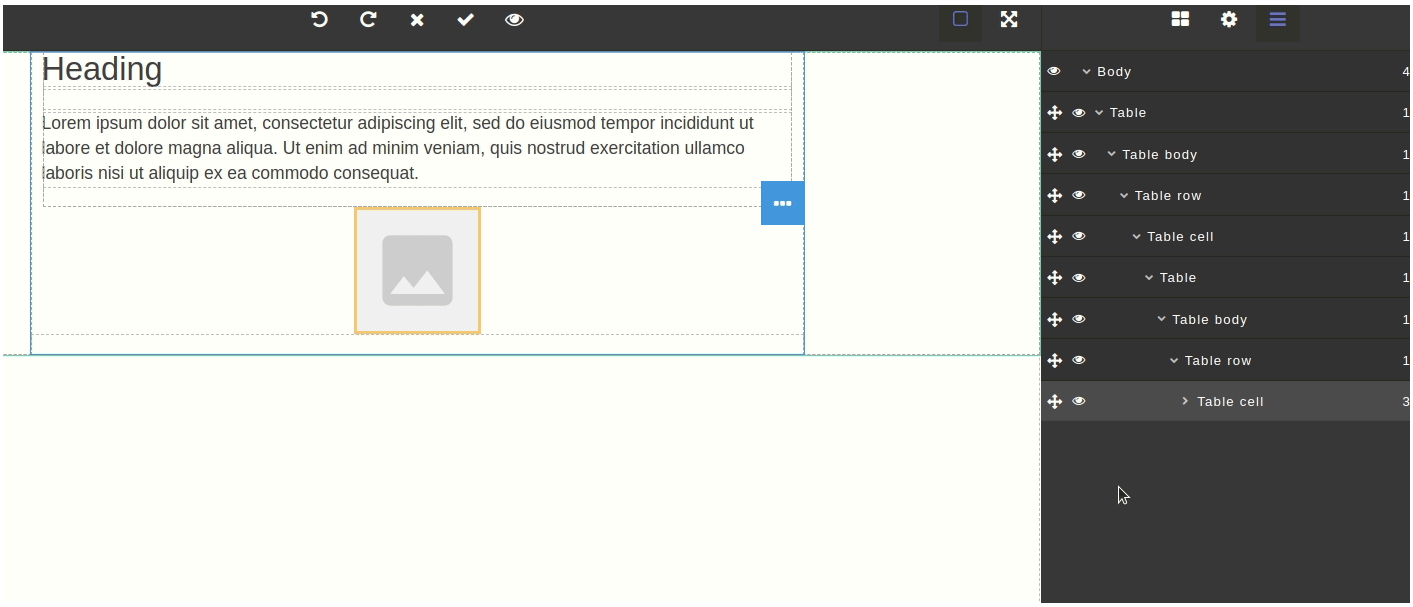
(1) Change the position of an element or the entire container. Drag and drop the element to its new position.



(2) Hide elements. Perhaps now you don't need a certain element of the message or a whole block, or you want to see how the template will look without it. Click

to hide the element. If you change your mind and want to return the element, click

.


Heading
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Body
Table
Table body
Table row
Table cell
Table
Table body
Table row
Table cell

Altcraft Variables and Functions

Profile Data

Description	Variable
Email address	`{lead.email}`
Phone numbers	`{lead.phones}`
First name	`{lead._fname}`
Last name	`{lead._lname}`
Date of birth	`{lead._bdate}`
Gender	`{lead._sex}`
Registration date	`{lead._regdate}`
Registration IP	`{lead._regip}`
Registration city	`{lead._regcity}`
Registration country	`{lead._regcountry}`
Registration URL	`{lead._regurl}`
IP address	`{lead._ip}`
City	`{lead._city}`
Country	`{lead._country}`
Region	`{lead._region}`
Postal code	`{lead._postal_code}`
Time zone	`{lead._tz}`
Provider	`{lead._vendor}`
Profile ID	`{lead._id}`
XXM hash	`{lead._xxm}`
XX hash	`{lead._xxh}`
MD5 hash	`{lead._md5}`
* **Custom field**	`{lead.custom_field_name}`

Subscriptions

Subscription variables contain data related to a specific subscription and the delivery channel being used. For example, it is not possible to pass a phone number when sending via email using these variables.

Description	Variable
Subscription email	`{subscription.email}`
Email domain	`{subscription.email_domain}`
List of subscription emails	`{subscription.emails}`
Subscription phone	`{subscription.phone}`
List of subscription phones	`{subscription.phones}`
List of push subscriptions	`{subscription.pushes}`
MD5 hash of email	`{subscription.email_md5}`
MD5 hash of phone	`{subscription.phone_md5}`
Subscription hash	`{subscription.hash_id}`
Subscription field (by channel)	`{subscription_field("channel" "field_name" resource_id)}` **channel — string indicating the channel (“email”)** **field_name — name of the field to insert (“domain”)** **resource — resource ID (e.g., 10)**

The `subscription_field` function lets you insert fields from subscriptions regardless of the channel used by the current message. For example, if a profile has an SMS subscription, you can insert their phone number into the email version of the message:

```
{subscription_field("sms" "phone" 1)}
```

Available fields:

Channel	Field	Description
email	**email**	**Subscription email address**
	domain	**Email domain of the subscription**
	domain_group	**Domain group**
	md5	**MD5 hash of the subscription email**
sms	**phone**	**Subscription phone number**
	md5	**MD5 hash of the subscription phone**
push	**subscription_id**	**Push subscription ID**
	bundle_id	**iOS APNs app bundle ID**
	provider	**Push provider**
telegram_bot/telegram_group	**id**	**Chat ID**

TIP

Similarly, you can insert data from custom channel subscriptions into templates. Fields of the main channels correspond to the field SIDs you set when creating the channel.

UTM

If a profile has UTMs in their card, you can insert them into the message template.

Description	Variable
Mailing UTM	`{lead._utm_campaign}`
Traffic source UTM	`{lead._utm_source}`
Medium UTM	`{lead._utm_medium}`
Content UTM	`{lead._utm_content}`
Keyword UTM	`{lead._utm_term}`

External JSON

External JSON content is a way to create dynamic messages and automate work. It lets you add information generated by your server right at the moment each message is sent.

Read more in "Using JSON content in messages".

In message text, JSON content appears as variables like `{json.variable}`. Data types `list` and `object` support loops; for `object` you can also access elements by key: `{json.object.key_name}`.

Market

Using market variables

To use order and product information in mailings, add market variables to your message template. See "Using market variables in messages" for details.

Date and Time

Date and time variables

Description	Variable	Example
Formatted date	`{format}`	`{format datenow "%Y-%m-%D %H:%M:%S timezone, %I am"}` **2018-12-25 18:56:10 +03:00, 6 pm**
Raw date	`{datenow}`	**2018-12-25T18:56:10+0300**
Date offset	`{adddate()}`	`{adddate(datenow 0 0 3)}` **2018-12-27T18:56:10+0300**
Full year	`{yearlong}`	**2018**
Year	`{year}`	**18**
Month	`{month}`	**12**
Day	`{day}`	**10**
24-hour format hours	`{hours}`	**18**
12-hour format hours	`{hours12}`	**06**
AM/PM	`{ampm}`	**PM**
Minutes	`{minutes}`	**56**
Seconds	`{seconds}`	**10**

Output Functions

Output functions

Description	Variable	Example
Number formatting	`format`	`{format 1000000000 ","}` **1,000,000,000**
Variable length	`length`	`{length("abc")}` **3**
Array of values	`array`	`{array[1 "2" 3.3]}` **[1 2 3.3]**
Random array element	`randomize`	`{randomize(array[1 2 3])}` **1** **Example: a profile custom field `Favourite_genres` stores `["Detective", "Humor", "Sci-Fi"]`. `{randomize(lead.Favourite_genres)}` will output one random element, e.g., `"Detective"`. Works with Phones, Tags, Array of Objects, as well as arrays in API content and external JSON content.**

Description	Variable	Example
Limit array	`limit`	`{limit(array[1 2 3] 2)}` **[1 2]**
Number is even	`iseven`	`{iseven(2)}` **=> true** `{iseven(3)}` **=> false**
URL encode	`urlencode`	`{urlencode(lead._regurl)}` **http:%2F%2Ftest.testdomain.com%2Fsomething** **Use to encode invalid characters in URL parameters.**
URL decode	`urldecode`	`{urldecode(lead._regurl)}` **http://test.testdomain.com/something**
Uppercase	`uppercase`	`{uppercase("text")}` **TEXT**
Uppercase first letter	`uppercasefirst`	`{uppercasefirst("john")}` **John**

Encryption

Encryption and hashing

Template variable encryption is used when you need to pass data along with a click to a target site but don't want the data to appear in web server logs or be visible to the user. For example, form prefill data can be encoded in base58 and decoded on the site.

To pass template content and variables in encrypted or hashed form, use the `crypt` function. Click the `</>` button in the template editor and choose Encryption.

```
{crypt(algorithm field key)}
```

You can change encoding or encrypt data with these algorithms:

- **base58**
- **base64**
- **sha1**
- **sha256**
- **aes**
- **blowfish**

For sha1, sha256, aes and Blowfish you must specify a key that will be used on the other side to decrypt the value.

Details for AES and Blowfish:

	AES	Blowfish
Initialization vector	**First 16 bytes of the string**	**First 8 bytes of the string**
Cipher mode	**CFB (Cipher Feedback)**	**CBC (Cipher Block Chaining)**
Key length	**128-bit — 16 chars**	**64-bit — 8 chars**
Padding	**zero**	**zero**

TIP

Encryption/decryption uses Base64URL encoding.

Examples of syntax:

```
{crypt(base58 lead._fname)}
{crypt(base64 lead._lname)}
{crypt(sha256 lead.custom_field "encrypt_key")}
{crypt(sha1 lead._city "encrypt_key")}
{crypt(aes lead._fname "key1234567891234")}
{crypt(blowfish lead._fname "key12345")}
```

If you need to encrypt a string, wrap it in quotes. For a number, quotes are not needed:

```
{crypt(base58 "value")}
{crypt(base64 42)}
```

Hashing algorithms (each supports an optional salt):

- **md5 + salt**
- **xxh32 + salt**

- **xxh64 + salt**

Syntax matches the encryption examples:

```
{crypt(md5 subscription.email "salt")}
{crypt(xxh32 loyalty.new2.promocode "salt")}
{crypt(xxh64 json.text "salt")}
```

Note that encrypted data is already Base64-encoded.

Logical Expressions

Logical expressions

Usage is described in a separate article. For a simple condition:

```
{if lead.age gte 18}
    If 18 or older.
{else}
    If younger than 18.
{end}
```

Fragments

Message fragments

Fragments are message blocks reused across messages (e.g., header, footer, social links).

Call a fragment with `{fragment.<short_fragment_name>}`**.**

Call a random fragment from a list: `{randomfragment[fragment.name1 fragment.name2 fragment.name3]}`**.**

See more in [this article].

TIP: Fragments and Parts

The platform also includes similar functionality called parts. Parts differ from fragments in that they are configured and used within a single email. To start working with parts, add the variable `{part.example_part}` **to the template code. After that, the Content tab will appear at the bottom of the page, where you can configure the template parts.**

Read more about parts [here].

Parameters

Description	Variable
Sent message ID (SMID)	`{send_message_id}`
Resource token	`{resource_token}`
Template ID	`{msgid}`
Mailing ID	`{campid}`
Resource ID	`{resid}`
Database (profiles) ID	`{listid}`
Mailing SUBID	`{subid}`
Resource name	`{resname}`
Database name	`{listname}`
Message type	`{msg_type}`
Resource URL	`{resurl}`
Resource domain	`{resdomain}`
Message name	`{msgname}`
Mailing name	`{campname}`
SMS sender name	`{from_name_sms}`
Segment ID	`{segid}`
Segment name	`{segname}`
Attribute value	`{attribute_value("attribute")}` **where** `"attribute"` **is the API name of the attribute**

Service

Description	Variable
API content	`{apicontent.field_name}`
Cancel message sending	`{cancel}`

Tracking

Description	Variable
Tracking domain	`{trkdomain}`
Pixel recording an open	`{pixel}`
Pixel recording a read	`{read}`
Subscription manager link	`{preferences}`
Unsubscribe link	`{unsubscribe}`
Global unsubscribe link	`{globalunsubscribe}`
Add to suppression list	`{suppress}`
Web version link	`{webversion}`

Workflow

Description	Variable
Workflow name	`{workflow_name}`
Workflow ID	`{workflow_id}`
Node (element) ID	`{node_id}`
Node (element) type	`{node_type}`

Events

Event variables let you work with user actions (page views, add-to-cart, purchases, etc.) collected via pixels and goals.

How it works

- **Real data is populated only when a trigger actually fires or within a running workflow.**
- **Events are context-bound — in triggers they are events of that trigger; in workflows they are events since entry.**

> INFO
>
> **These variables are only available for trigger-based mailings and scenarios that use the “Pixel activation” method.**

Variables

Description	Variable	Notes
First event	`{event.first}`	**The oldest event in the current context**
Last event	`{event.last}`	**The most recent event in the current context**
All events	`{event.all}`	**Full list of events in reverse arrival order; index `[0]` is the last**
Event count	`{event.count}`	**Number of events in the queue**

Using in templates

To get a specific event field, append the parameter with a dot:

```
{event.first.goal} // goal of the first event
{event.last.value} // value of the last event
{event.first.geo_city} // city from the first event
{event.last.utm_source} // source from the last event
{event.first.browser} // browser from the first event
{event.first.categories} // categories from the first event
```

To iterate over all events, use a loop:

```
{for $index $event = event.all}
            {if $event.goal equal "cart"}
                  {formatdate $event.date "%Y-%m-%D to %H:%M"} added to the cart {$event.data.count}x {$event.data.product}.
              {end}
{else}
    <p>No events...</p>
{end}
```

Examples

Trigger message after add-to-cart:

```
You left an item: {event.first.data.product}
Price: {event.first.data.price}
Buy: {event.first.data.link}
```

Segmentation by geolocation:

```
A sale is happening in your city {event.first.geo_city}!
Source: {event.first.utm_source}
```

Mailing performance analysis:

```
Source: {event.last.utm_source}
Mailing: {event.last.utm_campaign}
Browser: {event.last.browser}
```

For use in scenarios with the Event Waiting node:

```
Hello, {user.name}!

You recently added a new item to your cart but didn’t complete the purchase.

Your item is waiting for you:
{event.last.product_name}
```

Event fields

Basic fields

Field	Type	Description	Example
`id`	**string**	**Unique event ID**	`{event.first.id}`
`type`	**string**	**Platform event type**	`{event.last.type}`
`date`	**DateTime**	**Event date and time**	`{event.first.date}`
`test`	**bool**	**Test event flag**	`{event.last.test}`

Geo data

Field	Type	Description	Example
`geo_country`	**string**	**Country (full name)**	`{event.first.geo_country}`
`geo_tld`	**string**	**Two-letter country code**	`{event.first.geo_tld}`
`geo_region`	**string**	**Region**	`{event.first.geo_region}`
`geo_city`	**string**	**City**	`{event.first.geo_city}`
`geo_address`	**string**	**Address**	`{event.first.geo_address}`
`geo_zip`	**string**	**Postal code**	`{event.first.geo_zip}`
`geo_time_zone`	**string**	**Time zone**	`{event.first.geo_time_zone}`
`geo_lat`	**float**	**Latitude**	`{event.first.geo_lat}`
`geo_lon`	**float**	**Longitude**	`{event.first.geo_lon}`

UTM tags

Field	Type	Description	Example
`utm_source`	**string**	**Traffic source**	`{event.first.utm_source}`
`utm_medium`	**string**	**Traffic medium**	`{event.first.utm_medium}`
`utm_campaign`	**string**	**Mailing name**	`{event.first.utm_campaign}`
`utm_content`	**string**	**Content identifier**	`{event.first.utm_content}`
`utm_term`	**string**	**Keyword**	`{event.first.utm_term}`
`utm_keyword`	**string**	**Alternative keyword**	`{event.first.utm_keyword}`

Pixel event (Pixel: `goal`)

Field	Type	Description	Example
`pixel_id`	**int**	**Pixel ID**	`{event.first.pixel_id}`
`goal`	**string**	**Goal name**	`{event.first.goal}`
`value`	**float**	**Goal value**	`{event.first.value}`
`data`	**object**	**Custom goal data**	`{event.first.data.order_id}`
`browser`	**string**	**User’s browser**	`{event.first.browser}`
`user_agent`	**string**	**Browser User-Agent**	`{event.first.user_agent}`

Field	Type	Description	Example
`accept_language`	**string**	**Browser language**	`{event.first.accept_language}`
`count_items`	**int**	**Number of items**	`{event.first.count_items}`
`categories`	**string[]**	**Array of categories**	`{event.first.categories}`
`send_message_id`	**string**	**Message send ID**	`{event.first.send_message_id}`

API call event (API: `api_call`)

Field	Type	Description	Example
`content`	**object**	**Data from API call**	`{event.first.content.order_id}`
`custom_data`	**object**	**Custom API data**	`{event.first.custom_data.field_name}`

Event types

Group	Type	Description
Pixel	`goal`	**Pixel/goal activation (including via API)**
API	`api_call`	**API call through a workflow or trigger start**

Note: All fields of different event types are available at the same level. For example, `{event.last.content}` refers to the `content` field of an API call event if that is the current event.

See the developer documentation for full technical details and limits. This helps you set up API data collection precisely and use it correctly in templates.

Loyalty programs

When inserting loyalty program variables via the insert dialog in the editor, a context menu opens where you can select a loyalty program and a points type (as well as the transaction count limit and transaction activation period for the last two variables):

Description	Variable
Member points balance	`{loyalty_program.<LOYALTY_PROGRAM_ID>.points.<POINTS_ID>.balance}`
Member active points balance	`{loyalty_program.<LOYALTY_PROGRAM_ID>.points.<POINTS_ID>.active_balance}`
Member inactive points balance	`{loyalty_program.<LOYALTY_PROGRAM_ID>.points.<POINTS_ID>.inactive_balance}`
Balance with currency name	`{loyalty_program.<LOYALTY_PROGRAM_ID>.points.<POINTS_ID>.balance_named}`
Active balance with currency name	`{loyalty_program.<LOYALTY_PROGRAM_ID>.points.<POINTS_ID>.active_balance_named}`
Inactive balance with currency name	`{loyalty_program.<LOYALTY_PROGRAM_ID>.points.<POINTS_ID>.inactive_balance_named}`
List of ближайших expirations (where x is the transaction count limit and y is the transaction activation period)	`{loyalty_expiration_points(<LOYALTY_PROGRAM_ID> "<POINTS_ID>" x y "desc")}`
List of nearest points activations (where x is the transaction count limit and y is the transaction activation period)	`{loyalty_activation_points(<LOYALTY_PROGRAM_ID> "<POINTS_ID>" x y "acs")}`

Emoji

You can also insert emoji into your message templates. Choose the emoji picker in the variables menu and click the emoji you want.

Profile Relations

General variable formula:

`{relation.[relation_short_name].[direct for direct or reverse for reverse].[property_name].[count – count, total – sum, top – top relations]}`

Count of direct relations with `profit` property:

```
{relation.managers_loyal.direct.profit.count}
```

Top reverse relations by `profit`. A loop is used to output multiple values:

```
{for $index $item = relation.managers_loyal.reverse.profit.top}
{$item.lead._fname} {$item.lead._lname} ({$item.value} profit)
{else} There is no ‘profit’ for you :(
{end}
```

Sum of `profit` across direct relations:

```
{relation.managers_loyal.direct.profit.total}
```

Additional Editor Functions

Price formatting

A number formatting function splits numbers into groups of three with a given delimiter.

```
Price 8.000 RUR
```

```
<p>Price {format json.price "."} RUR</p>
```

To insert, click `</>` in the editor and choose Output functions —> Formatting.

Date offset

The `adddate(date, year, month, day)` function returns a date after adding the specified interval.

```
<p>Event starts on {adddate(lead.notify_date 0 0 10)}</p>
```

`adddate()` can be used inside `format()`:

```
<p>Event starts on {format adddate(lead.notify_date 0 0 10) "%Y-%m-%D"}</p>
```

To insert, click `</>` —> Date & time —> Date offset.

You can format and shift the date at the same time using the following function:

```
{format adddate(datenow 0 0 3) "%Y-%m-%D"}
```

Date formatting

Date values differ by locale. Use date formatting for subscriber-friendly dates.

For example, to show the subscriber's date of birth as:

```
Date of birth 02.11.1998
```

Use:

```
<p>Date of birth {format lead._bdate "%D.%m.%Y"}</p>
```

You can format and shift the date at the same time using the following function:

```
{format adddate(datenow 0 0 3) "%Y-%m-%D"}
```

All date formatting parameters:

```
# years
"%y": "06"
"%Y": "2006"
"yy": "06"
"YYYY": "2006"

# months
"%m": "01"
"MM": "01"
"%B": "January"
"month": "January"
"%ru_month": "январь"

# days
"%d": "2"
"%D": "02"
"DD": "_2"
"%A": "Monday"
"weekday": "Monday"

# hours
"%H": "15"
"%I": "3"
"hh": "15"
"hours": "15"
"twelve_h": "3"

# mins
"%M": "04"
"mm": "04"

# seconds
"%S": "05"
"ss": "05"

# am/pm indicator
"am": "pm"
"AM": "PM"
"%p": "pm"

# timezone
"tzone": "-07"
"timezone": "-07:00"
"%Z": "Z0700"
```

To insert, click `</>` —> Date & time —> Formatted date.

INFO

If the template uses content from external JSON (`apicontent` variable or dynamic JSON content), the platform can convert RFC 3339 strings into date/time.

Formatting example:

Mode	Format
JSON	`"date_time": "2023-02-21T11:10:35.141Z"`
Template	`{format apicontent.date_time "%Y-%m-%d %H:%M:%S"}`
Preview	`2023-02-21 11:10:35`

Logical expressions in messages

Description

TIP

If you're using multiple logical operators in a single `if` expression, make sure to wrap each pair of conditions in parentheses — this is required for correct parsing.

Correct syntax example:

```
<html>
{if param1 and (param2 and (param3 and param4))} <!--true-->
if true
{else}
if false
{end}
</html>
```

Example that will cause a parsing error:

```
<html>
{if param1 and param2 and param3 and param4} <!--false-->
if true
{else}
if false
{end}
</html>
```

The core of highly personalized targeted messages are conditional expressions.

They are based on if → then → else logic and display different content depending on conditions.

```
{if lead.field in json.promo_actions1}
```

You can use data variables both in conditions and in output blocks. This significantly reduces message duplication for segmented audiences.

Example: setting special discounts for VIP customers of a specific shop, while all other recipients receive a default offer.

- `Congratulations! Your premium discount is 42%!` **— VIP customers**
- `Special offer! Get a 20% discount for your second purchase!` **— all others**

Two conditions inside a single message are enough.

1. **Check if the customer has the `"best_buy_type"` field and belongs to the required shop.**
2. **Check whether premium products were purchased.**

```
<!-- Condition 1: profile has "shop_name" and "best_buy_type" -->
{if lead.shop_name and lead.best_buy_type}

    <!-- Condition 2: shop_name equals "my_shop" -->
    {if lead.shop_name equal "my_shop"}

        <!-- Condition 3: best_buy_type contains "premium" -->
        {if lead.best_buy_type contains "premium"}
            <p>Congratulations! Your premium discount is 42%!</p>
        {else}
            <p>Special offer! Get a 20% discount for your second purchase!</p>
        {end}

    {else}
        <p>Special offer! Get a 20% discount for your second purchase!</p>
    {end}

{else}
    <p>Special offer! Get a 20% discount for your second purchase!</p>
{end}
```

Examples

Conditions

Simple condition

```
{if param1}
        if true
{else}
        if false
{end}
```

Complex condition

Example 1:

```
{if param1 in array["par1" "par2"] and ( 5 equal length("par 3") or false )}
        if true
{else}
        if false
{end}
```

Example 2 using `apicontent`:

```
{if apicontent.field_1 not_equal apicontent.field_2 and ( apicontent.field_1 or apicontent.field_2 )}
        if true
{else}
        if false
{end}
```

Example 3:

```
{if crypt("md5" lead._fname "salt") like "123b003aa39d110d1333b37af9889e10" and crypt("base58" lead.lottery_id) like json.crypted_winner}
        if true
{else}
        if false
{end}
```

Example 4:

```
{if iseven(lead.count) and iseven(lead.count2) equal false}
        if true
{else}
        if false
{end}
```

Check field and cancel sending if missing

```
{if lead.some_field}
some_field is not empty
{else}
{cancel}
{end}
```

Loops

Simple loop

```
{for array[1 "2" 3.3]}
        {.}
{else}
        if empty array
{end}
```

Loop with variables

```
{for $index, $el = array[1 "2" 3.3]}
        {$index}: {$el}
{else}
        if empty array
{end}
```

Value matching conditions

Equal

```
{if param1 equal param2}
        if true
{else}
        if false
{end}
```

Not equal

```
{if param1 not_equal param2}
        if true
{else}
        if false
{end}
```

String is equal (case insensitive)

```
{if param1 like param2}
        if true
{else}
        if false
{end}
```

String is not equal (case insensitive)

```
{if param1 not_like param2}
        if true
{else}
        if false
{end}
```

String contains (case insensitive)

```
{if param1 contains param2}
        if true
{else}
        if false
{end}
```

String doesn’t contain (case insensitive)

```
{if param1 not_contains param2}
        if true
{else}
        if false
{end}
```

Numbers and dates comparison

More than number/date

```
{if 0 gt 1}
        if true
{else}
        if false
{end}
```

More than or equal to number/date

```
{if 0 gte 1}
        if true
{else}
        if false
{end}
```

Less than number/date

```
{if 0 lt 1}
        if true
{else}
        if false
{end}
```

Less than or equal to number/date

```
{if 0 lte 1}
        if true
{else}
        if false
{end}
```

Arrays and tags fields

Element is in array

```
{if param1 in array["par1" "par2"]}
        if true
{else}
        if false
{end}
```

Element is in tags field

```
{if param1 in lead.TagsField}
        if true
{else}
        if false
{end}
```

Element is not in array

```
{if param1 not_in array["par1" "par2"]}
        if true
{else}
        if false
{end}
```

Element is not in tags field

```
{if param1 not_in lead.TagsField}
        if true
{else}
        if false
{end}
```

All elements are in first array

```
{if array["par1" "par2"] all_in array["par1" "par2"]}
        if true
{else}
        if false
{end}
```

Not all elements are in first array

```
{if array["par1" "par2"] not_all_in array["par1" "par2"]}
        if true
{else}
        if false
{end}
```

At least one match in the arrays

```
{if array["par1" "par2"] any_of array["par1" "par2"]}
        if true
{else}
        if false
{end}
```

No matches in the arrays

```
{if array["par1" "par2"] not_any_of array["par1" "par2"]}
        if true
{else}
        if false
{end}
```

Loops in messages

The message supports loops for JSON variables of type `"list" []` and `"object" {}`, as well as access to an object element by key {json.object.name}.

If additional profile fields are inserted as variables into the template, fields of the “Array of Objects” and “Tag” (list of strings) types can also be used in loops.

Nested loops are also possible.

Variables example

```
{
  "Addresses": ["Barclay street, 244 building", "Green street, 555 building"],
  "Information": {
    "LJ": [
      { "name": "Optimyss X1", "price": "$10" },
      { "name": "Optimyss X2", "price": "$20" }
    ],
    "Samsung": [{ "name": "Galaktiko 17", "price": "$50" }]
  }
}
```

Loops example

```
<!-- Loop by object elements -->
{for $vendor $models = json.Information}
    <p>{$vendor}</p>

    <ol>
        <!-- Nested loop -->
        {for $index $model = $models}

            <!-- Call elements by key -->
            <li>{$model.name} {$model.price}</li>

        {else}
            <p>Empty array!</p>
        {end}
    </ol>

{else}
    <p>Empty array!</p>
{end}

<h5>Addresses:</h5>
<ol>
<!-- Simple loop by strings list -->
{for $index $address = json.Addresses}
    <li>{$address}</li>
{else}
    <p>Empty array!</p>
{end}
</ol>
```

Result

```
<h5>New:</h5>

    <!-- Loop by object elements -->
    <p>LJ<p>
    <ol>
        <!-- Nested loop. Call elements by key -->
        <li>Optimyss X1 $10</li>
        <li>Optimyss X2 $20</li>
    <ol>

    <!-- Loop by object elements -->
    <p>Samsyng<p>
    <ol>
        <!-- Nested loop. Call elements by key -->
        <li>Galaktiko 17 $50</li>
    <ol>

<h5>Addresses:</h5>
    <ol>
        <!-- Simple loop by strings list -->
        <li>Barclay street, 244 building</li>
        <li>Green street, 555 building</li>
    </ol>
```

Market variables in templates

Variables

There are four variables for Market:

- `products` — **function selects products**

```
{products("<filter>" 10 "manufacturer_name" "desc")}
```

- `sku` — **function selects SKUs**

```
{sku("<filter>" 10 "manufacturer_name" "desc")}
```

- `orders` — **the function selects orders from the list of profile orders**

```
{orders("<filter>" 10 "status" "asc")}
```

- `order` — **the function selects one order from the list of profile orders**

```
{order("<filter>")}
```

Function arguments

`("<filter>" 10 "manufacturer_name" "desc")`

Argument	Required	Description
`"<filter>"`	**Yes**	**The filter is a string of conditions. Allows you to select products, SKUs, orders, or order lines that meet specified conditions.** **For example, you can select products from the same manufacturer or orders that exceed a specific cost.** **See below for details.**
`10`	**Yes**	**Limit. How many array elements to iterate over. The maximum value is 30.** **For example, if you specify `10`, then the filter will select the first ten elements from the array.**
`"manufacturer_name"`	**No**	**Sort objects by the specified field.** **By default — no sort.**
`"desc"`	**No**	**Sort direction.** `asc` **— ascending (by default)** `desc` **— descending**

Filter

The filter consists of fields and operators. Each object in the market has fields. For example, products and SKUs have a `name` field (product name). Orders have a `total_price` field (total cost of the order). In a filter, you specify a field and a value that it should match. If the field of the object matches the value, this object will be selected and its data can be substituted into the template.

Filter example

Let's add the "Orders" variable to the template:

```
{orders ("order.total_price == '200 USD' && (order.custom_fields.field == 'lead._region')" 10 "id" "desc")}
```

The bold part of the variable is the filter. Let's consider it in parts:

- `order.total_price == '200 USD'` **: the condition will select all orders with a cost equal to $200.**
- `order.custom_fields.field == 'lead._region'` **- the condition will select orders with the additional 'field'. You can specify any additional (custom) field instead of `"field"`. The specified equality means that the value of the `"field"` must match the value of the region in the customer's profile (`"lead._region"`).**
- `&&` **is an operator. It indicates that both conditions must be met.**
- **The number 10 indicates the number of output objects, `"id"` indicates the filter field, and `"desc"` indicates the descending sort.**

Below are the fields and operators available in the filter.

Operators and functions in a filter

Operator	Description	Example	What will be selected
>	**greater**	`order.delivery_cost > '10 USD'`	**Orders with shipping costs over $10**
>=	**greater or equal**	`order.delivery_cost >= '10 USD'`	**Orders with a shipping cost of $10 or more**
<	**less**	`sku.count < '10'`	**The number of SKUs for sale is less than 10**
<=	**less or equal**	`sku.count <= '10'`	**The number of SKUs for sale is 10 or more**

`==`	**equals**	`order.status == '1'`
		
		`order.custom_fields('date') == '2024-12-25'`
			Orders with “new” status
			
			Orders with date December 25, 2024

!=	**not equal**	`product.manufacturer_name != 'Apple'`	**Products not manufactured by Apple**
&&	**AND (both conditions match)**	`product.manufacturer_name == 'Apple' && product.price < '200 USD'`	**Apple products cost more than $200**
||	**OR (one of the conditions matches)**	`sku.manufacturer_name == 'MSI' || sku.manufacturer_name == 'Acer'`	**SKU with manufacturer MSI or Acer**
`in('<field_name>', 'value1', 'value2'...'valueN')`	**The field value (** `<field_name>` **) matches one of the specified values (‘** `value1` **’, ‘** `value2` **’)**	`in(product.external_id,'1','2','3')`	**Products with ID equal to 1, 2 or 3**
`contains('<field_name>', 'value1', 'value2'...'valueN')`	**One of the array elements matches one of the specified values.** `<field_name>` **— array** `'value1', 'value2'` **— values to match.** **Used if the field is an array, i.e. contains several elements (for example: categories, images, additional fields).**	`contains(product.categories,'smartphone')`	**Products included in the category “smartphone”**
`len('<field_name>', <operator>, <number> )`	**Checks the length of an array. Compares the number of elements in the array with the specified number.** `<field_name>` **— array field (e.g.,** `product.pictures` **)** `<operator>` **— comparison operator (** `>` **,** `>=` **,** `<` **,** `<=` **,** `==` **,** `!=` **)** `<number>` **— integer for comparison.**	`len(product.pictures, !=, 0)`	**Products that have a number of pictures not equal to zero (i.e., have at least one picture)**

Working with custom fields

Two syntaxes are available for filtering by custom fields:

Universal syntax (recommended)

```
(product.custom_fields('external_id'))
```

INFO

We recommend using the universal syntax `product.custom_fields('external_id')` **because it supports all types of identifiers — both numeric and string, including special characters. This option guarantees correct filtering regardless of your custom field’s** `external_id` **format.**

Examples:

```
{for $product in products("product.custom_fields('123') == 'red'")}
{for $product in products("in(product.custom_fields('color'), 'red', 'blue')")}
{for $product in products("sku.custom_fields('45') > '100'")}
```

String syntax (limited)

```
(product.custom_fields.field_name)
```

INFO

The old syntax `product.custom_fields.field_name` **can only be used if the field’s** `external_id` **consists exclusively of alphabetic characters and does not contain numbers, hyphens, or other special characters.**

Example:

```
{for $product in products("product.custom_fields.brand == 'Apple'")}
```

List of available fields in filters

Products

Prefix — `product.`

Field	Description
`product.external_id`	**Product ID**
`product.name`	**Product name**
`product.manufacturer_name`	**Manufacturer's name**
`product.url`	**URL of the product page on the store website**
`product.pictures`	**Product Image URL**
`product.count`	**Number of products available for sale**
`product.is_available`	**Is the product in stock at the store?**
`product.delivery`	**Possibility of delivery (for all regions to which the store delivers)**
`product.barcode`	**List of product barcodes from the manufacturer**
`product.categories`	**Product category ID**
`product.custom_fields.<field_external_id>`	**Custom product field ID. For numeric IDs use** `custom_fields('123')`

SKU

Prefix — `sku.`

Field	Description
`sku.external_id`	**SKU ID**
`sku.name`	**SKU name**
`sku.manufacturer_name`	**Manufacturer's name**
`sku.url`	**URL of the SKU page on the store website**
`sku.pictures`	**SKU image URL**
`sku.count`	**Number of SKUs available for sale**
`sku.is_available`	**Is the SKU in stock at the store?**
`sku.delivery`	**Possibility of delivery (for all regions to which the store delivers)**
`sku.barcode`	**List of SKU barcodes from the manufacturer**
`sku.categories`	**SKU category ID**
`sku.custom_fields.<field_external_id>`	**Custom sku field ID. For numeric IDs use** `custom_fields('123')`

Orders

Prefix — `order.`

Field	Description
`order.external_id`	**Order ID**
`order.status`	**Order status ID** **1 - new** **2 - delivered** **3 - paid** **4 - cancelled** **5 - partial refund**
`order.delivery_cost`	**Shipping cost**
`order.total_price`	**Total cost of the order.** **It consists of the number of each item in the order and the cost of delivery.**
`order.custom_fields.<external_id поля>`	**Custom order field ID**
`order.real_create_time`	**Order creation time**
`order.real_update_time`	**Order update time**
`order.region_external_id`	**Region ID**

Sorting by Custom Fields

Sometimes you need to sort objects by their custom field values (the `custom_fields` **object).**
Custom fields are stored in the database as a JSON string. Therefore, to access a field inside the JSON for sorting, you can use the `->` **operator.**

Example of accessing a key:

```
custom_fields -> 'priority'
```

Usage in a sort parameter function (using `products` **as an example):**

```
{for $index $el = products("contains(product.categories,'smartphone')" 3 "custom_fields -> 'priority'" "desc")}
{$el.name} — {$el.price}
{else}
if empty array
{end}
```

In this example, the product list is first filtered by the `smartphone` category, then sorted by the value of the `priority` key inside `custom_fields` in descending order, and the first 3 results are displayed.

INFO

Important

- **The `->` operator is used only in the sort parameter to access a JSON key.**
- **For filtering by custom fields, use the dot notation from the "Filter" section, for example: `order.custom_fields.field == 'value'`.**
- **The `->` operator is not a market operator — it is a database operator (PostgreSQL) used for the query, so this sorting method should be considered a workaround.**

Use cases

Any variable in the market is an array. For example, `products` contains an array of all products uploaded to the platform, and `orders` contains an array of orders for one profile. To iterate over an array, use loops.

Searching the array of products (products)

Outputs 5 Apple products in an email:

```
You might be interested in these products:
{for $index $el = products("product.manufacturer_name == 'Apple'" 5 "external_id" "desc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

7 cheapest smartphones:

```
The best smartphone prices for you and your loved ones:
{for $index $el = products("product.categories == 'smartphone'" 7 "price" "asc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

Goods with delivery:

```
Delivered to your door:
{for $index $el = products("product.delivery == true" 10 "name" "desc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

Search SKU (sku) array

Smartphones under 20000:

```
Smartphones you will like:
{for $index $el = sku("contains(sku.categories,'smartphone') && sku.price < '20000 RUB'" 3 "price" "asc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

IPhone with more than 128GB of storage:

```
Enough memory for everything:
{for $index $el = sku("contains(sku.name,'IPhone') && sku.custom_fields.memory >= '128'" 10 "name" "desc")}
{$el.name} - {$el.price}
{else}
if empty array
{end}
```

Searching the array of all orders (orders)

Orders at pickup points:

```
Orders are waiting for you at the pickup point:
{for $index $el = orders("order.status == '2'" 10 "external_id" "desc")}
Order # {$el.external_id}
Due: {$el.total_price}
{else}
if empty array
{end}
```

List of new orders:

```
New orders in the last week:
{for $index $el = orders("order.status == '1'" 20 "external_id" "desc")}
Order # {$el.external_id}
Due: {$el.total_price}
{else}
if empty array
{end}
```

Search for items in order with id = ‘0001’ (order + lines)

```
{for $index $el = order("order.external_id == '0001'")}
 {if $index equal "lines"}
  {for $i $line = $el}
    {$line.name} position name: {$line.name}
    {$line.price} Item price: {$line.final_price_of_line}
  {end}
 {end}

 {if $index equal "delivery_cost"}
   {$el} {$delivery_cost} {$el}
 {end}
 {if $index equal "total_price"}
   To pay: {$el}
 {end}

{else}
if empty array
{end}
```

List of available fields for substitution in the template

products/sku

Substitution field	Description
external_id	**Product/SKU ID**
name	**Product/SKU name**
manufacturer_name	**Manufacturer’s name**
url	**URL of the product page on the store website**
pictures	**List of URL links to product images (array)**
count	**Quantity of products/SKU available for sale**
available	**Is the product/sku in stock at the store?**
delivery	**Possibility of delivery (for all regions to which the store delivers)**
barcode	**List of product barcodes from the manufacturer (array)**
categories	**List of product/SKU category IDs (array)**
custom_fields.<external_id поля>	**Custom field ID. For numeric IDs and special characters use custom_fields(‘external_id’) syntax.**
price	**Current product price**
old_price	**Previous product price**
regional_data.<region_external_id>.count	**Quantity of products/SKU available for sale in the specified region.** `regional_data` **is product/sku data that varies by region.** `<region_external_id>` **is region ID.**
regional_data.<region_external_id>.delivery	**Possibility of delivery to the specified region**
regional_data.<region_external_id>.available	**Is the product/SKU in stock in the specified region?**
regional_data.<region_external_id>.price	**Product/SKU price in the specified region**

orders / order

Substitution field	Description
external_id	**Order ID**
status	**Order status ID**
delivery_cost	**Shipping cost**
total_price	**Total cost of the order.** **It consists of the cost of each item in the order and the cost of delivery.**
custom_fields.<external_id поля>	**Custom field ID. For numeric IDs and special characters use custom_fields('external_id') syntax.**
create_time	**Order creation time**
update_time	**Order update time**
region.external_id	**Region ID**
lines	**Order line. Only for the `order` function.**

lines

Substitution field	Description
id	**Line ID**
external_id	**Order line additional ID**
custom_status_external_id	**Order line status ID**
base_price_per_item	**Base price**
final_price_of_line	**Final price**
count	**Quantity of products or skus in the line**
custom_fields.<field_external_id>	**ID for the custom field of the order line. For numeric IDs and special characters use custom_fields('external_id') syntax.**
product_external_id	**Line product ID**
name	**Line name**
manufacturer_name	**Manufacturer's name of the line product**
url	**URL of the page of the line product**
pictures	**List of URL links to images of the line product**
balance	**Stock quantity**
available	**Is the line product in stock at the store**
delivery	**Possibility of delivery of the line product (array)**
barcodes	**List of barcodes of the line product (array)**
categories	**List of category IDs of the line product**

INFO

Custom field override priority:
(defines which field values are applied first)

SKU > Product > Order-item

Using the JSONPath functionality

The Altcraft platform supports the functionality of the JSONPath query language when working with JSON content. JSONPath syntax allows you to quickly and easily select objects within code and use them message templates. It can be used when working with API content and with external JSON content.

API content allows you to change the content of the trigger message each time it is sent. The required variable can be specified in the trigger API request. You can learn more about how to run triggered mailings and how to work with API content in messages here.

To use external JSON content, go to the Options section when creating or editing a template and provide a link to the dynamic JSON content source. The platform will automatically determine the fields of the external JSON file. More information about using JSON when working with message templates can be found here.

The JSONPath variable on the Altcraft platform is specified as follows:

`{jsonpath(json.<root_object> "<path_to_object>" "<output_format>")}`

Function arguments

Argument	Description
Root object	**Accessing a JSON object that contains the desired parameter or object**
Path to object	**The path to the desired object can consist of segments separated by dots, logical operators, filters or functions**
Output format	**The final variable can be represented in two versions: item or array. `Item` will display the first result found for the request, and if there is no result, it will return an empty string. `Array` returns an array of all results found. This mode always outputs the result as an array, even if there is only one or no result.**

Usage examples

Let's look at examples of using JSONPath. Suppose we have a JSON file with the following content:

```
{
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "images": ["AbeLincoln.png","Churchill.jpg","Thatcher.png"]
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "images": ["MenAtArms.png","Gentlemen.jpg","Officers.png"]
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99,
        "images": ["Ahab.png","Ishmael.jpg","whale.png"]

      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ]
  }
}
```

With JSONPath functionality, a small expression can return an array containing all the authors of the books:

`{jsonpath(json.store "$.book[*].author" "array")}`

Another expression will return one object, the author of the first book:

`{jsonpath(json.store "$.book[*].author" "item")}`

You can build a more complex function, for example, in order to find out the name of a book whose price is higher than 10, we will compose the following code:

`{jsonpath(json.store "$.book[?(@.price > 10)].title" "item")}`

To find out the arithmetic average of the price of all books, we can use this function:

`{jsonpath(json.store "$.book[*].price.avg()" "item")}`

JSONPath variables can also be run in loops:

```
{for $index $imgs = jsonpath(json.store "$.book[*].images" "array")}
    <li>{$imgs}</li>
{else}
    <p>Empty array!</p>
{end}
```

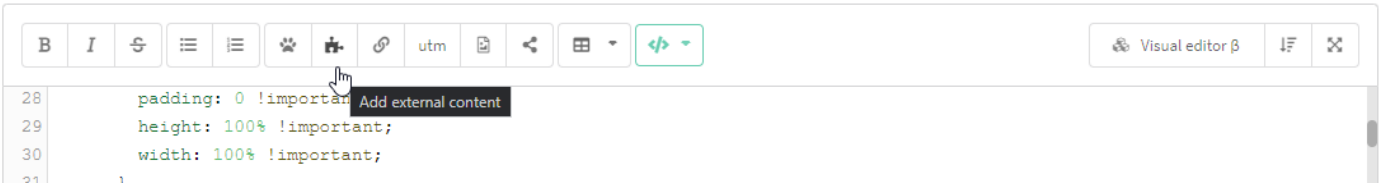
The full functionality of JSONPath on the Altcraft Marketing platform is listed in the table below:

Argument	Description	Use example	Result
`$`	Root element	`{jsonpath(json.store "$.book[0].title" "item")}`	Sayings of the Century
`@`	Current element	`{jsonpath(json.store "$.book[?(@.price > 10)].title" "item")}`	Sword of Honour
`..`	Recursive descent	`{jsonpath(json.store "$..title" "array")}`	[Sayings of the Century Sword of Honour Moby Dick The Lord of the Rings]
`*`	Match all objects	`{jsonPath(json.store "$.book[*].price" "array")}`	[8.95 12.99 8.99 22.99]
`?()`	Filter	`{jsonpath(json.store "$.book[?(@.category == 'reference')].title" "item")}`	Sayings of the Century
Boolean operator for filtering	Description	Use example	Result
`==`	Equal to	`{jsonpath(json.store "$.book[?(@.category == 'reference')].title" "item")}`	Sayings of the Century
`!=`	Not equal to	`{jsonpath(json.store "$.book[?(@.category != 'reference')].title" "array")}`	[Sword of Honour Moby Dick The Lord of the Rings]
`<`	Less than	`{jsonpath(json.store "$.book[?(@.price < 10)].title" "array")}`	[Sayings of the Century Moby Dick]
`>`	Greater than	`{jsonpath(json.store "$.book[?(@.price > 10)].title" "array")}`	[Sword of Honour The Lord of the Rings]
`<=`	Less than or equal to	`{jsonpath(json.store "$.book[?(@.price >= 12.99)].title" "array")}`	[Sword of Honour The Lord of the Rings]
`>=`	Greater than or equal to	`{jsonpath(json.store "$.book[?(@.price <= 12.99)].title" "array")}`	[Sayings of the Century Sword of Honour Moby Dick]
Function	Description	Use example	Result
`max`	Maximum value in array	`{jsonPath(json.store "$.book[*].price.max()" "item")}`	22.99
`min`	Minimum value in array	`{jsonPath(json.store "$.book[*].price.min()" "item")}`	8.95
`first`	First array element	`{jsonPath(json.store "$.book[*].title.first()" "item")}`	Sayings of the Century
`last`	Last array element	`{jsonPath(json.store "$.book[*].title.last()" "item")}`	The Lord of the Rings
`sum`	Sum of numbers in the array	`{jsonPath(json.store "$.book[*].price.sum()" "item")}`	53.92
`avg`	Average of the numbers in the array	`{jsonPath(json.store "$.book[*].price.avg()" "item")}`	13.48
`length`	Number of elements in the array	`{jsonPath(json.store "$.book.length()" "item")}`	4

Dynamic HTML content

External HTML content is one of the ways to create dynamic messages and automate your job. Using it, you can add into templates data, generated on a remote server exactly in the mailing moment.

To add the link to dynamic HTML content, press “Add external content” button in the message editor toolbar:



For example, if the server generates latest news, the message will be automatically maintained up to date.

You can use several links to HTML content in the message.

```
{[http://www.example.news?index=1&news_group=movies]}

{[http://www.example.news?index=2&news_group=movies]}
```

You can include different variables into dynamic HTML content requests. It can be for example profile data fields: subscriber’s name `{lead._fname}` or a containing database name `{listname}`. Thus instead of a static `&news_group=movies` you can use a request with a custom profile field variable: `&news_group={lead.NewsGroup}`.

Each time such message is sent it will request HTML content from the server and generate it for different recipients depending on their personal preferences.

Dynamic JSON content

Dynamic JSON content is good way to create dynamic messages and automate your work. It allows you to add information, generated exactly at the moment messages are sent.

You can include variables into JSON content queries, for example profile data fields like `{lead._fname}`.

Set the source of your external JSON content in message options tab.

> CAUTION
>
> **By default, the maximum size of dynamic JSON is 1 MB. In the on-premises version, this limit is set in the configuration file using the `MAX_REMOTE_CONTENT_BYTES` parameter.**

JSON data looks like this:

```
{
    "number": 10,
    "text": "Hello!",
    "list": ["One", "Two", "Three"],
    "object": {"model": "x17", "price": "$99"},
    "mixed": [
        {"city": "Los Gamburger", "street": "Florence Avenue"},
        {"city": "St. Potato", "street": "Glendale Boulevard"}
    ]
}
```

To add dynamic JSON content to your message template click the green `</>` variables button and select External JSON.

Thus JSON content is inserted into a message as a special variable type* `{json.variable}:`

Two data types: “list” and “object” are available.**

- **Objects are inserted as a one-value variable, like this: `{json.object.key_name}`**
- **Lists are JSON arrays — they can be inserted with the help of loops, like this one:**

```
{for $index $products= json.products}
    <li>{$products}</li>
{else}
    <p>Check for new products next week!</p>
{end}
```

Content from SQL database in templates

Description

You can use dynamic content from external data tables, which can be stored on MySQL, PostgreSQL, MSSQL, Oracle, Hive and ClickHouse database servers.

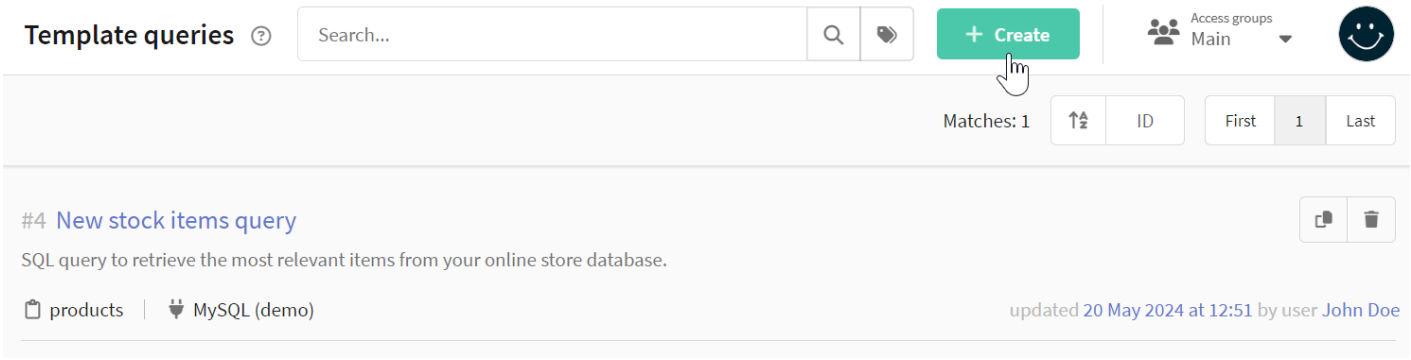
Connecting to SQL server

To connect to your database you would need an account with proper access privileges. The process depends on your version of Altcraft Marketing:

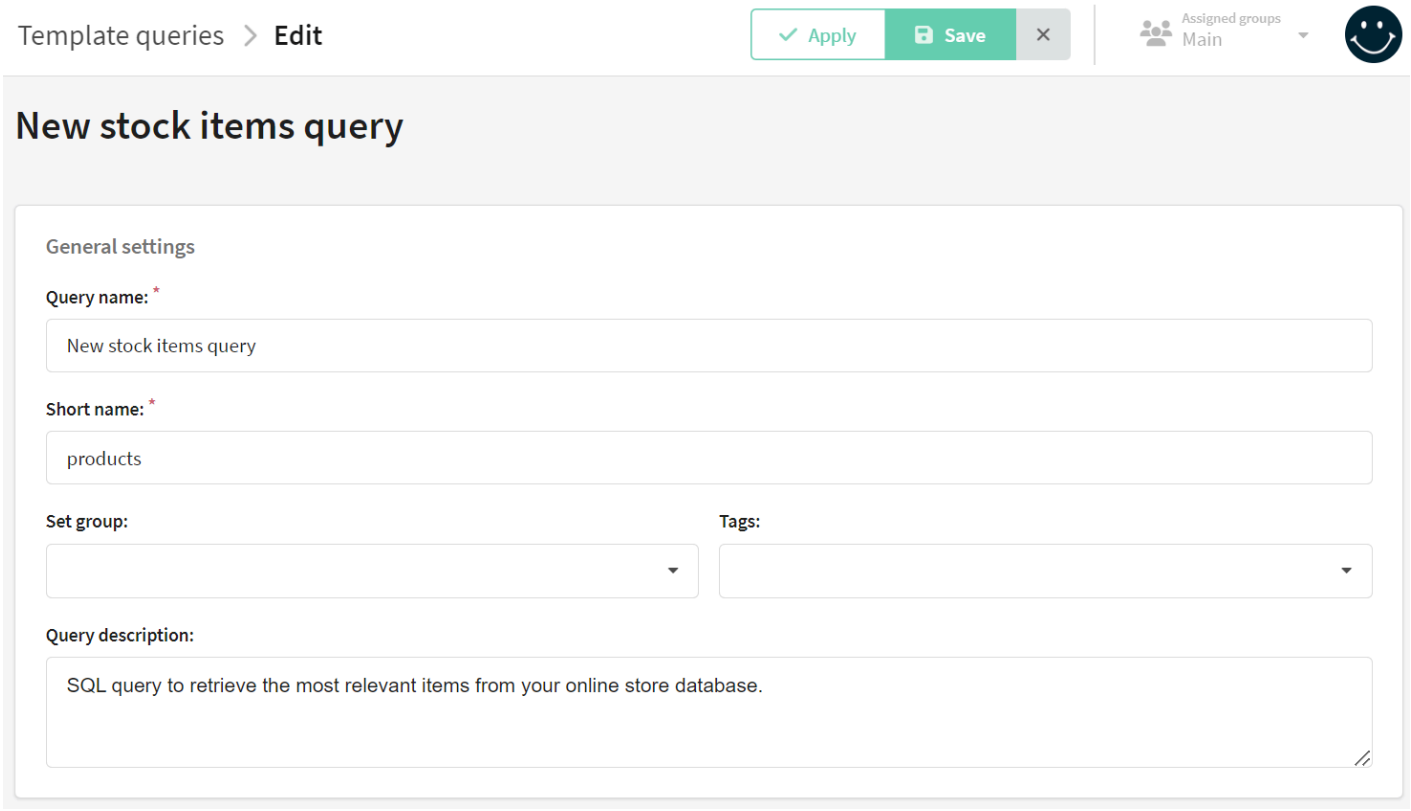
- **If you are a cloud account user — send access credentials to our support — support@altcraft.com. You can also contact this address for any questions related to connecting and configuring external services and databases.**
- **If you have an On-premise installation, platform administrator should create a database connector in Administrative panel — and assign it to your platform account.**

Creating query to external data table

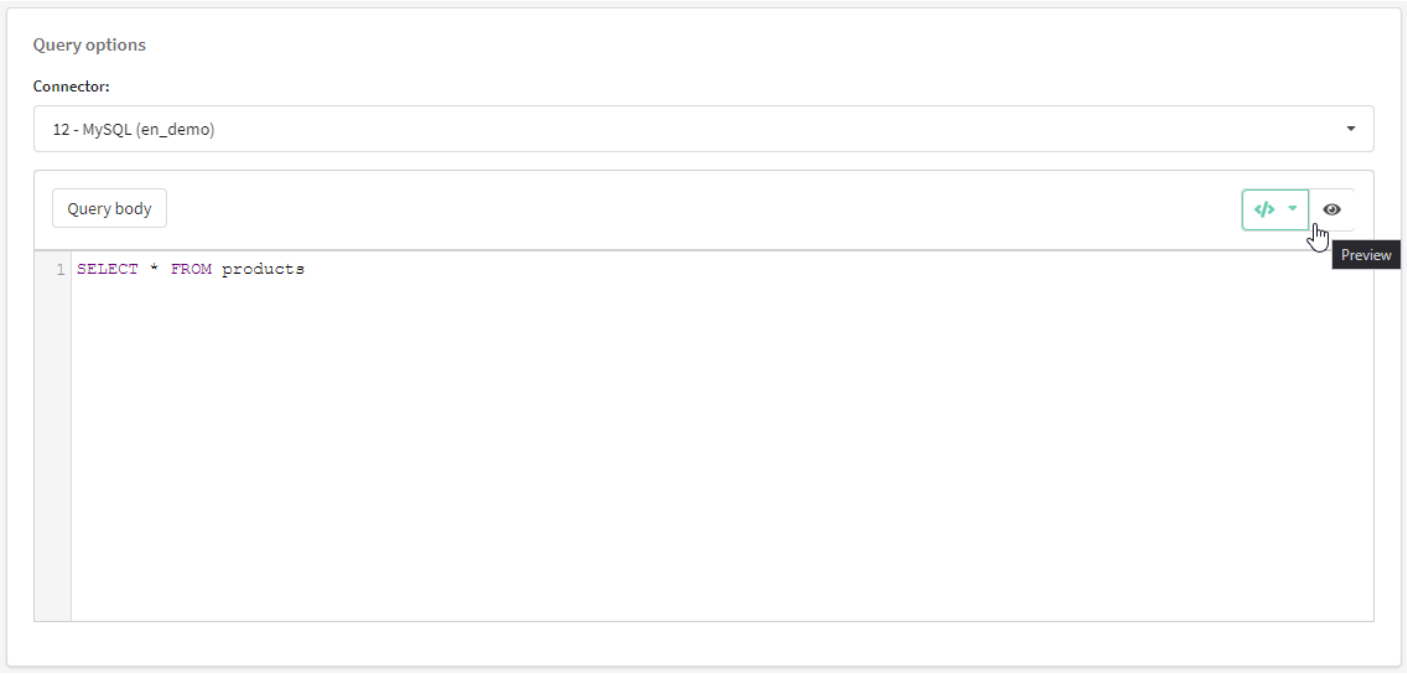
To create a query, go to Data —> Template queries and create a new query:



In general settings, you need to set Query name and Short name for using in the templates editor. If necessary, you can add a description, set groups to restrict other users from accessing the request, and attach tags for quick search.



Select the previously created database connector and enter query body depending on SQL management system you have. When done - you can preview the result:



TIP

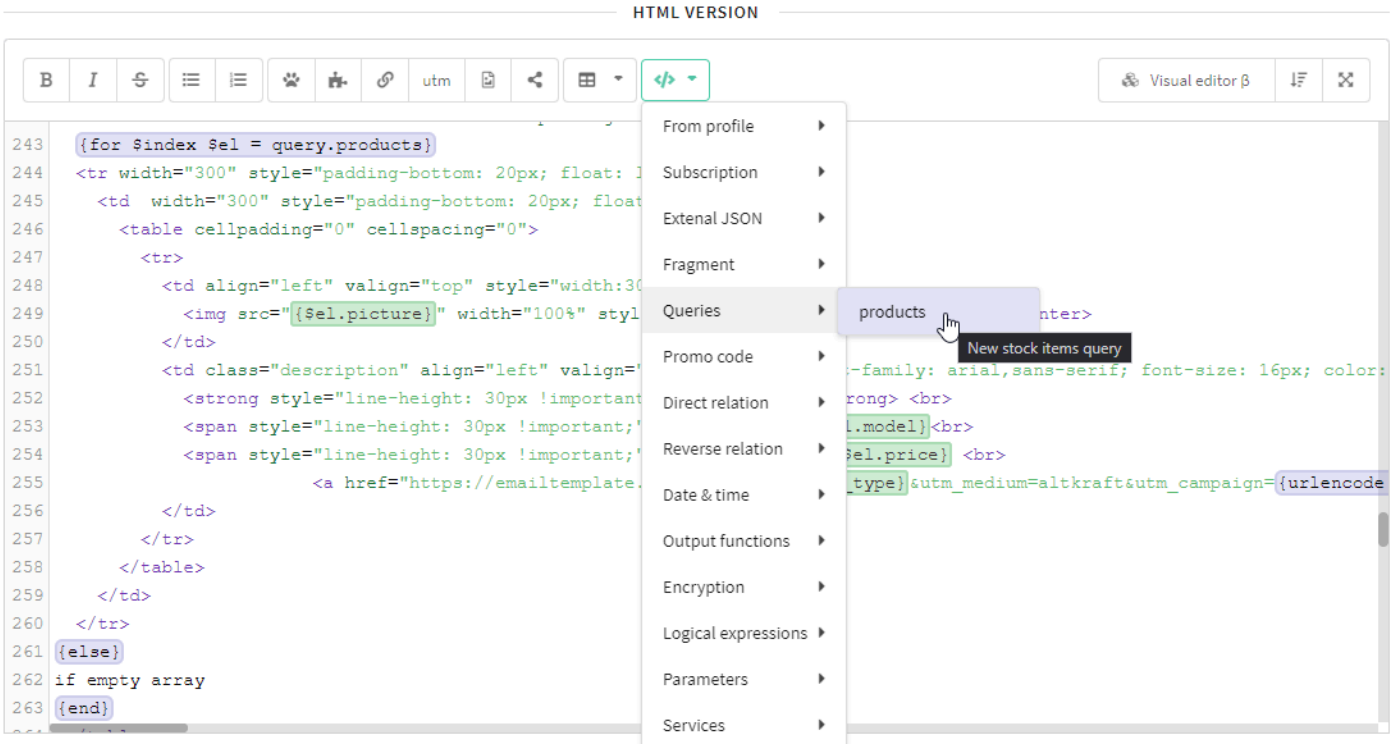
Also, you can create the query by selecting Data tables in the Data menu. Select Save as template query.

Using SQL query in message templates

In the message template queries are inserted like this: `query.your_query_name`**. To insert query contents into a template use variables menu (** `</>` **).**

TIP

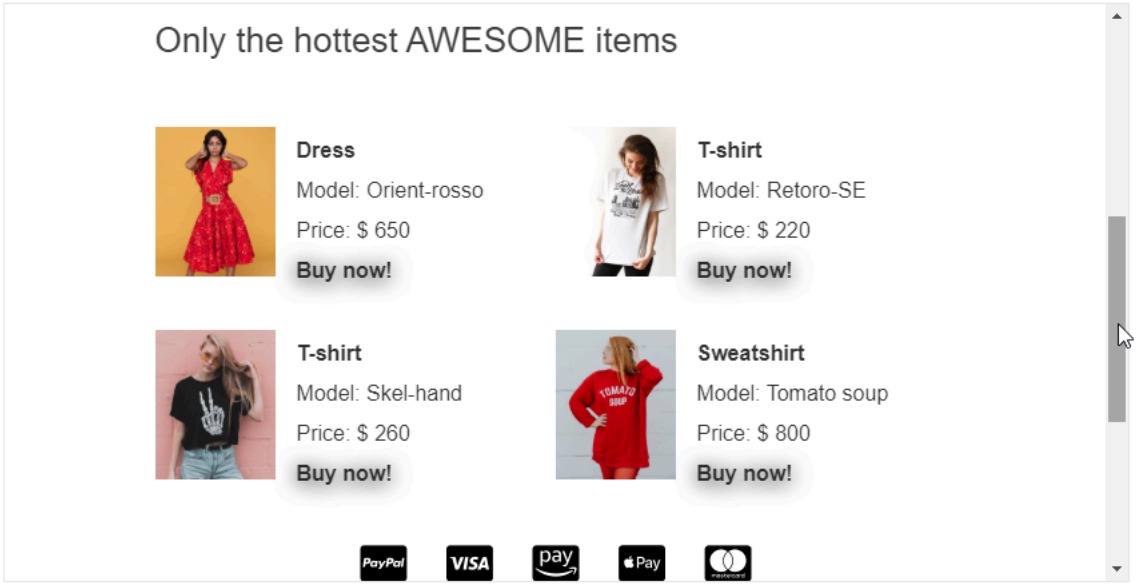
You can also include queries into [template loops] and [logical expressions].



Then you can style your query to be displayed properly:

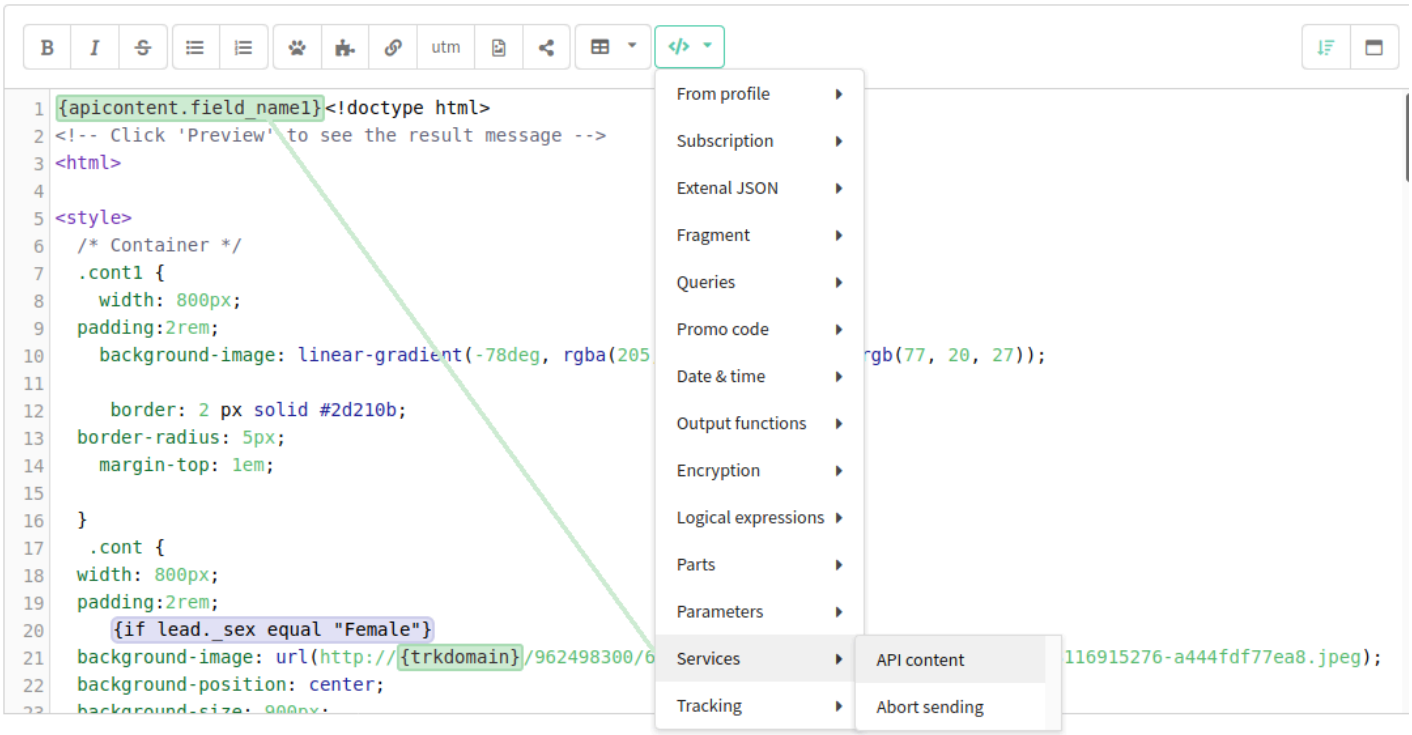
```
{for $index $el = query.products}
  <tr width="300" style="padding-bottom: 20px; float: left">
    <td  width="300" style="padding-bottom: 20px; float: left">
      <table cellpadding="0" cellspacing="0" class="doc-table">
        <tr>
          <td align="left" valign="top" style="width:30%;"><center>
            <img class="img" src="{$el.picture}" width="100%" style="display: block; border: 0;" /></center>
          </td>
          <td class="description" align="left" valign="middle" style="font-family: arial,sans-serif; font-size: 16px; color: #333; padding:
0 16px;">
            <strong style="line-height: 30px !important;">{$el.type} </strong> <br>
            <span style="line-height: 30px !important;">Model: </span>{$el.model}<br>
            <span style="line-height: 30px !important;">Price: </span>$ {$el.price} <br>
                        <a href="https://example.com style="display: block; text-decoration: none; font-weight: bold; margin-top: 15px;
text-shadow: 0 2px 15px #000000; color: #313131" name="link1" >Buy now!</a>
          </td>
        </tr>
      </table>
    </td>
  </tr>
{else}
if empty array
{end}
```

Subject: F - for fresh awesome offers



Dynamic API content

When sending messages using the [API Call Trigger](), starting from version 1.1, it is possible to transmit data for the message along with the API request.



Using `apicontent`, one message can be automatically personalized for each client. Using `apicontent`, you can transfer information about completed transactions, purchases, customer history, etc. This information is transmitted as a json-object.

An example of what an object for apicontent might look like:

```
{
  "product_id": 11,
  "product_name": "Graphic design: a beginner course",
  "product_price": 1000
 }
```

How to transfer API content to the platform

API request

Data can be transferred if you run a trigger mailing or scenario using an API request. In the request, the data is passed to the `content` field. Examples of requests in which the content field is available:

- **Trigger launch**
- **Profile Import + trigger mailing launch**
- **Bulk trigger launch**
- **Bulk profiles import + trigger mailing launch**
- **Engage profile in scenario**
- **Import and engage profile in scenario**
- **Batch import and engage profiles in a scenario**

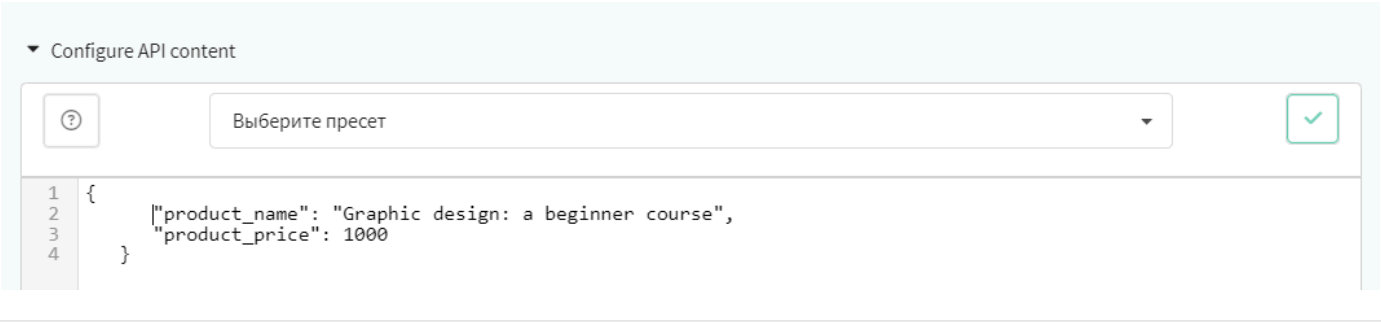
Next, this data can be inserted into the template using the `apicontent` variable:

HTML TEXT

```
<table align="center" role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%" style="margin: auto">
<tr>
<td style="background-color: #ffffff;">
<table role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%">
<tr>
<td style="padding: 20px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20px; color: #555555">
<h2 style="margin: 0 0 10px 0; font-family: sans-serif; font-size: 28px; color: #6778cd; font-weight: 800; line-height:
<h2 style="font-size: 28px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0 10px">Thank you for re
<p>Thank you for choosing our course:</p>
<ul>
    <li>{apicontent.product_name}</li>
    <li>{apicontent.product_price}</li>
</ul>
</td>
<tr>
<td style="padding: 0px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20px; color: #555555;">
<h2 style="font-size: 15px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0 10px">We appriciate yo
   <p>Thank you and good luck!</p>
</td>
</tr>
</table>
```

TIP

The preview of the message with `apicontent` is available only after configuring the section Configure API content. To do this, go to the preview section and paste the object containing the `apicontent` into the editor. Click on the check mark to apply the changes.



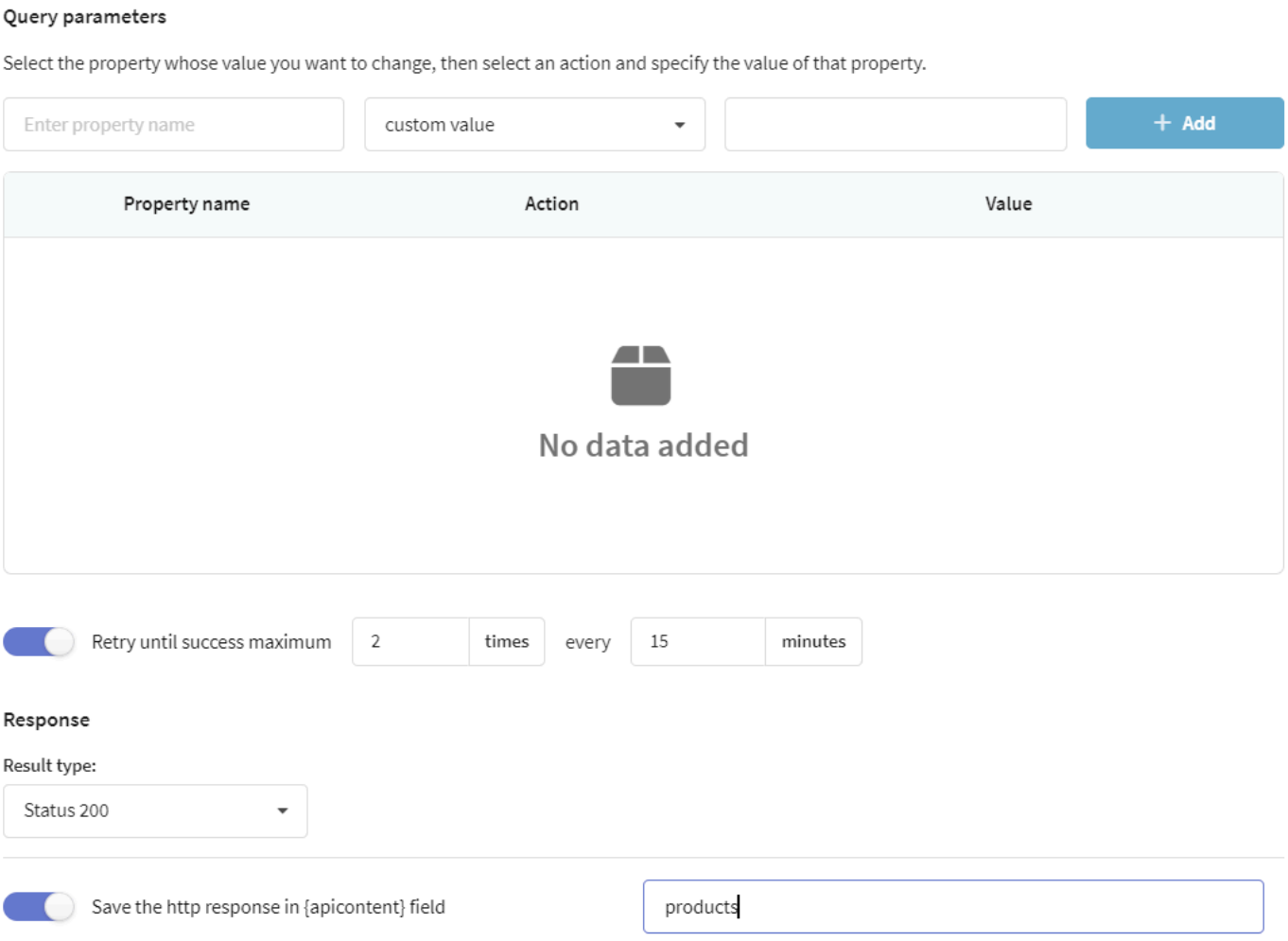
“API Request” element in the scenario

In the API Request scenario element, you can get data from an external source using a request and save the http response to the `apicontent` variable. The information from the request can later be used in the scenario.

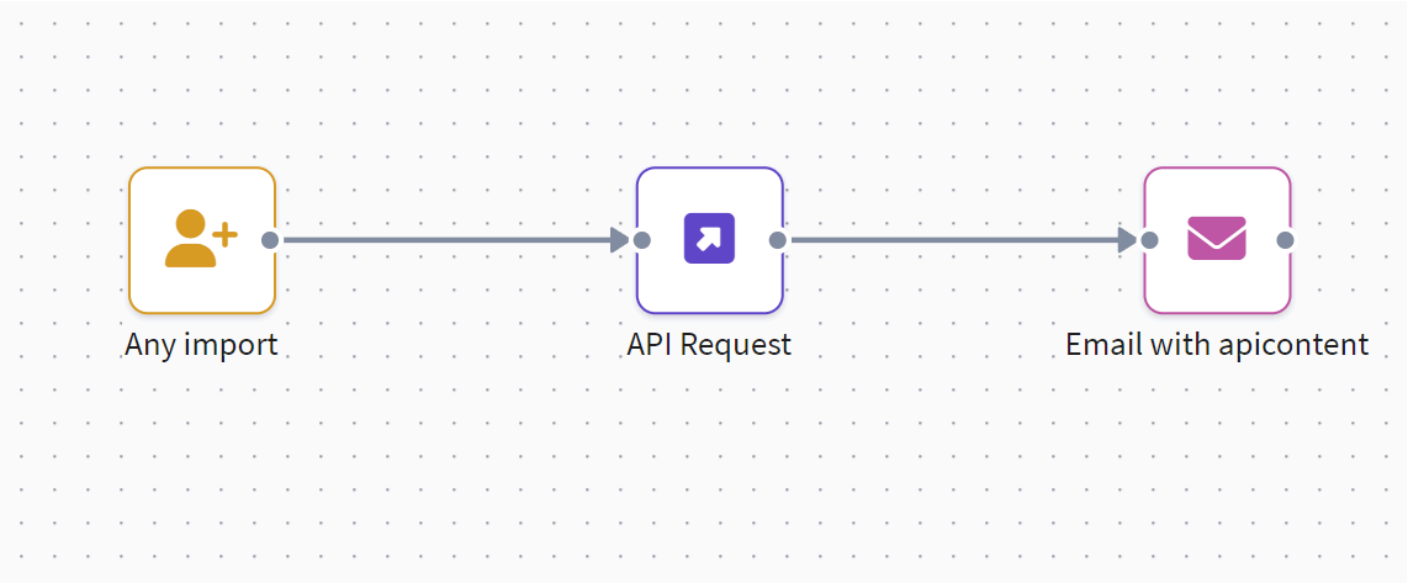
You can read about how to configure the “API Request” element in this article.

Select the type of request and paste the link to the service that will return the json-object:

Click on the switch Save the http response in apicontent field and enter the name of the field under which information from external sources will be recorded and which will be used in the message template in the future:



In order to understand how information will be transmitted to `apicontent` in the future, let's look at the following example. Add the Email element in the required place in order to send the data received from the external service to the client via email.



The path to the data in the template will change slightly:

HTML TEXT

```
<table align="center" role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%" style="margin: auto">
<tr>
<td style="background-color: #ffffff;">
<table role="presentation" cellspacing="0" cellpadding="0" border="0" width="100%">
<tr>
<td style="padding: 20px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20px; color: #555555">
<h2 style="margin: 0 0 10px 0; font-family: sans-serif; font-size: 28px; color: #6778cd; font-weight: 800; line-height: 30px"></h2
<h2 style="font-size: 28px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0 10px">Thank you for registering!p
<p>Thank you for choosing our course:</p>
<ul>
    <li>{apicontent.products.product_name}</li>
    <li>{apicontent.products.product_price}</li>
</ul>
</td>
<tr>
<td style="padding: 0px 20px 0px 20px; font-family: sans-serif; font-size: 14px; line-height: 20px; color: #555555;">
<h2 style="font-size: 15px; color: #6778cd; font-weight: 800; line-height: 1.2em; margin: 15px 0 10px">We appriciate your choice</
   <p>Thank you and good luck!</p>
</td>
</tr>
```

TIP

The preview of the message with `apicontent` is available only after configuring the section Configure API content. To do this, go to the preview section and paste the object containing the `apicontent` into the editor. Click on the check mark to apply the changes.

Using a loop in a template with apicontent

You can also transfer `apicontent` variable into json-object with an array.

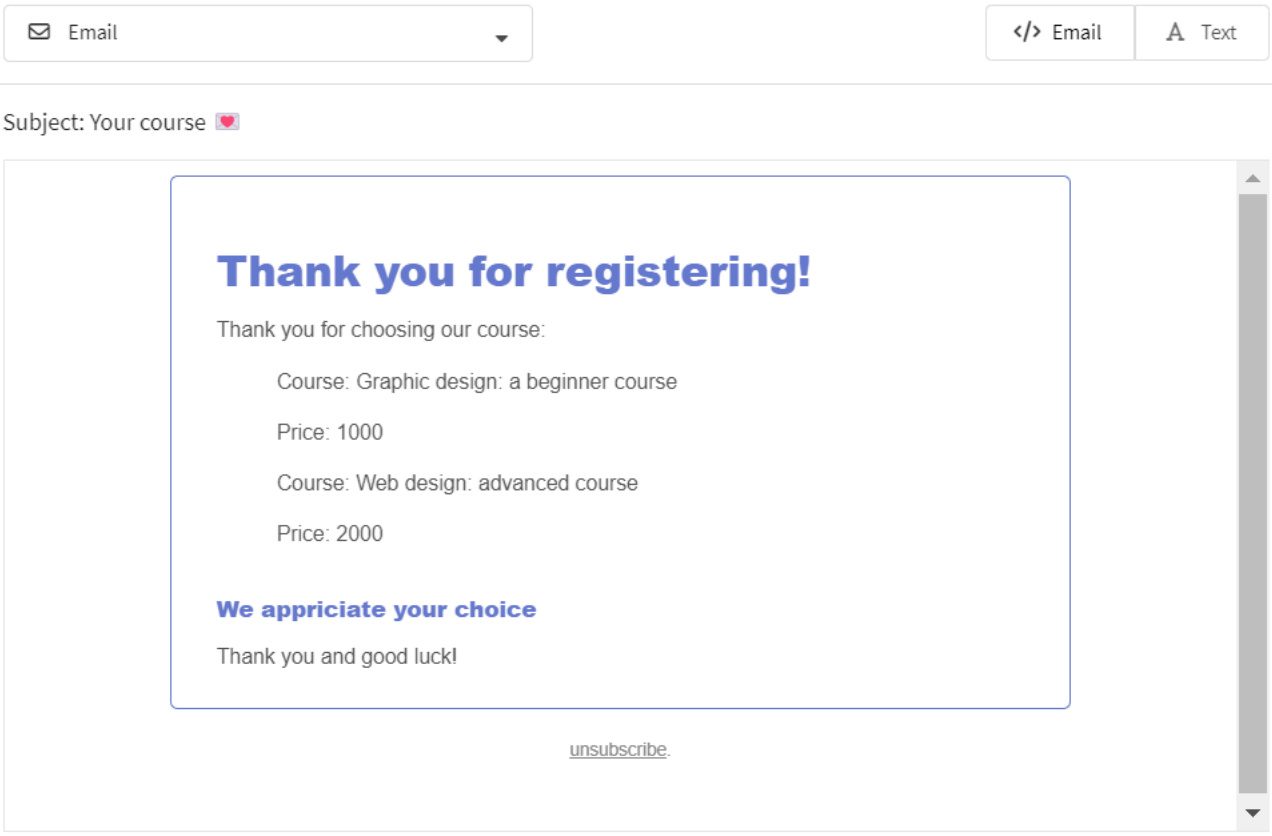
An example:

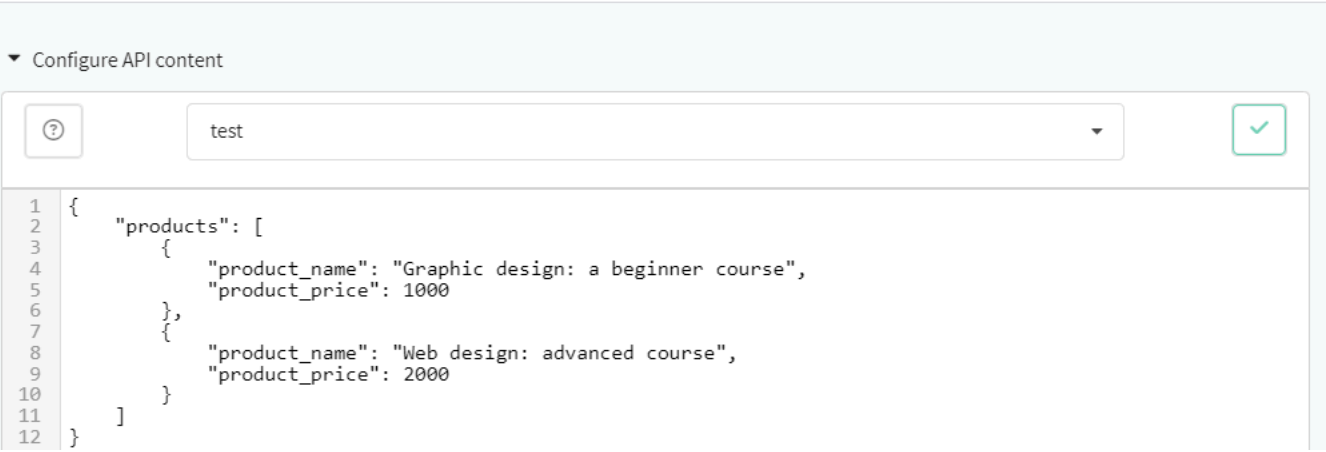
```
{
    "products": [
      {
        "product_id": 11,
        "product_name": "Graphic design: a beginner course",
        "product_price": 1000
      },
      {
        "product_id": 12,
        "product_name": "Web design: advanced course",
        "product_price": 2000
      }
  ]
}
```

To insert data from an array into a template, you need to iterate through your array. To do this, use the loop with variables.

```
{for $index $el = apicontent.products}
<div>
  <p>Course: {$el.product_name}</p>
  <p>Price: {$el.product_price}</p>
</div>
{else}
  if empty array
{end}
```

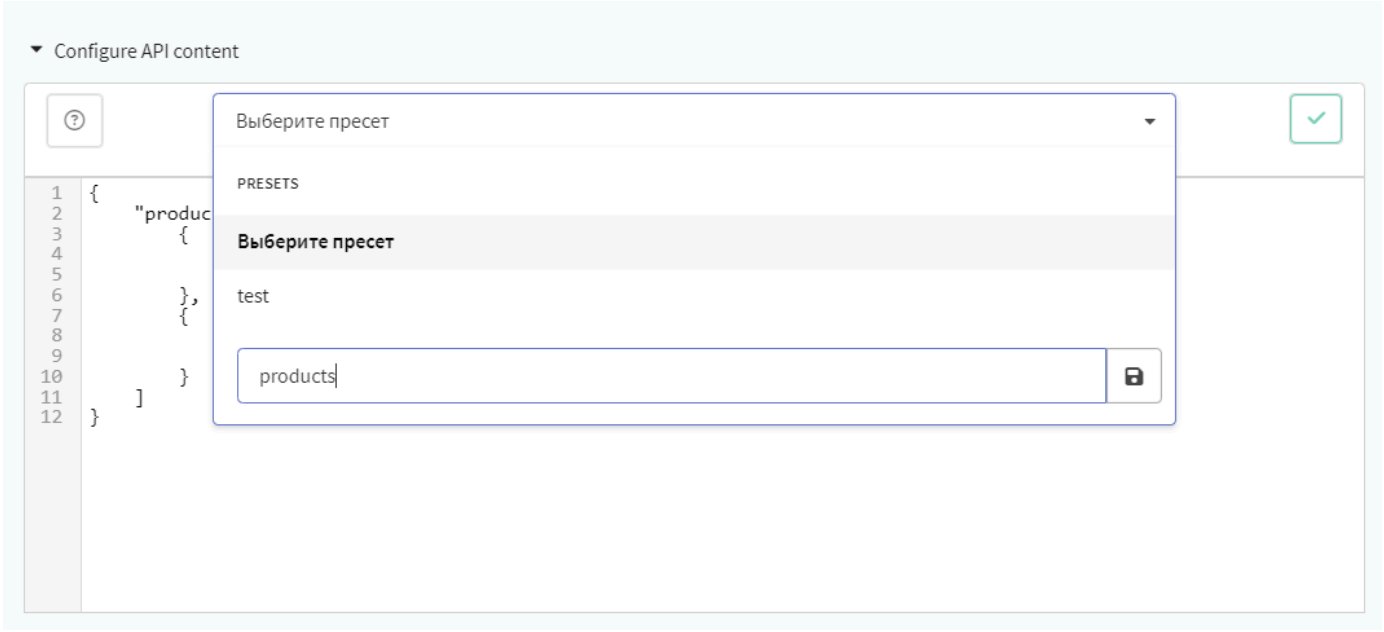
This code will insert the following data into the email:





TIP

If you regularly use the same json-bject in a variable, then you can save it as a preset and then select it in the preview window:



Cancel sending message if field is missing

You can configure the cancellation of the message sending if a specific field in the `apicontent` object is missing or contains an empty value. To do this, add a check for the field's existence and use the `{cancel}` variable.

Example:

```
{if apicontent.field_2}
field_2 не пустое
{else}
{cancel}
{end}
```

In this case, the mailing will not send the email if the `apicontent` object contains one of the following:

```
{
  "field_1": "example"
}
```

```
{
  "field_1": "example",
  "field_2": "" / or 0, null, false
}
```

This approach helps to avoid sending irrelevant emails and blocks sending if incomplete data was received from an external source.

Use case example

Suppose you have a marketing mailing intended only for users who have started or completed a certain course. The request includes an `apicontent` object with information about the course progress status:

```
{
  "user_id": 123,
  "course_id": "abcadefg",
  "course_name": "Graphic Design: Beginner Course",
  "course_status": "completed"
}
```

However, if the `course_status` field is missing or empty in the request, it may indicate that the user has not started or completed the course. In this case, the email should not be sent. The email template:

```
{if apicontent.course_status}
<div>
  <span>You showed interest in the course {apicontent.course_name}.</span>
  <span>You might also like these courses: <course list></span>
</div>
{else}
{cancel}
{end}
```

Mailings

Description

In Altcraft Platform, Mailings are used for sending messages to the customers.

Mailing creates message for certain resource subscribers, based on selected template, and adds customer information from the database, dynamic content, parts and fragments.

If dynamic segment is used as mailing audience, it will be counted just before every launch. Also, before every launch, mailing excludes profiles in suppression lists and blocked by sending policy.

Mailing types

	Broadcasts **This mailing performs one mailing to certain subscribers audience. If the audience is too big, mailing can be divided into several parts by schedule. Subscribers base or segment can be used as audience.**
	Regular mailings **This mailing sends certain content to certain audience regularly. This mailing will never stop by itself. Subscribers base or segment can be used as audience.**
	Multivariate testing **This mailing allows you to test marketing efficiency of sent messages. You can add up to 16 messages in the mailing.**
	Trigger mailings **This mailing allows you to send messages to subscribers after certain action. Mailing launch conditions can be changing subscriber information, profile import, or direct API request.**

Mailings calendar

Altcraft mailings calendar is our ultimate solution for organizing your marketing mailings.

It empowers your business with a strategical overview of the situation and an efficient marketing communications planning framework.

	Mailings calendar в displays launch time for all your marketing mailings during the custom period of time: • **Broadcasts — one-time massive mailings** • **Regulars — periodic mailings** • **Multivariate testing (A/B/n) —massive mailings that allow you to select content mailing by best efficiency**

Also, from the calendar, you can create a new mailing of any type, including a trigger mailing, which does not have a fixed launch time.

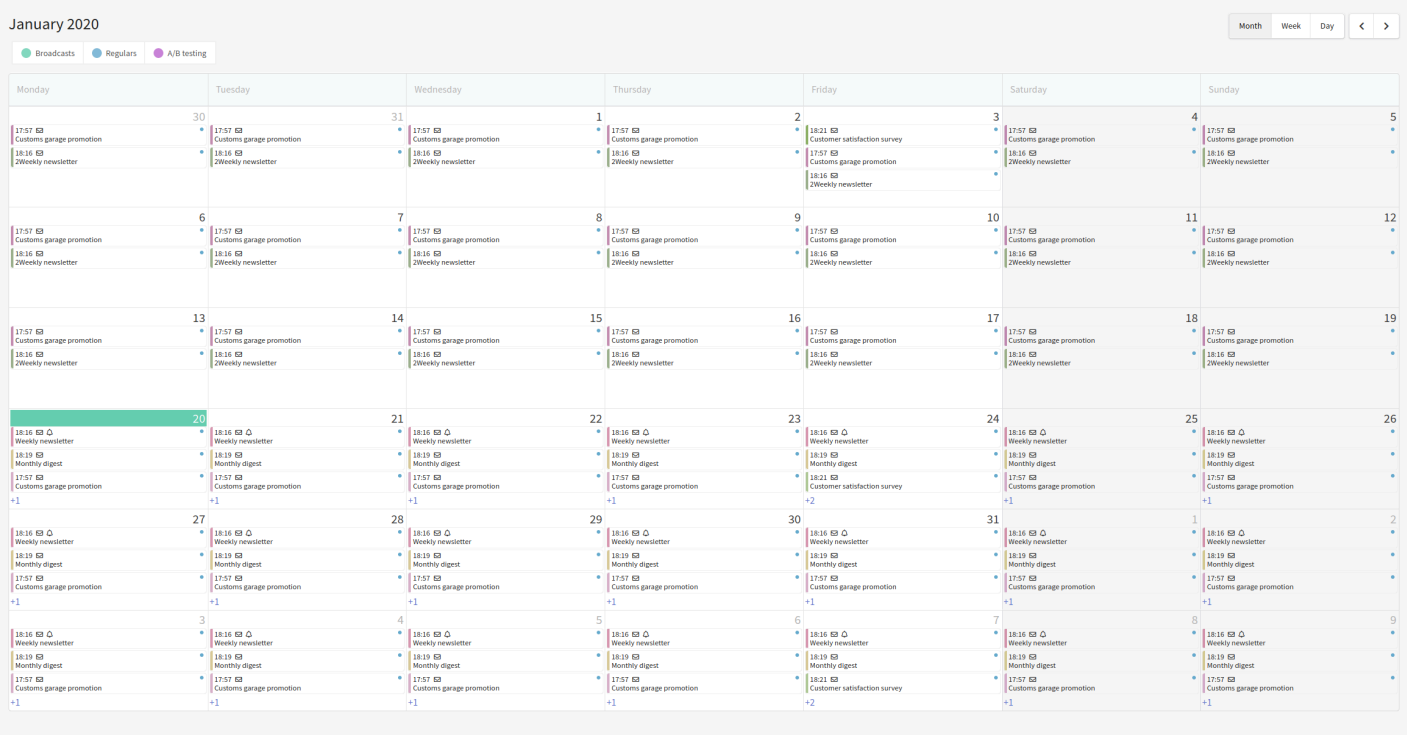
To access the calendar, in the main menu select Mailings → Calendar.

Viewing mailings information

The calendar displays information about future and past mailings. If you perform massive mailing divided into several parts, every part will be displayed in the calendar. It can help you to control the loading of your sender and network, and email provider limitations.

For every mailing, launch time, mailing type and name, and used channels are displayed. Three display modes are available:

- **Month — this option displays information about all launches in a month. For every day, three first mailings are shown. If there are more than three launches in a day, a link to detailed information will be shown in the bottom part of the tile.**
- **Week — this option displays information about all launches in a week. Mailings are grouped by launch day and time.**
- **Day — this option displays information about all launches on a separate day.**

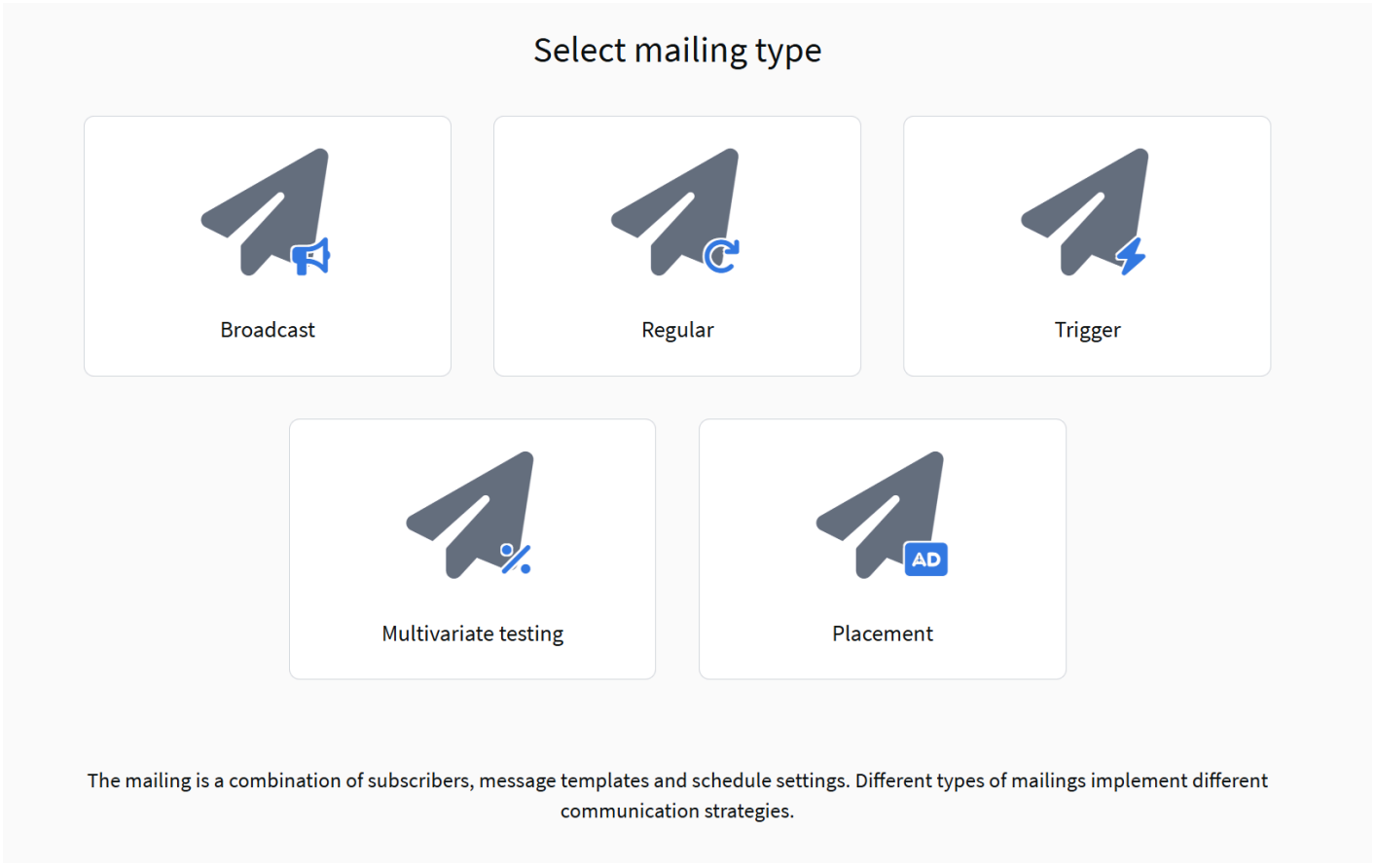


	Click a mailing label to view details of a mailing. You can clone or edit the mailing and see the mailing summary: • **Mailing name** • **Message template** • **Database or segment** • **Activation date and time** • **Communication channel** • **Mailing creator** • **Creation date** • **Resource** **In month view you can see all active mailings for a specific month. Use < and > buttons to switch months.**

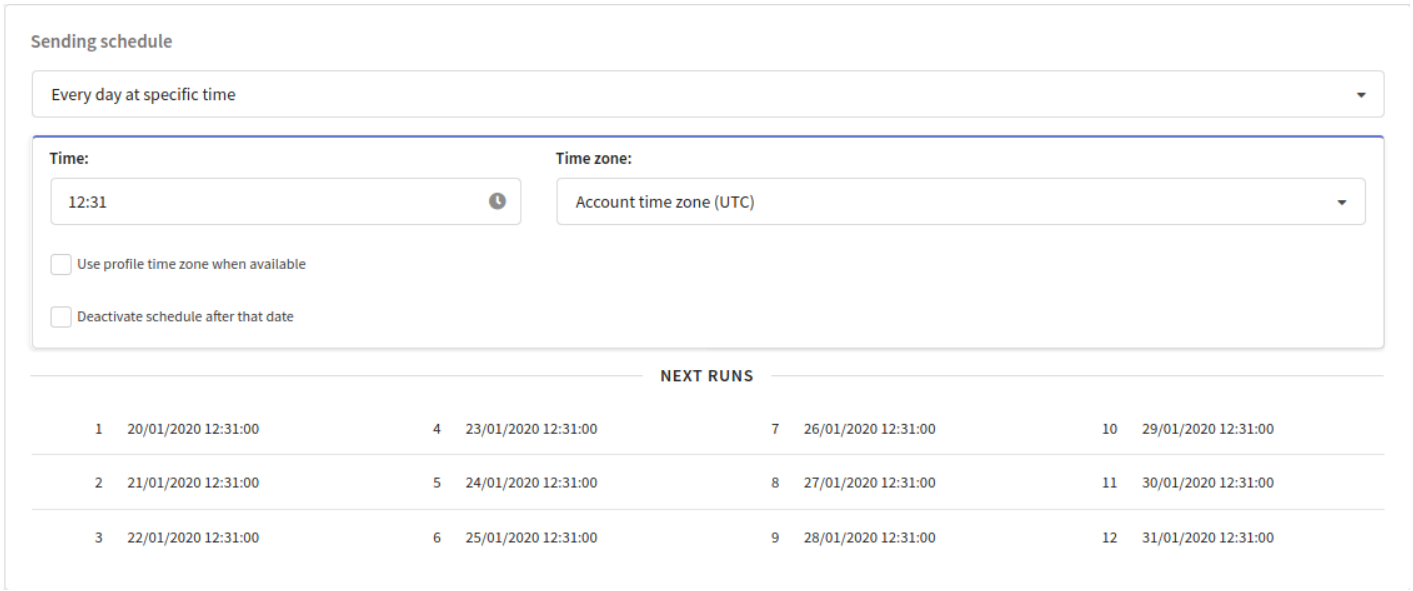
Creating a new mailing

	You can create a mailing at any time from the calendar interface. Select first launch day, click an empty space in the calendar for that date, and press + Add mailing in the popup.

The selection panel with all 4 mailing types will open. When you press on the type you want, the editing form will open.



Sending date will be preset in the mailing editor, though you can change it as you wish.



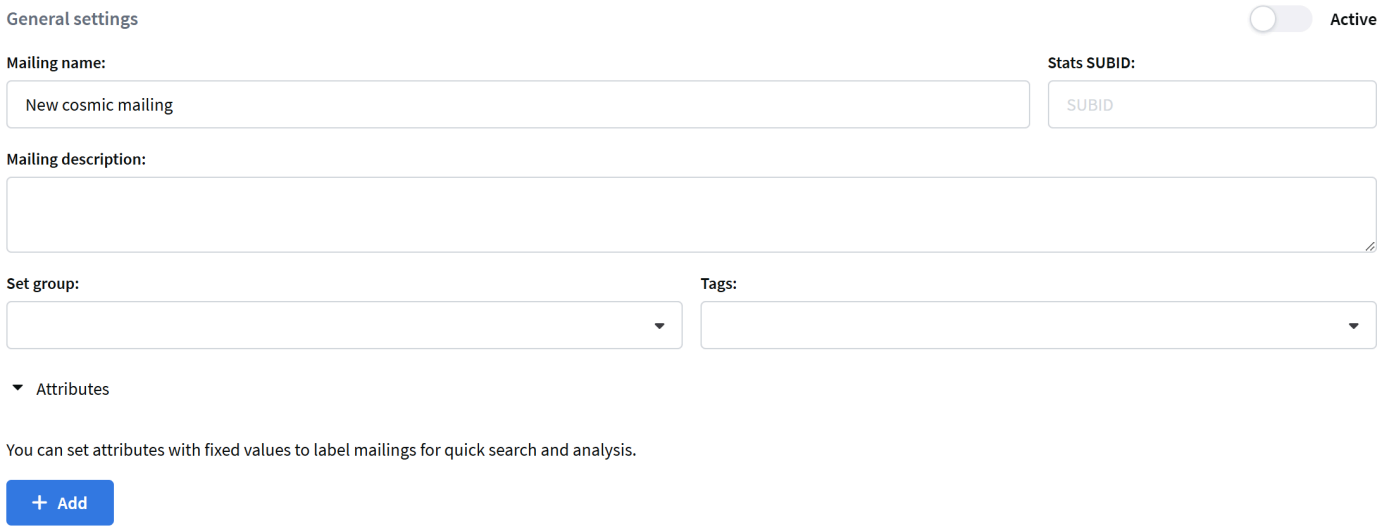
Broadcasts

Description

	Broadcast mailings are used when you need to communicate to a specific audience one single time. It can be a seasonal offer email, or usage terms change notification — any content that is sent to an audience rather than to separate customers on a non-regular basis. At the end of a broadcast mailing, every recipient is sent one message. Thus broadcast messages mostly use static content, as it is sent once. Personalization options can still be a great asset to make your mass communications more human-like. To create a new broadcast, go to Mailings → Broadcasts and click + Create button. Also, you can use the calendar to create a new broadcast. </td> </tr>

General settings

In the main settings, specify the name and description of the new mailing, and if needed, set the access group, add tags and attributes for additional grouping. You can also specify a statistics SUBID for the mailing here. In the upper right corner, there's a toggle to activate the mailing.



Mailing settings

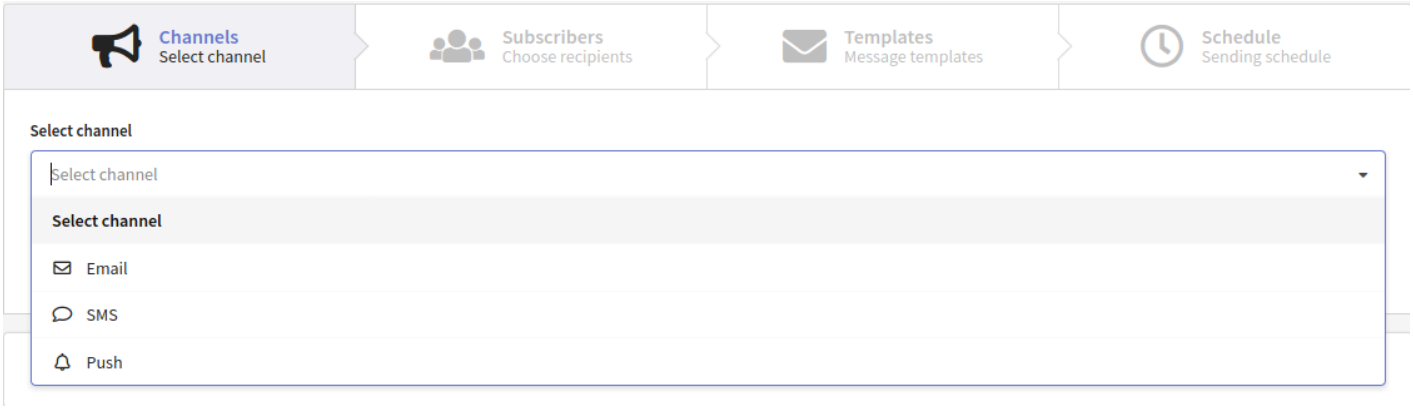
The following information is defined in mailing settings: communication channels, subscribers, templates with content, and schedule for mailing.

Channels

On the first tab, you can select a communication channel you want to use. Channel you want to use must be available in at least one subscriptions resource, which you can select on the next tab.

INFO

For Email sending, you can use the field with the main subscriber contact information – the Email field.



TIP

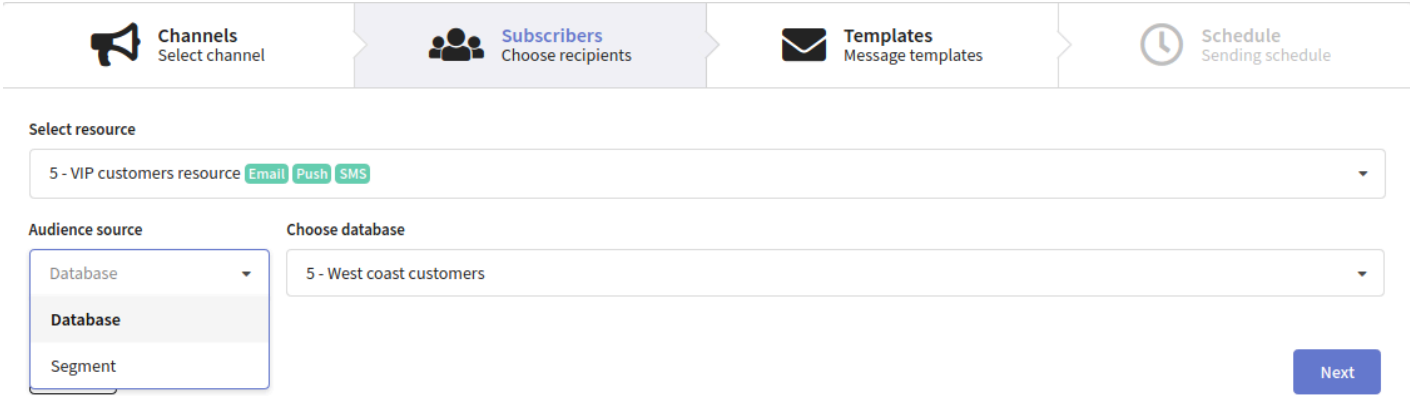
When using a push channel, you can select specific push providers for sending, as well as the sending mode - fanout, sending to all subscriptions taking into account filters, or last, sending to the subscription with the most recent creation date. If there are several of these (created at the same time), then one will be selected randomly. If the mailing has filters by provider, custom fields or categories, then the filters are applied first, and then the most recent subscription is searched.

Subscribers

On this tab, select the resource form drop-down menu. You can use a database or segment as an audience.

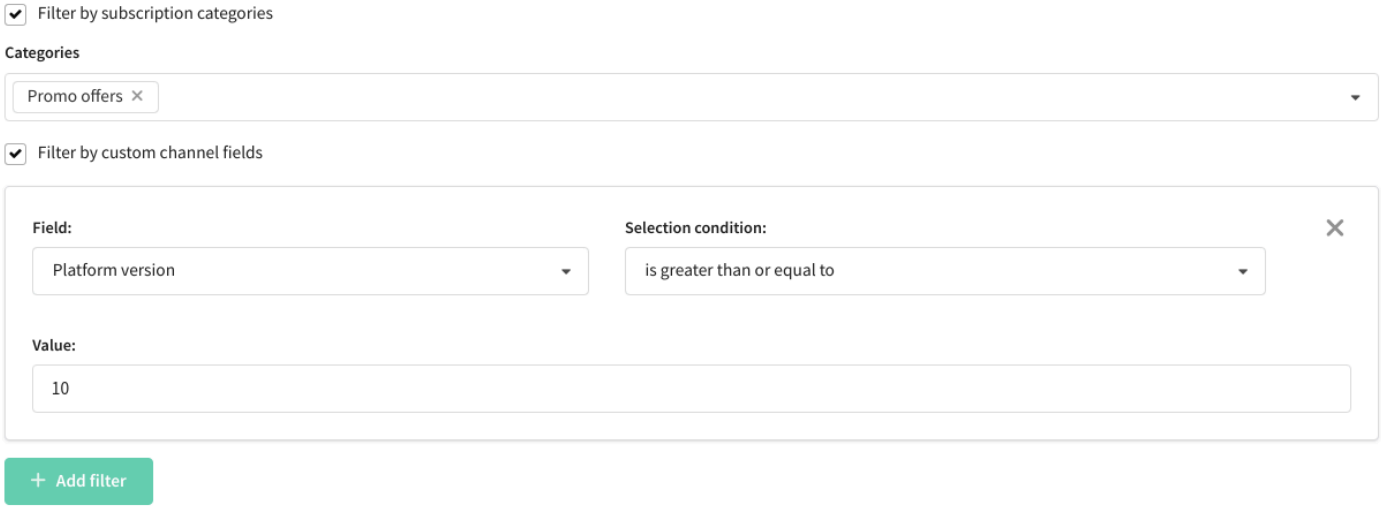
TIP

Mailings are sent to customers who subscribe to a certain resource. Resources can be a separate website, product, promotion, or another direction of your communications with customers. Select a resource. Make sure the recipients in your database are subscribed to it.



Additionally, you can filter recipients by resource categories. To enable this functionality, you need to have categories created in the resource. In mailing settings, select the categories to which the profile should be subscribed to receive a message.

You can also filter recipients by custom channel fields. To do this, select either a standard or custom subscription field. In the Selection condition field, choose a logical operator and, if necessary, specify a value. You can combine rules using the logical AND (all conditions must be met) and OR (at least one condition must be met).



Templates

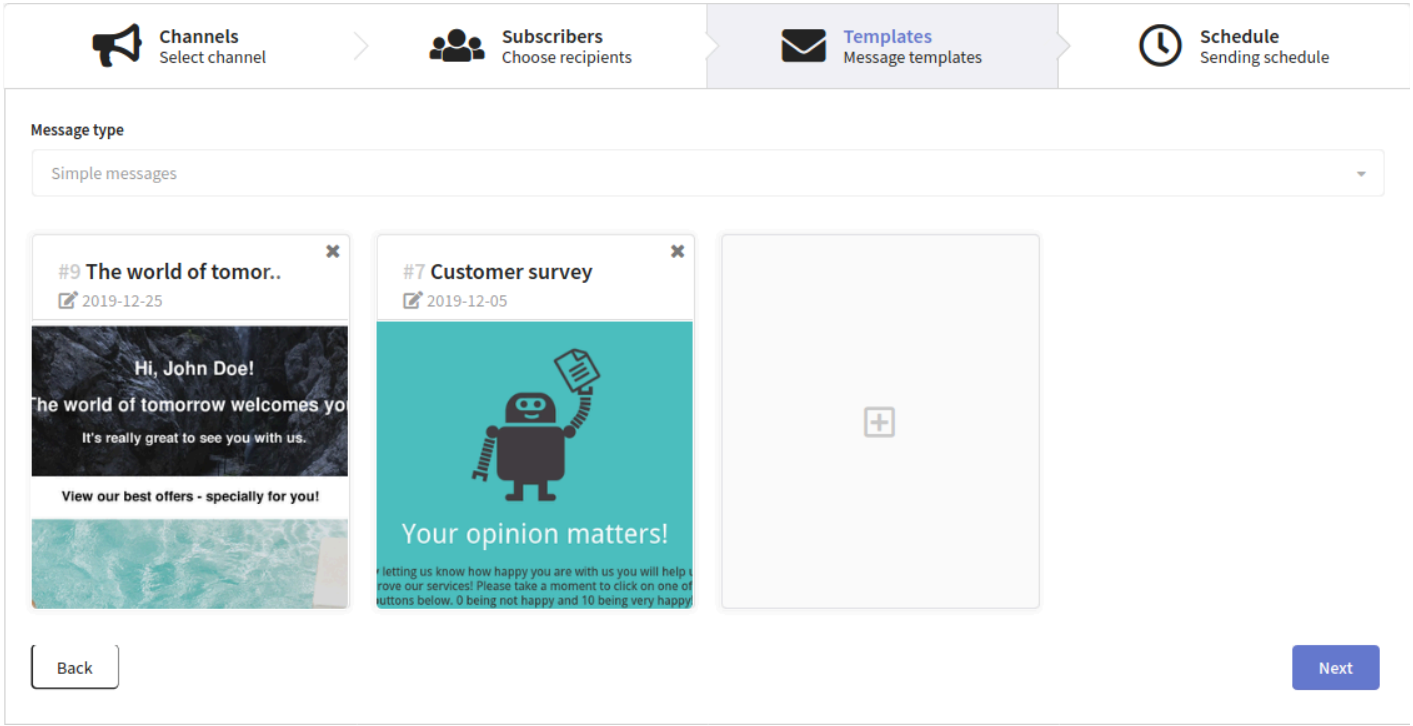
On this tab, you can select the content that you want to use in the mailing. In Altcraft, messages templates combine content for mailing by all available channels.

TIP

Please, make sure, that the template has content for the selected channel.

Clcik

and select a prepared template. Any mailing type can use up to 8 templates. The broadcast mailing will send one message to each customer using selected templates in random order in equal parts. Click on the thumbnail to preview, or click on the template name to open template editor:



Schedule

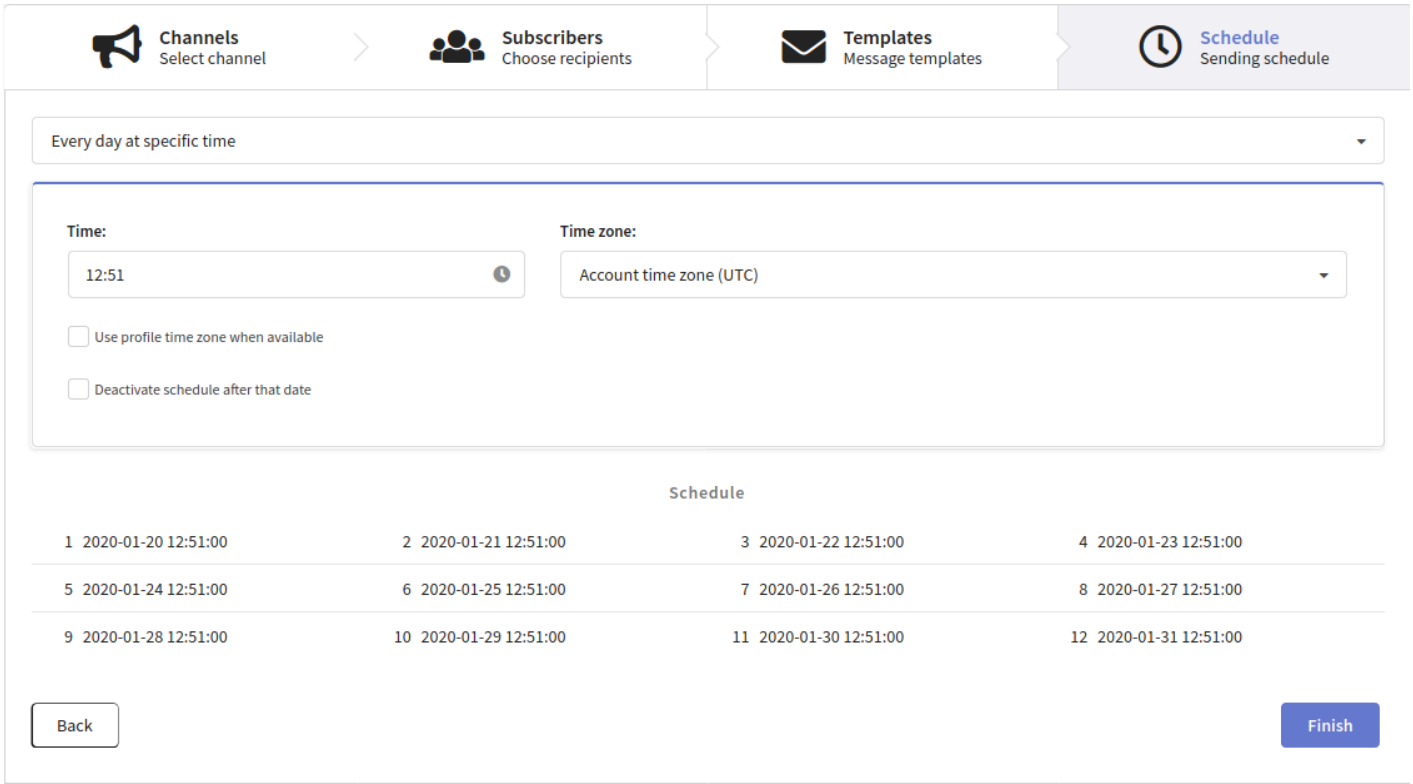
TIP

The mailing schedule is described in detail on [this page](#).

On this tab, you can configure sending policy. During massive mailings, you need to remember about sender reputation. Email providers set limits for massive mailings that depend on sender reputation and the number of sent messages. Usually, providers set limits for messages received in 24 hours.

Sending all messages at once

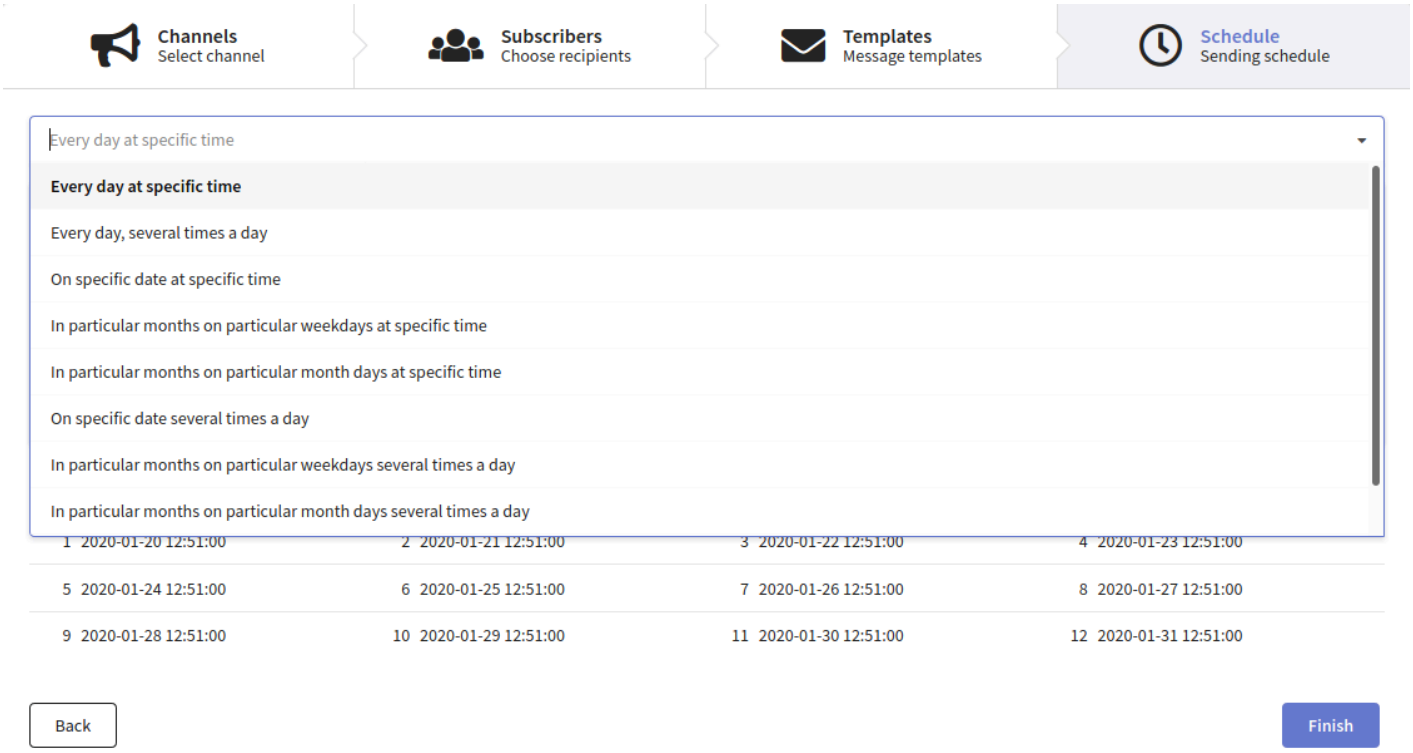
If your sender reputation is high enough, you can send all messages in 24 hours. To do this, select On specific date at specific time.



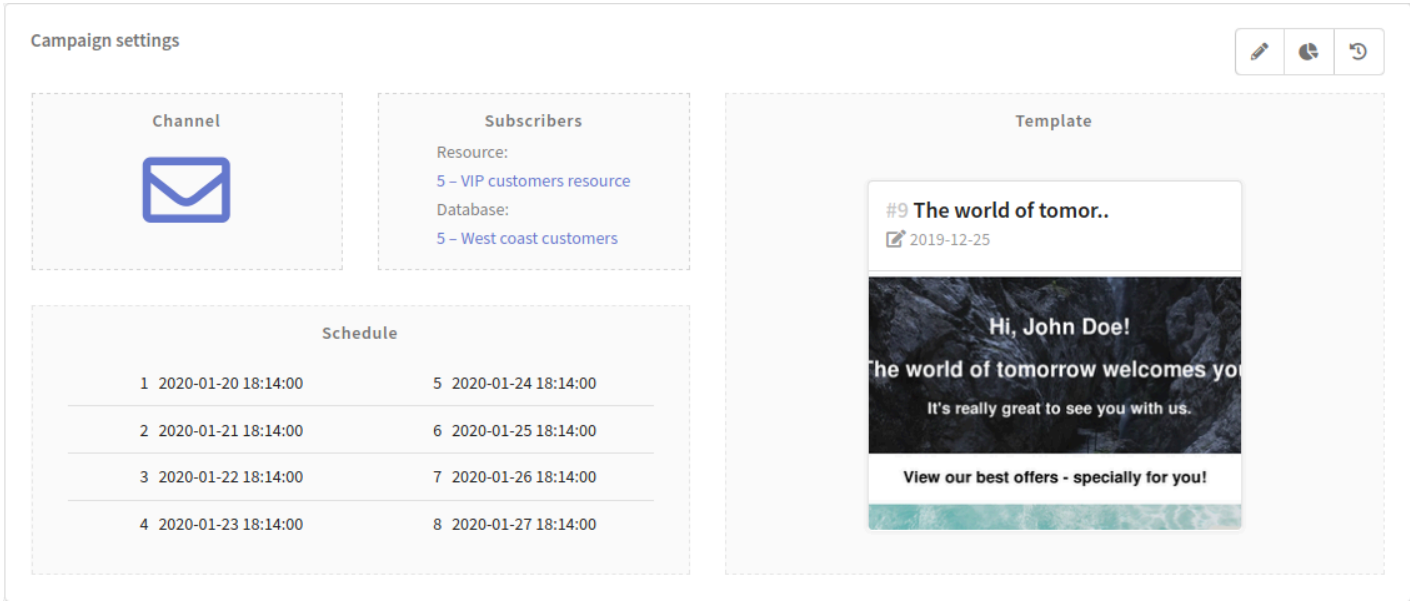
Also, you can select a time zone for sending, or Use profile time zone when available.

Partial sending

Most email providers limits are valid for 24 hours. If your mailing is over limits, we recommend you to use Every day at specific time option. You can configure the number of sent messages in the Limitations block.



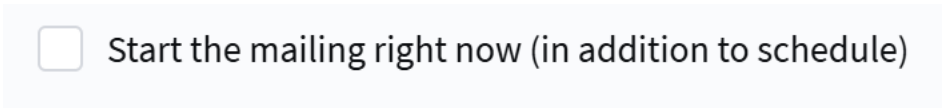
After you made your configurations, click the Finish button. Selected parameters will be displayed in the Mailing settings block. You can use appropriate buttons to edit settings or go to the [mailing report](#).



Instant mailing launch

If you want to launch a mailing right after its creation, toggle the appropriate option at the bottom part of the page.

- **If the limits were not configured, messages will be sent to all customers.**
- **If limits were configured, the mailing will be launched besides schedule.**



Limitation

If the number of messages in the mailing is more than the number of messages received by email providers in a day, you can separate your broadcast mailing into several parts.

INFO

You can also use mailing limitations to control the loading of your sender or your company network architecture.

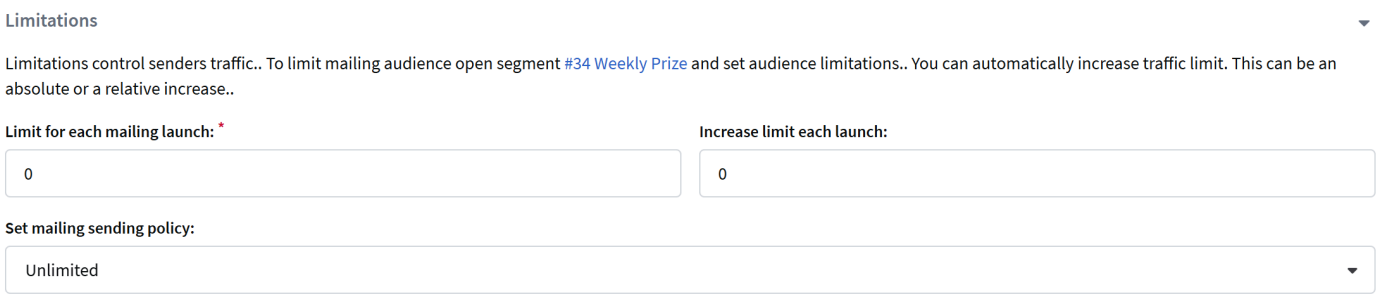
You can configure limits using these options:

- **Limit for each mailing run — the number of messages, that will be sent during the next launch by schedule**
- **Increase limit each run — the number of messages or percent value that will be added during next launch.**

CAUTION

When the “Use profile time zone when available” option is enabled, the limit settings and instant mailing launch are unavailable.

Communication policy sets limits for communication with one subscriber. By default, the mailing will use account policy or resource policy.



Smart settings

The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages from the platform’s real senders).**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify each domain on a separate line.**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of campaign recipients based on subscription data.**

Mailing watchers

You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mailing recipients. To do this, you need a watchers database. It is created the same way as a regular database, but you must enable the corresponding option in General settings.

In Smart Settings, select the watchers database and set the message sending limit for each mailing launch:

Наблюдатели за рассылкой

Наблюдателям будут отправлены письма каждый запуск рассылки если лимит не установлен, иначе будет выбрана новая порция наблюдателей в пределах лимита.

Письма будут отправлены на основной email профиля наблюдателя, подписки не учитываются.

БД наблюдателей: 2 - Mailing watchers database 1000

Лимит: 100

Broadcast mailing sequence

The Activate this mailing automatically after another one is finished option is available only for broadcast mailings. It allows to automatically launch a mailing after the previous mailing is finished. For this, select the mailing, after:

Broadcast mailing sequence

Activate this mailing automatically after another one is finished:

Select a mailing

CAUTION

If you have massive mailings, please, make sure that your mailing fits the limits of email service providers.

Message Time-To-Live (TTL) Restriction

Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the delivery queue. If the message is not delivered within this time, sending stops and the message is removed from the queue.

TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is related to a promotion, an event reminder, a one-time code, or an offer that quickly loses relevance.

To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, hours, minutes, or seconds.

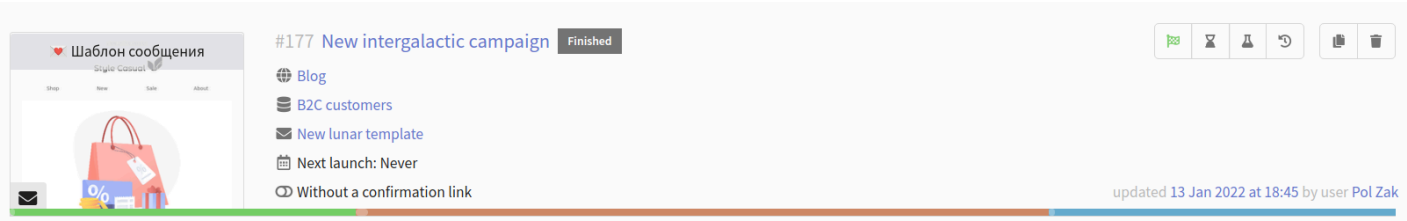
After the specified time expires, the system stops delivery attempts for the undelivered message and registers a delivery failure event.

TIP: Message TTL specifics

- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration parameters.**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API parameter > the value specified in Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefore, even within a single mailing, different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to deliver the message.**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and custom channels.**

Mailing progress

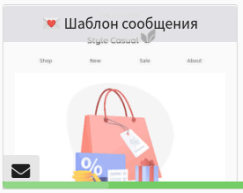
In the list of broadcast mailings, you can see the progress bar of each mailing.



Each message status has its own colour:

- **sends – the number of profiles for which messages were generated**
- **delivered – the number of profiles to which the message was delivered**
- **delivering – the number of messages in the sender queue**
- **undeliveries – the number of profiles that failed to deliver the message**
- **policy offences – the number ofprofiles affected by sending policy (see more)**
- **supressions – the number of profilesin suppression lists (see more)**

Click on the progress bar to open a table with mailing statistics:



#177 New interga

Blog

B2C customers

New lunar templat

Next launch: Never

Without a confirma

Total expecting	Sends	Delivered	Delivering
4	4	1	1
Undeliveries	Policy offences	Suppressions	Errors
2	0	0	0

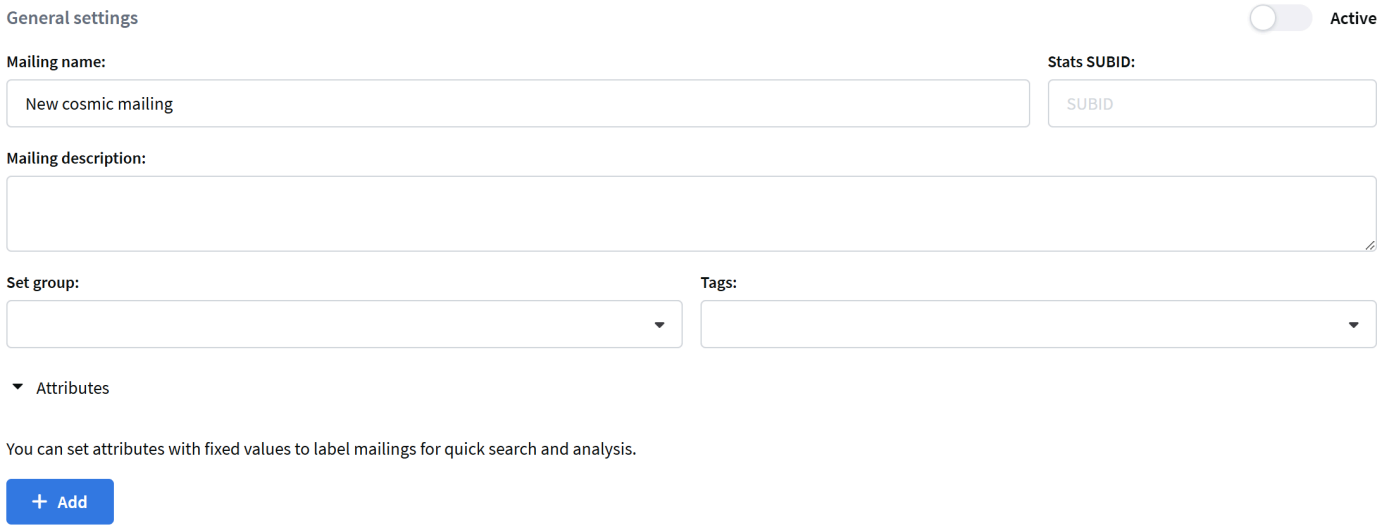
updated 13 Jan 2022 at 18:45 by user Pol Zak

Regular mailings

	Regular mailing is a tool for organizing** massive multiple** mailings to your subscribers. **At every launch moment,** this mailing sends one message to each subscriber from the selected audience. Unlike a broadcast mailing, which processes the database **only once**, a regular mailing never finishes by itself. To create a new mailing, select **Mailings → Regulars** and press **+ Create** button, or you can use the mailings calendar. </td> </tr>

General settings

In the main settings, specify the name and description of the new segment, and if needed, set the access group, add tags and attributes for additional grouping:



Mailing settings

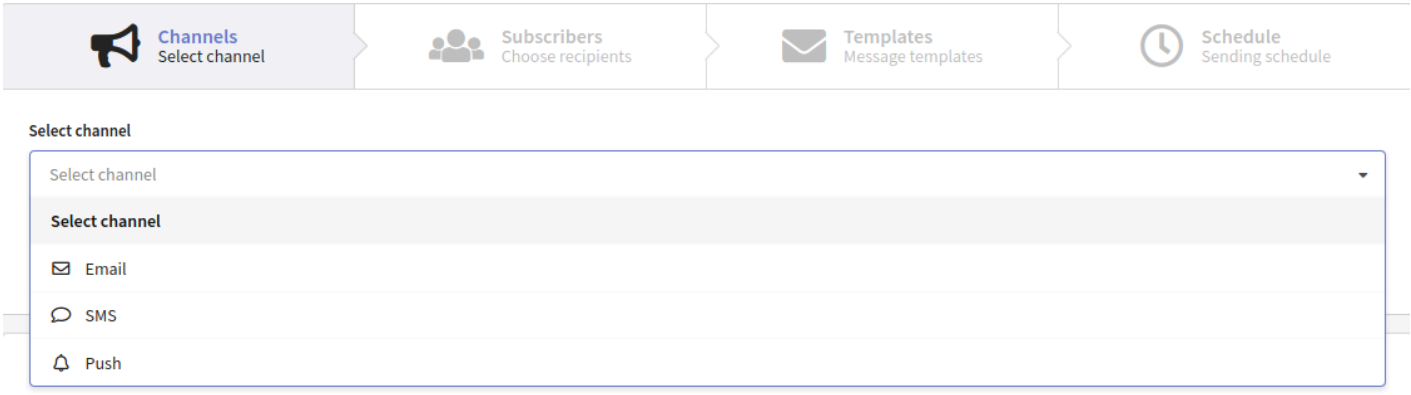
These settings are communication channel, messages recipients, templates with content and mailing schedule.

Channels

On this tab, you can select a communication channel for the mailing. This channel must be available in at least one resource, which you can select on the next tab.

NOTE

For Email sending, you can use the main subscriber contact data, which is the Email field in the database.



TIP

When using a push channel, you can select specific push providers for sending, as well as the sending mode - fanout, sending to all subscriptions taking into account filters, or last, sending to the subscription with the most recent creation date. If there are several of these (created at the same time), then one will be selected randomly. If the mailing has filters by provider, custom fields or categories, then the filters are applied first, and then the most recent subscription is searched.

Subscribers

On this tab, you can select Resource for mailing from the drop-down menu. The resource must contain a channel, which was selected on the previous step.

Also, you can select an Audience source, which can be a database or segment with profiles from one or several databases.

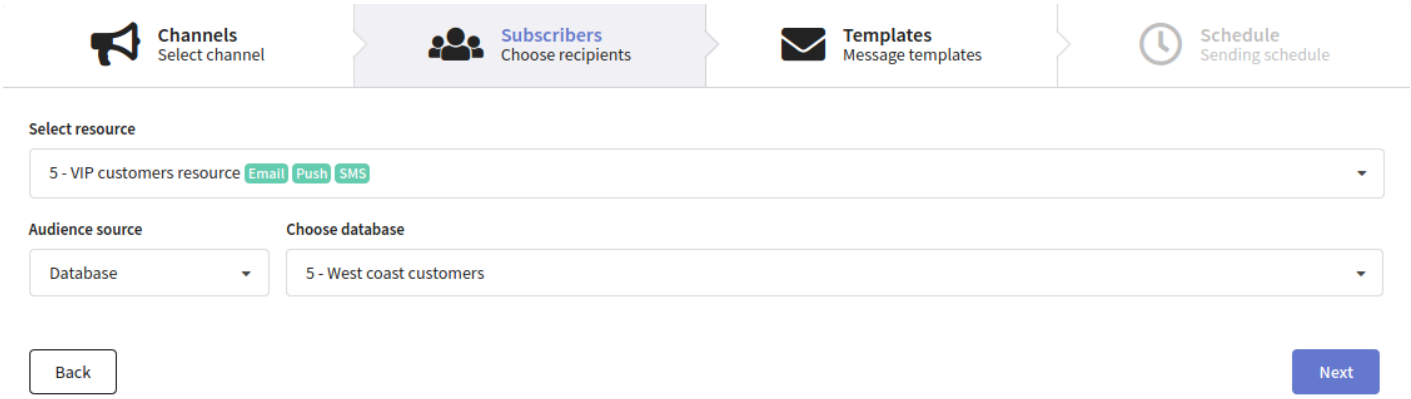
TIP

The audience source for a regular mailing depends on its purpose:

- **For periodic mailings, you can use a database, a static or dynamic segment. The mailing will send messages to the subscribers of the configured audience.**
- **For activity-based mailing, it is necessary to use a dynamic segment with selection based on some date, like subscriber birthday.**

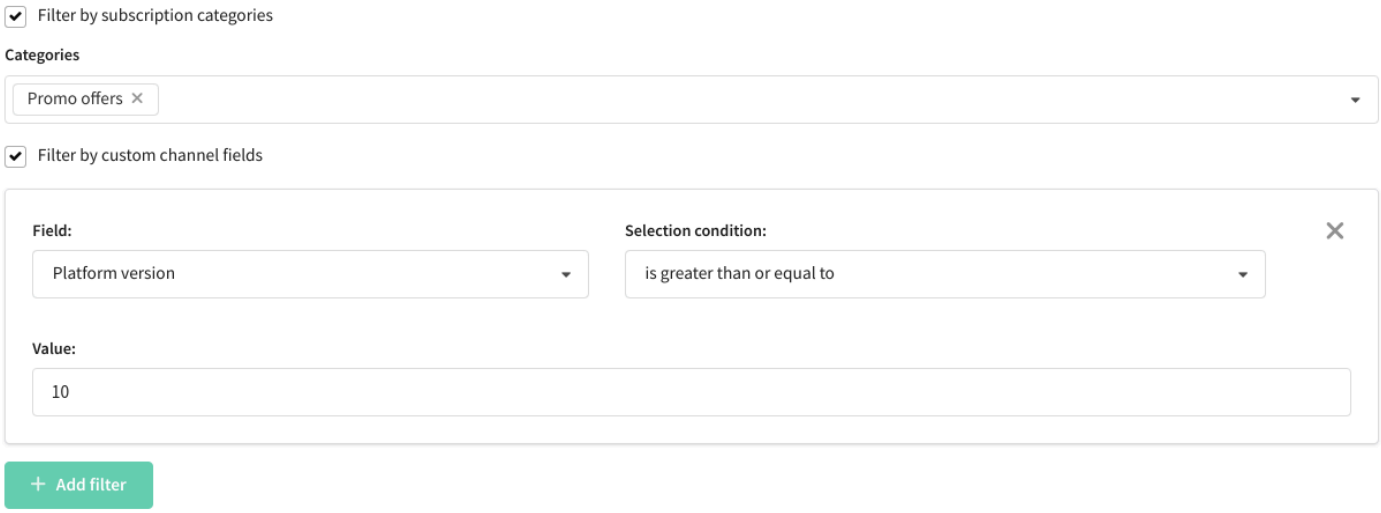
TIP

Mailings are sent to customers who subscribe to a certain resource.Please, make sure your customers are subscribed to this resource before launching a mailing.



Additionally, you can filter recipients by resource categories. To enable this functionality, you need to have categories created in the resource. In mailing settings, select the categories to which the profile should be subscribed to receive a message.

You can also filter recipients by custom channel fields. To do this, select either a standard or custom subscription field. In the Selection condition field, choose a logical operator and, if necessary, specify a value. You can combine rules using the logical AND (all conditions must be met) and OR (at least one condition must be met).



Templates

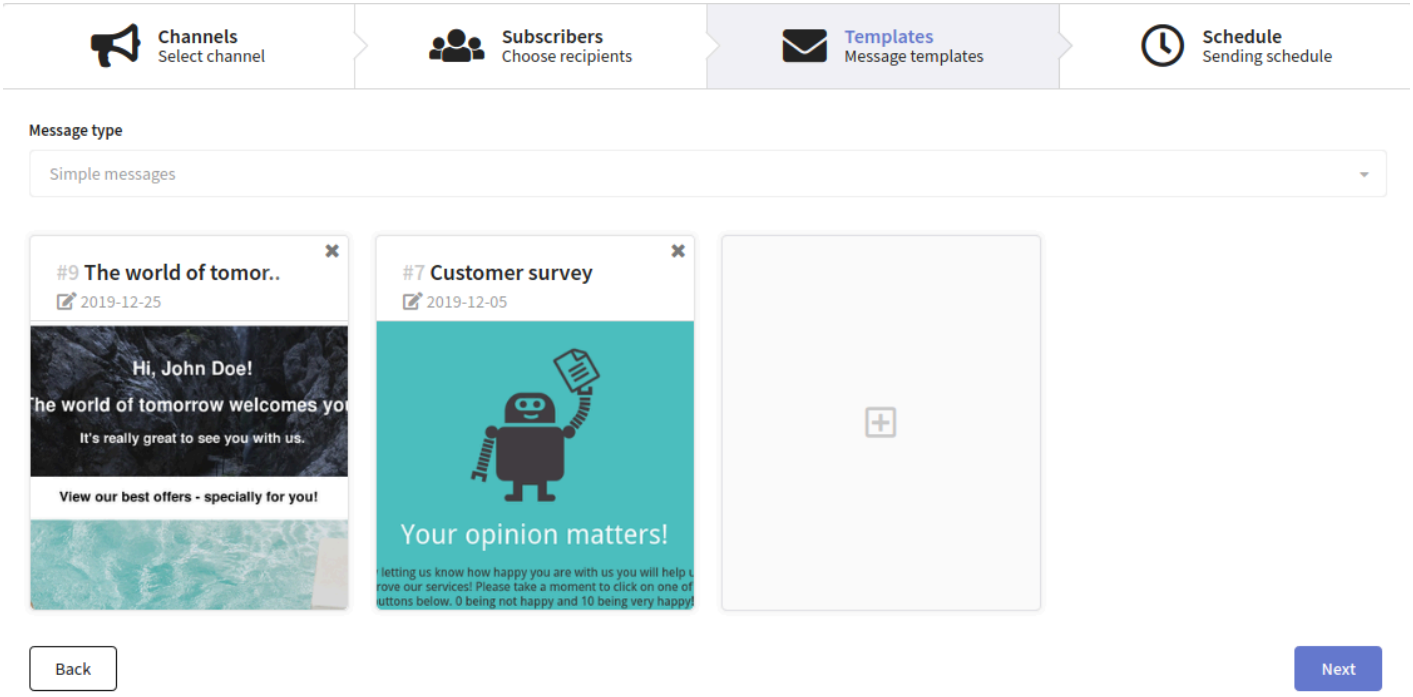
On this tab, you can select the content you want to use in your mailing. In Altcraft, templates combine content for sending by all available channels.

TIP

Before mailing, please, make sure that the content you want to use is contained in the selected template.

Press Add button and select the template. You can use up to 8 templates in any type of mailing. A regular mailing will send one message to each subscriber in random order in equal parts.

Click on the template miniature to open the preview.** Click on the template name to open it for editing in a new tab:**



Schedule

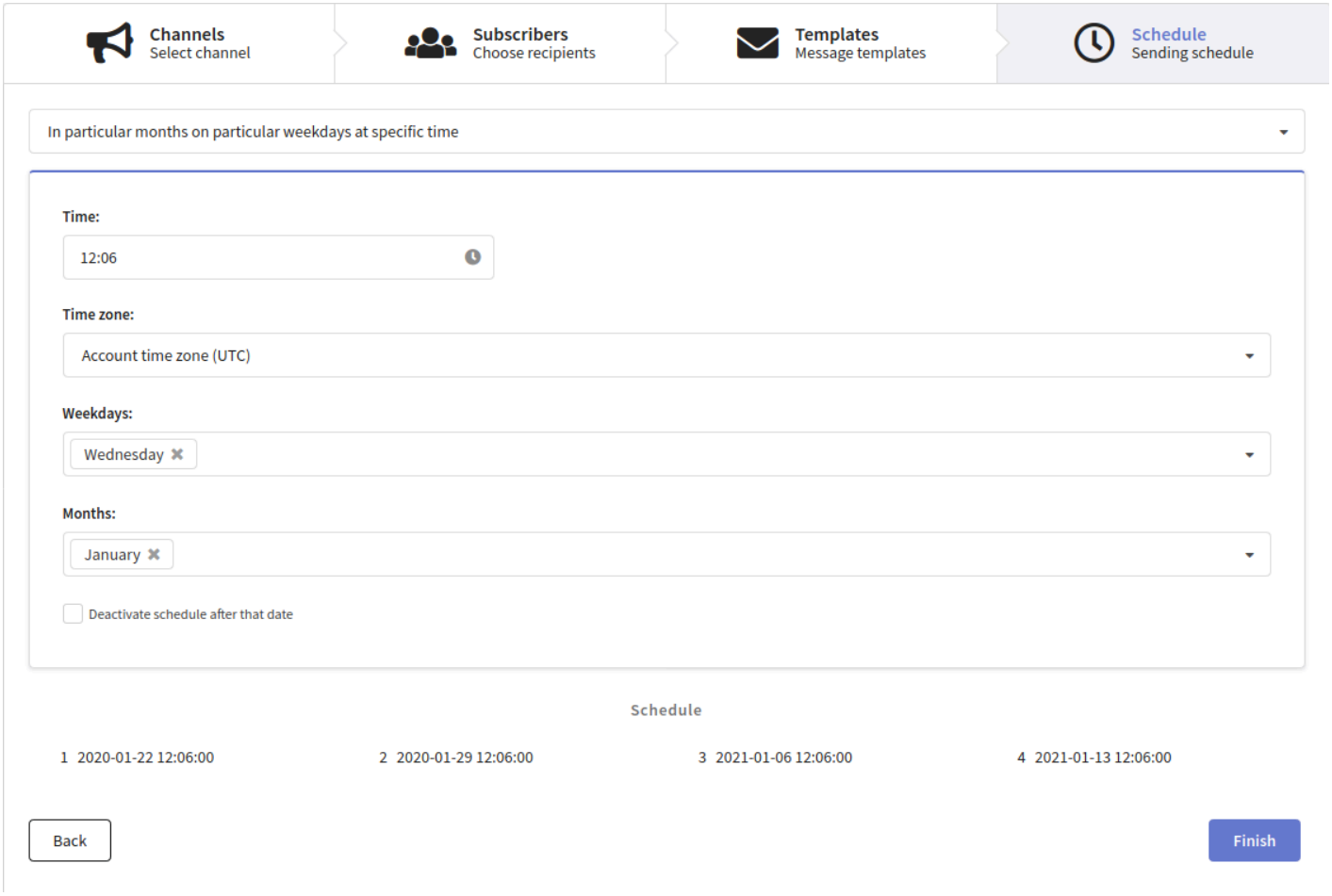
TIP

The mailing schedule is described in detail on [this page](#).

On this tab, you can configure sending policy. During massive mailings, you need to remember about sender reputation. Email providers set limits for massive mailings that depend on sender reputation and the number of sent messages. Usually, providers set limits for messages received in 24 hours.

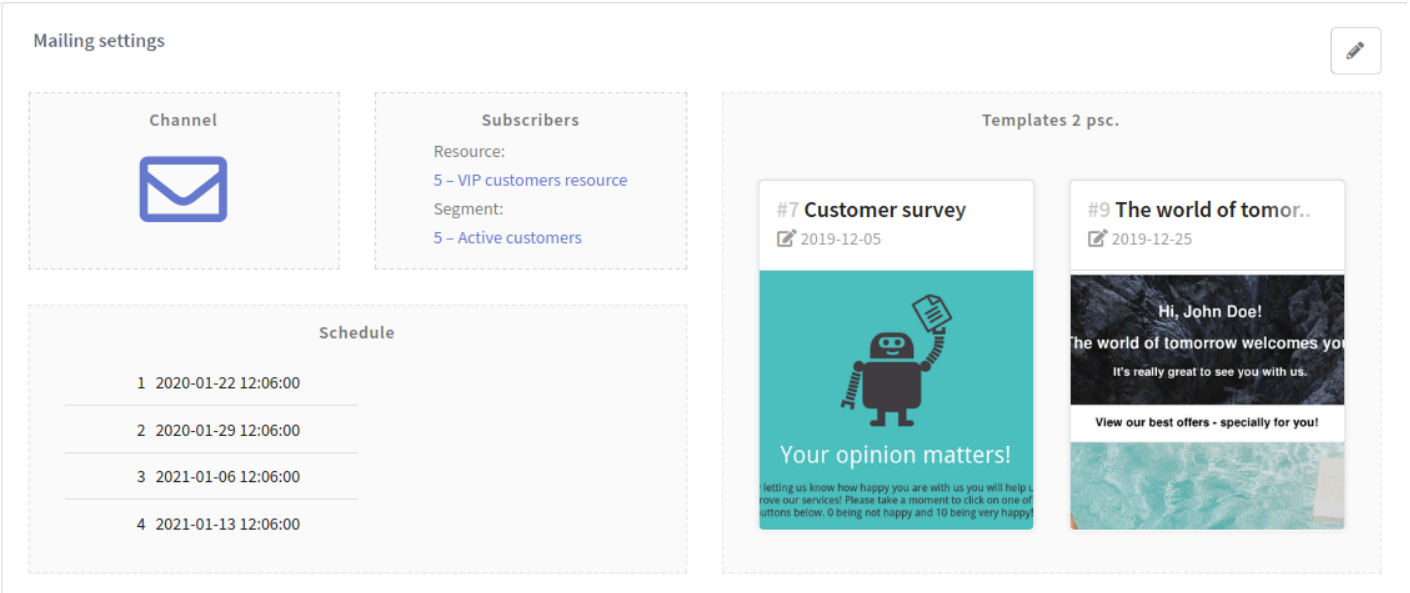
Regular mailing schedule

A regular mailing schedule is based on what kind of mailing you want to organize. You can set minutes, hours, days of the week, months, and even certain days of the week relative to the beginning of the month:



Also, for some schedule types, you can set a time zone, or use profile time zone, if this information is available. If you need to stop any mailings after a certain time, you can use Deactivate schedule after that date option.

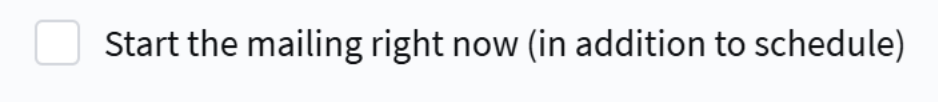
After you made your configurations, click the Finish button. Selected parameters will be displayed in the Mailing settings block. You can use appropriate buttons to edit settings or go to the mailing report.



Instant mailing launch

If you want to launch a mailing right after its creation, toggle the appropriate option at the bottom part of the page.

- **If the limits were not configured, messages will be sent to all customers.**
- **If limits were configured, the mailing will be launched besides schedule.**



Limitations

In this block, you can configure limits and sending policy.

Using limits in the regular mailing

> TIP
>
> **The regular mailing will not be stopped automatically after sending one message to each subscriber. If you want to organize massive mailing that must be sent in parts, we recommend that you use a broadcast mailing.**

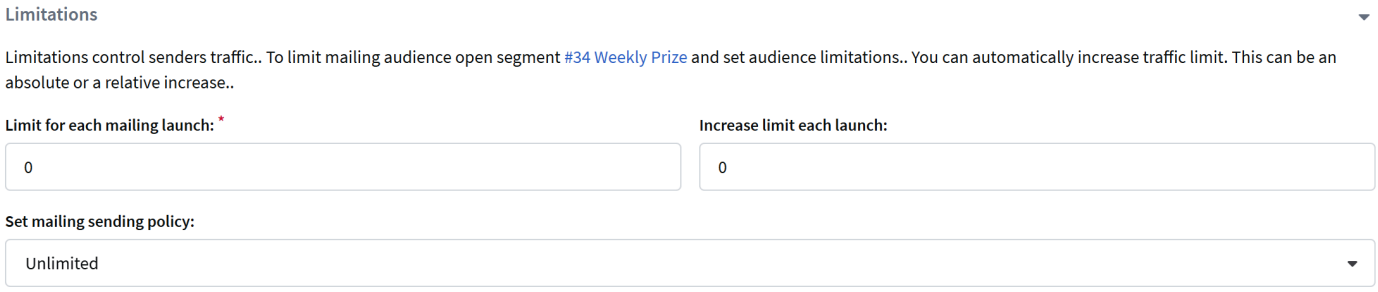
You can use limits in regular mailings when the number of mailings does not fit email providers limits.

> NOTE
>
> **You can also use limits to control the loading of your sender or your company network.**

Set limitations for the mailing:

- ****Limit for each mailing run** — the number of messages that will be sent during the next mailing run by schedule.**

> DANGER
>
> **Increase limit each run option is not recommended to use in regular mailings.**



In this case, during launch, the mailing will send as many messages, as you selected in the limit. When the number of sent messages is more than one set in the limit, mailing will start from the beginning of the list. To exclude re-sending one message, set Sending policy.

> CAUTION
>
> **Please, remember that limitation can act for no more than 16 days, and the mailing period must not be more than this value.**

> CAUTION
>
> **When the “Use profile time zone when available” option is enabled, the limit settings and instant mailing launch are unavailable.**

Communication policy sets limits for communication with one subscriber. By default, the mailing will use account policy or resource policy.

Smart settings

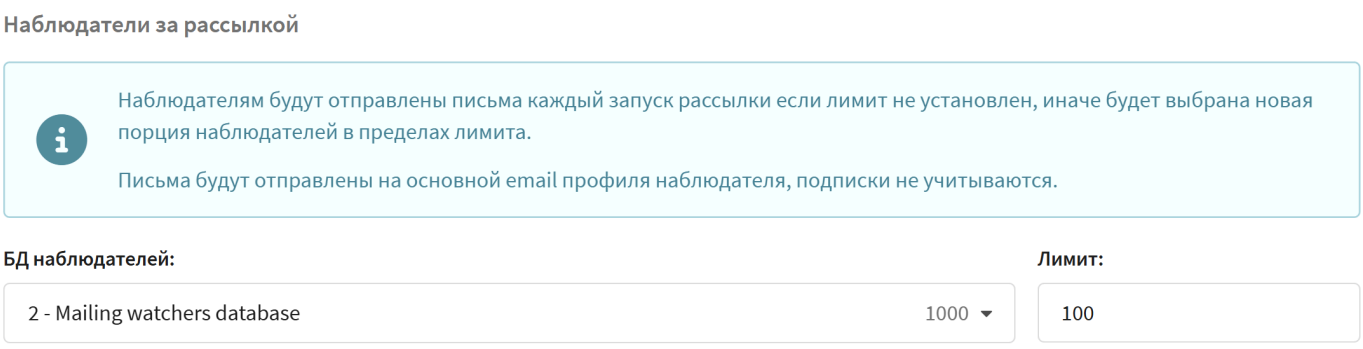
The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages from the platform’s real senders).**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify each domain on a separate line.**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients based on subscription data.**

Mailing watchers

You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mailing recipients. To do this, you need a watchers database. It is created the same way as a regular database, but you must enable the corresponding option in General settings.

In Smart Settings, select the watchers database and set the message sending limit for each mailing launch:



Message Time-To-Live (TTL) Restriction

Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the delivery queue. If the message is not delivered within this time, sending stops and the message is removed from the queue.

TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is related to a promotion, an event reminder, a one-time code, or an offer that quickly loses relevance.

To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, hours, minutes, or seconds.

After the specified time expires, the system stops delivery attempts for the undelivered message and registers a delivery failure event.

TIP: Message TTL specifics

- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration parameters.**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API parameter > the value specified in Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefore, even within a single mailing, different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to deliver the message.**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and custom channels.**

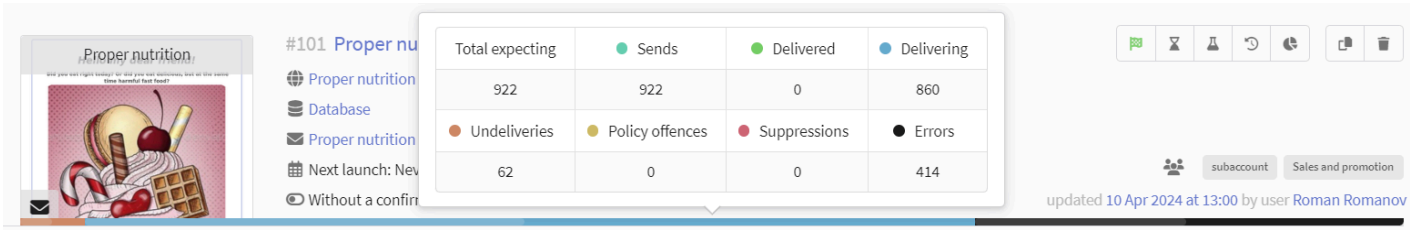
Mailing progress

In the list of regular mailings, you can see the progress bar of each mailing.

Each message status has its own colour:

- **sends – the number of profiles for which messages were generated**
- **delivered – the number of profiles to which the message was delivered**
- **delivering – the number of messages in the sender queue**
- **undeliveries – the number of profiles that failed to deliver the message**
- **policy offences – the number ofprofiles affected by sending policy (for more information more right there)**
- **supressions – the number of profilesin suppression lists (for more information more right there)**

Click on the progress bar to open a table with mailing statistics:



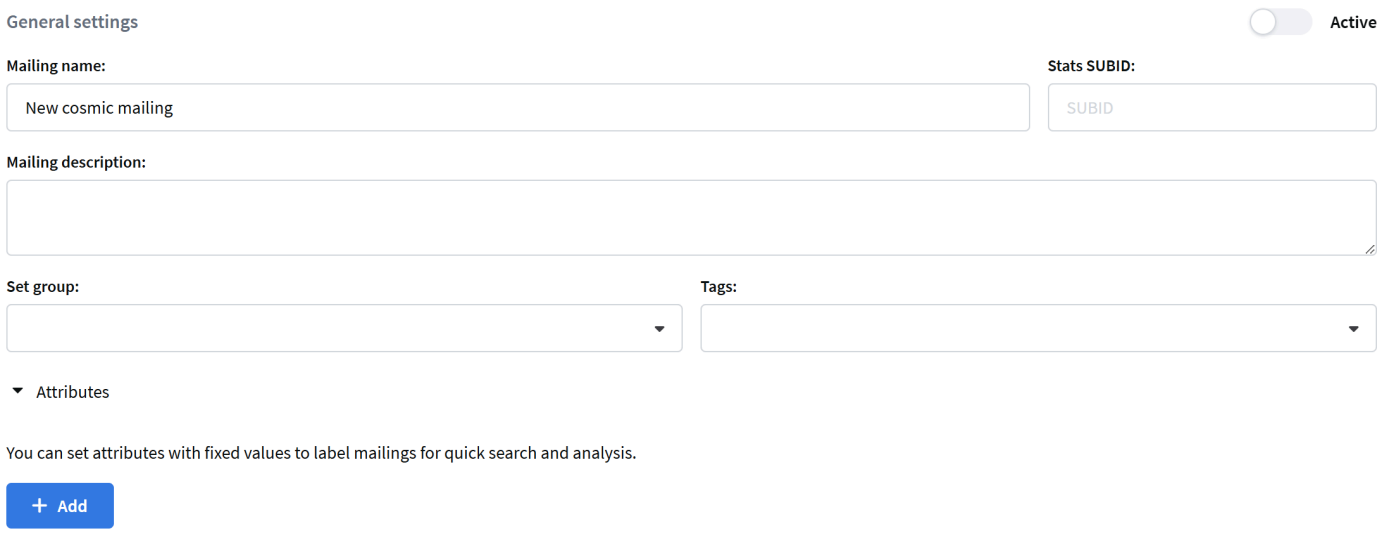
Trigger mailings

Description

	Trigger mailing is our ultimate tool for composing transactional messaging. In other words, if you select this mailing, a marketing message is automatically delivered after a specific event has occurred. Trigger mailings send one message to each subscriber who fits trigger launch conditions. It can be used as a standalone feature or as a part of a Scenario tool – a state of the art solution for composing marketing automation and creating customer journey paths. To create a new mailing, select Mailings → Triggers and click on the "Create" button, or you can use the mailings calendar. </td> </tr>

General settings

In the main settings, specify the name and description of the new mailing, and if needed, set the access group, add tags and attributes for additional grouping. You can also specify a statistics SUBID for the mailing here. In the upper right corner, there's a toggle that activates the trigger, putting it into standby mode:



Mailing settings

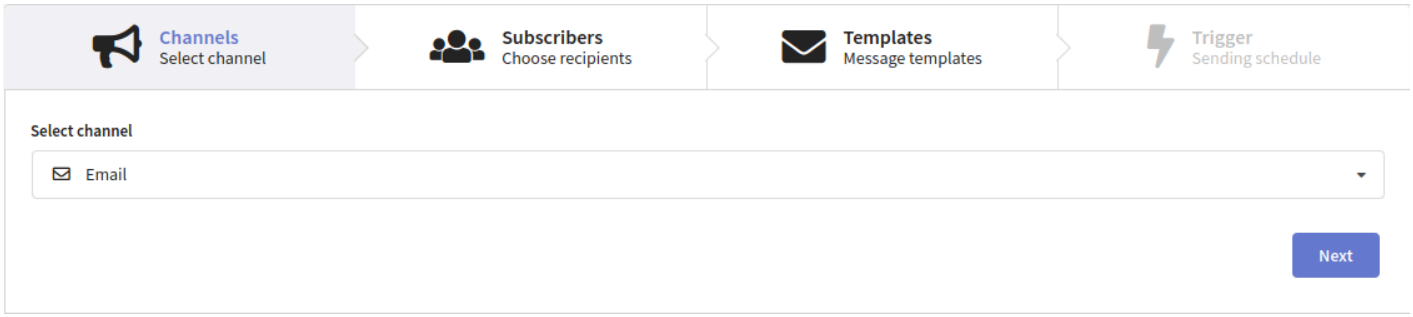
The following information is defined in mailing settings: communication channels, subscribers, templates with content, and schedule for mailing.

Channels

On the first tab, you can select a communication channel you want to use. Channel you want to use must be available in at least one subscriptions resource, which you can select on the next tab.

INFO

For Email sending, you can use the field with the main subscriber contact information – Email field.



Subscribers

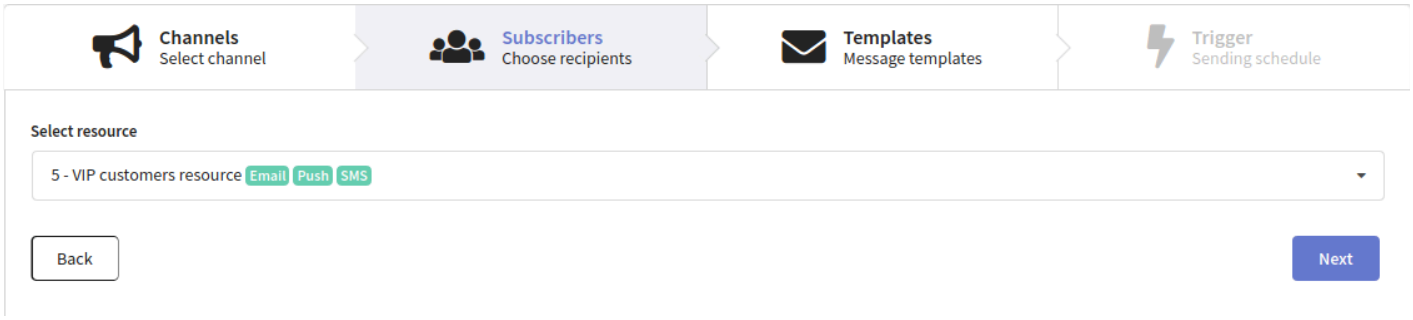
On this tab, select the resource form drop-down menu. There is no audience source in trigger mailings. The mailing audience depends on the event trigger.

INFO

Mailings are sent to customers who subscribe to a certain resource. Make sure that recipients are subscribed to a selected resource.

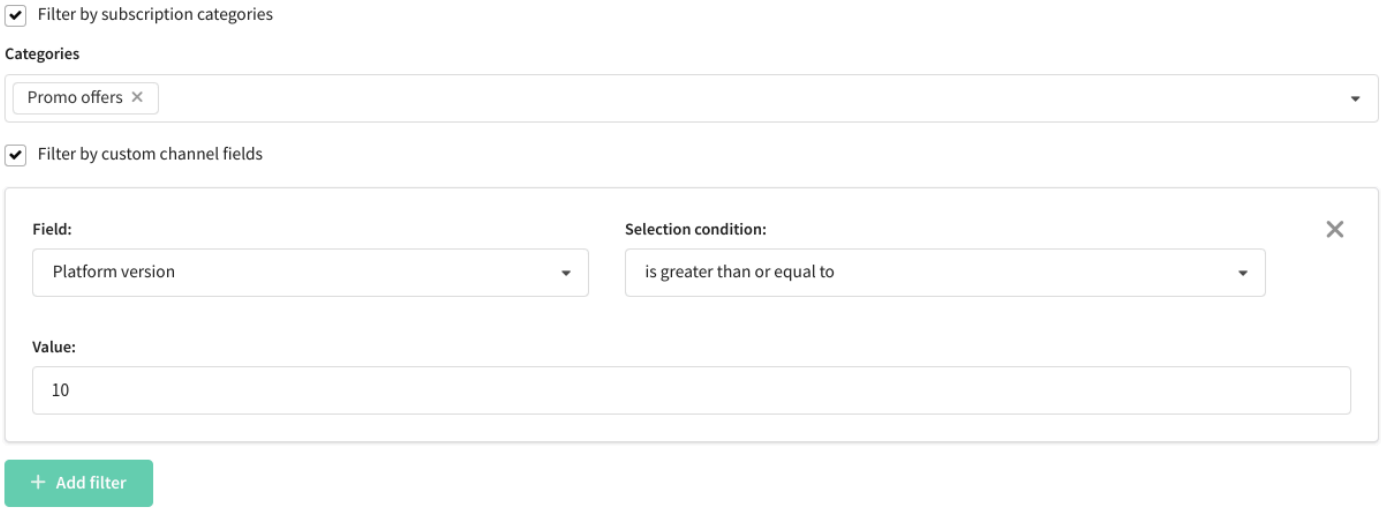
CAUTION

If database import is used as a trigger event, make sure that the database is available in the resource selected for the mailing. Otherwise, messages will not be sent.



Additionally, you can filter recipients by resource categories. To enable this functionality, you need to have categories created in the resource. In mailing settings, select the categories to which the profile should be subscribed to receive a message.

You can also filter recipients by custom channel fields. To do this, select either a standard or custom subscription field. In the Selection condition field, choose a logical operator and, if necessary, specify a value. You can combine rules using the logical AND (all conditions must be met) and OR (at least one condition must be met).



Templates

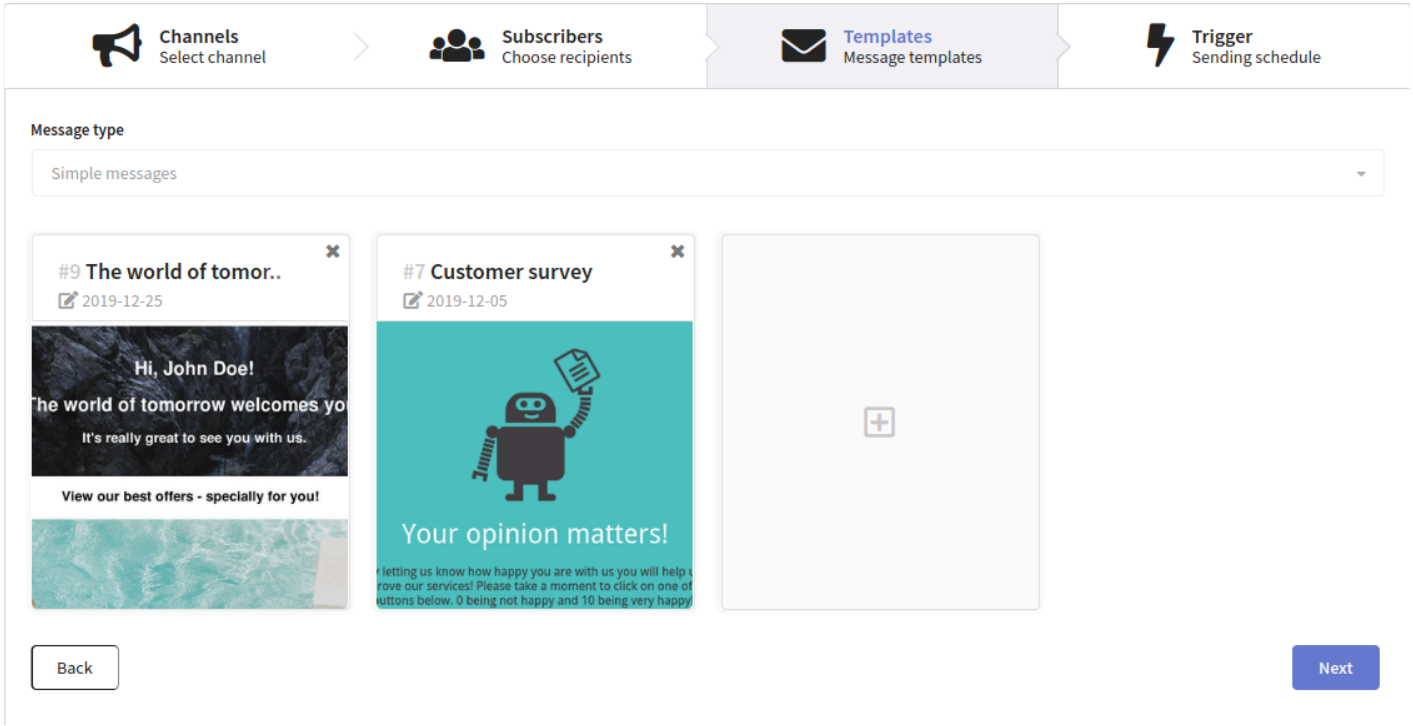
On this tab, you can select the content that you want to use in the mailing. In Altcraft, messages templates combine content for mailing by all available channels.

TIP

Please, make sure, that the template has content for the selected channel.

Click the "Add" button and select a prepared template. Any mailing type can use up to 8 templates. The mailing will send one message to each customer using selected templates in random order in equal parts.

Click on the template miniature to see a preview, or click on the template name to open the template editor:



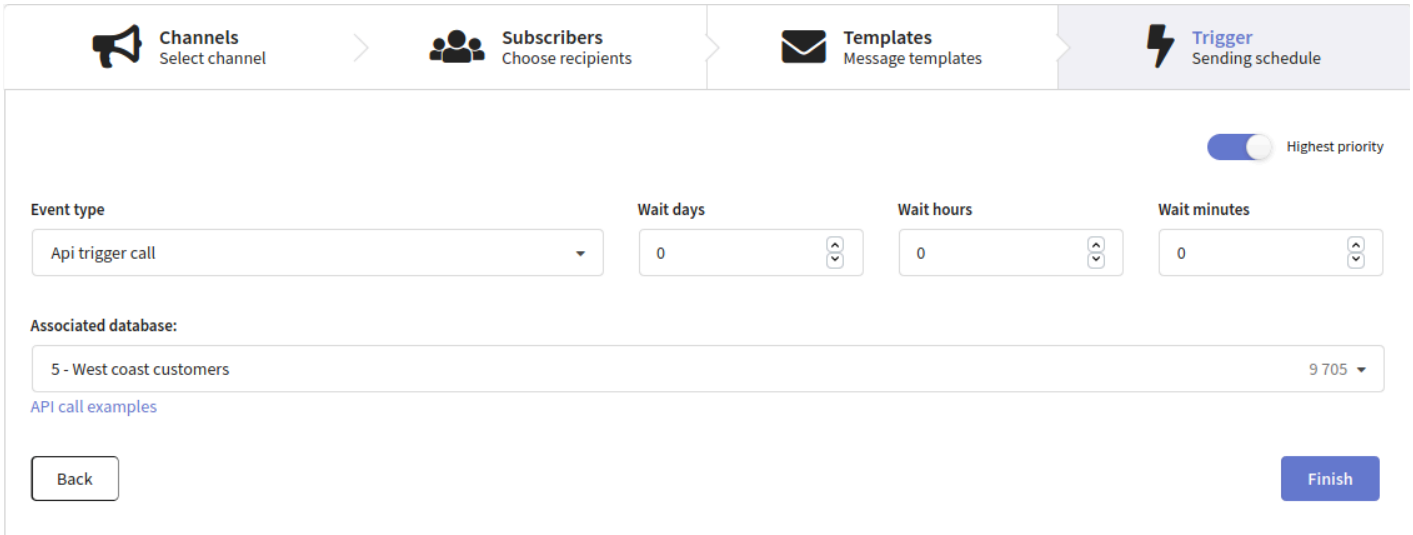
Trigger

Trigger mailings use the sender's priority channel to deliver messages instantly after a trigger event occurs.

You can also enable the Highest priority — ideal for urgent transactional messages. In this case, content generation tasks go to a dedicated priority queue in the message broker (`oxy_triggers_prior`), so the message is prepared faster.

After content is generated, the message is moved to the general sending queue, where priority is no longer considered — messages are sent in the order they arrive. The only exception is the email channel with AKMTA. In this case, a dedicated virtual queue is used on the sender side to deliver high-priority messages faster. For all other channels, priority doesn't affect the sending order.

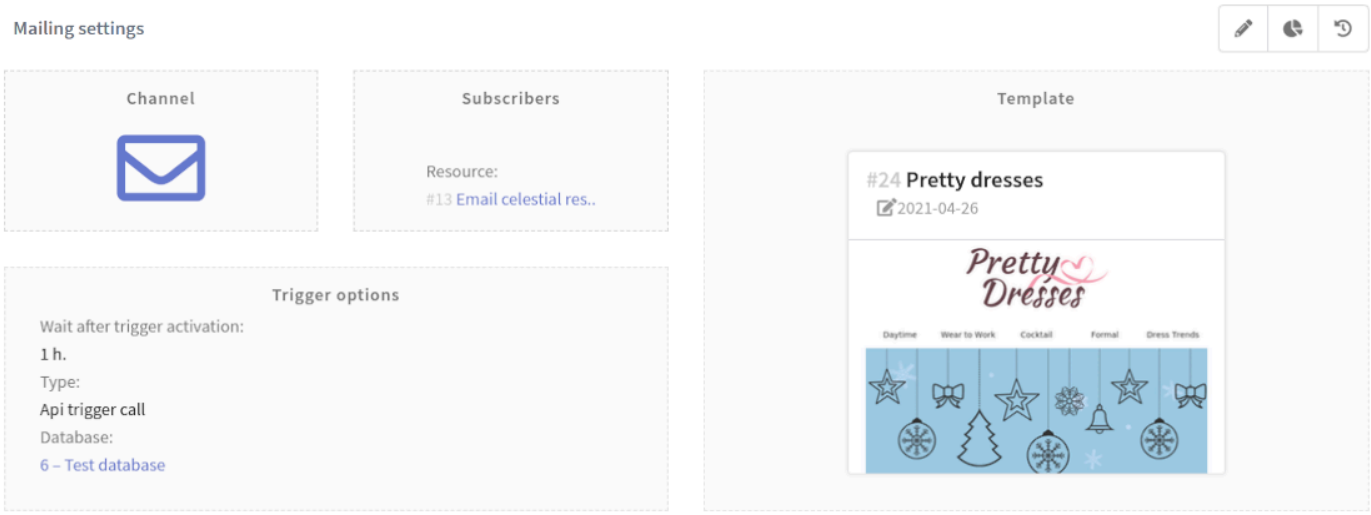
Additionally, you can set a delay for each trigger in days, hours, and minutes.



TIP

The types of trigger events are described in the next paragraph of this article.

When you've done with settings, click "Finish". The parameters you've chosen will be displayed in the "Trigger option" section. Use the buttons in the upper right corner to edit settings, check the log of the mailing or see the channel report.



NOTE

In the mailing log for trigger mailings, the Total time spent field may display a value of 30 seconds, even if the message was sent almost immediately after the triggering event occurred.

Time	Type	Value
2026-01-19 11:29:53	Trigger activated	
2026-01-19 11:29:53	Database ID	6
2026-01-19 11:30:23	Number of subscribers received from the source	1
2026-01-19 11:30:23	Number of subscribers found	1
2026-01-19 11:30:23	Number of email recipients	1
2026-01-19 11:30:23	Trigger finished	
2026-01-19 11:30:23	Mailing finished	
2026-01-19 11:30:23	Total time spent	30.066781361s

For trigger mailings, the Total time spent value is recorded when the mailing is completed, not at the moment a specific message is actually sent. During this interval, the trigger continues to accept new events, after which the campaign is completed and aggregated statistics for the period are recorded.

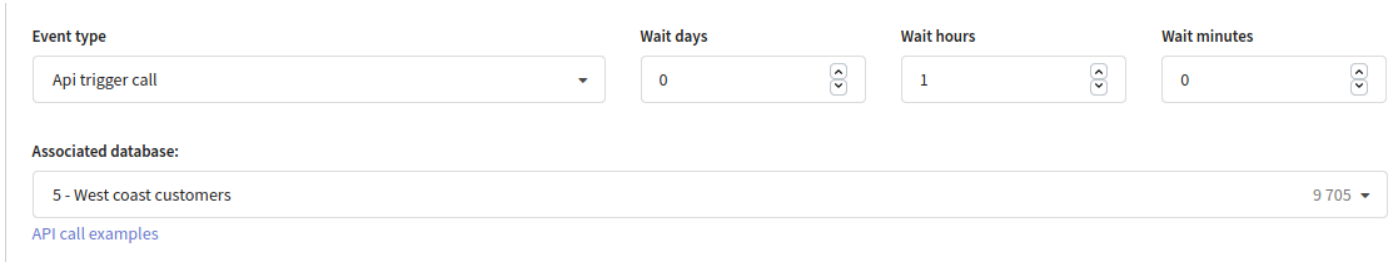
As a result, the same time value is shown in the log regardless of when a particular message was actually sent.

To find the actual time of message generation and sending, use the profile history, where the corresponding events are recorded.

Trigger types

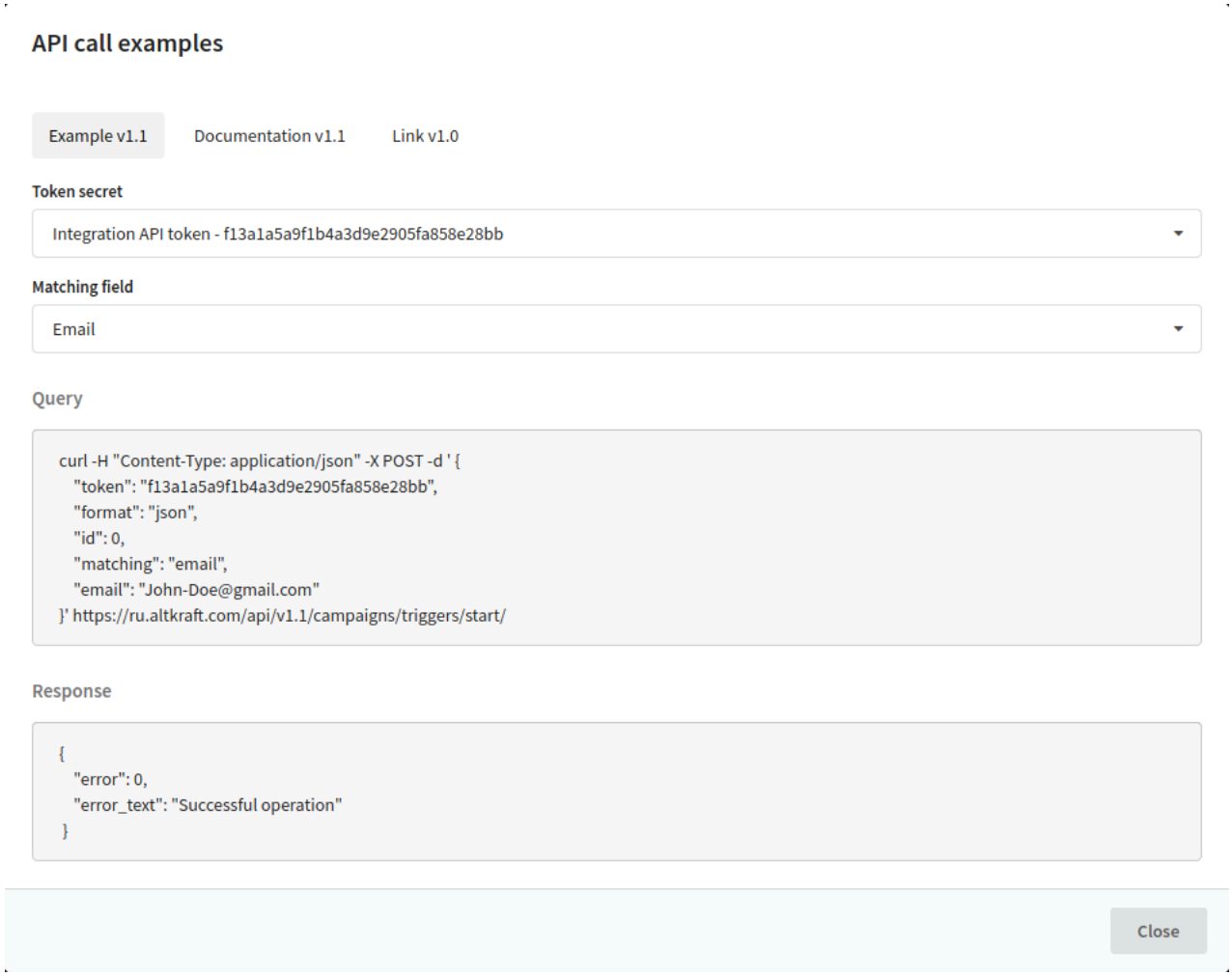
API trigger call

API-triggered mailings are effective means of integration with your websites, apps, CRMs or other network software. They can be assigned to specific steps of your digital workflow. Such triggers can work with one specific profiles database.



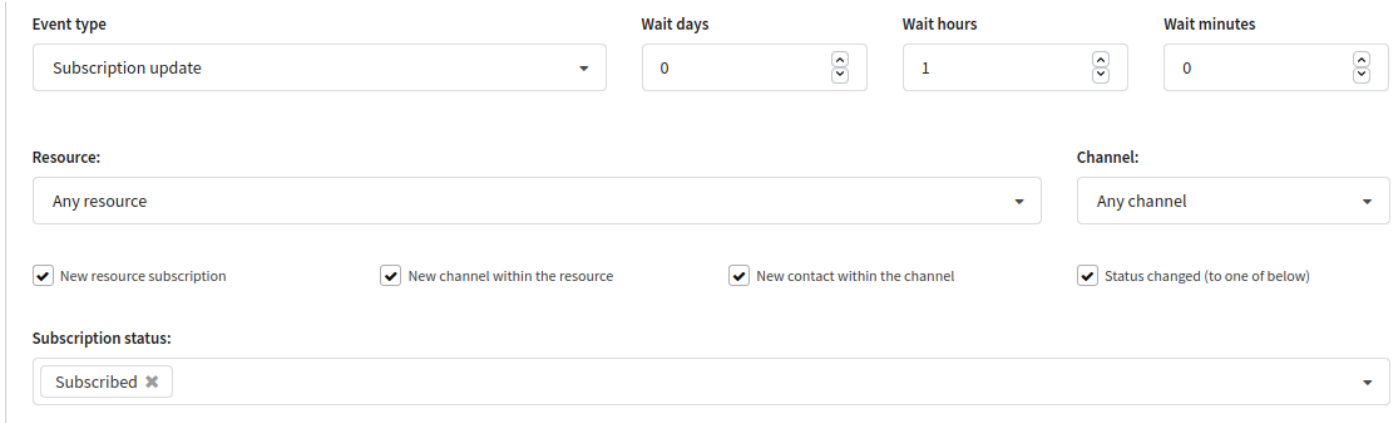
There are two options of API for trigger mailings:

- **v1.0 is a simple GET request with no additional parameters. Profiles are matched by md5 hash, xxhash or email. Request URL is generated when you save a trigger mailing and can be then implemented by your developers.**
- **v1.1 uses POST requests with JSON body. It is better for security: the access token is in the request body, and these requests can contain custom parameters. You can tune Example v1.1 as you need and see [API trigger call documentation] for further details. If you need to delegate configuration to your IT department — download the example and forward it.**



Subscription update

The mailing will be activated when profile subscriptions are updated.



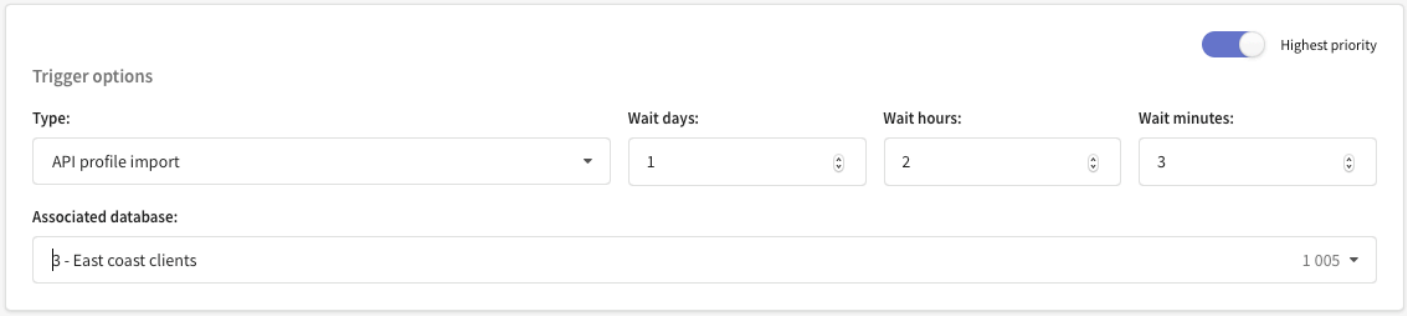
- **New resource subscription** — when a profile gets subscribed to a resource for the first time.
- **New channel within the resource** — when a profile gets subscribed to another channel of communication within an existing resource for the first time.
- **New contact within the channel** — works for each new contact entry within a channel of a resource, including the first contact for this channel.
- **Status changed** (to one of the below) — when subscription status is changed for a profile.
- **The first 3 options are selected** — when any subscription entry is added to a profile.
- **No option selected** — nothing would happen.

CAUTION

Please, take note that this trigger works only for subscriptions with a “subscribed” status.

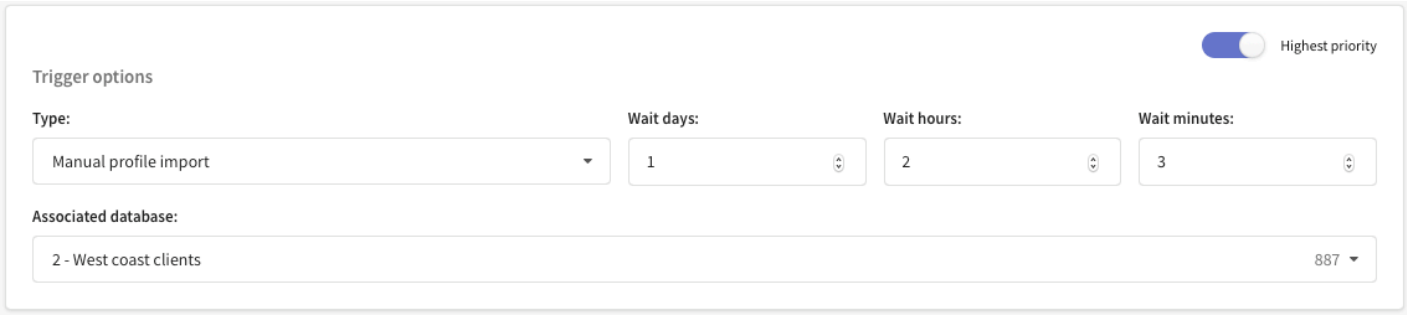
API profile import

The mailing will be activated when a new profile is created by your site, app or CRM API request.



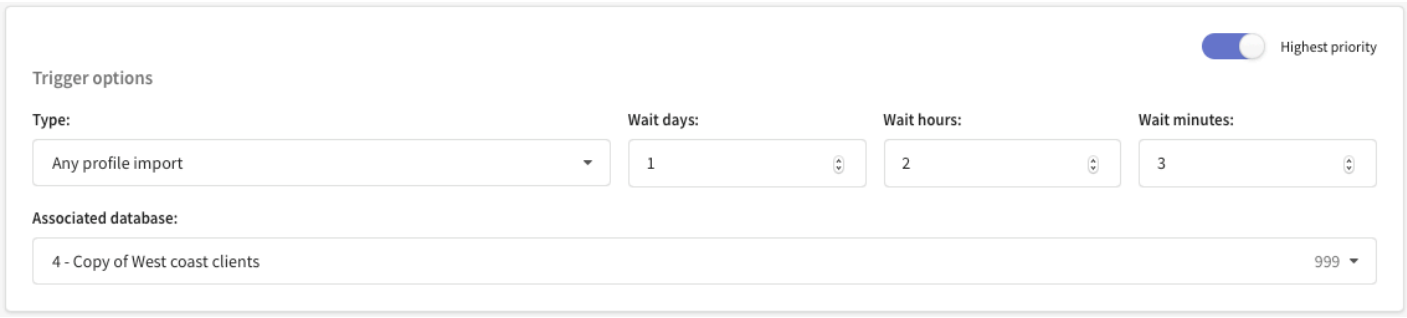
Manual profile import

This trigger will activate a mailing when you add customer profiles manually.



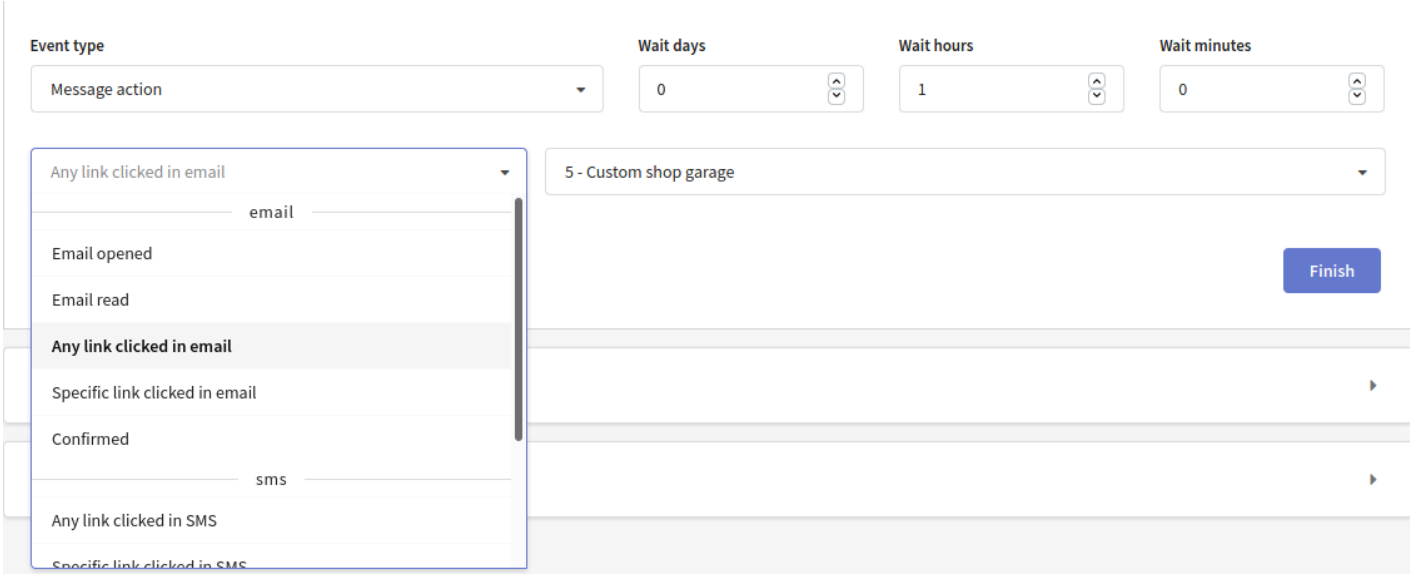
Any profile import

This trigger will activate a mailing when you import a profile to a certain database by any means.



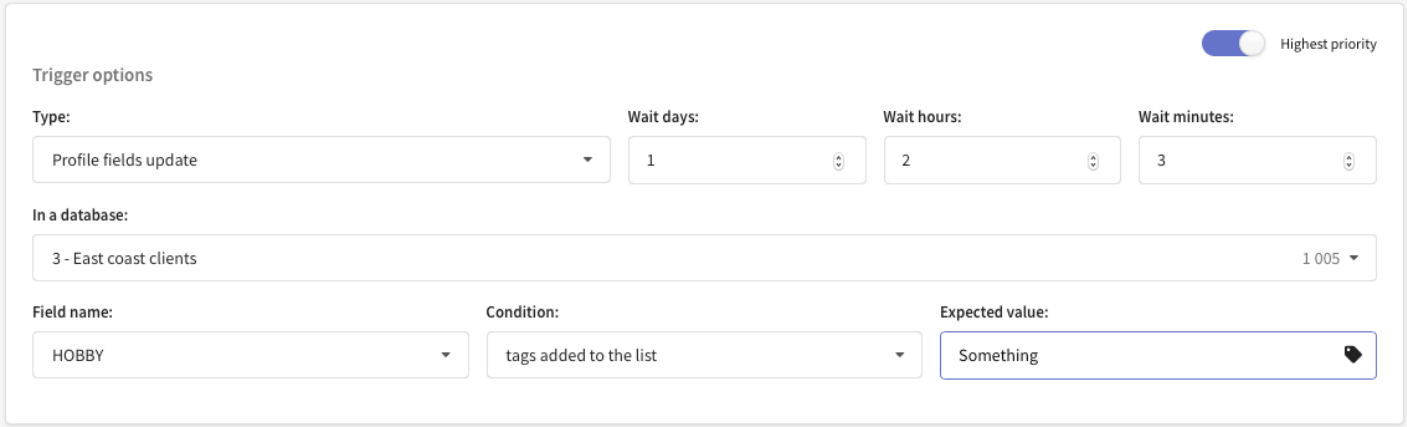
Message action

This trigger is based on default Altcraft message tracking. Mailings can be activated by various systems and message recipient actions. For link clicks, you can specify a link or select Any link clicked.



Profile fields update

Data update trigger detects changes in a customer profile. When a field value is changed to meet specific conditions mailings will be launched.



Conditions vary depending on the type of field:

Condition	Field type	Description
changed	**all types**	**Any change in the profile field**
equal	**all types except *tags, object, array of objects***	**The new value is equal to the expected one**
not equal	**all types except *tags, object, array of objects***	**The new value is not equal to the expected one**
not empty	**all types except *tags***	**The field takes any non-empty value**
empty	**all types except *tags***	**All values have been deleted for the profile field**
tags added to the list	***tags***	**The list of tags is formed in the trigger**
tags removed from the list	***tags***	**The list of tags is formed in the trigger**
property changed	***object***	**The property of the field whose changes should be observed is specified.**
property equal	***object***	**The field property and the expected value are specified.**
property not equal	***object***	**The field property and the expected value are specified.**

Condition	Field type	Description

element's property changed	***array of objects***	**The path to an array element is specified. If the property of this element changes, the trigger will be activated.**
		
		When choosing the path to the element property, you also select the data type to which the compared value should be converted. Comparison is available for data types such as string, int, float, and bool. It is not possible to compare complex data types (object, array). In the example below, you won't be able to track changes in the properties of elements `key_1` and `key_4`.
		`[`
		`{`
		`"key_1": [1,2,3,4,5],`
		`"key_2": 2`
		`},`
		`{`
		`"key_3":"value",`
		`"key_4": [1,2,3,4,5]`
		`}`
		`]`
		Please note that the dropdown list contains available paths to the field property, but the index of any element is set to 0 by default — `$[0].key_3`. To specify the path to the desired element, you can manually enter the index in the JSONPath format — `$[1].key_3`.

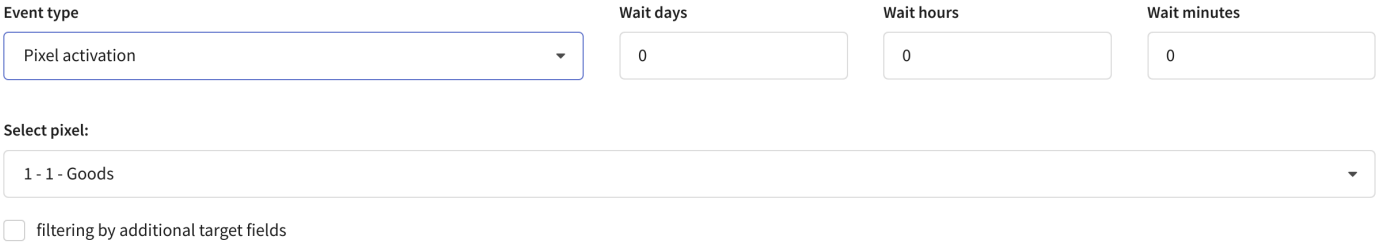
element's property equal	***array of objects***	**The field property and the expected value are specified.**
element's property not equal	***array of objects***	**The field property and the expected value are specified.**
property added	***array of objects***	**An element is added to the array.**
property deleted	***array of objects***	**An element is deleted from the array.**

CAUTION

This type of trigger can be used with links in templates that are configured to update profile fields. Please note that if some other triggers are attached to the link clicks (see above), they will also work.

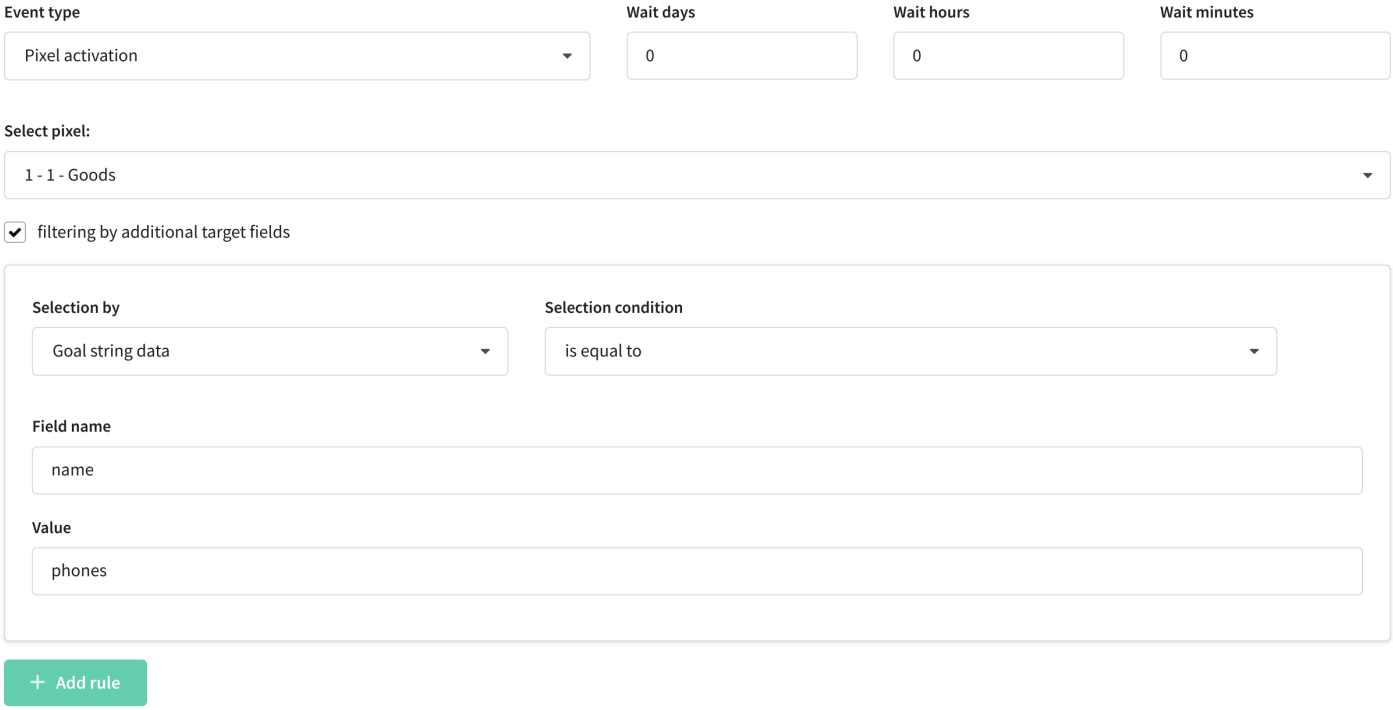
Pixel activation

Altcraft pixel is a highly customizable tracking technology stack, that can be easily integrated with your digital ecosystem. Pixel activation can as well trigger a mailing: when your customers browse your website or use your applications pixels will track their behavior and trigger mailings and workflows when it is needed:



It is also possible to track not only the general event of pixel activation. When activated, the pixel can register goals. Within the goal registration event, you can pass the goal value and other additional information to the platform.

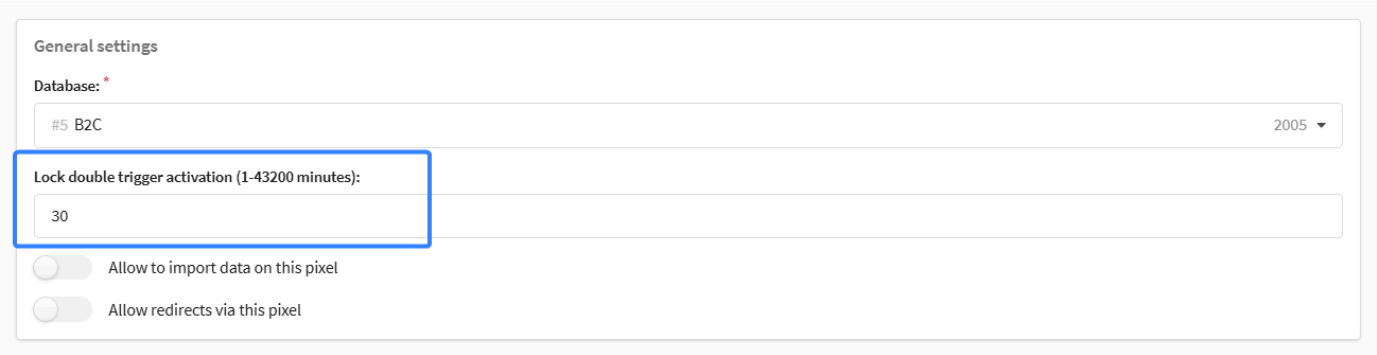
You can specify selection rules for a goal and its additional fields, for example, the values of some fields or the location of the client. This allows you to more accurately customize a mailing for a specific group of customers or specific products:



When sending messages in templates, you can add variables with data about the pixel trigger event that launched the mailing. You can learn more about these variables here.

TIP

If the trigger does not activate in mailings or scenarios, although the trigger conditions are met, it is recommended to check pixel trigger interval.

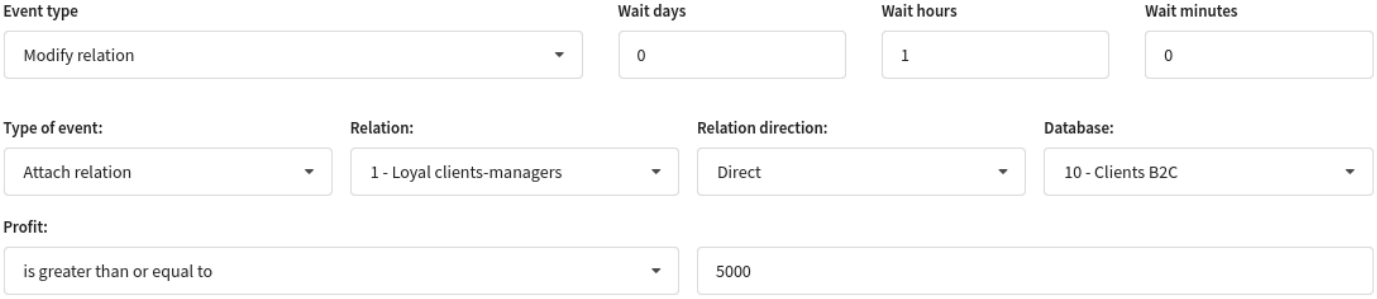


Modify relation

The event can be triggered when the relations between the profiles are changed: both direct and reverse.

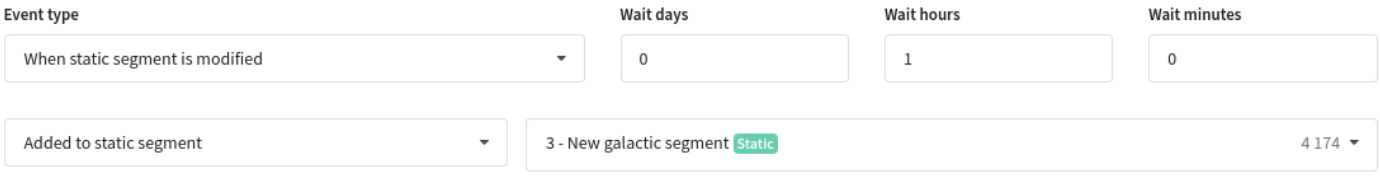
The event types:

- **attach relation,**
- **change relation,**
- **reaching relation "x" values,**
- **delete relation.**



When static segment is modified

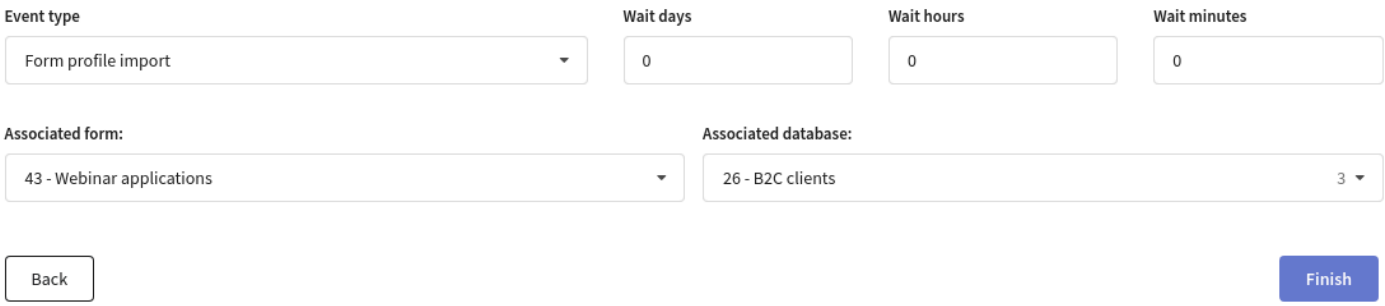
The event can be triggered when a change occurs in a static segment: a profile can be added to the selected static segment or removed from it.



Form profile import

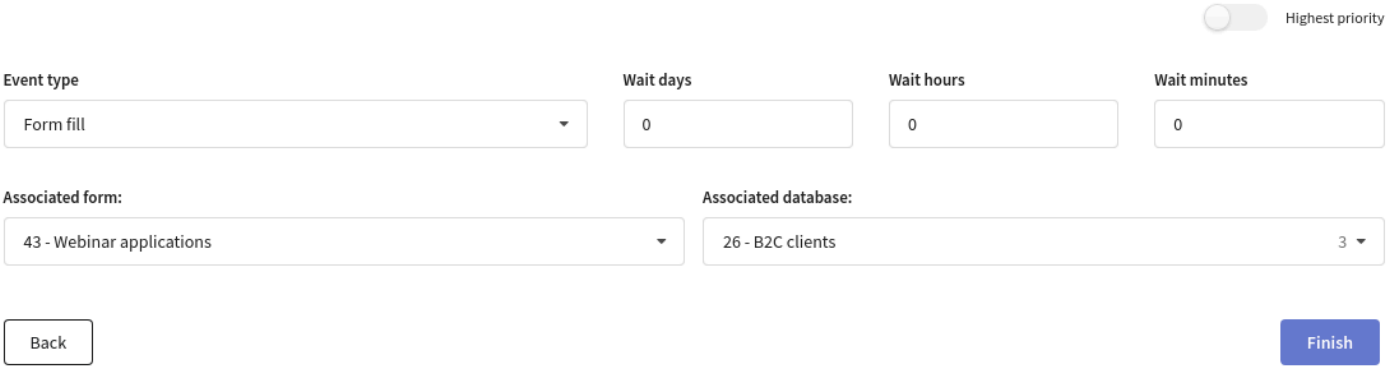
The trigger is activated if the profile has filled out the form and for this reason, has been added to the database. To do this, when setting up the form, activate the Import Profiles option on the Actions tab.

Select the form and a profile database:



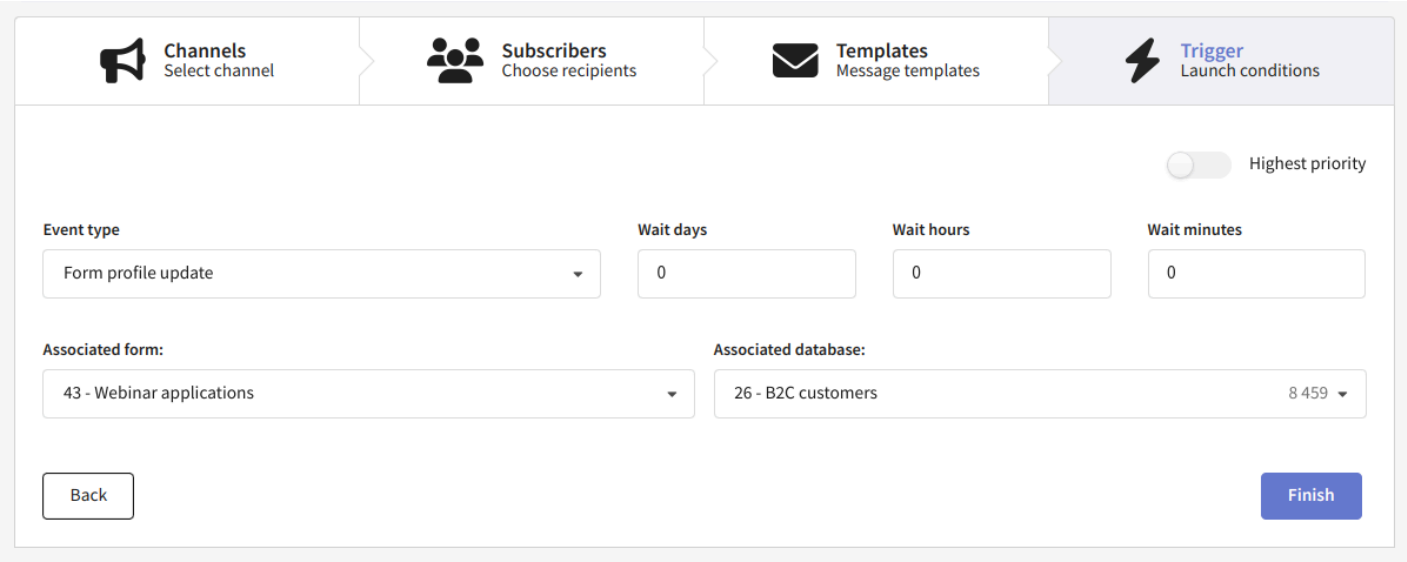
Form fill

The trigger is activated only if the profile fills out the form after clicking a link in an email. Select the form and the database containing the profiles that will fill out the form:



Form profile update

The trigger is activated when the data of a profile, which is already in the database, is changed as a result of filling out a form. Select a form. If you want to track changes to profile data only in a specific database, also specify the database.



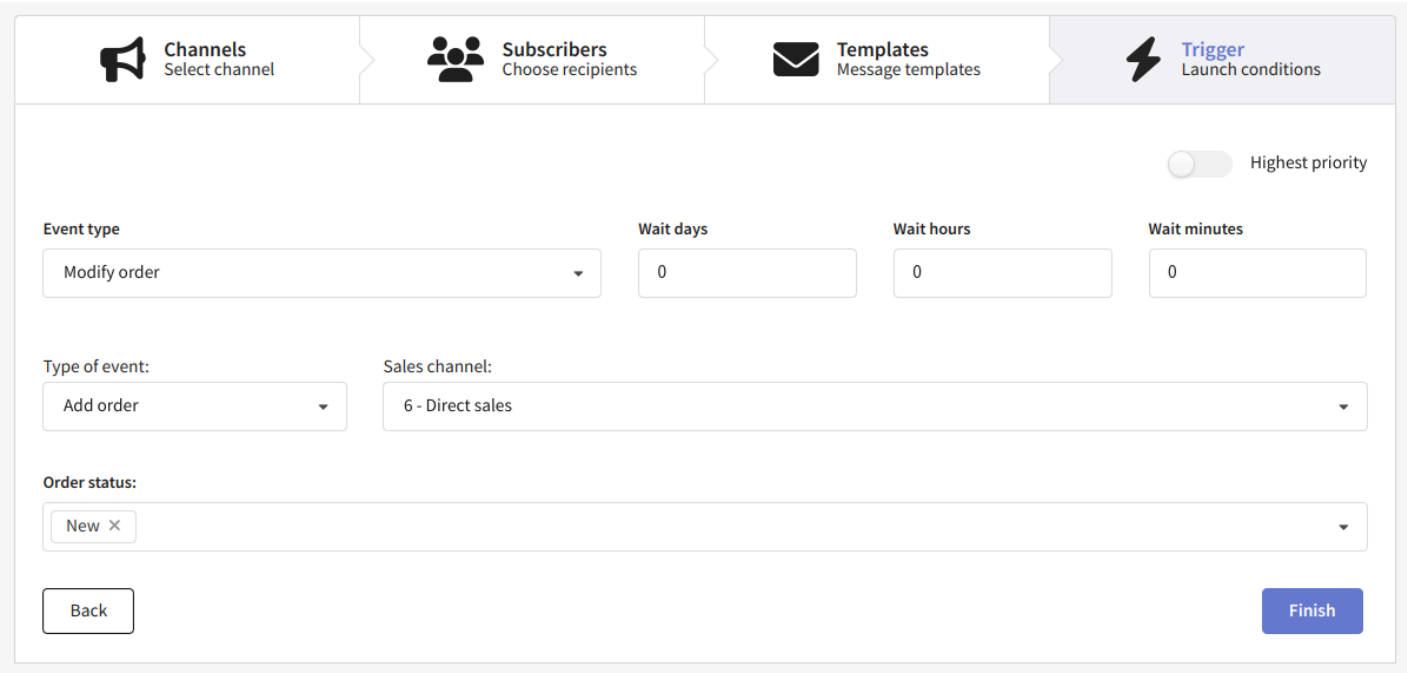
Modify order

The trigger will br activated if order in the Market has been changed. Select a sales channel and an event type. Two types available:

- **add order (a new order has been added to the Market)**
- **change order (the status of an existing order has changed)**

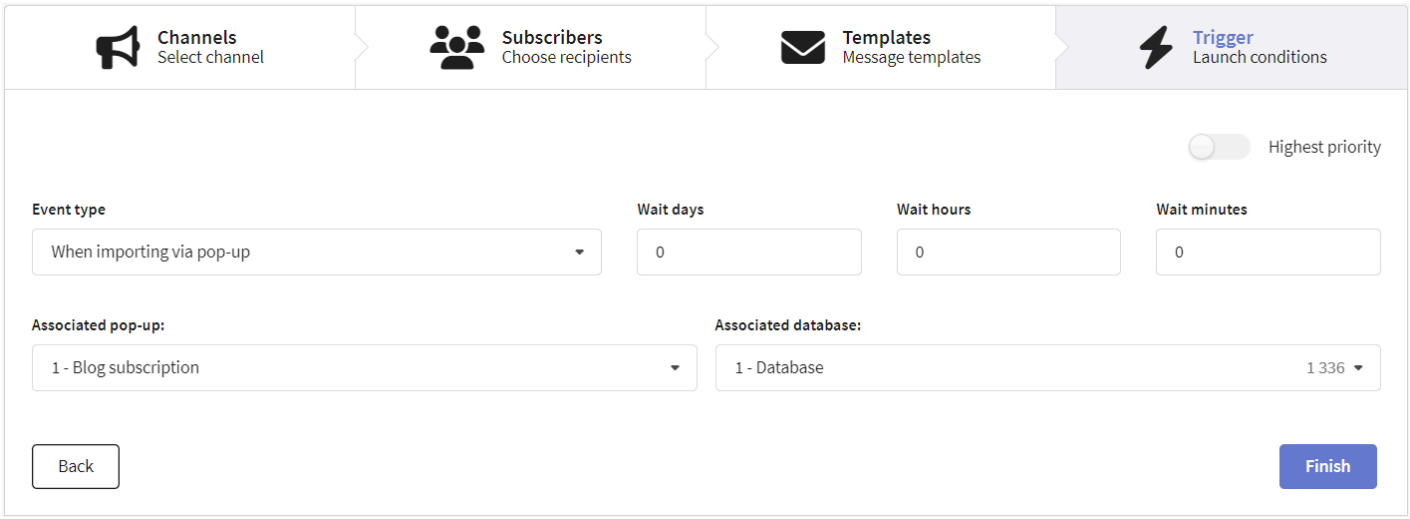
Available order statuses:

- **new**
- **delivered**
- **paid**
- **cancelled**
- **partial refund**
- **in delivery**
- **in process**



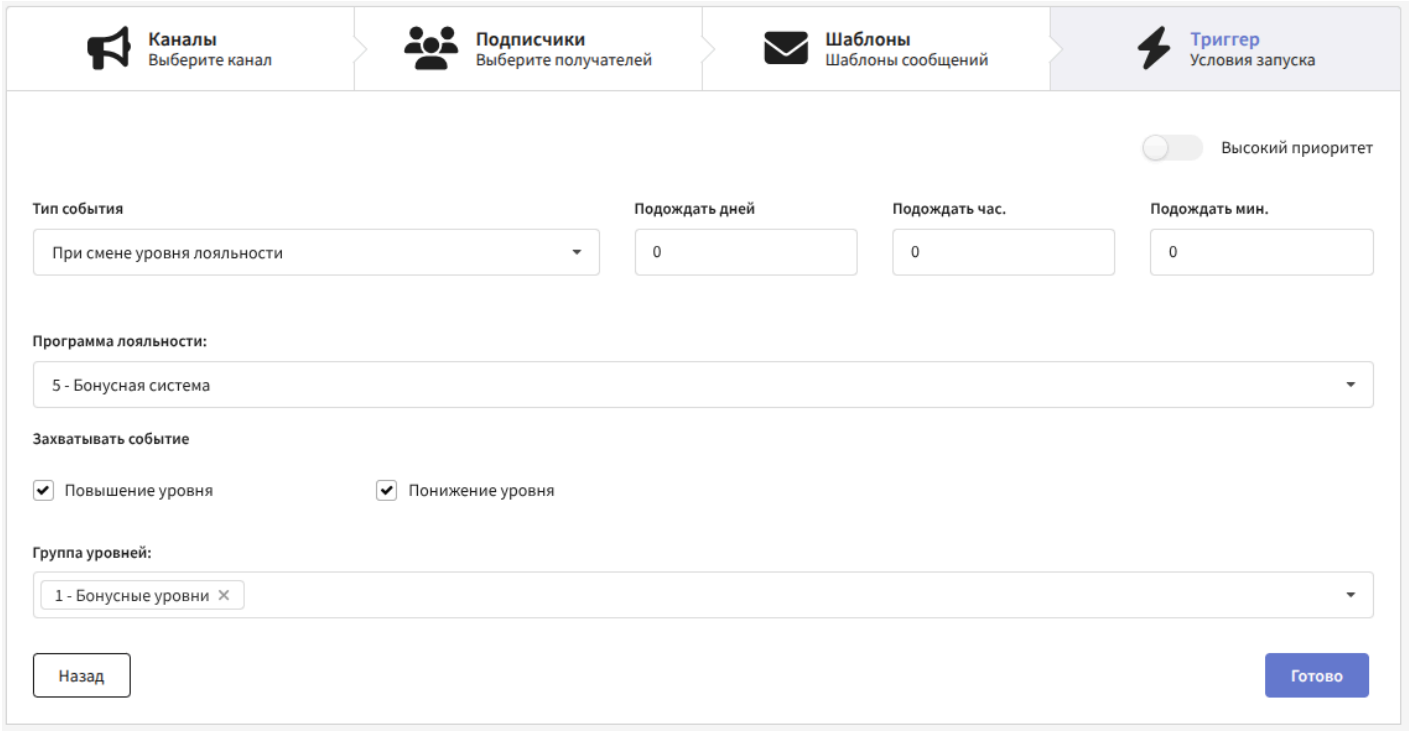
Import via pop-up

The trigger will be activated if the profile data gets into the database as a result of filling out and submitting the popup form. Select the popup you want to track:



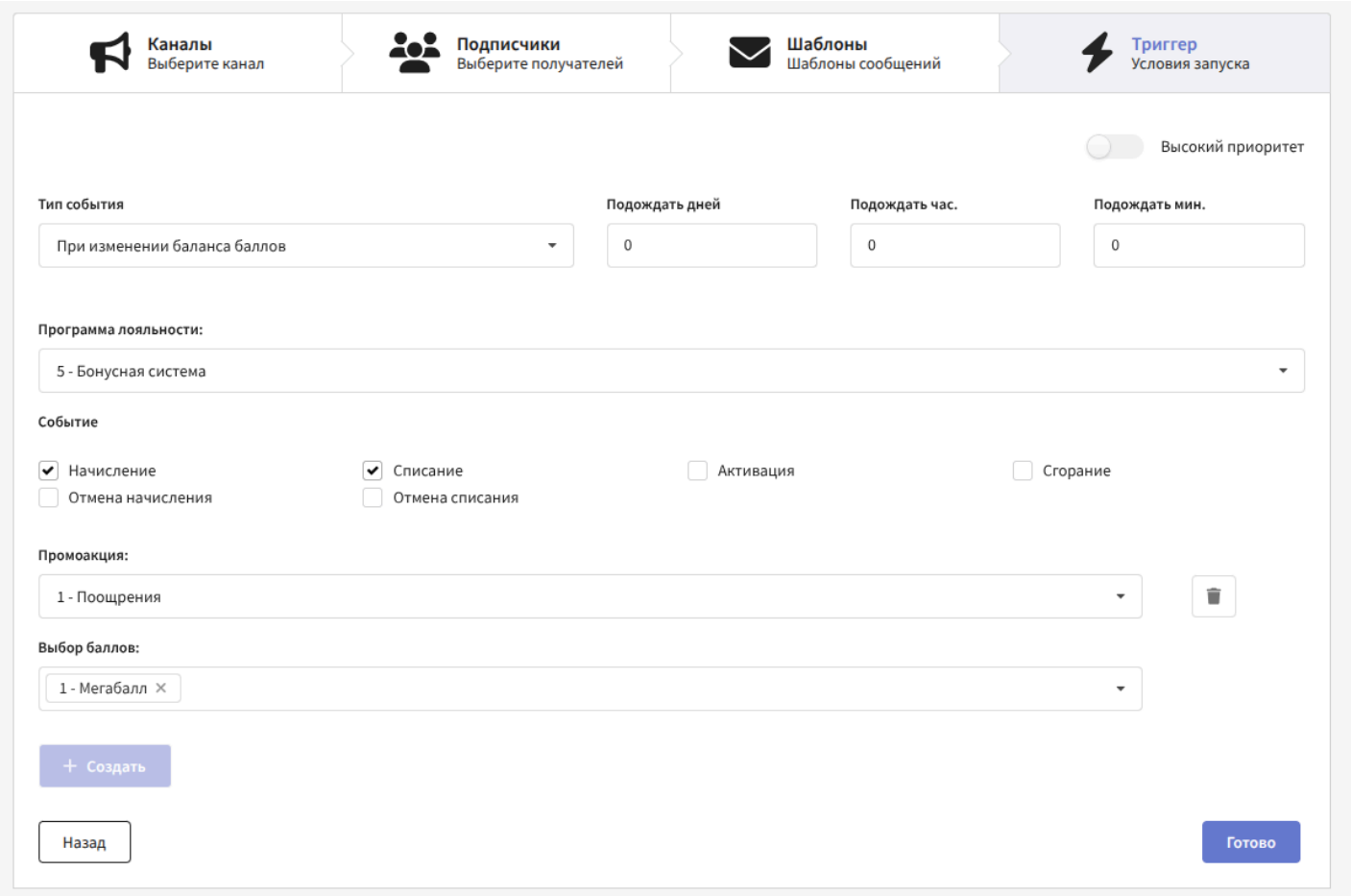
On loyalty level change

The trigger fires when a profile moves to a different level within a loyalty program.



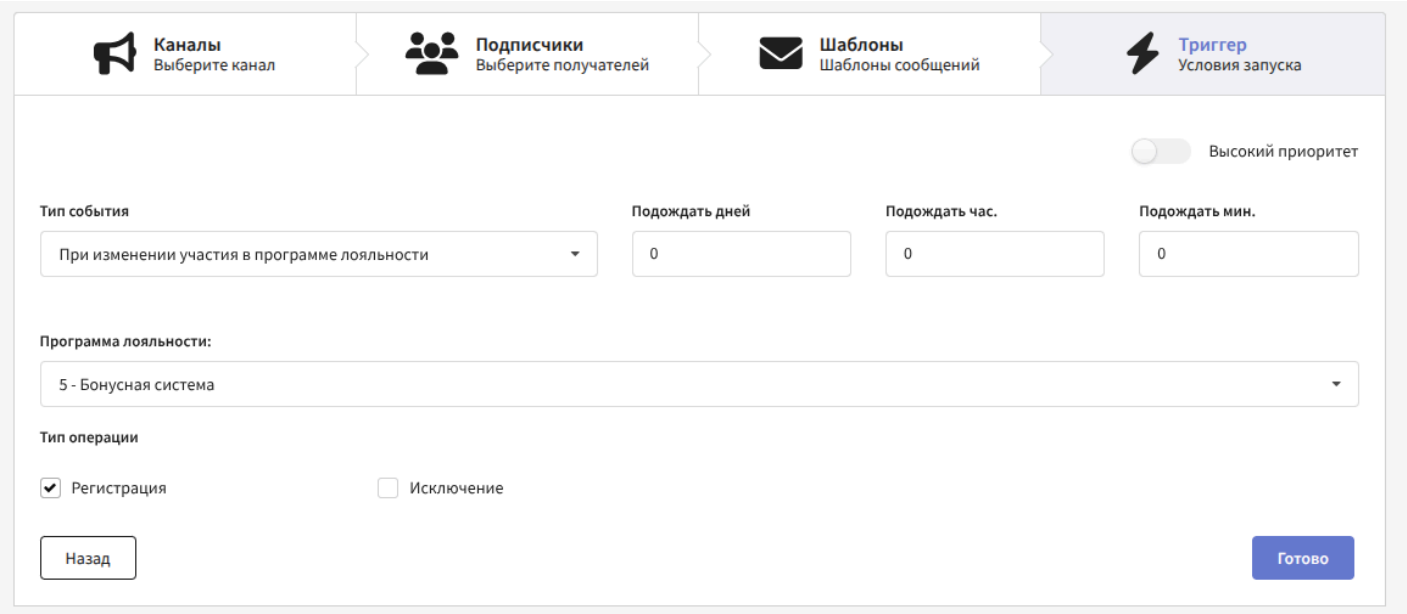
On points balance change

The trigger fires when the number of bonus points on a profile changes in a loyalty program.



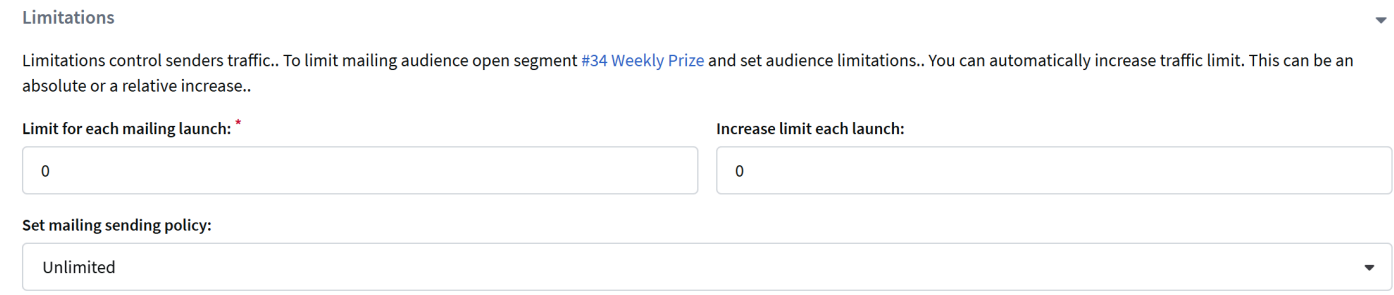
On loyalty program participation change

The trigger fires when a profile is registered in or removed from a loyalty program.



Limitations

As no mass audience is processed by trigger mailings — no limitations are even necessary except for communication policies, limiting individual messaging per time period.



Smart settings

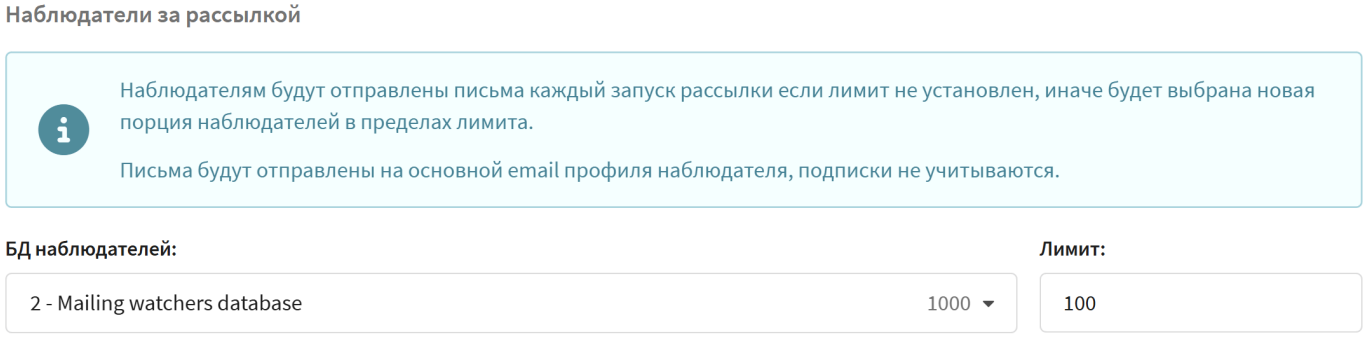
The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages from the platform's real senders).**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify each domain on a separate line.**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients based on subscription data.**

Mailing watchers

You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mailing recipients. To do this, you need a watchers database. It is created the same way as a regular database, but you must enable the corresponding option in General settings.

In Smart Settings, select the watchers database and set the message sending limit for each mailing launch:



Message Time-To-Live (TTL) Restriction

Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the delivery queue. If the message is not delivered within this time, sending stops and the message is removed from the queue.

TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is related to a promotion, an event reminder, a one-time code, or an offer that quickly loses relevance.

To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, hours, minutes, or seconds.

After the specified time expires, the system stops delivery attempts for the undelivered message and registers a delivery failure event.

TIP: Message TTL specifics

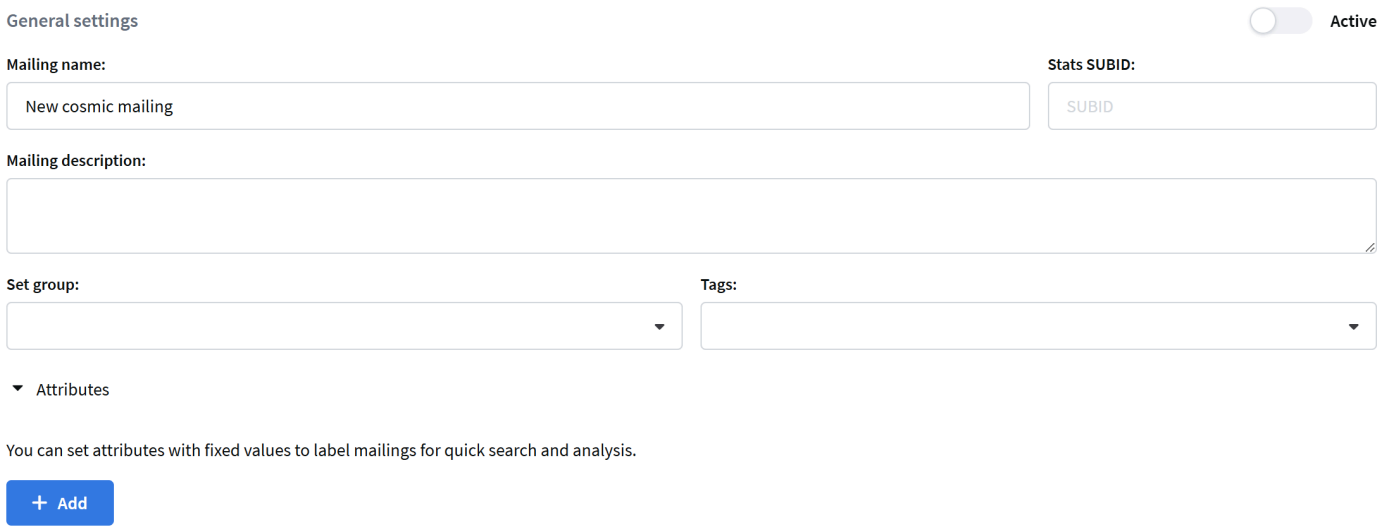
- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration parameters.**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API parameter > the value specified in Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefore, even within a single mailing, different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to deliver the message.**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and custom channels.**

Multivariate testing

	Multivariate testing, also known as MVT, or A/B/n testing, is a mailing that allows you to send multiple message templates (up to 8) to a part of the audience to compare their effectiveness. After that, the winning template is selected and sent to the rest of the audience. When the message is sent to each of the recipients, the mailing will end in the same way as a broadcast. To create a new multivariate test mailing, select Mailings → Multivariate testing and press the + Create button. Also, you can use the mailings calendar to create a new multivariate testing mailing. </td> </tr>

General settings

In the main settings, specify the name and description of the new segment, and if needed, set the access group, add tags and attributes for additional grouping:



Mailing settings

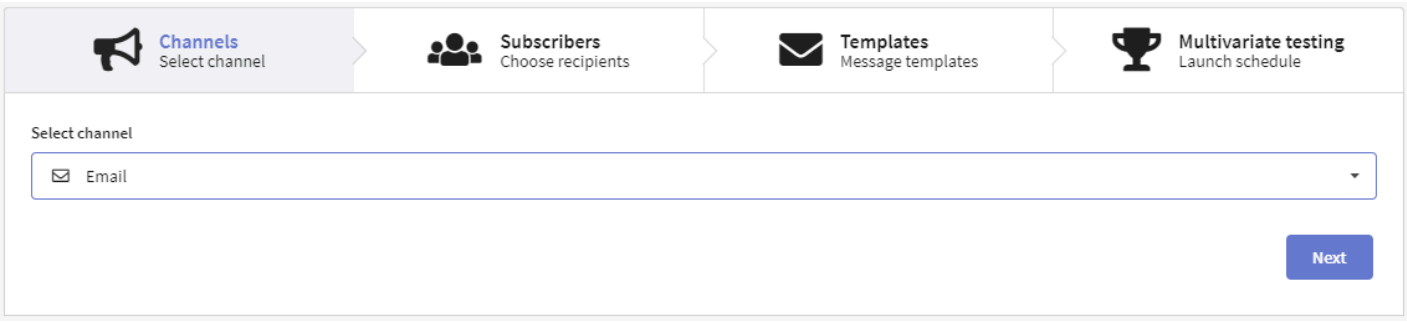
The following information is defined in mailing settings:

- **communication channels (Email, SMS or Push),**
- **subscribers,**
- **several templates with content,**
- **audience size for testing,**
- **selection of winner template.**

Channels

On the first tab, you can select the communication channel you want to use. This channel must be available in at least one subscription resource. The resource is selected on the next tab.

> NOTE
>
> **For Email sending, you can use the field with the main subscriber contact information — Email field.**



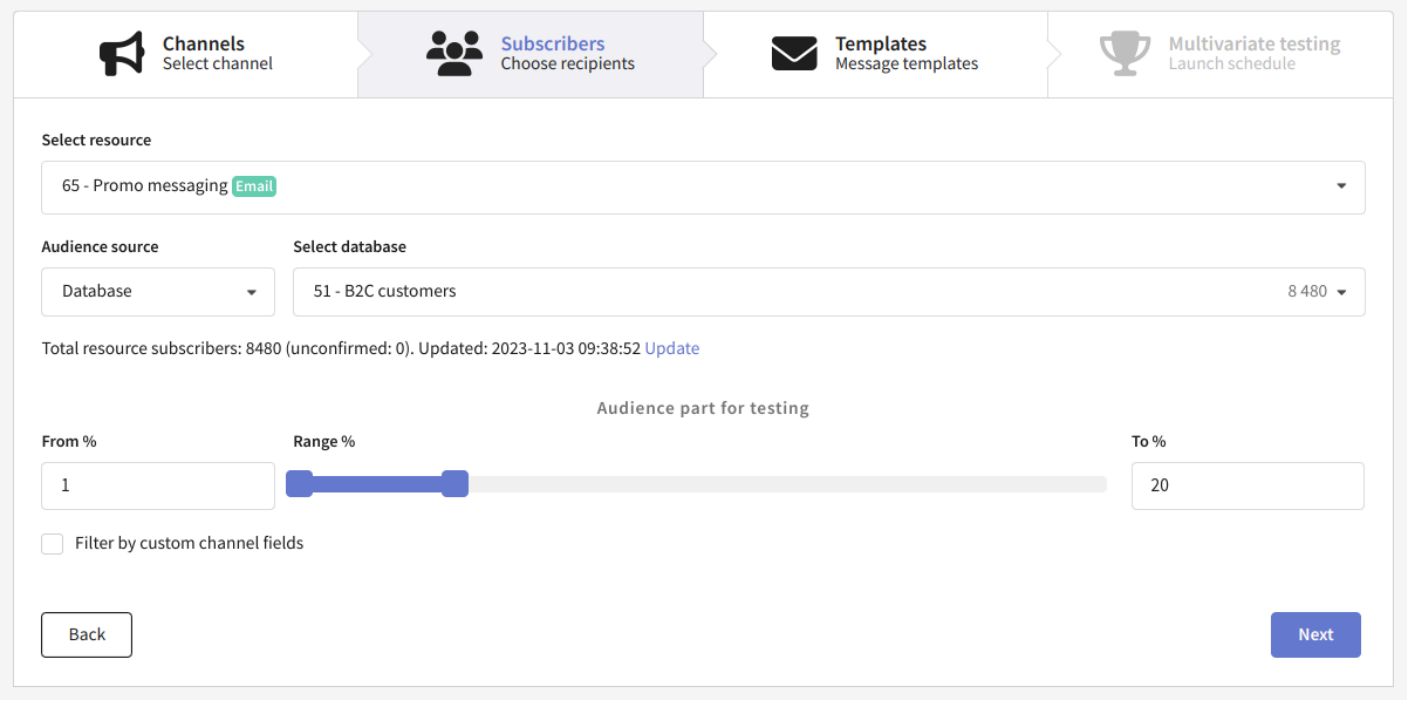
Subscribers

On the Subscribers tab you can choose a resource to send messages from and an audience you want to communicate to. It can be a database or a segment.

TIP

Mailings are sent to customers who subscribe to a certain resource. Please, make sure that message recipients are subscribed to the resource before mailing.

In the Audience part for testing block, you set an audience part, that will receive several content kinds for comparing templates efficiency.



Templates

On this tab, you can select the content that you want to use in the mailing. In Altcraft, messages templates combine content for mailing by all available channels.

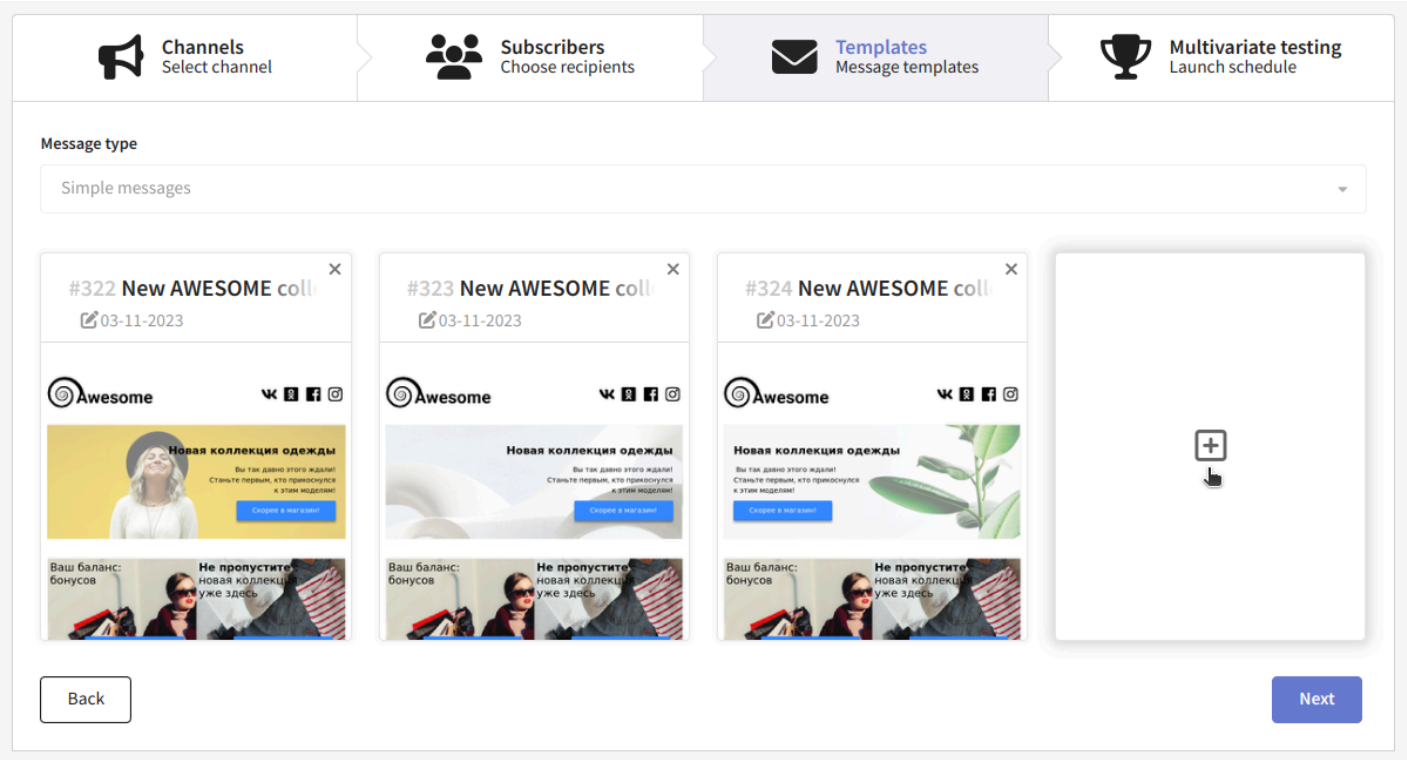
TIP

Please, make sure, that the template has content for the selected channel.

Press Add button

and select the template. Any mailing type can use up to 8 templates. A/B testing mailing will send one message to each customer using selected templates in random order in equal parts.

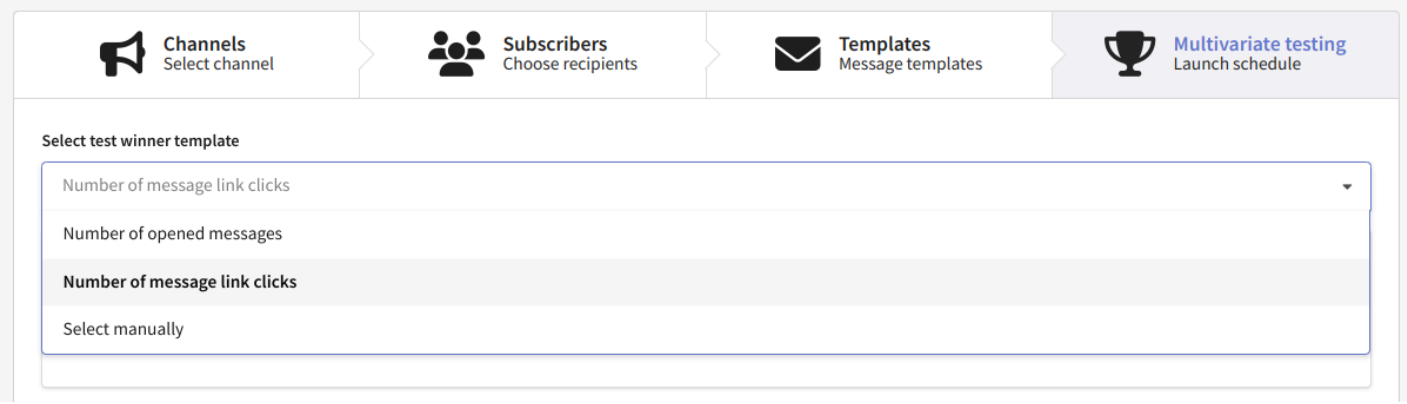
Click on the template miniature to see a preview, or click on the template name to open the template editor:



Test settings

On this tab, you can configure how the winning template will be selected. The following options are available:

- **Number of opened messages**
- **Number of message link clicks**
- **Select manually**

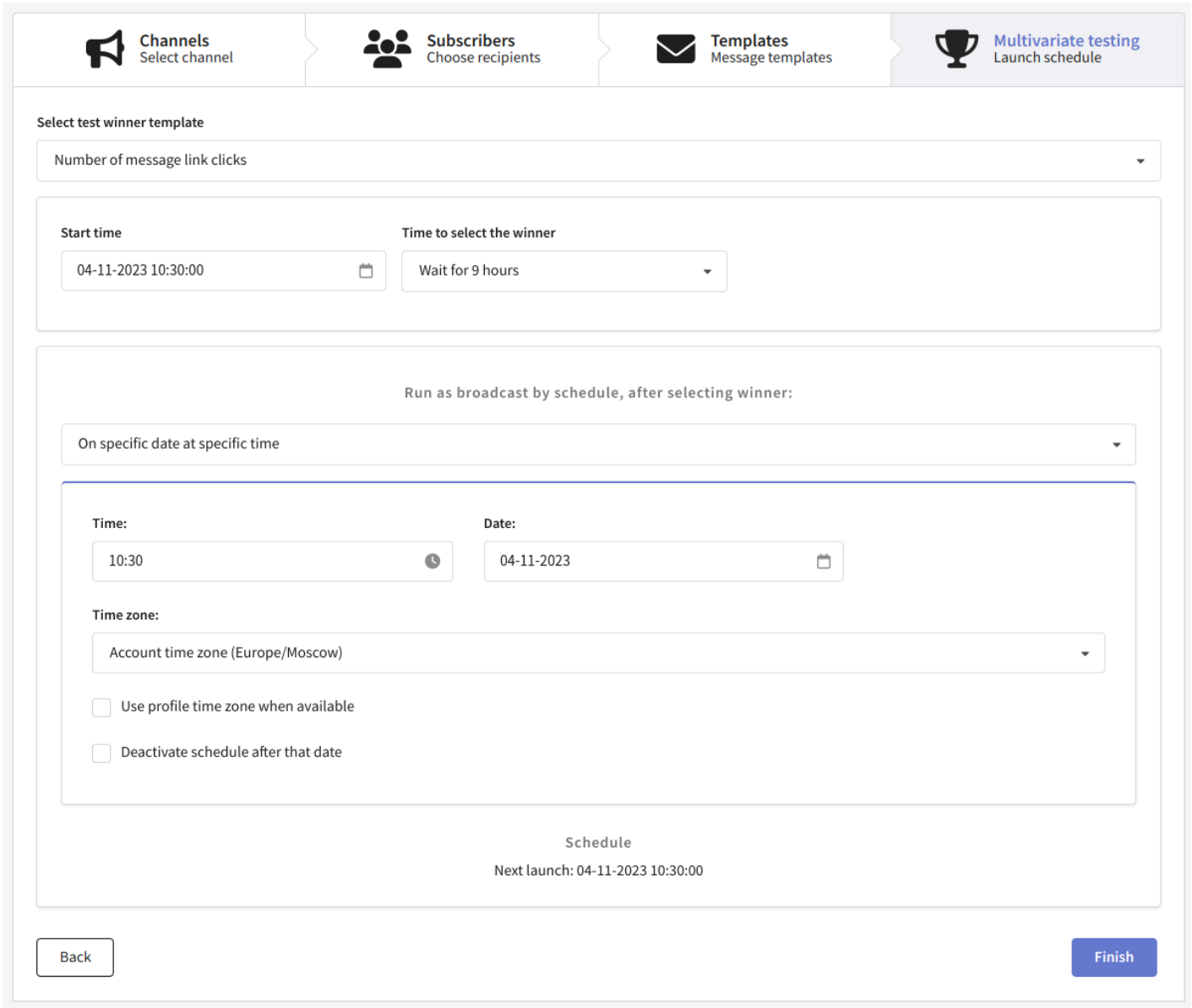


Selection winner by openings/clicks

If you want to select the winning template automatically, set the mailing Start time. This is the time and date for the mailing test launch.

After launch, the mailing will automatically collect statistics by the total number of messages openings or link clicks. You can set a time for collecting statistics using Time to select the winner parameter.

After the mailing collects statistics on opens or clicks, the message template that is most effective based on the selected indicator will be sent to the main part of the audience according to the specified schedule:



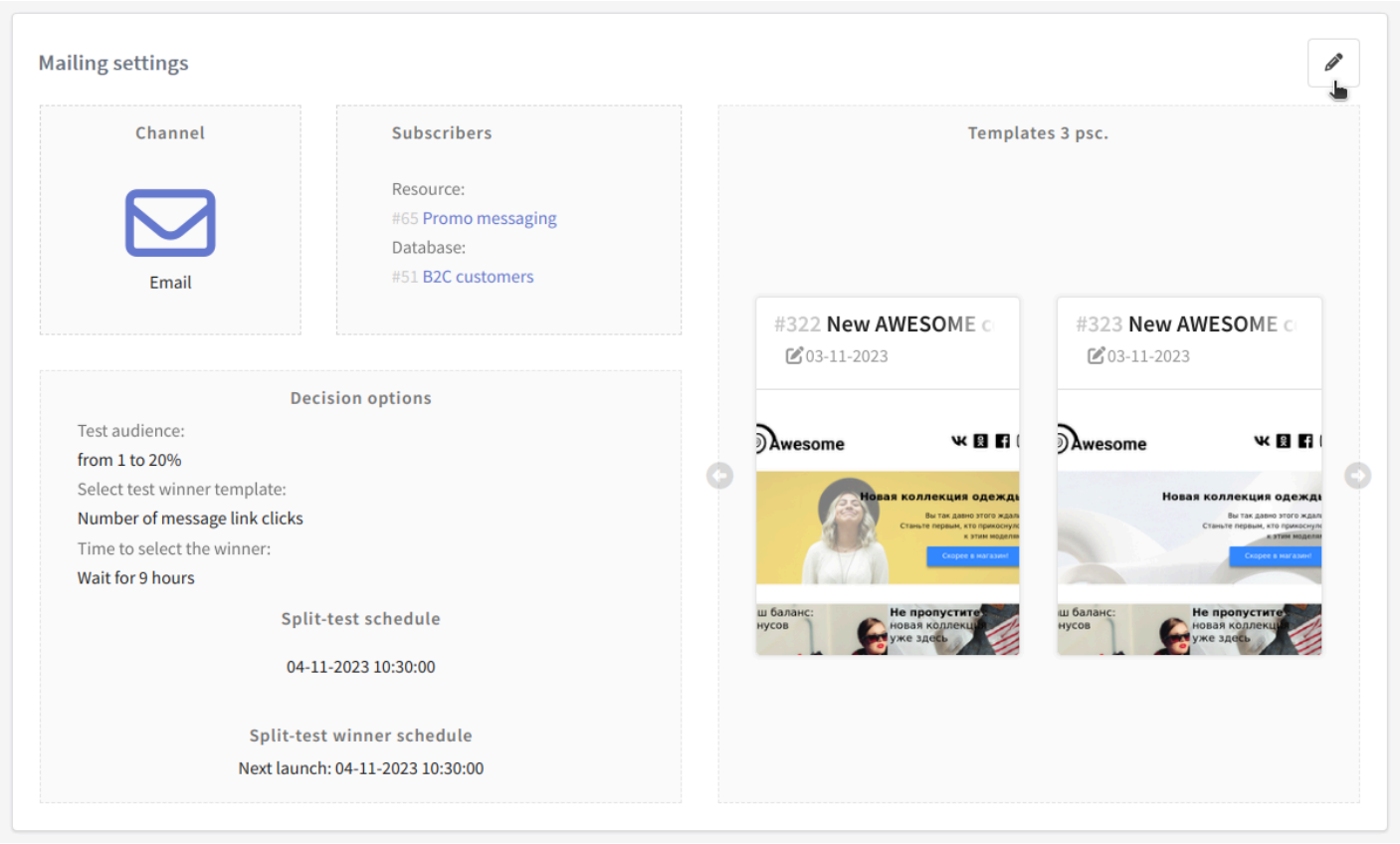
TIP

The mailing schedule is described in detail on this page.

CAUTION

If you select a time zone other than your account zone in the schedule settings, only the winner template will be sent according to that time zone. Test messages will be sent according to the account's time zone.

To complete the setup, click Finish. The options you select will be displayed in the Mailing settings block. Using the button in the upper right corner of the block you can edit the settings:



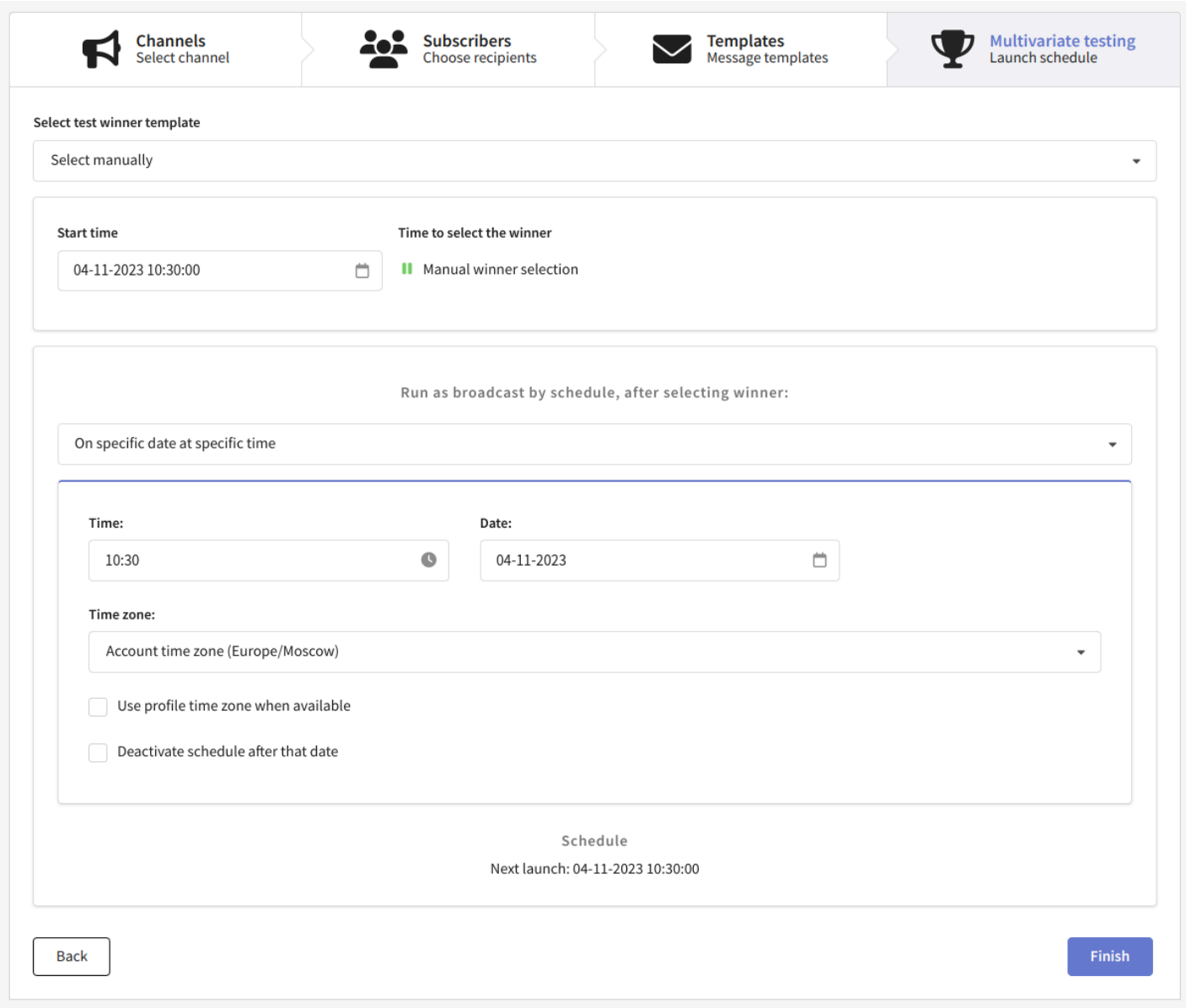
Save the mailing to activate it.

Manual winner selection

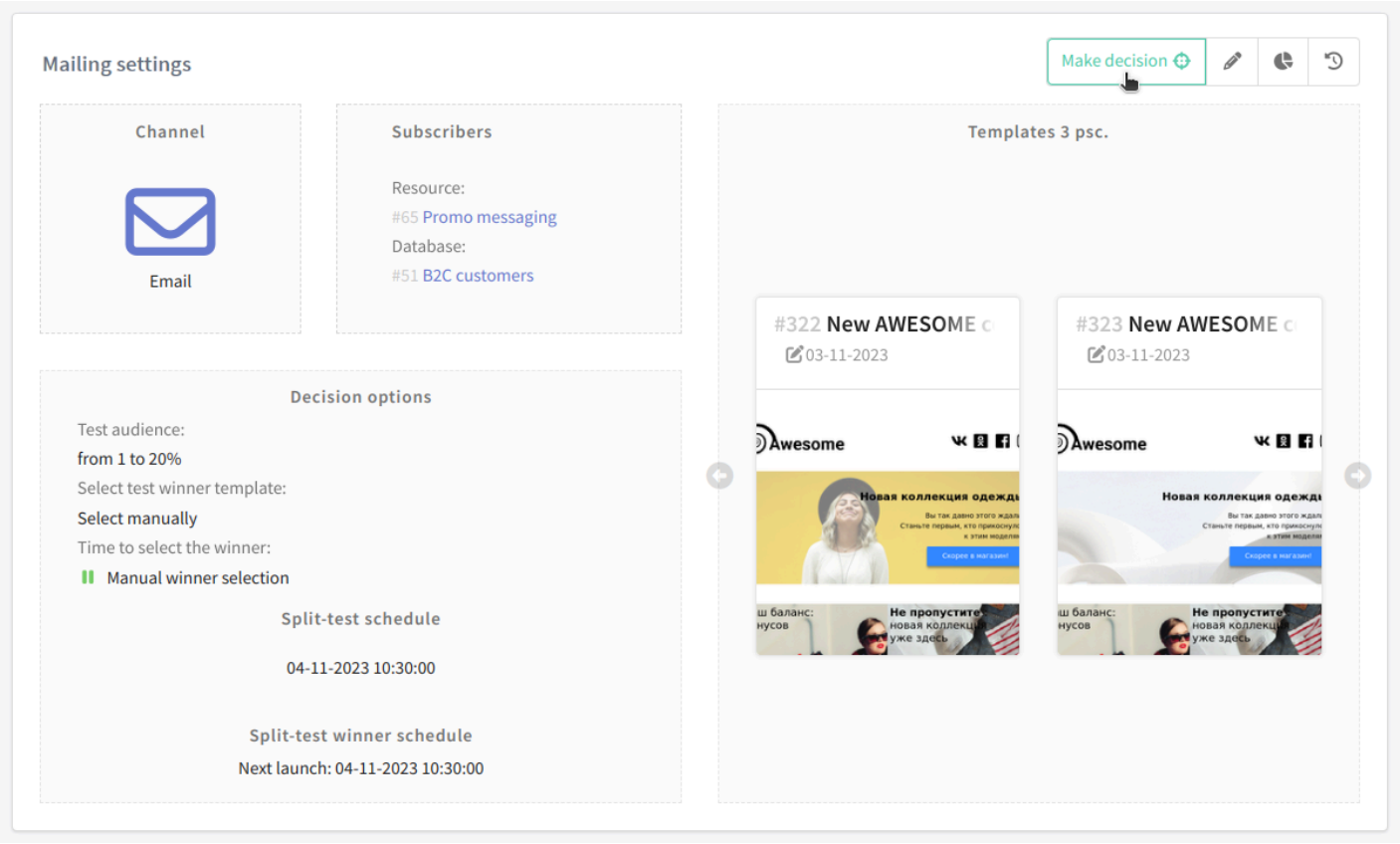
If you want to use the Select manually option, set the Start time. Mailing start is the time and date of launch of the test part of the mailing.

Next, set the mailing schedule. According to this schedule, the winner template will be sent to the main part of the audience.

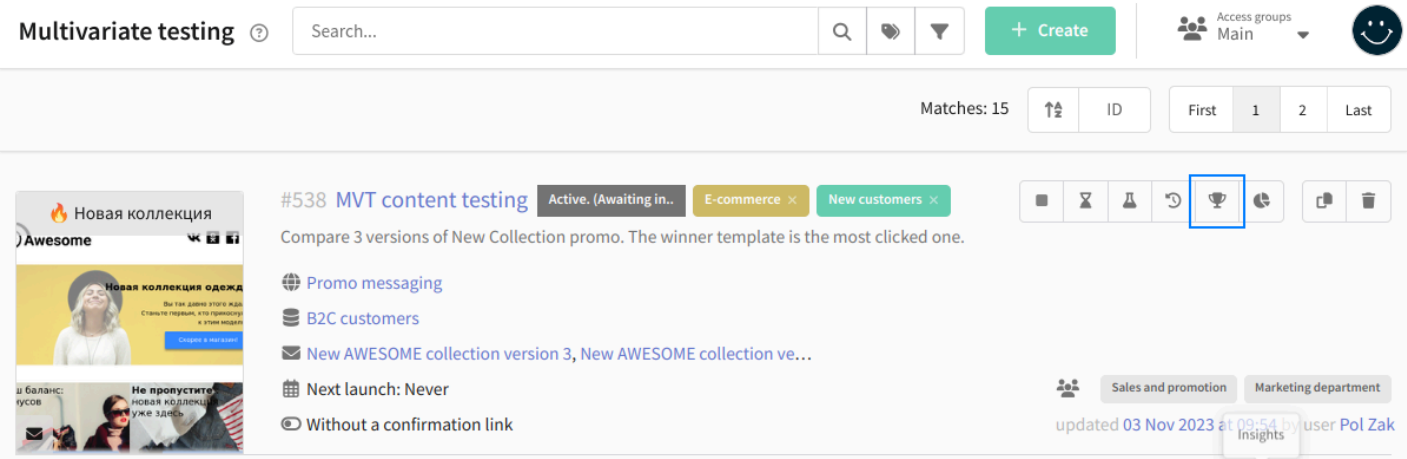
Save the mailing to activate it. After the mailing sends test messages, it will switch to Active (Waiting for you) status.



At this step, you need to analyze the performance of each template and choose the winner template. To do this, click on the Make a decision button in the mailing editing window:

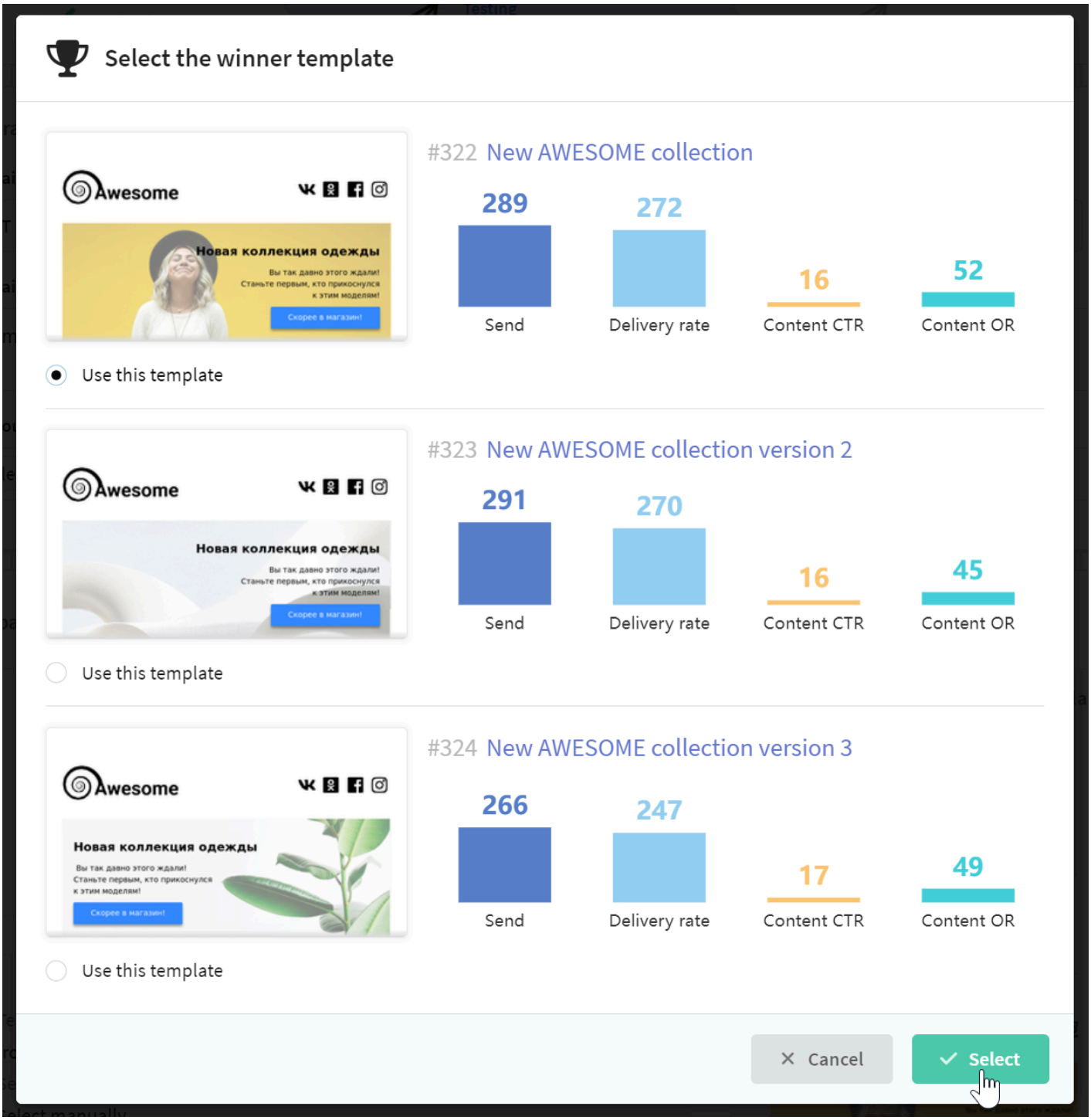


You can also select the winning template in the window with the list of mailings by clicking on the corresponding button in the context menu:

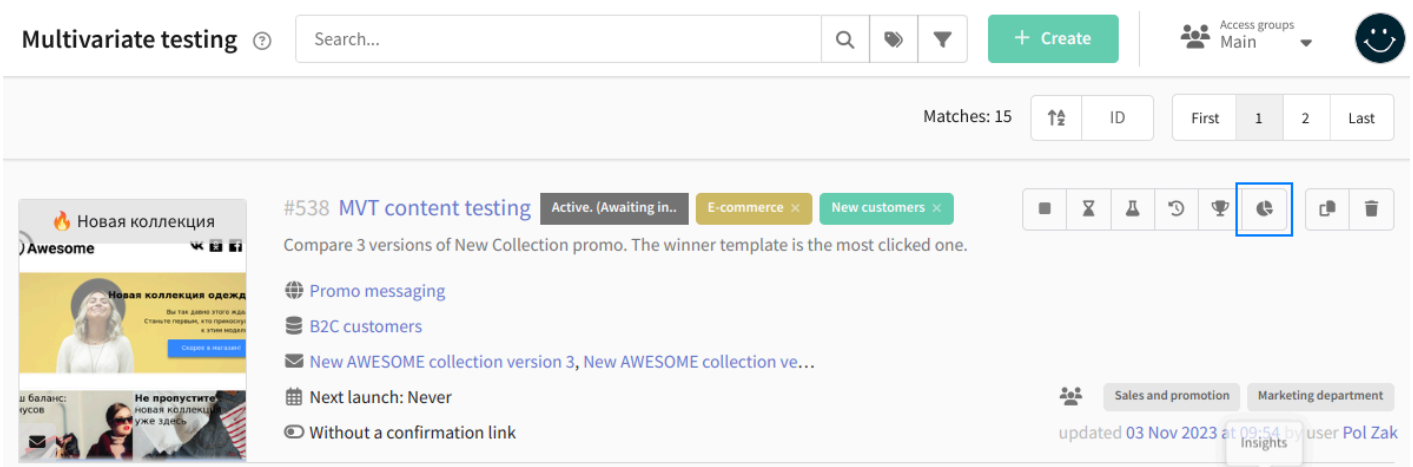


You will see a window with statistics for each template:

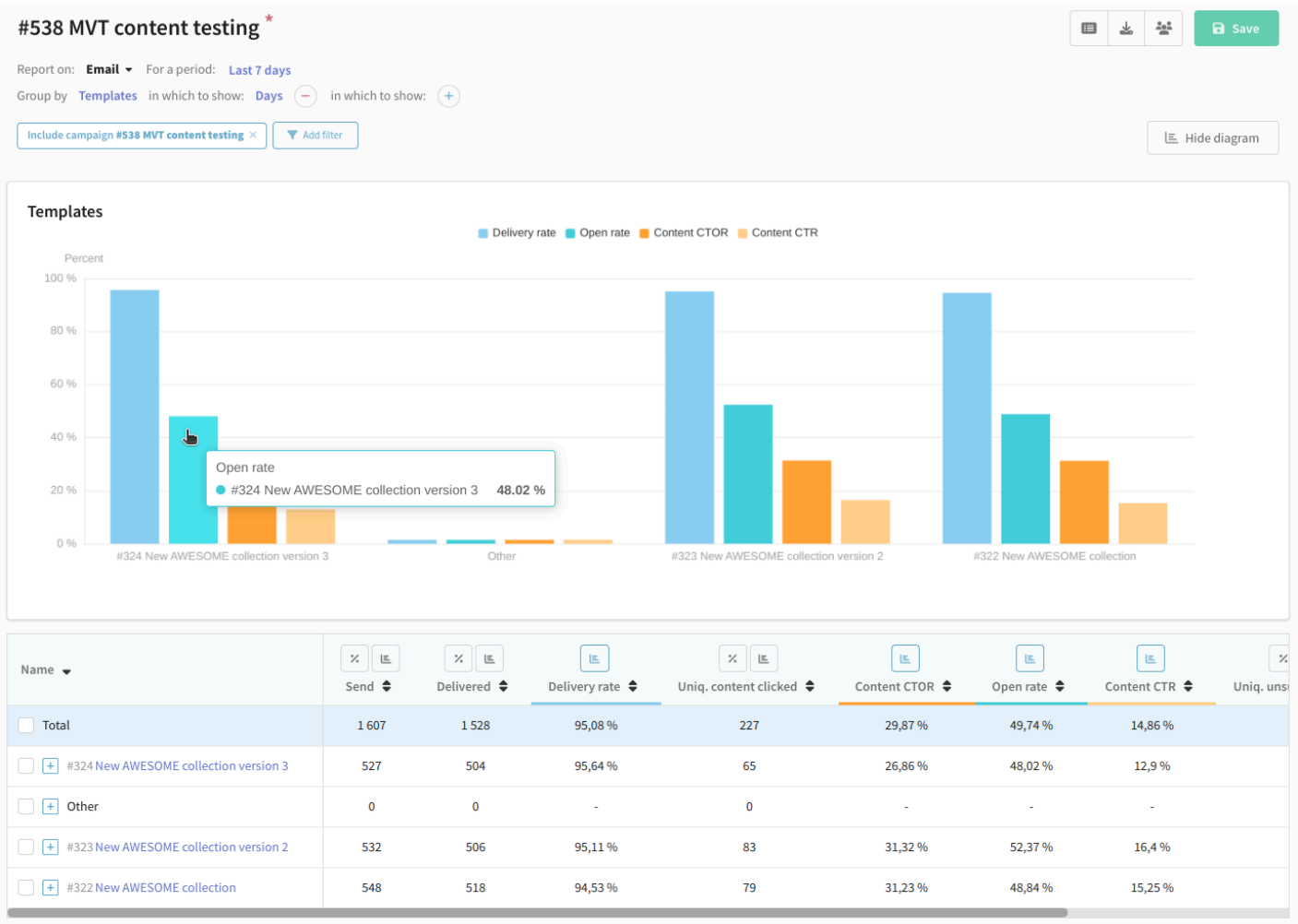
- **Send is the number of messages sent.**
- **Delivery rate is the number of messages delivered.**
- **Content CTR is the percentage of clicks in a message. Calculated using the formula: (Number of clicks / Number of messages delivered) * 100%.**
- **Content OR is the percentage of openings in a a message. Calculated using the formula: (Number of messages opened / Number of messages delivered) * 100%.**



If you don't have enough data to choose the winning template, you can use the [Channel report](). Click on the analytics icon to open it:



In the report, select the parameters necessary to evaluate performance. These can be absolute or relative values (such as Open Rate or CTR/CTOR). You can also check negative reactions by the number of complaints and unsubscriptions.

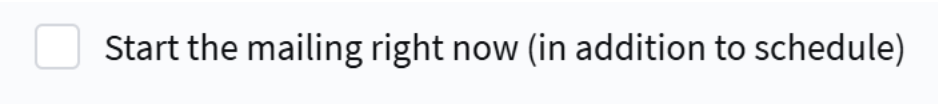


Name	Send	Delivered	Delivery rate	Uniq. content clicked	Content CTOR	Open rate	Content CTR	Uniq. uns
Total	1 607	1 528	95,08 %	227	29,87 %	49,74 %	14,86 %	
#324 New AWESOME collection version 3	527	504	95,64 %	65	26,86 %	48,02 %	12,9 %	
Other	0	0	-	0	-	-	-	
#323 New AWESOME collection version 2	532	506	95,11 %	83	31,32 %	52,37 %	16,4 %	
#322 New AWESOME collection	548	518	94,53 %	79	31,23 %	48,84 %	15,25 %	

Instant mailing launch

If you want to launch a mailing right after its creation, toggle the appropriate option at the bottom part of the page.

- **If the limits were not configured, messages will be sent to all customers.**
- **If limits were configured, the mailing will be launched besides schedule.**



Communication policy

Communication policy sets limits for communication with one subscriber. It is configured in the Limitations block. By default, the mailing will use account policy or resource policy.

Smart settings

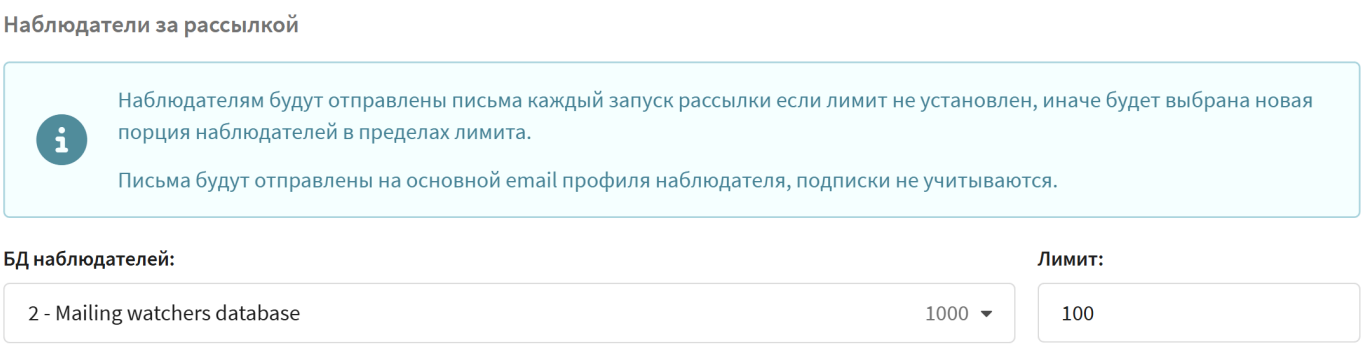
The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages from the platform's real senders).**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify each domain on a separate line.**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients based on subscription data.**

Mailing Watchers

You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mailing recipients. To do this, you need a watchers database. It is created the same way as a regular database, but you must enable the corresponding option in General settings.

In Smart Settings, select the watchers database and set the message sending limit for each mailing launch:



Message Time-To-Live (TTL) Restriction

Message Time-To-Live (TTL) is the period during which the system attempts to send a message from the delivery queue. If the message is not delivered within this time, sending stops and the message is removed from the queue.

TTL should be used for campaigns that are relevant only for a limited time. For example, if a message is related to a promotion, an event reminder, a one-time code, or an offer that quickly loses relevance.

To configure TTL, enable the Limit message lifetime (TTL) option and specify the waiting time in days, hours, minutes, or seconds.

After the specified time expires, the system stops delivery attempts for the undelivered message and registers a delivery failure event.

TIP: Message TTL specifics

- **The TTL value must be within the limits defined by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration parameters.**
- **If TTL is set in multiple places, the value with the highest priority is applied: the `expiration` API parameter > the value specified in Smart settings > the `MTA_TTL_DEFAULT` configuration parameter.**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefore, even within a single mailing, different recipients may have different message TTL values.**
- **After TTL expires, the system stops delivery attempts even if the channel is still technically able to deliver the message.**
- **TTL does not work for channels that do not return message delivery statuses: Telegram, MAX, and custom channels.**

Mailing testing

Description

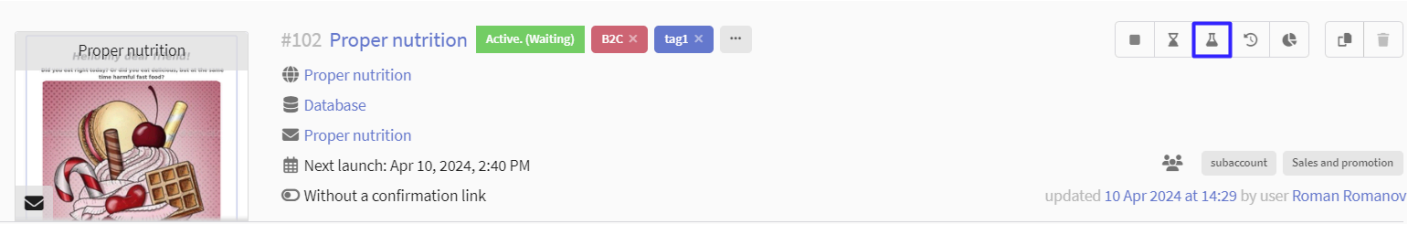
Sometimes you need to test your mailings before activation due to several reasons. This way, you can be sure that:

- **content will be properly displayed in different mail clients;**
- **the technical part of sending and delivering the message is all right;**
- **all substitutions and complex logic inside the template work correctly;**
- **email and SMS providers do not block messages because of the content.**

Using mailing testing, you can send letters for verification to the person responsible for your mailings or use services that evaluate the quality of messages, delivery infrastructure, etc.

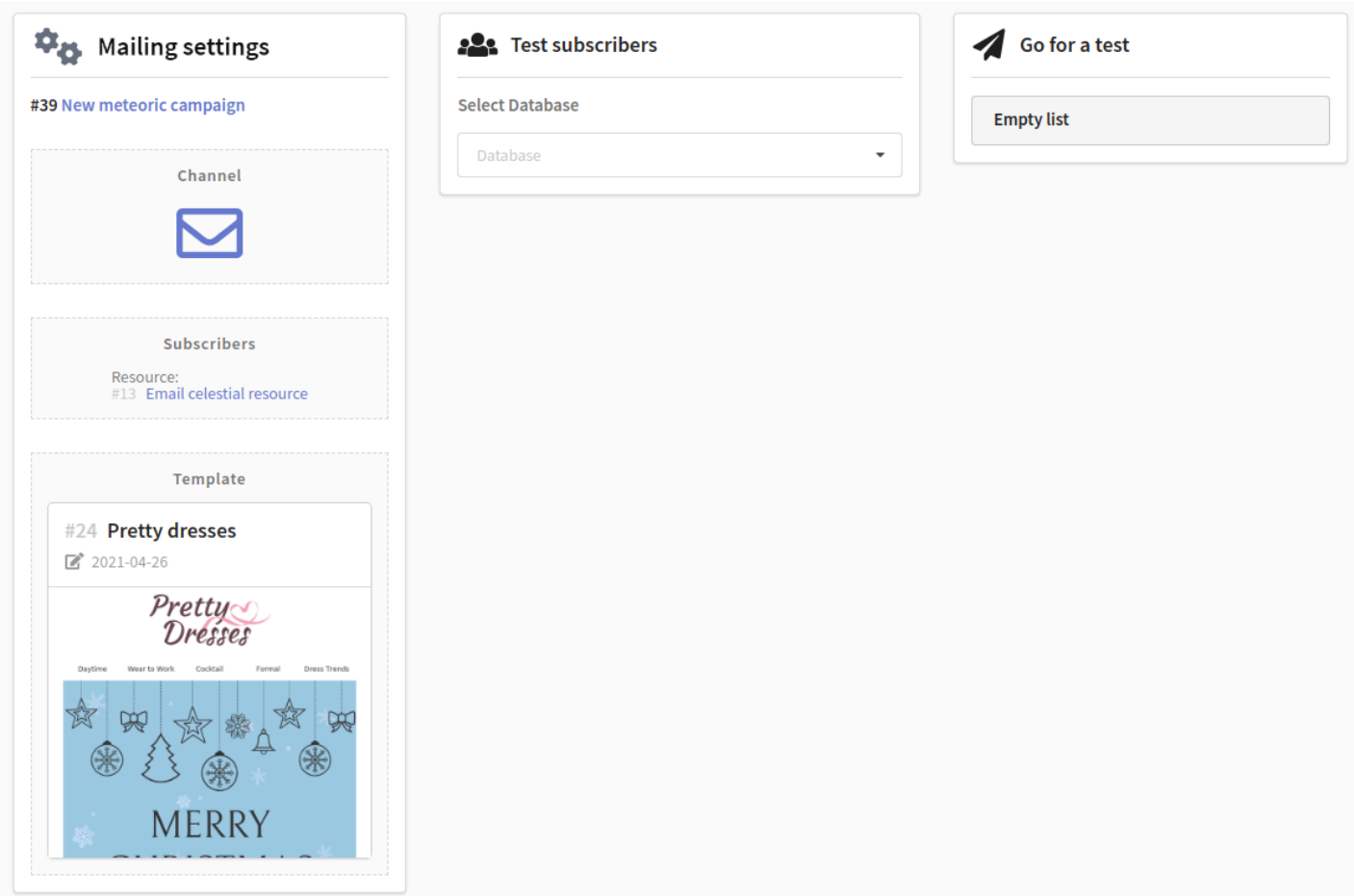
Choose a mailing of a certain type and click the test icon

:

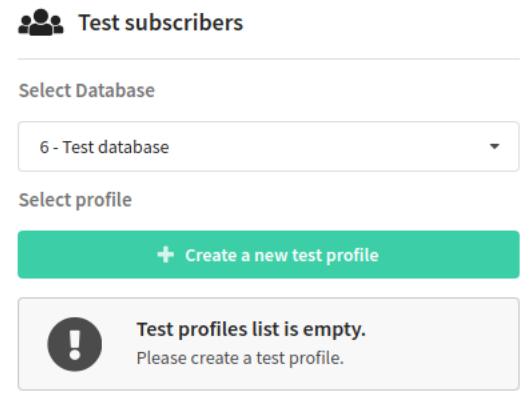


Contacts of test mailing recipients

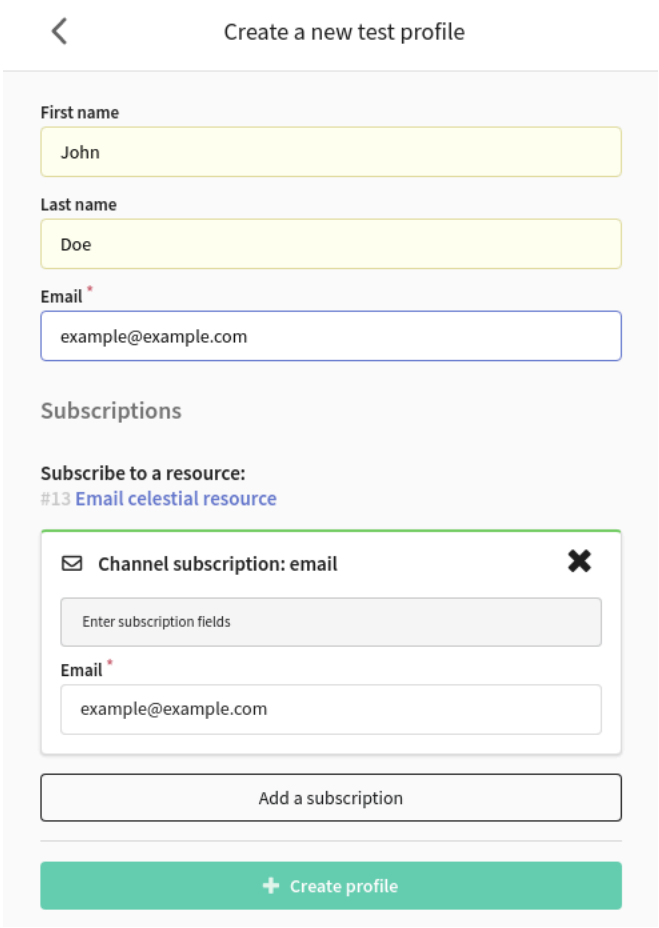
In the new window, you'll see the settings for the mailing you are testing and test subscribers.



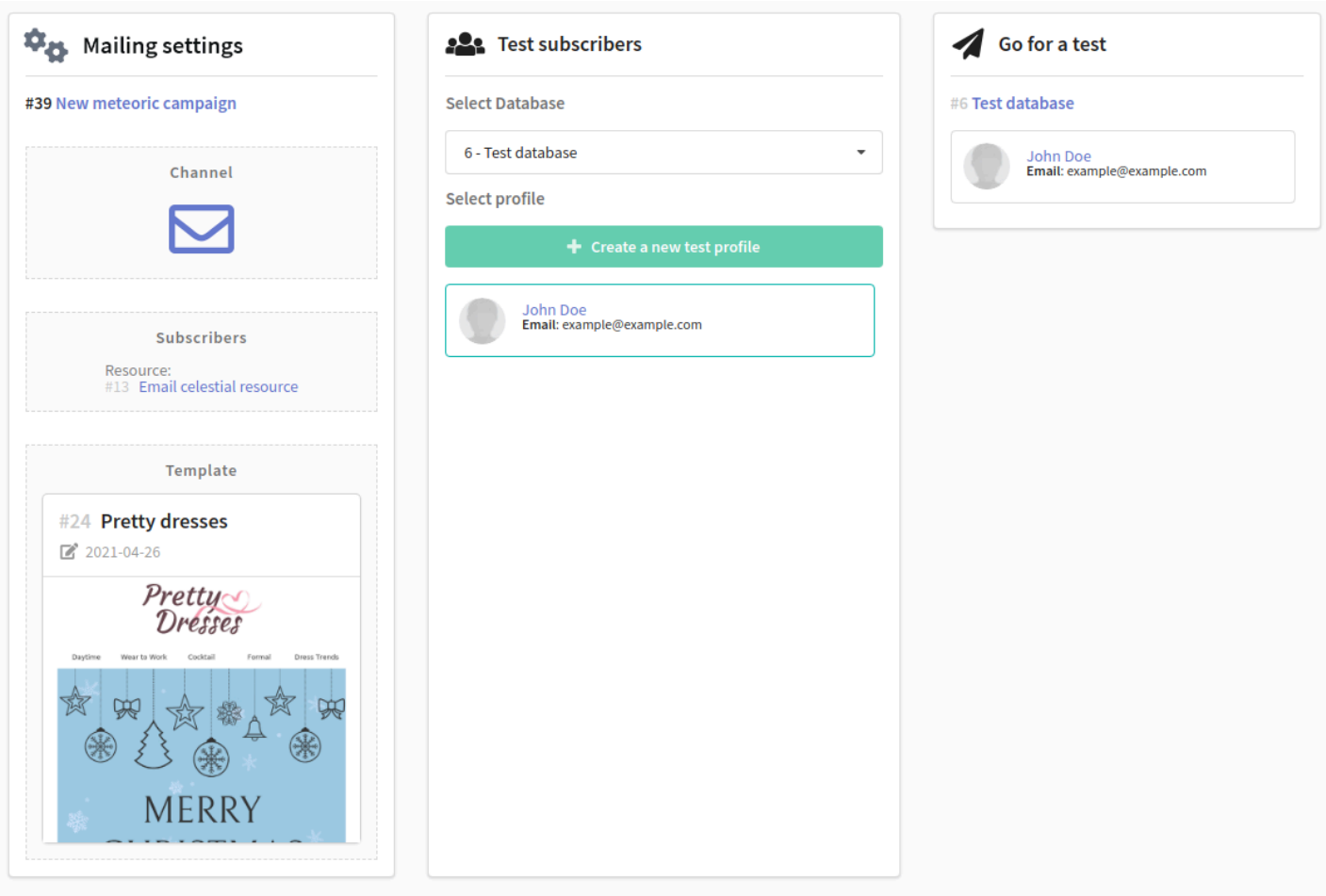
If you don't have test subscribers yet, you can add them. To do this, select the profile base in the drop-down list, then you will be prompted to create a test subscriber profile in this database if it has not yet been created. To do this, click on the Create a new test profile button:



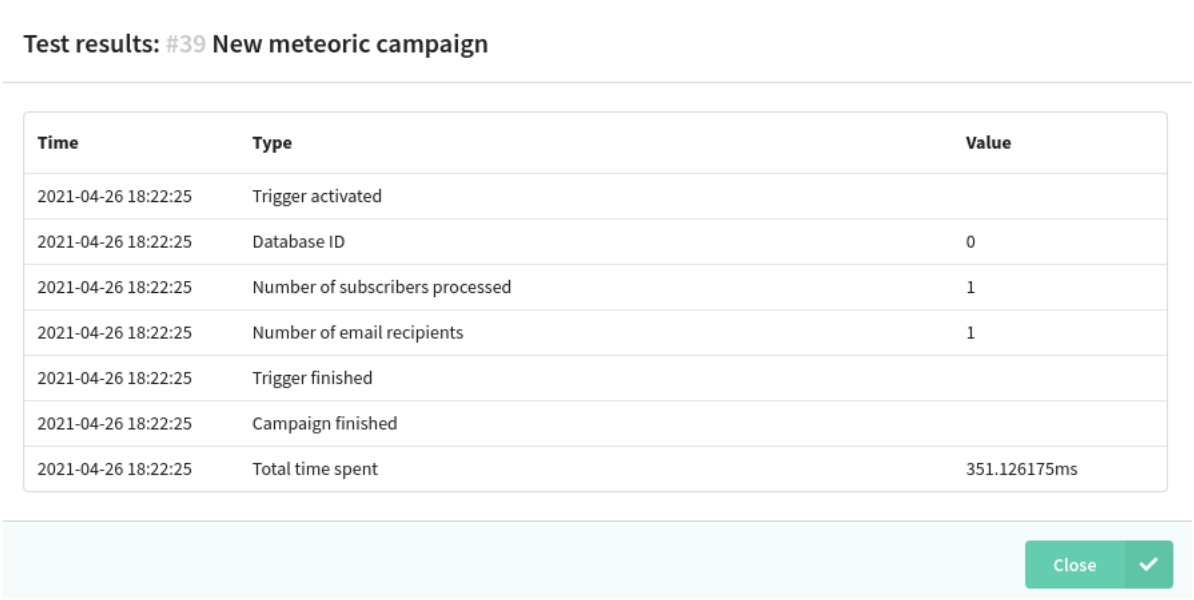
Then you'll see the interface for creating a test profile:



After all the data has been entered in the appropriate fields, click the Create profile button. The test profile will be displayed in the test list and will be available for selection:



Click the Test button and a test mailing will be sent. You'll get the test result in the new window:



Test results: #39 New meteoric campaign

Time	Type	Value
2021-04-26 18:22:25	Trigger activated	
2021-04-26 18:22:25	Database ID	0
2021-04-26 18:22:25	Number of subscribers processed	1
2021-04-26 18:22:25	Number of email recipients	1
2021-04-26 18:22:25	Trigger finished	
2021-04-26 18:22:25	Campaign finished	
2021-04-26 18:22:25	Total time spent	351.126175ms

Close

TIP

It is definitely a good practice to have test contacts for all most used email and SMS providers.

Mailing schedule

The schedule is applied to broadcast and regular mailings, MVT mailings, scenarios, scheduled imports and integrations

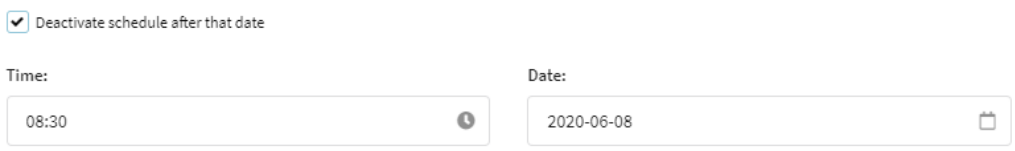
Several plans are available — for different cases. They are described in this article.

INFO

For every schedule plan, you can override the account time zone.

INFO

Deactivate schedule option will help you stop mailing on specific time:

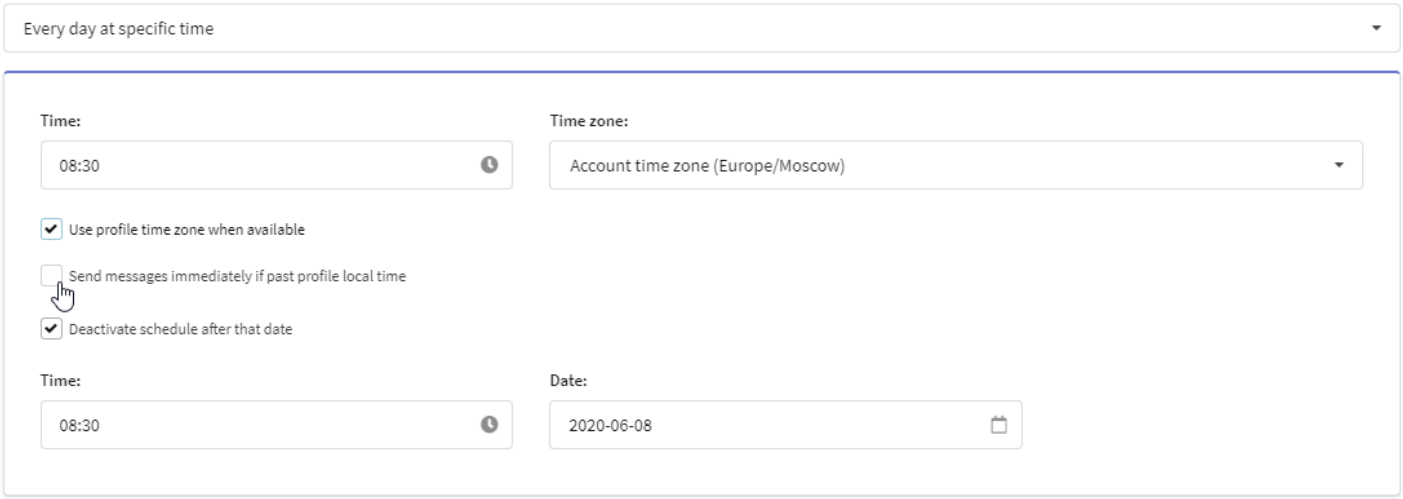


Every day at specific time

This plan will help organize dynamic regular mailings and scenarios: the mailing will start every day for subscribers having an upcoming birthday or pay day.

TIP

This schedule plan can use customer profile time zone - for delicate mailings. If you are past local time for some recipients - use the option to send immediately.

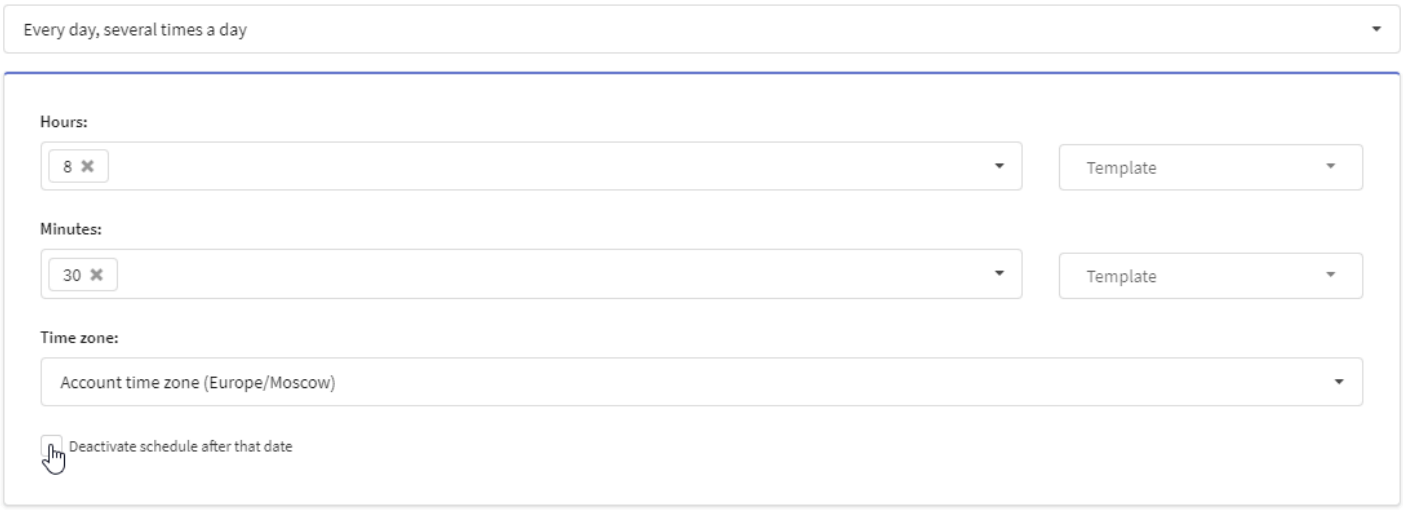


TIP

This plan is also good for large broadcasts, divided by audience limitations.

Every day, several times a day

Can be launched several times every day: specify hours and minutes. You can use templates at the right side of the form:

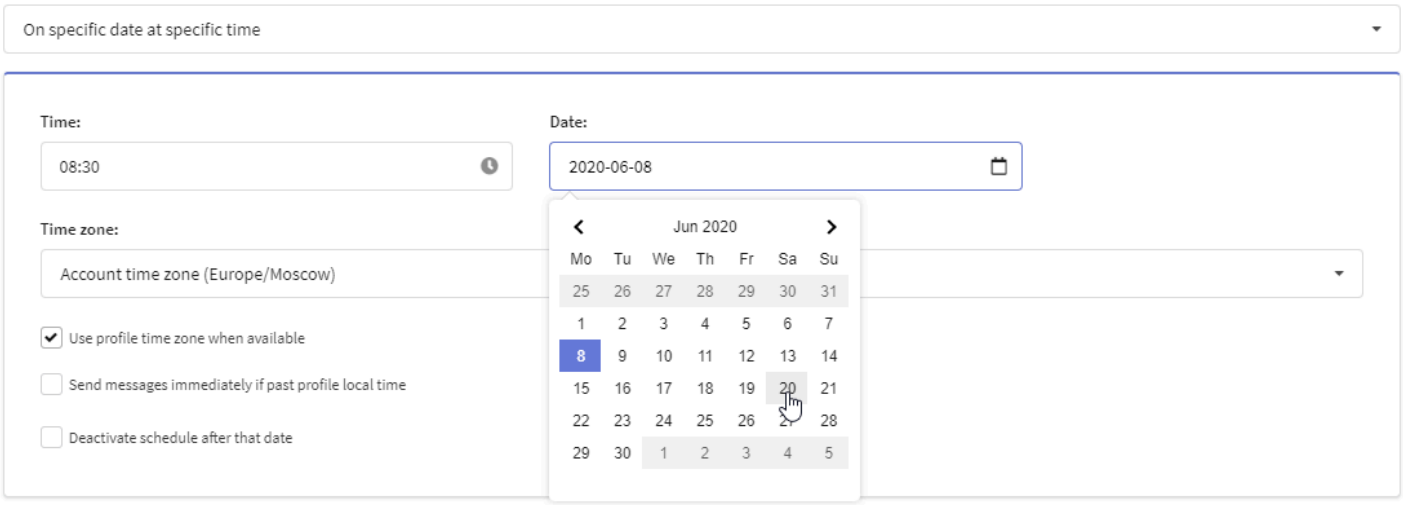


On specific date at specific time

This plan is great for one-time mailings.

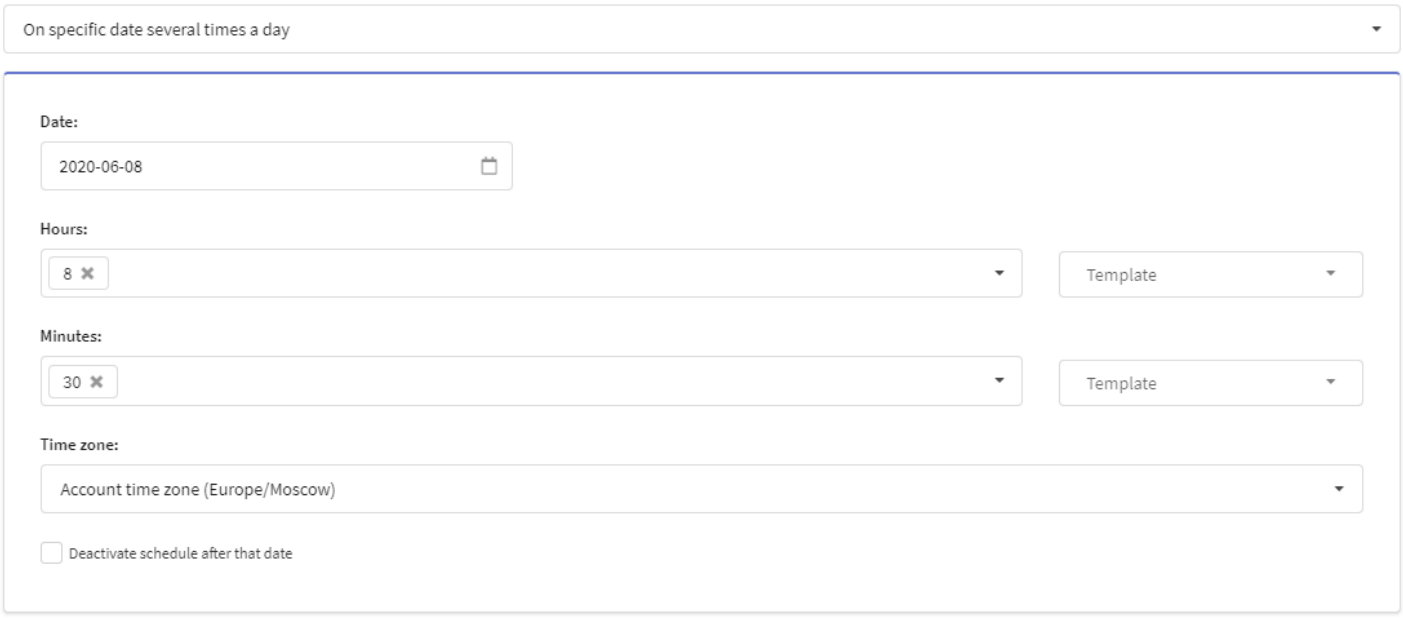
TIP

This schedule plan can use customer profile time zone - for delicate mailings. If you are past local time for some recipients - use the option to send immediately.



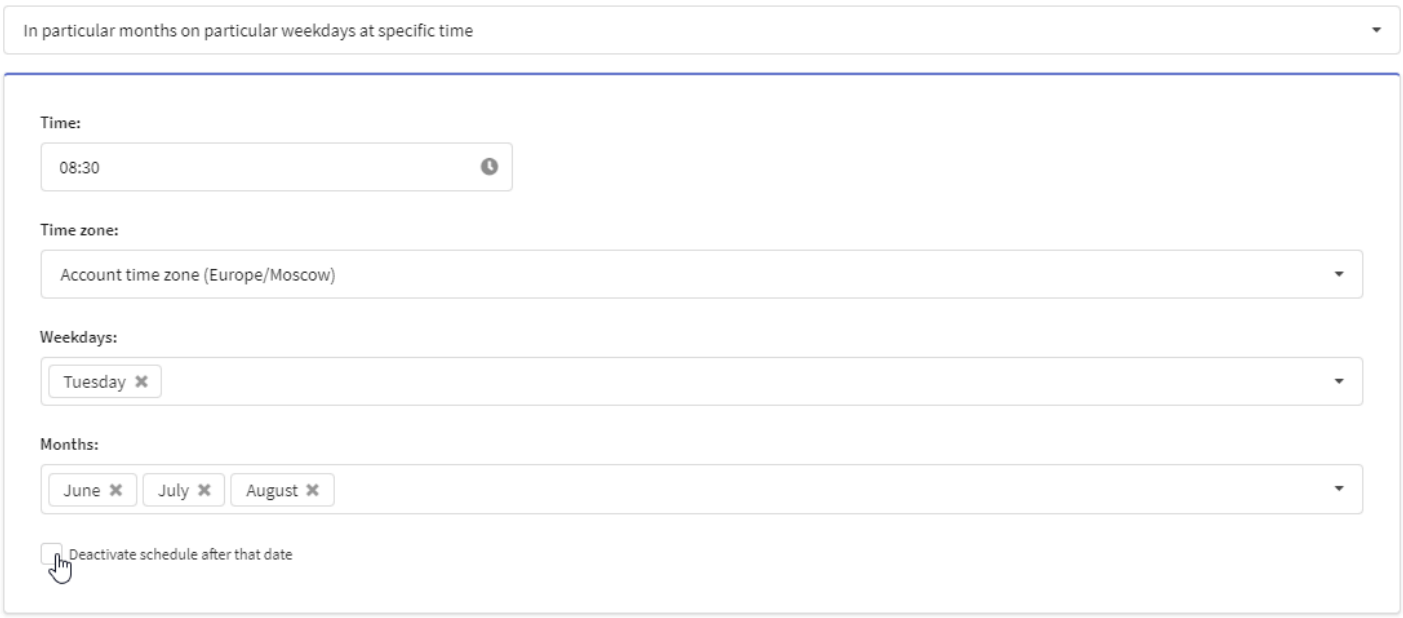
On specific date several times a day

With this plan you can launch mailings several times on one day.



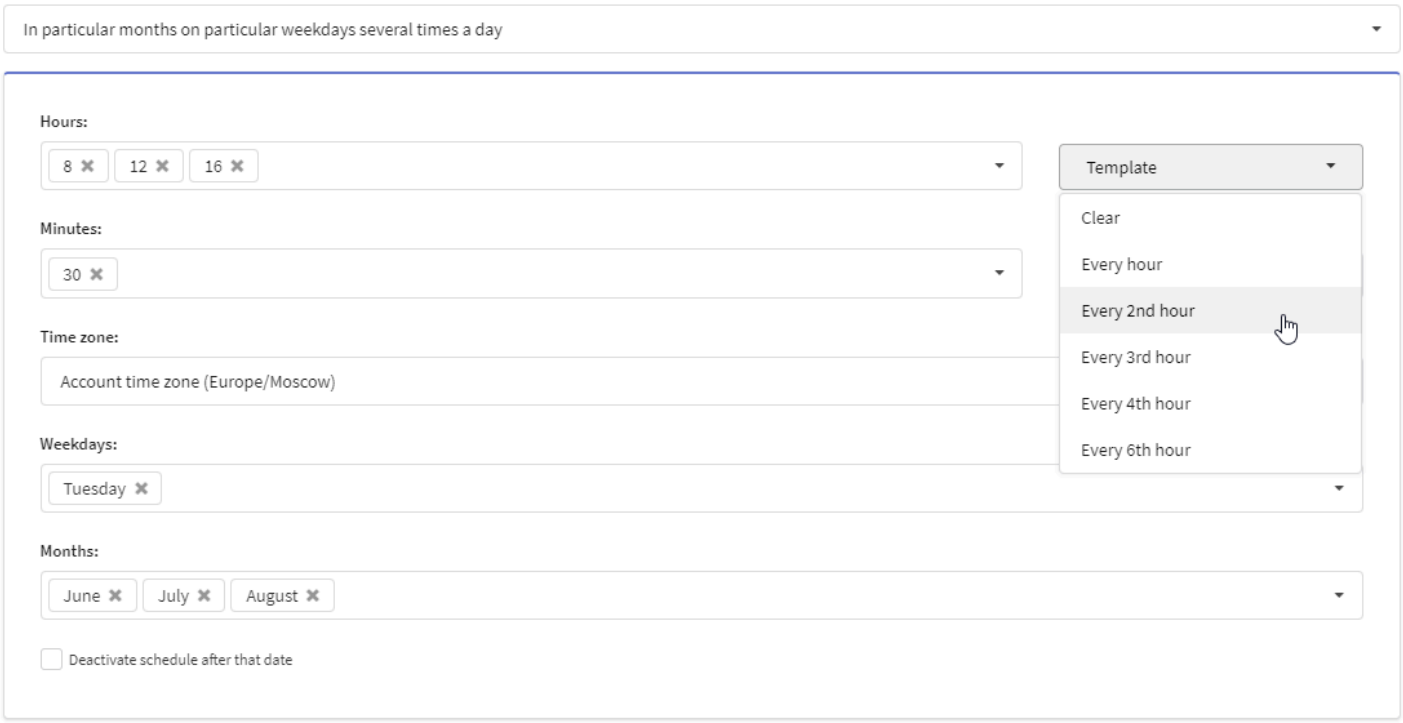
In particular months on particular weekdays at specific time

Use this plan for your weekly mailings: the mailing will start on every weekday specified - in specific months.



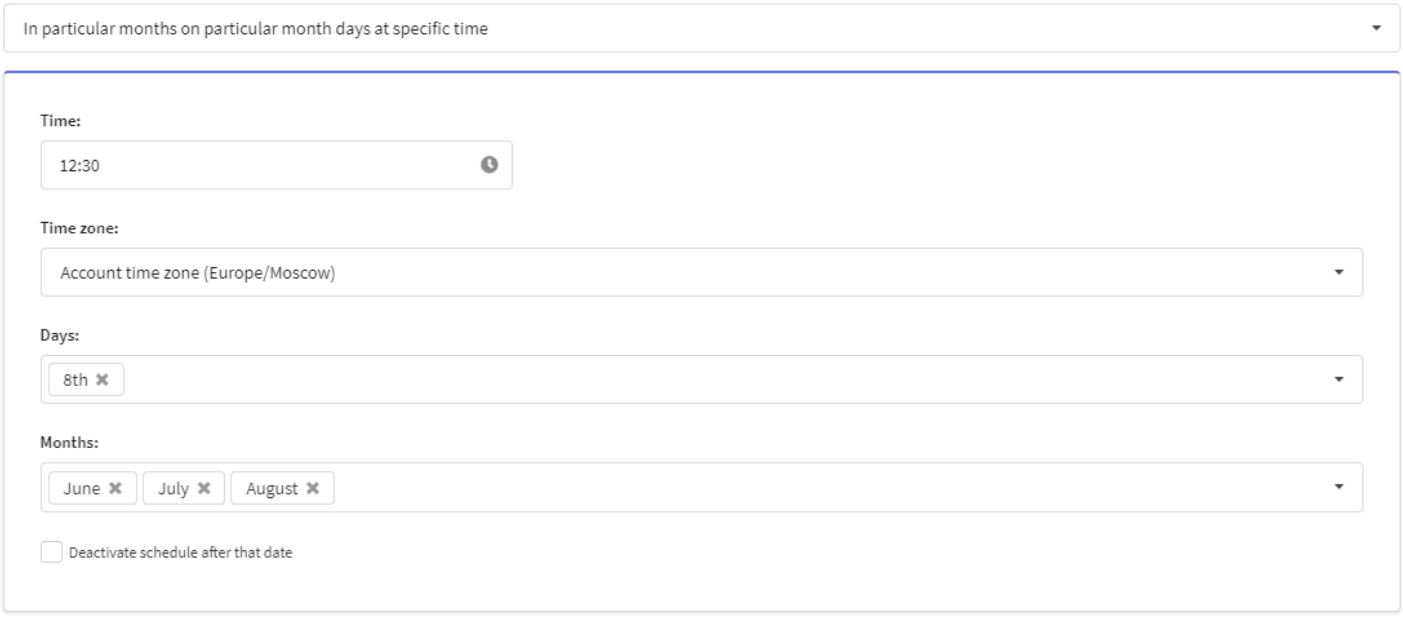
In particular months on particular weekdays several times a day

Here you can also specify multiple hours and minutes.



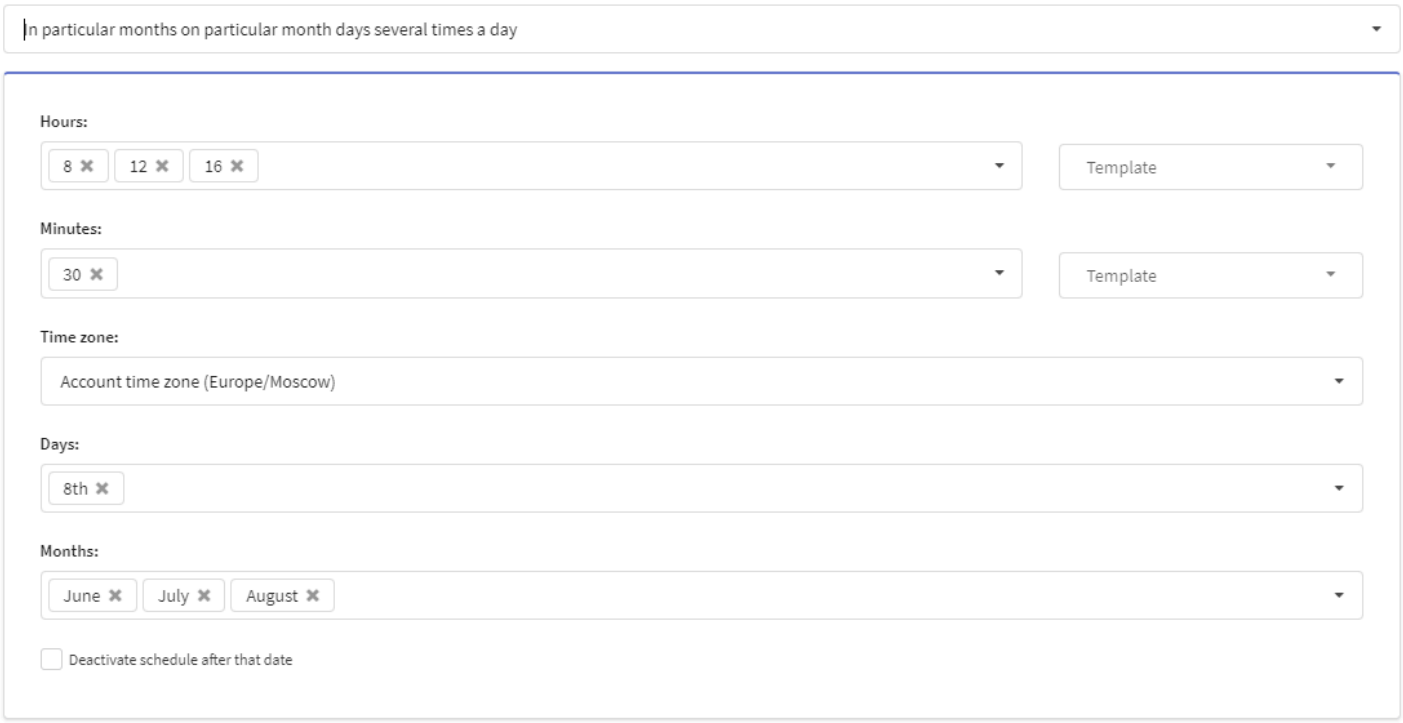
In particular months on particular month days at specific time

This plan uses not weekdays but month days. Thus it is quite good for monthly or quarterly mailings.



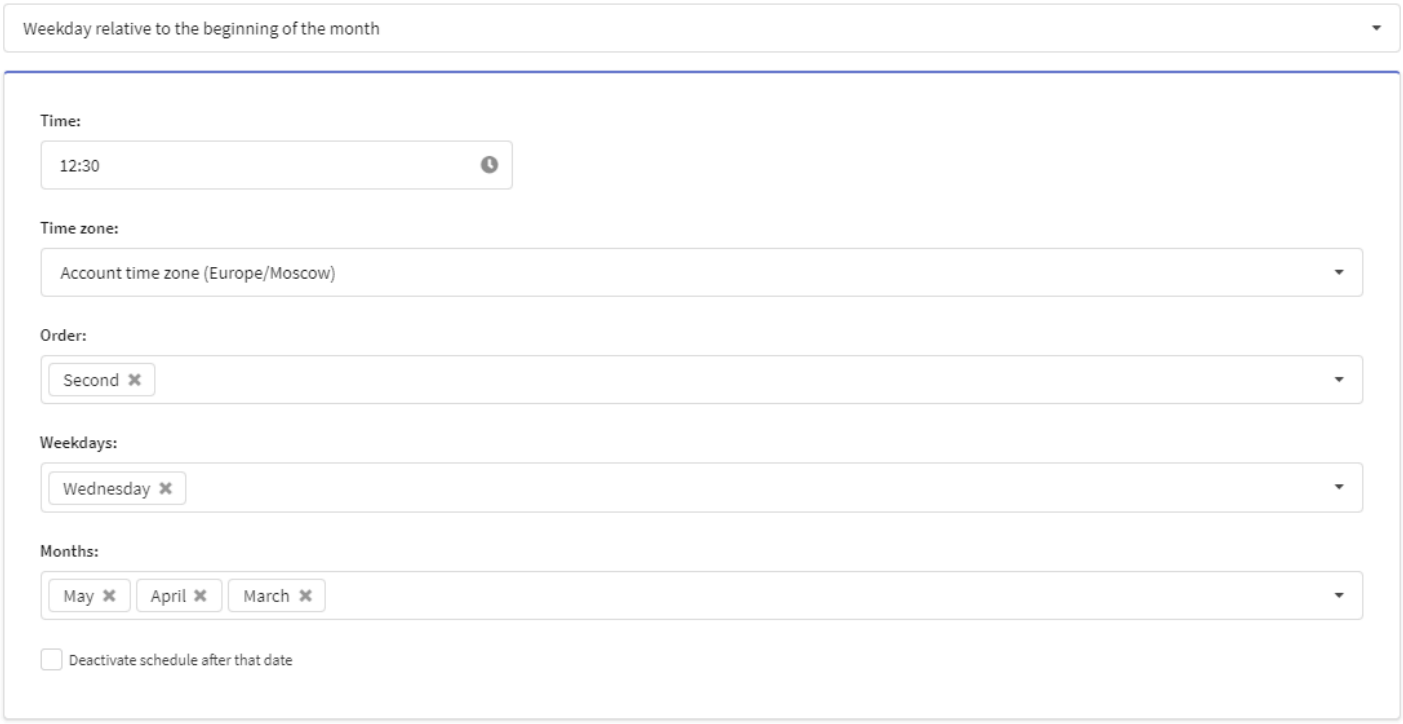
In particular months on particular month days several times a day

Here you can organize several launches a day.



Weekday relative to the beginning of the month

If you need a mailing for e.g. every second Wednesday during springtime - this plan can fit you best:



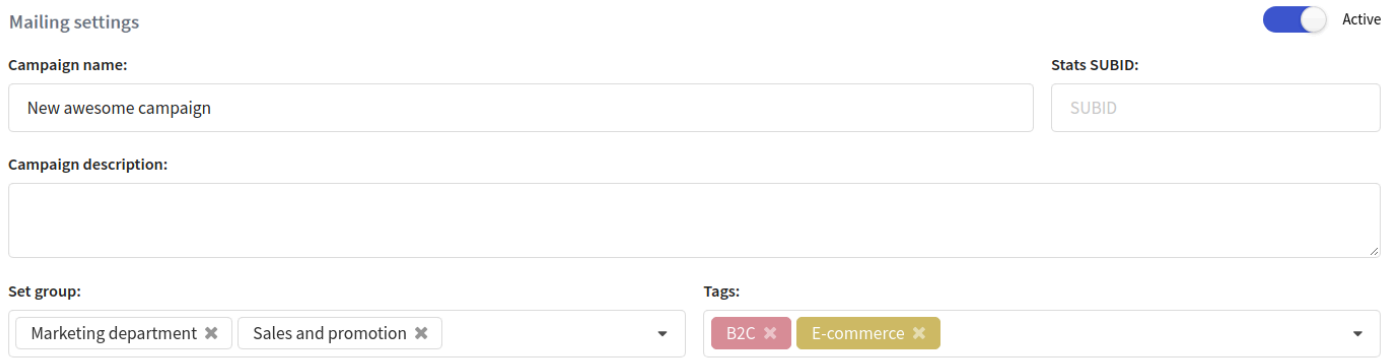
Placement

Description

	Placement mailing is a tool for placing content for subscribers on a specific resource with the possibility of cancellation at any time. After launch, the mailing sends content to each subscriber in the data source (in the case of a scheduled start) or to a specific lead that initiated a certain event (in the case of a triggered start condition). If the channel is configured to unsubscribe, the placed content can be removed from the resource manually or upon reaching a certain date and time. To create a new mailing, go to "Mailings" → "Placements" in the main menu. Click + Create in the top panel. </td> </tr>

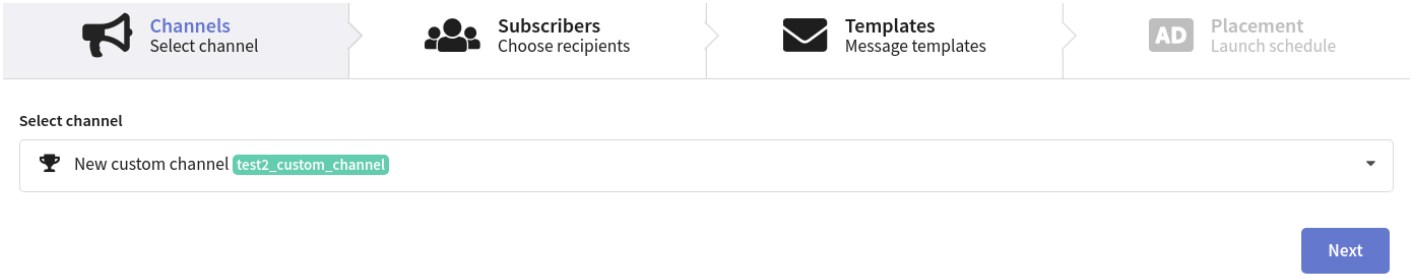
General settings

In the main settings, specify the name and description of the new segment, and if needed, set the access group, add tags and attributes for additional grouping:



Channel

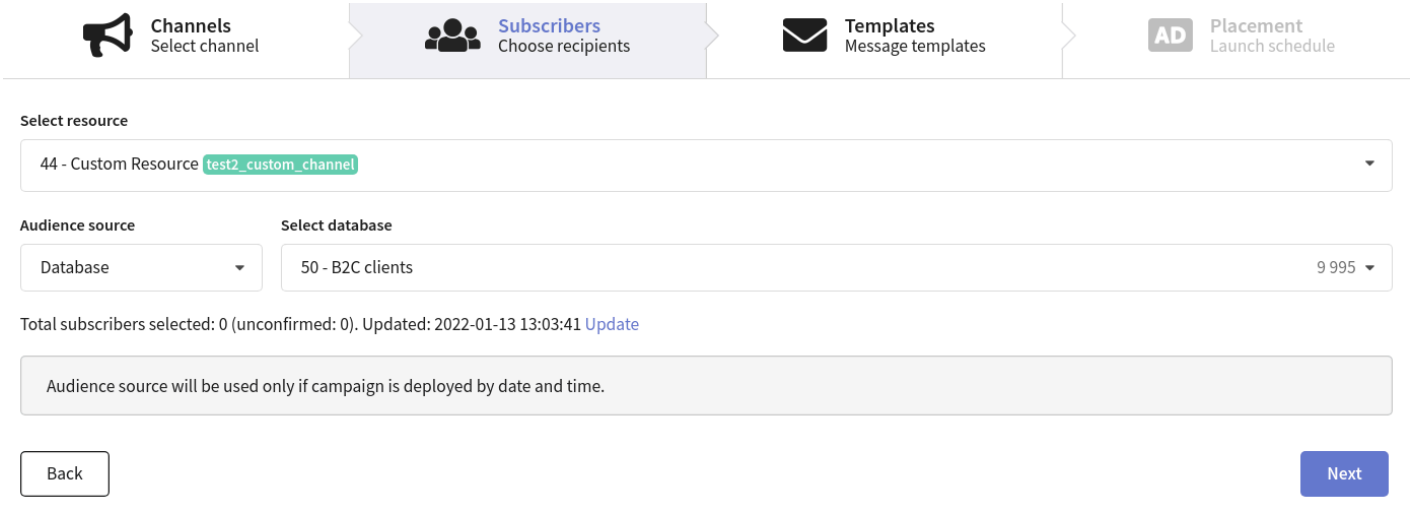
In the drop-down list, select a channel that can be used to send messages for this mailing type.



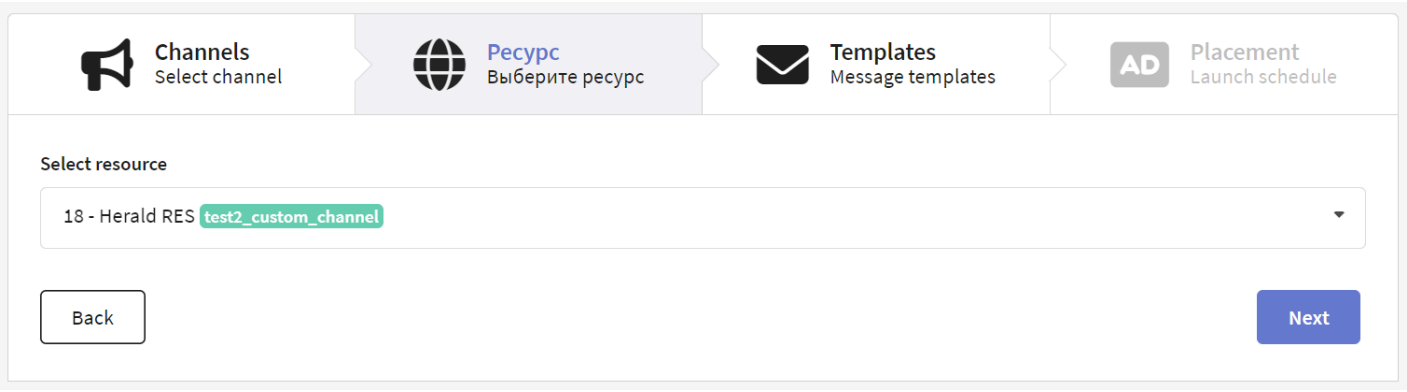
Subscribers

Mailings are sent to customers who subscribe to a certain resource. Resources can be a separate website, product, promotion, or another direction of your communications with customers. Select a resource. Make sure the recipients in your database are subscribed to it.

Select an audience source. Placement mailing recipients can be a profile database or a segment consisting of profiles from one or more databases.



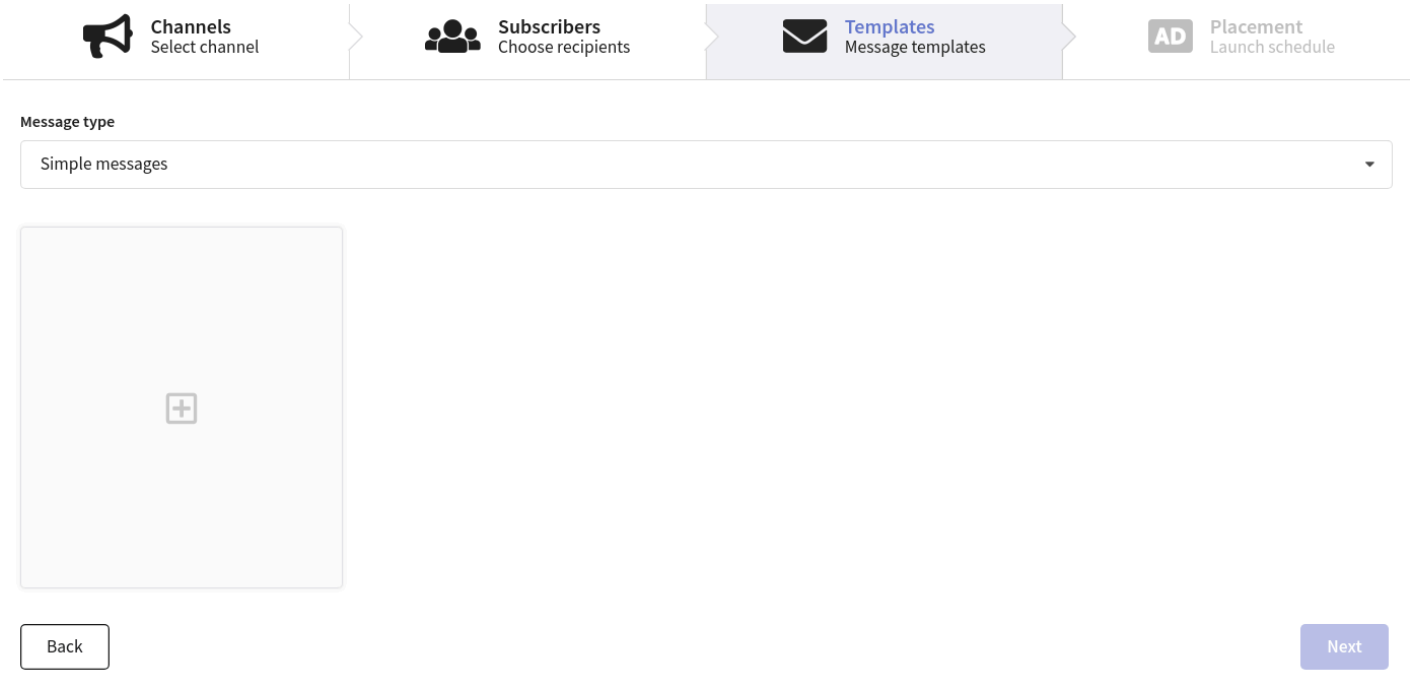
If you are using a placement mailing to send messages through a custom channel, the mailing settings may vary depending on the channel configuration. For example, some channels may require selecting only the resource without specifying the subscriber source.



Templates

Select the content you want to use in your placement mailing. Click

to add a new template. Any mailing type supports up to 8 message templates. The mailing will send each recipient one randomly selected message in equal proportions. If you want to preview the message, click on the template thumbnail.



Schedule

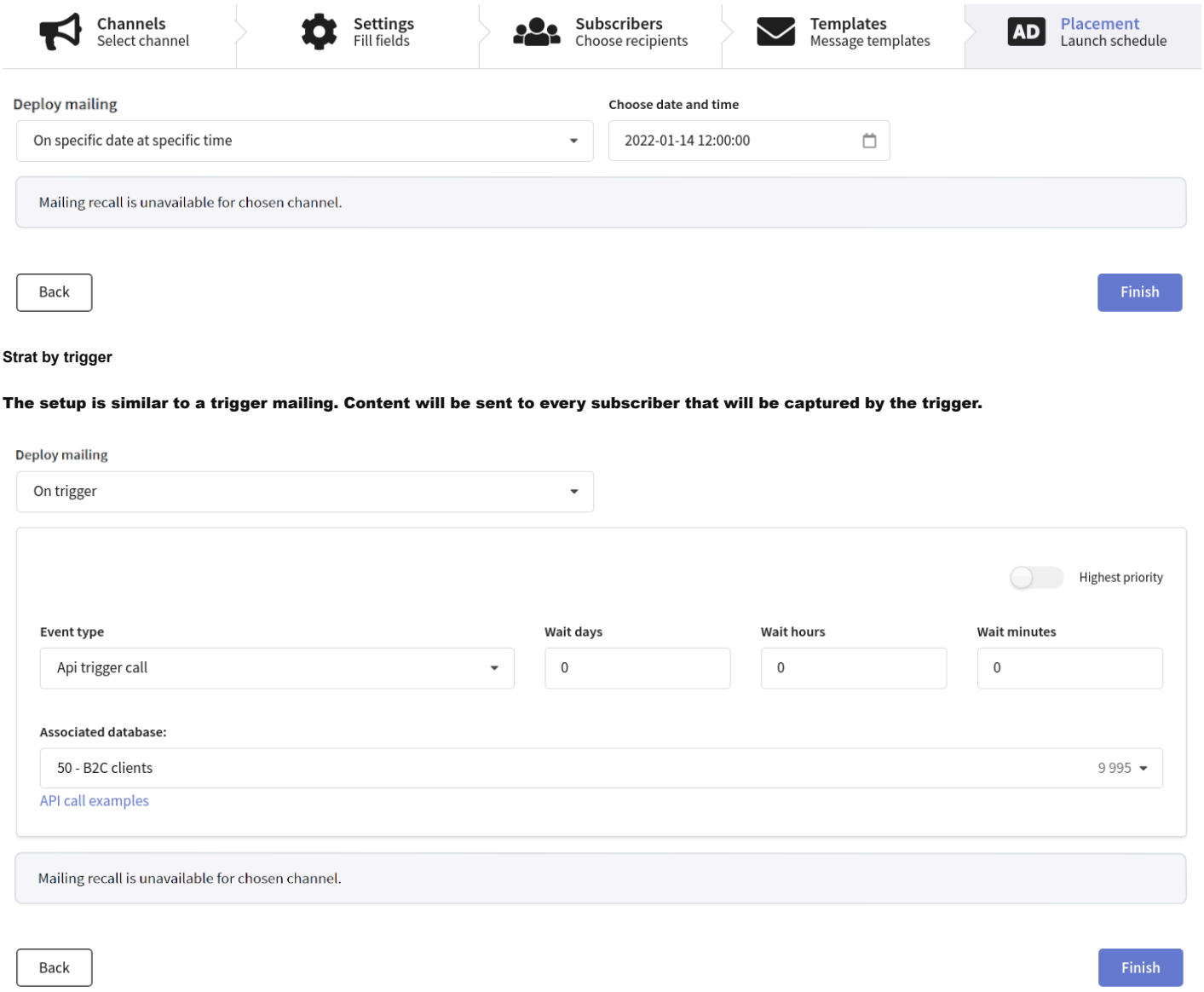
Start and stop settings

A placement mailing can be launched in two ways:

- **on specific date at specific time**
- **on trigger for one subscriber**

Start at a specific date and time

The mailing will run once at the specified time. After that, the mailing will not be available for restart.



Strat by trigger

The setup is similar to a trigger mailing. Content will be sent to every subscriber that will be captured by the trigger.

INFO: Direct communications and trigger launch for placement

Event-based launch is available only for placements in direct channels. Telegram Group and MAX Group are indirect channels, so trigger-based launch is not available for them.

You can read more about the differences between direct and indirect channels here.

Cancellation of placement

Cancellation of placement is only available for channels where it's configured. If the cancellation was not provided by the administrator when configuring the channel, this option will not be available for editing in the mailing.

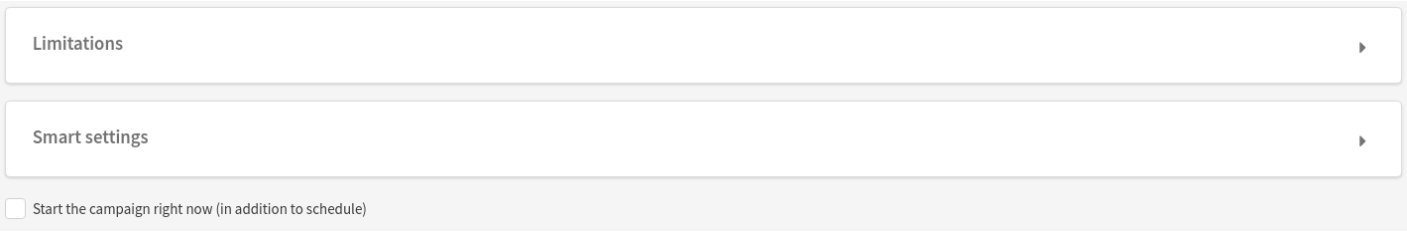
There are two types of cancellation:

- **on specific date at specific time**
- **manually**

Both options will cancel placements for all subscribers to whom the content was sent. It does not depend on the start type. If the start was set to a trigger, the trigger will be deactivated before cancelling.

Placement mailing can be canceled manually if its status is "Deployed".

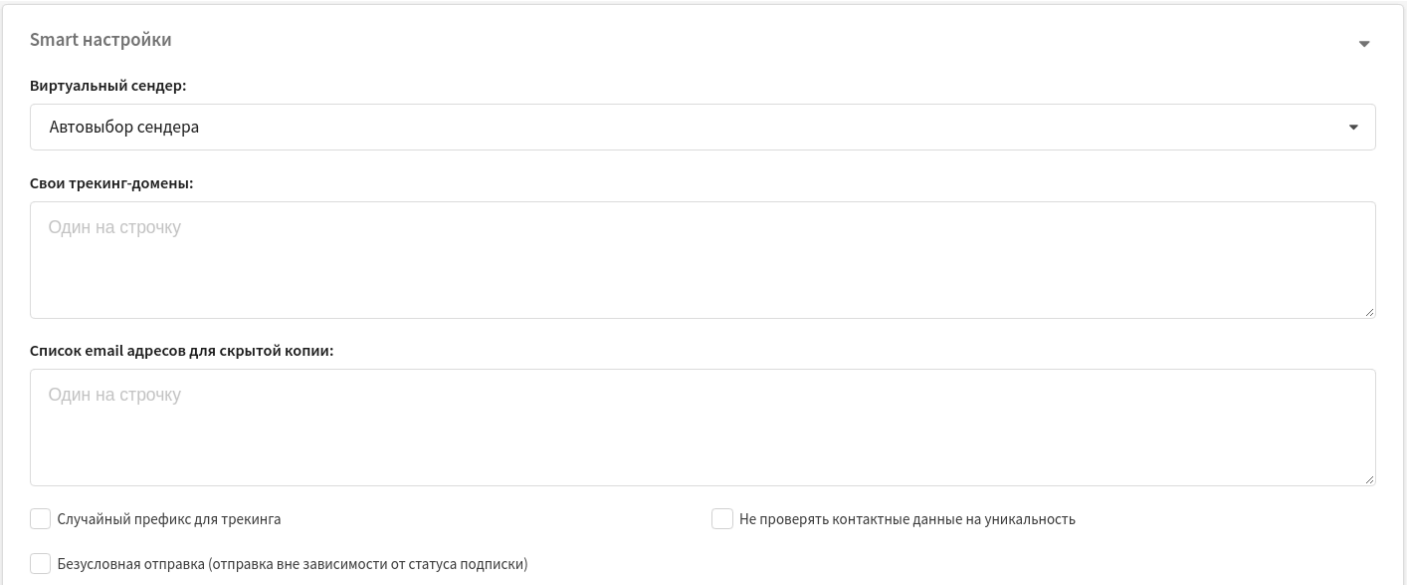
Additional settings



In the Limitations block, you can configure limits and sending policy.

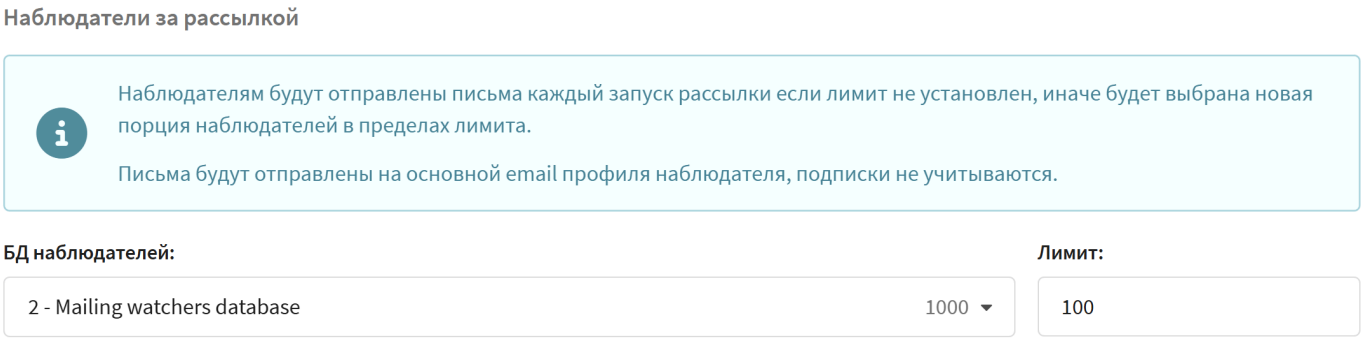
The Smart Settings block includes the following additional mailing options:

- **Virtual sender — lets you select a virtual sender for the mailing (a set of rules for sending messages from the platform's real senders).**
- **Custom tracking domains — tracks message interactions using additional tracking servers. Specify each domain on a separate line.**
- **BCC email list — BCC recipients for outgoing messages. Specify each address on a separate line.**
- **Do not check contact data for uniqueness — disables uniqueness validation of mailing recipients based on subscription data.**



You can also configure mailing watchers in Smart Settings. Watchers receive the same messages as mailing recipients. To do this, you need a watchers database. It is created the same way as a regular database, but you must enable the corresponding option in General settings.

In Smart Settings, select the watchers database and set the message sending limit for each mailing launch:



If you want to launch a mailing right after its creation, toggle the option Start the mailing right now at the bottom part of the page.

Campaigns

INFO: Terminology Update

Previously, all types of marketing broadcasts were called Campaigns in the platform. In the new platform version, they have been renamed to Broadcasts, while the new Campaigns represent a broader tool.

Campaigns are a tool for the comprehensive management of marketing activities within a single workspace.

It allows you to configure audiences, segment databases, manage user profiles, control groups, and scenarios.

Within campaigns, you can both create and transfer existing marketing activities: scenarios, broadcasts, recurring, triggered, multivariate campaigns, and placements.

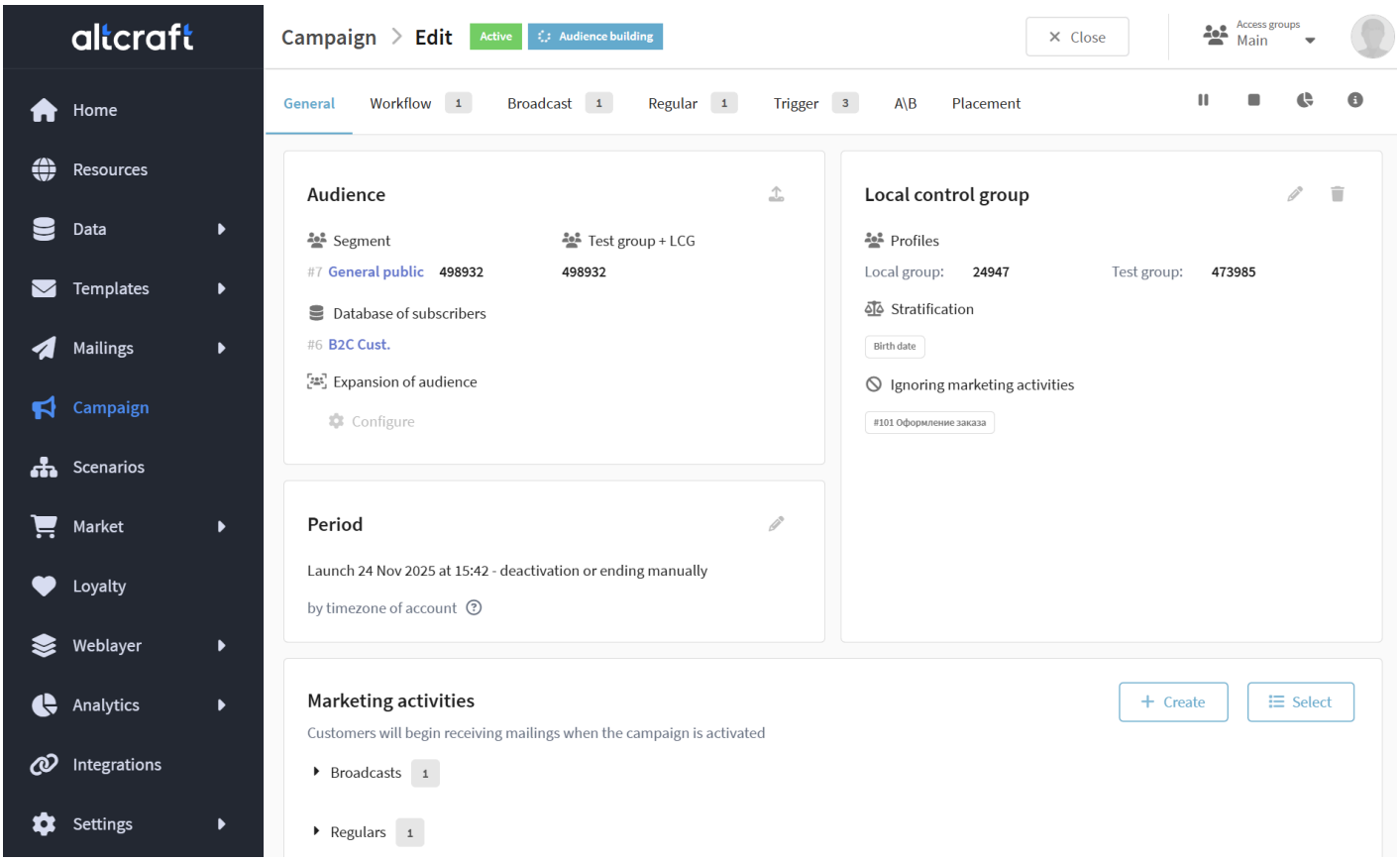
Working with Campaigns

Description

INFO: Terminology update

Previously, all types of marketing mailings in the platform were called Campaigns. In the new version they have been renamed to Mailings, and the new Campaigns are a broader tool.

Campaigns are a tool for uniting marketing activities into a single strategy and managing them in one place. A campaign can contain broadcasts, regular and triggered mailings, scenarios, A/B tests and placements. You can configure test and control groups, track strategy performance and manage the lifecycle of activities from a single interface:



Creating a campaign

Before creating a campaign, prepare a static or updatable segments for the target audience. It defines who will join the activities.

The campaign audience is the current composition of the segment at the time of campaign activation and each subsequent segment update.

- **Profile not in segment = profile not in campaign audience**

For example:

The campaign is configured with an updatable segment.
The segment is set to weekly update on Mondays with the condition “purchase within the last 7 days”:

A profile makes a purchase on April 9 (Sunday).

- **April 10 (Monday) — the nearest update. The profile enters the segment and therefore the campaign, since the purchase was within the last 7 days.**
- **April 17 (Monday) — the next update. Since there were no purchases from April 10 to April 17, the profile no longer meets the conditions and is removed from the segment and, accordingly, from the campaign.**

For a static segment
A profile will also be removed from the campaign audience if it ceases to be a member of the static segment. This can happen when:

- **manually deleted from the segment;**
- **deleted by conditions;**
- **deleted via API;**
- **deleted via a scenario;**
- **any other way of excluding a profile from a static segment.**

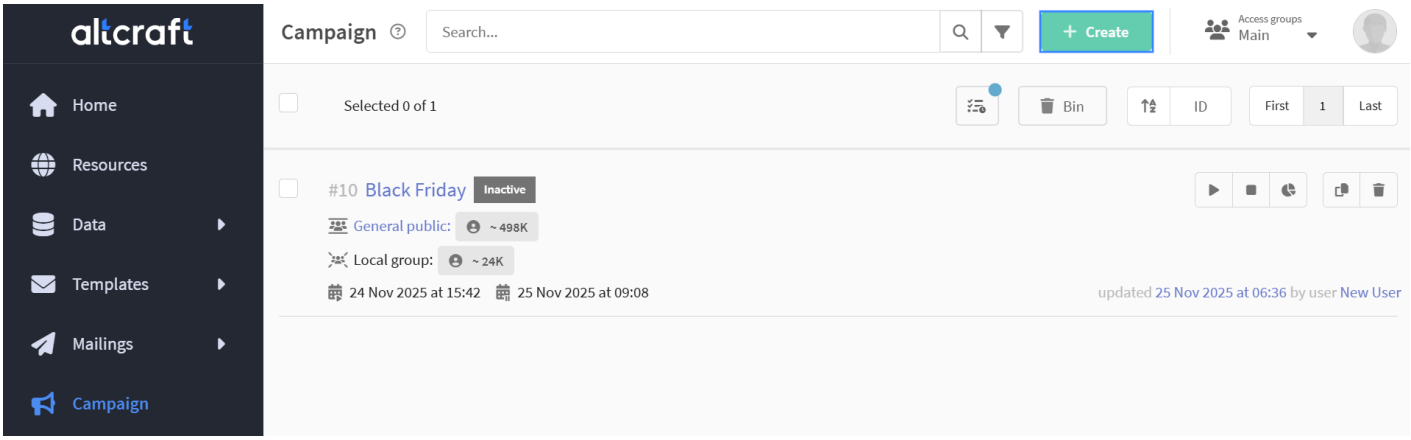
Profile exit from campaign
If a profile leaves the segment audience, it will also leave the campaign audience. In this case:

- **The profile receives a “Campaign exit” event;**
- **All active campaign scenarios for this profile are deactivated;**
- **If the profile later re-enters the segment, it will be added to the campaign as a new one and will undergo re-labeling.**

INFO: Please note:

When using a dynamic segment as the main campaign segment, the audience expansion function is not available.

Open the Campaigns section and click `+ Create`:



Specify the name, profile database and static or updatable segment.
If mandatory attributes exist, select them as well.

NOTE

The segment defines the campaign audience. On activation the system splits it into:

- **Test group — profiles that receive activities;**
- **Local control group (LCG) — profiles excluded from activities, if LCG is enabled.**

CAUTION

After creation the segment and database cannot be changed.

Managing marketing activities

The Marketing activities section contains all activities linked to the campaign.

You can create new ones or attach existing ones.

NOTE

Unavailable for selection:

- **activities from another database;**
- **mailings and scenarios with statuses “Active” or “Completed”;**
- **scenarios with active profiles;**
- **activities already used in other campaigns;**
- **broadcasts and regular mailings launched at least once.**

Available activity types:

- **Broadcasts**
- **Regular mailings**
- **Triggered mailings**
- **A/B tests**
- **Scenarios**
- **Placements**

You can specify a mailing segment. The profile database must always match the campaign. If the activity segment differs from the campaign segment, only profiles present in both segments receive the activity.

Intersection examples

Partial overlap

Campaign: database `CRM_Base`, segment `Loyal customers` — 100 profiles.
Broadcast segment: `Recent buyers` — 60 profiles, 40 intersect.
Result: message goes to 40 profiles.

No overlap

Campaign: `CRM_Base`, segment `Loyal customers` — 100 profiles.
Broadcast: `New registrations` — 50 profiles with no intersection.
Result: 0 recipients.

Moving an activity

Find the activity → click Move:

Choose destination — external list or another campaign:

> INFO
>
> **You cannot move:**
>
> - **active or previously launched activities;**
> - **scenarios with profiles;**
> - **activities into a completed campaign.**

Local control group (LCG)

What it is

The Local control group (LCG) is a part of the audience that does not receive campaign activities. Used for precise audience control. LCG analytics is not available yet.

> TIP: LCG vs GCG
>
> **LCG works inside a single campaign.**
> **GCG excludes profiles from all activities on the platform.**
>
> **If a profile is in GCG, it cannot join the campaign unless added to exceptions.**
> **GCG is configured in database settings.**

Basic setup

Set the percentage of profiles to exclude.
Option Audience limit lets you set an exact number:

> TIP
>
> **Set up LCG before activation.**
> **If you change LCG parameters later, the platform recalculates the audience and pauses broadcasts and regular mailings.**
> **After activation only new profiles can enter LCG.**
>
> **If the campaign is configured for a dynamic segment, audience tagging will be triggered with each recalculation.**

Stratification

Stratification splits the audience into groups by selected fields (gender, age, city).
A share of each group is placed into LCG:

Usage example: A marketer needs to include men and women from two age groups in the control group:

- **from 20 to 25 years old (date of birth from 01.01.2000 to 01.01.2005);**
- **from 25 to 40 years old (date of birth from 01.01.1985 to 31.12.1999).**

To do this, they select the Gender and Date of birth fields, define two date-of-birth ranges, and add gender values (male and female). The system will create four strata — men 20–25, women 20–25, men 25–40, and women 25–40 — and distribute participants between the test and control groups according to the configured settings.

> CAUTION: Note
>
> **It is not recommended to change stratification settings in an active campaign or for an already formed LCG.**
>
> **Changing values inside fields (adding new cities, expanding age ranges, etc.) may lead to loss of LCG homogeneity — proportions between strata will change, and some current members may move to the test group.**

If you change the set of stratification fields (for example, adding stratification by the `age` field or removing stratification by the `region` field) — this is a completely different division grid. Current LCG members do not match the new strata, the system cannot map old members to new strata, and moves all of them to the test group. The control group then starts forming from scratch and only from new users.

Ignoring activities

Some activities can send messages even to LCG.
Useful for transactional messages like order notifications:

Status management

Activation

> CAUTION
>
> **A campaign must contain at least one activity to be activated.**

Click ▶ to open activation settings:

Two options:

- **Activate now**
- **Activate on a specific date**

Be careful when selecting the date and time for launching the campaign. If you specify a past date, the campaign will wait for launch indefinitely and no warning will be displayed.

> NOTE
>
> **Activating a campaign does not auto-start its activities.**
> **Inactive mailings or scenarios must be launched manually.**

Completion settings

- **Deactivation — temporary stop, profiles keep group membership.**
- **Completion — full stop, profiles are removed.**

> NOTE
>
> **A completed campaign cannot be activated again.**

Status table (updated)

Status	Description and impact on activities
Inactive	**Assigned after a campaign is created and kept until it is launched. Activities do not run.**
Audience building	**Audience calculation and distribution of profiles between test and control groups is in progress. During this phase broadcasts and regular mailings are paused.**
Waiting	**The campaign is activated with a delayed start. The campaign starts automatically at the scheduled time.**
Active	**The campaign is running, the audience is built. Activities run according to their settings.**
Audience exclusion	**The system removes profiles from the campaign and detaches them from scenarios. New mailings do not run.**
Completed	**The campaign is fully stopped. Activities no longer run, relaunch is impossible.**

Working with profiles

Test and control groups

After activation, audience building starts.
The system distributes profiles into the test group and LCG (if enabled).

If LCG is disabled, all profiles go to the test group.

Only the selected database is used.

> NOTE
>
> **Audience building starts only after activation.**

Audience expansion

Audience expansion adds new users to the campaign if they participate in selected activities.

INFO: Please note:

When using an updatable segment as the main campaign segment, the audience expansion function is not available. New profiles are added to the campaign audience only as a result of segment recalculation.

When an activity runs, the system checks whether the recipients are already in the campaign and adds missing ones.
New profiles are then distributed into test group and LCG:

Examples

Without expansion
No intersection → message sent to no one.

With expansion
Recipients are added to the campaign and distributed into groups.
Test group receives the message; LCG does not.

Manual profile addition

In the profile card → Interactions → Campaigns → choose campaign:

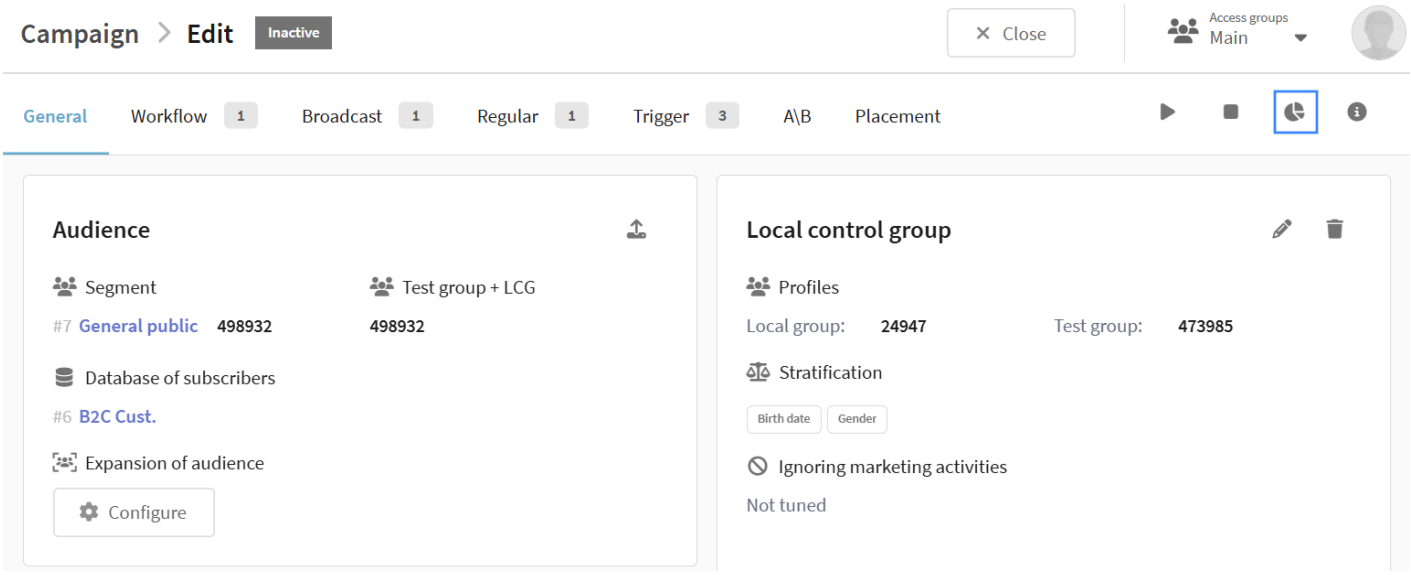
INFO

Manual profile addition is available only for static segments.

To force a profile into the test group, mark Send profile to test group even if it is in the control group.

Analytics

Click Analytics on the top panel:



The system opens the analytics section filtered by the campaign:

Additional features

Profile export

Done in the Audience section:

Available parameters:

- **subscription status filter**
- **audience group (Test, LCG, GCG)**
- **field selection**
- **export limit**

> INFO
>
> **Export is available after audience building is finished.**

Cloning

Cloning creates a full copy with all activities and settings:

Database cannot be changed.
Segment can be changed.
New campaign is Inactive.

Deleting

Accessible in statuses Completed or Inactive with no allocated profiles.

> CAUTION
>
> **Deleted campaigns go to the recycle bin.**
> **When restored, they return as Completed and cannot be activated.**

Local control groups (LCG)

Description

A Local control group (LCG) is the part of a campaign's audience that does not participate in marketing activities. It is used for precise audience management and for assessing how the campaign affects recipients. At the moment, LCG reporting is not available.

A Global control group (GCG) works across the entire profile database and excludes selected profiles from all activities on the platform. If a profile belongs to the GCG, it cannot participate in a campaign unless that campaign is added to the GCG exception list. GCG is configured in the profile database settings. Details are in the separate GCG article.

LCG setup

Configure the local control group via the Local control group widget on the campaign editor's main screen by clicking Configure:

Group size setup

Specify the size of the control group in percent. Fractional values are supported:

If you enable Audience limit, you can set the maximum number of profiles for the control group. The actual group size will not exceed this value but may be lower depending on the percentage and available audience.

> INFO
>
> **Recommended LCG size — 5% of campaign audience. Too small a group may not give statistically valid results; too large lowers campaign reach.**

Stratification

Description

Stratification is a method of random sampling after splitting the audience into subgroups (strata) based on selected characteristics. The audience is divided so that each subgroup reflects the composition of the full audience.

A share of profiles is selected from each stratum proportional to its size. This creates a control group with the same distribution of characteristics as the test group, which improves accuracy. The maximum number of strata depends on the number of fields and conditions configured in the system.

A stratum is a set of conditions defined by one or more criteria. Each unique combination forms a separate stratum.

A field is a characteristic used to form strata.

> CAUTION: Note
>
> **It is not recommended to change stratification settings in an active campaign or for an already formed LCG.**
>
> **Changing values inside fields (adding new cities, expanding age ranges, etc.) may lead to loss of LCG homogeneity — proportions between strata will change, and some current members may move to the test group.**
>
> **If you change the set of stratification fields (for example, adding stratification by the `age` field or removing stratification by the `region` field) — this is a completely different division grid. Current LCG members do not match the new strata, the system cannot map old members to new strata, and moves all of them to the test group. The control group then starts forming from scratch and only from new users.**

Stratification example

Suppose the marketer needs to split the audience by three criteria:

- **City: Moscow, Saint Petersburg**
- **Gender: male, female**
- **Age: 18–30**

The system forms strata for all combinations:

- **Men aged 18–30 from Moscow**
- **Men aged 18–30 from Saint Petersburg**
- **Women aged 18–30 from Moscow**
- **Women aged 18–30 from Saint Petersburg**

Stratum share calculation
Suppose your target audience consists of:

- **50% men, 50% women**
- **40% aged 18–30**
- **60% from Moscow, 40% from Saint Petersburg**

The share of “Men 18–30 from Moscow” in the entire audience:
0.5 × 0.4 × 0.6 = 0.12 (12%)

Profiles for the control group are selected from each stratum proportionally to its share.

Stratification setup

Stratification settings are in the same window as the main LCG parameters and are available via a dropdown menu:

Set strata by one or multiple fields, defining conditions such as date ranges, numeric ranges, or lists of values. The setup is done in the UI, where you can add strata and define parameters for each:

Supported field types

You can use core and additional fields of these types:

- **enum — predefined list of values**
- **bool — true/false**
- **int/float — numeric values**
- **string — text**
- **date — date field**

Field conditions

For each type, different conditions are available:

- **Number — range (“from–to”)**
- **Float — range with inclusive/exclusive borders**
- **String — “equals”, with multiple values**
- **Boolean — true, false, missing**
- **List — “equals”, multi-select**
- **Date — fixed or relative range**

Stratification limits

- **Minimum conditions per field — 2**
- **Maximum conditions — 64 total**
- **Maximum fields — 32**

The limit is calculated by multiplying the number of conditions in selected fields.
Example: City (3 values) × Age (2 values) = 6 strata.

TIP: Note

Values in one field must not fully overlap:

- **Dates — ranges must not overlap**
- **Numbers — ranges must not fully cover one another**
- **Strings/lists — no full duplication of values**

Strata recommendations

- **Up to 100k profiles — 5–10 strata**
- **100k–1M profiles — 10–20 strata**
- **1M+ profiles — 20–30 strata**

Recommended number of conditions per field: 2–4. More than 5 may create overly narrow strata.

Stratification Error

The system may issue a warning if the LCG limit is too small to proportionally fill all strata. The warning does not stop the campaign but indicates that proportions may be violated. For example:

Parameter	Value
Segment	**10,000 profiles**
LCG	**10% (should be 1,000 profiles)**
Stratification	**Region: Moscow, Saint Petersburg**
LCG limit	**500 profiles**

Initial distribution

Stratum	In segment	Share	Expected in LCG (with limit 500)
Moscow	**5,000**	**50%**	**250**
Saint Petersburg	**5,000**	**50%**	**250**

Proportions are met, no error. Then 5,000 new profiles from Moscow are added to the segment. Now the segment has 15,000 profiles: 10,000 from Moscow, 5,000 from Saint Petersburg.

Stratum	In segment	Share	Should be in LCG (no limit)	Expected in LCG (with limit 500)
Moscow	**10,000**	**66.7%**	**1,000**	**333**
Saint Petersburg	**5,000**	**33.3%**	**500**	**167**

With an LCG limit of 500, the required LCG size should be 1,500 (10% of 15,000 profiles in the segment). Based on shares of 66.7% and 33.3%, the correct distribution for 500 spots is 333 Muscovites and 167 Petersburgers. The actual LCG composition (250 and 250) does not match this. The LCG cannot be expanded due to the limit, so the system issues a warning.

How to fix: increase the limit to the actual LCG size (at least 1,500) or disable it entirely. In that case, the LCG will be able to add the necessary profiles for the correct proportion during subsequent audience updates.

Ignoring marketing activities

Description

The Ignore marketing activities setting lets you choose activities that must send messages to all profiles, including the control group. Useful for critical communications that must reach everyone.

TIP: Examples

- **Transactional notifications — order status messages**
- **Legal updates — changes to terms or privacy policy**
- **Technical alerts — system maintenance messages**
- **Emergency notifications**

Ignore setup

Located in the same LCG settings window:

To edit the list of activities that ignore LCG, click

Edit:

Audience building process

When you configure LCG and activate the campaign, the system starts the audience building process.
It also starts when LCG settings change in an active campaign.

During audience building:

- **Profiles are distributed between the test group and the control group**
- **Mailings are paused until the process completes**
- **The campaign receives an additional Audience building status while its main status stays unchanged**

If LCG is enabled after the campaign is already active, the control group will contain only profiles added to the campaign after activation, such as through audience expansion. Profiles initially assigned to the test group remain in the test group.

CAUTION

If broadcasts or regular mailings are active when audience building starts, they are paused until it finishes.

LCG management

Once LCG is configured, the following actions are available in the campaign interface:

- **Edit settings — change LCG parameters, stratification, and ignored activities;**
- **Delete LCG — completely remove the local control group. All related settings (Stratification and Ignored Activities) will be deleted along with the LCG. As a result, profiles from the control group will be moved to the test group;**
- **Export profiles — export LCG participants. After audience building completes, you can select local control group participants for export using filters in the export settings. Export is available in the Audience widget. For more details, see here.**

Features of the LCG with segment recalculation

When the audience of the segment on which the campaign is based changes, the LCG mechanism has several specific features:

Labeling New Profiles
Audience labeling is triggered automatically after each segment update. The system distributes only the new profiles added to the segment between the test and control groups. Profiles already in the campaign retain their group.

Changing LCG Settings
When control group parameters (percentage or stratification) are changed, or when the LCG is completely deleted, the new settings apply only to new profiles that may be added to the campaign as a result of audience expansion or segment audience updates. Already distributed profiles do not change their group.

Stratification Violation Error

A stratification violation error occurs when, due to an imposed limit, the system cannot maintain proportions between strata when forming a local control group (LCG). The warning does not stop the campaign but indicates that the control group may be formed with violated proportions.

The examples below use a database of 100 profiles. This small size is chosen intentionally — it makes it easier to track the logic behind the error. In real projects, databases can be much larger, but the principle of stratification remains the same.

Database fields and their distribution:

Field	Values	Distribution
`int_field`	**20 / 80**	**20% / 80%**
`region`	**Moscow, SPb, Kazan**	**~34 / ~33 / ~33 profiles**
`age`	**18–54**	**uniform**

DANGER: Important

The error notification appears only when the limit is reached during LCG selection. During LCG formation, deviations may occur that the system tries to correct using new profiles added to the campaign, but cannot do so because of the imposed limit.

Example 1. Limit too small for the number of strata

Initial data

Parameter	Value
Database	**100 profiles**
Stratification	`int_field` **(20 / 80) and** `region` **(Moscow, SPb, Kazan)**
Number of strata	**6**
LCG size	**15% (should be 15 profiles)**
LCG limit	**12 profiles**

Stratum distribution in the database

Stratum	In database	Share	Expected in LCG (share × limit)
int_field=20 × Moscow	**6**	**6%**	**0.72 → ~1**
int_field=20 × SPb	**7**	**7%**	**0.84 → ~1**
int_field=20 × Kazan	**7**	**7%**	**0.84 → ~1**
int_field=80 × Moscow	**28**	**28%**	**3.36 → ~3**
int_field=80 × SPb	**26**	**26%**	**3.12 → ~3**
int_field=80 × Kazan	**26**	**26%**	**3.12 → ~3**

Problem

Due to the limit of 12 profiles, the expected value for the stratum “20 × Moscow” is 0.72 — less than 1. The system cannot reliably fill such a stratum.

How to fix

- **Increase the limit (e.g., to 35 profiles) — in this case, the LCG size must also be increased proportionally;**
- **Remove the limit entirely, leaving only the LCG percentage;**
- **Simplify stratification by removing fields with rare values (e.g., leave only `int_field` without region breakdown).**

Example 2. Critical heterogeneity in three-field stratification

Initial data

Parameter	Value
Database	**100 profiles**
Segment	**Profiles under 25 years old — 21 profiles**
Stratification	`int_field` **(20 / 80),** `region` **(Moscow, SPb, Kazan),** `age` **(18-21, 22-25)**
Number of strata	**2 × 3 × 2 = 12**
LCG size	**10%**
LCG limit	**8 profiles**

Stratum distribution in the segment and expected values in LCG

Combination	In segment	Share	Expected in LCG
20 × Moscow × 18-21	**0**	**0%**	**0**
20 × Moscow × 22-25	**1**	**4.8%**	**0.38 → ~0**
20 × SPb × 18-21	**1**	**4.8%**	**0.38 → ~0**
20 × SPb × 22-25	**1**	**4.8%**	**0.38 → ~0**

Combination	In segment	Share	Expected in LCG
20 × Kazan × 18-21	**0**	**0%**	**0**
20 × Kazan × 22-25	**0**	**0%**	**0**
80 × Moscow × 18-21	**3**	**14.3%**	**1.14 → ~1**
80 × Moscow × 22-25	**2**	**9.5%**	**0.76 → ~1**
80 × SPb × 18-21	**2**	**9.5%**	**0.76 → ~1**
80 × SPb × 22-25	**5**	**23.8%**	**1.90 → ~2**
80 × Kazan × 18-21	**2**	**9.5%**	**0.76 → ~1**
80 × Kazan × 22-25	**4**	**19.0%**	**1.52 → ~2**

Problem

Due to the limit of 8 profiles, the expected value for strata with `int_field=20` is 0 or close to 0. Empty strata themselves are not an error — they may reflect the actual state of the database if such profiles simply do not exist. The error occurs because the expected number of profiles in a stratum is too small and cannot be increased even when the campaign audience expands, as the limit restricts it, causing a loss of stratification homogeneity.

How to fix

Remove the limit — then the system will be able to add profiles from new arrivals. Or simplify stratification: remove age or broaden age ranges.

Example 3. Very small limit

Initial data

Parameter	Value
Database	**100 profiles**
Stratification	**by `int_field` (20 / 80)**
LCG limit	**1 profile**
LCG size	**10% (10 profiles available, but limit restricts to 1)**

Problem

With a limit of 1 profile, the system cannot maintain proportions of 20% and 80%. In a sample of one profile, only two outcomes are possible: 0%/100% or 100%/0%. The deviation from the ideal proportion is 100%.

How to fix

Remove the limit or increase it to at least 10-20 profiles.

Audience expansion

Description

Audience expansion is a mechanism that automatically adds new profiles to a campaign from the segments of activities included in it. This mechanism increases campaign reach and keeps its audience up to date without manually updating the initial segment.

Each campaign is created using a static segment that defines its target audience. However, inside a campaign, each activity can be configured for any segment, even if it does not intersect with the main one.

For example, a campaign targets the Loyal customers segment, but inside it you create a mailing aimed at the New registrations segment. Without audience expansion, these users will not receive the message because they are not part of the campaign audience. But if audience expansion is enabled for this mailing, the system will automatically add recipients from New registrations to the campaign and distribute them between the test group and the local control group, if LCG is enabled.

INFO: Key limitations

- **Only activities of this campaign — you cannot select a mailing or scenario from another campaign**
- **Only non-completed activities — activities with the "Completed" status cannot be selected**
- **Only profiles from the campaign's Profile database — even if the activity segment contains profiles from other databases, only profiles from the campaign's database will be added**
- **Only new profiles — only profiles not yet in the campaign audience will be added**

How audience expansion works

Audience expansion enriches the campaign with new profiles directly from the audience of launched activities. Its core principles:

- **When an activity is launched, the system determines its audience (segment or entire base) and automatically adds profiles that are not yet part of the campaign;**
- **New participants are immediately distributed between the test group and the control group;**
- **Users outside the campaign's profile database will not receive the message;**
- **If a profile appears in the segments of several audience expansion activities within the same campaign, it is added only once. Its group assignment (test or control) is fixed at that moment, preventing accidental reassignment later;**
- **Audience expansion works only for active campaigns.**

CAUTION

Audience expansion adds profiles to the static segment of the campaign (its initial audience). Any recalculation of that static segment removes all profiles added through audience expansion.
To avoid this, it is not recommended to enable Update segment daily for the campaign segment.

Profile addition process

For every profile in the activity audience, the system checks:

- **Profile database match:**
  The profile must belong to the same profile database as the campaign.
  Otherwise, it will not enter the campaign audience and will not receive the message.
- **Campaign participation:**
  The system checks whether the profile is already in the campaign audience.
  - **If yes — it receives the message (unless it's in LCG).**
  - **If no — it is distributed into the test or control group based on LCG settings.**
    Profiles placed into the test group will receive the message; those placed into the control group will not.

Usage example

Scenario without audience expansion

The campaign uses the `CRM_Base` profile database and the `Loyal customers` segment (100 profiles).
A broadcast is created with the segment `New registrations` (50 profiles). These users are not part of the campaign and do not intersect with the main segment.

When the broadcast is launched, no one receives the message because there is no overlap — these profiles are not in the campaign.

Scenario with audience expansion

If you add this broadcast to the Audience expansion list in the campaign settings, the system includes all recipients from `New registrations` in the campaign.
After that, profiles are automatically distributed between the test group and the local control group.

Audience expansion setup

Audience expansion is configured in the Audience section of the campaign editor, in the Audience expansion block:

Selecting activities

The following types of activities are available:

- **Broadcasts — one-off mass sends**
- **Regular mailings — scheduled communications**
- **Triggered mailings — automated event-based sends**
- **A/B tests — variant comparison**
- **Scenarios — multi-step automations**
- **Placements — on-site content placements**

TIP

You can select multiple activities. Each activity independently adds new users to the campaign when it is launched.

Audience building

Process overview

Audience building is the process of distributing campaign profiles between the test group and the control group according to the configured settings.

The system processes only profiles belonging to the profile database selected when the campaign was created.

The process runs in the following cases:

- **When the campaign is activated — the initial distribution of profiles based on Local control group (LCG) settings**
- **When modifying an active campaign — editing LCG settings triggers recalculation**
- **When the segment is updated — newly added profiles go through incremental audience building. This happens in the background.**

Working with inactive campaigns

In an inactive campaign, profile-related actions behave differently. When a profile is removed from the segment:

- **The Left segment event is recorded in the profile card immediately**
- **The Left campaign event appears only after the campaign is activated**

If a profile is removed from the segment and then added back before activation, the system considers only the final state — being in the segment. Campaign-level “enter/exit” events for these intermediate actions are not recorded, but the profile card still shows full segment history.

LCG changes in active campaigns

Enabling the Local control group in an active campaign starts audience building. Scheduled mailings (broadcasts, regular mailings) are paused during the process. Only new profiles entering the campaign audience (for example, through audience expansion) can be placed into LCG.

Removing LCG in an active campaign redistributes the audience — all profiles from LCG are moved to the test group.

Campaign completion

When a campaign is completed, the system automatically records the Left campaign event for all participating profiles.

Automation scenarios

Description

Scenarios are sequences of actions used for marketing workflow automation. Scenarios can be convenient to control your customers' loyalty and their actions during the certain time periods. You can also use them to make your marketing processes interactive and thus engaging. Welcome sequences, abandoned carts, reactivation, cross-sales and many other mechanics can be easily automated using scenarios.

Scenarios can be launched:

- **Individually for each customer after a certain event, same as a trigger mailing.**
- **On schedule for a segment of customers, same as a regular mailing.**

In this section

Managing scenarios

INFO

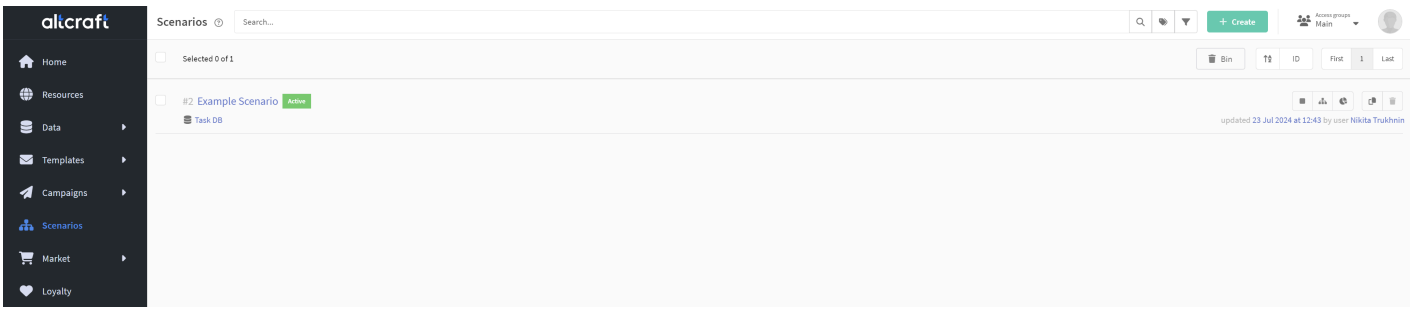
Elements (steps) of the scenario are described in a separate article.

INFO

Automation scenarios can be managed via API. Information is available at this link.

Scenario management

To start working with customer interaction scenarios, select Scenarios from the main menu. The list that opens displays brief information on existing scenarios, their activity statuses, and the number of profiles that are currently in the scenarios.



You can use the context menu to:

— activate the scenario.

— stop the scenario. If there are still profiles in the scenario at the moment of stopping, they can be saved. At the next scenario activation they will “wake up” and continue to go through the scenario from the element where they stopped.

— send profiles from the dynamic segment to the scenario (the segment must use the same profile database as the scenario).

— clone the scenario.

— move the scenario to the trash (available only for inactive scenarios).

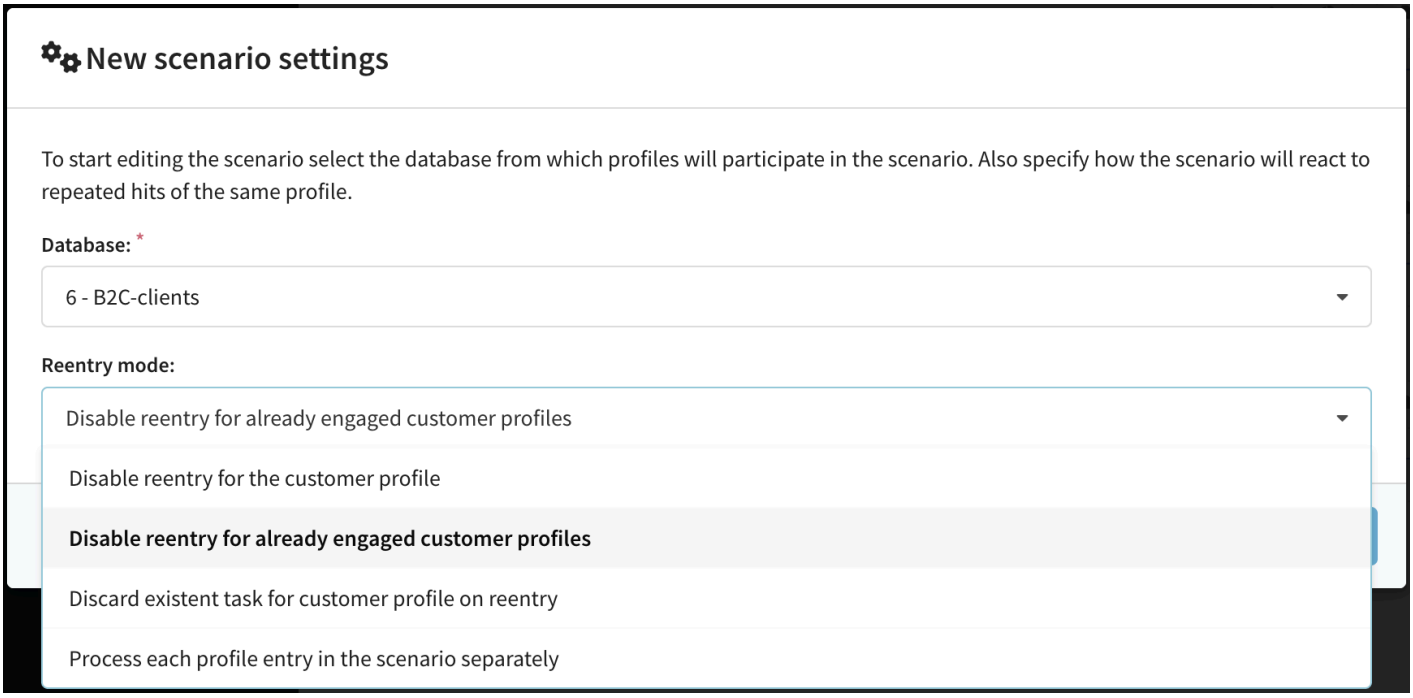
— go to the scenario report.

Getting started with the scenario

To add a new scenario, click + Create at the top of the page.

Pre-setting

In the scenario pre-configuration window that opens, specify the database that will be captured in the scenario, and re-entry mode (they can be changed). Then press the Continue button. You can change these settings later.



Reentry mode defines the behavior of the scenario when repeatedly capturing a client profile. Select one of the modes:

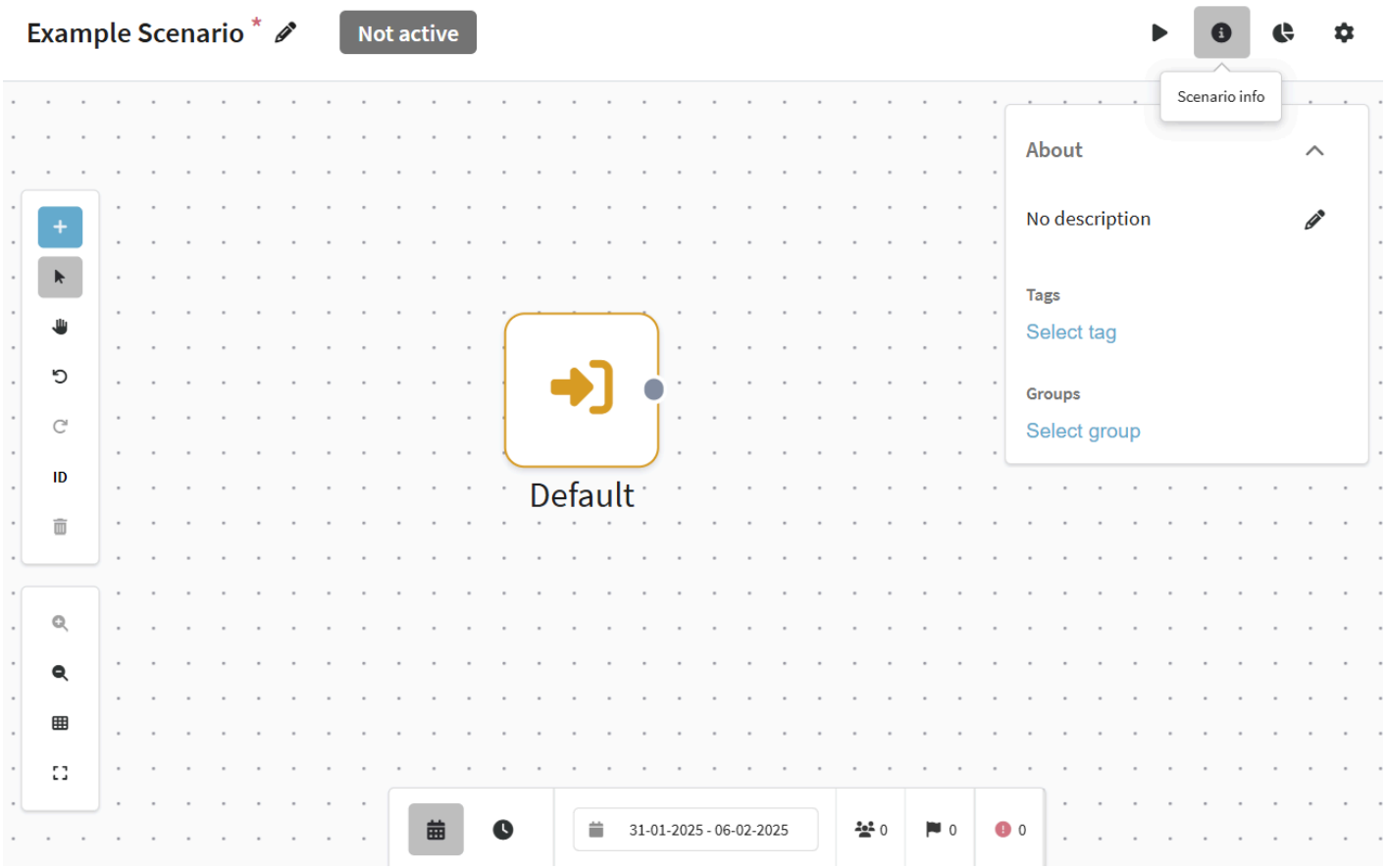
- **Disable reentry for the customer profile — in this case, the profile can only enter the scenario once, and even if it leaves it, it cannot enter it again. Never.**
- **Disable reentry for already engaged customer profiles — If the profile is currently already in the scenario, it cannot enter it again, but as soon as it exits, it can.**
- **Discard existent task for customer profile on reentry — if a client profile is recaptured by the scenario, its current task in the scenario will be immediately terminated and replaced by a new one. A task is an individual process of going through a scenario by an individual profile. All data associated with the current job (`apicontent`, etc.) will be deleted. The profile will start going through the scenario from the beginning, under the new task in the scenario.**
- **Process each profile entry in the scenario separately — no matter how many times the profile is present in the scenario, it will be captured again when the capture condition is met. Both captures will be processed completely.**

General settings

You will be taken to the editor. In the top panel, enter the name of the scenario. In the About section you can add a description, attach tags or attributes for quick search and assign groups for access.

To display or hide the About section, use the

button on the toolbar:



If you want to change the base of profiles in the scenario or choose a different re-entry mode, then on the top panel click

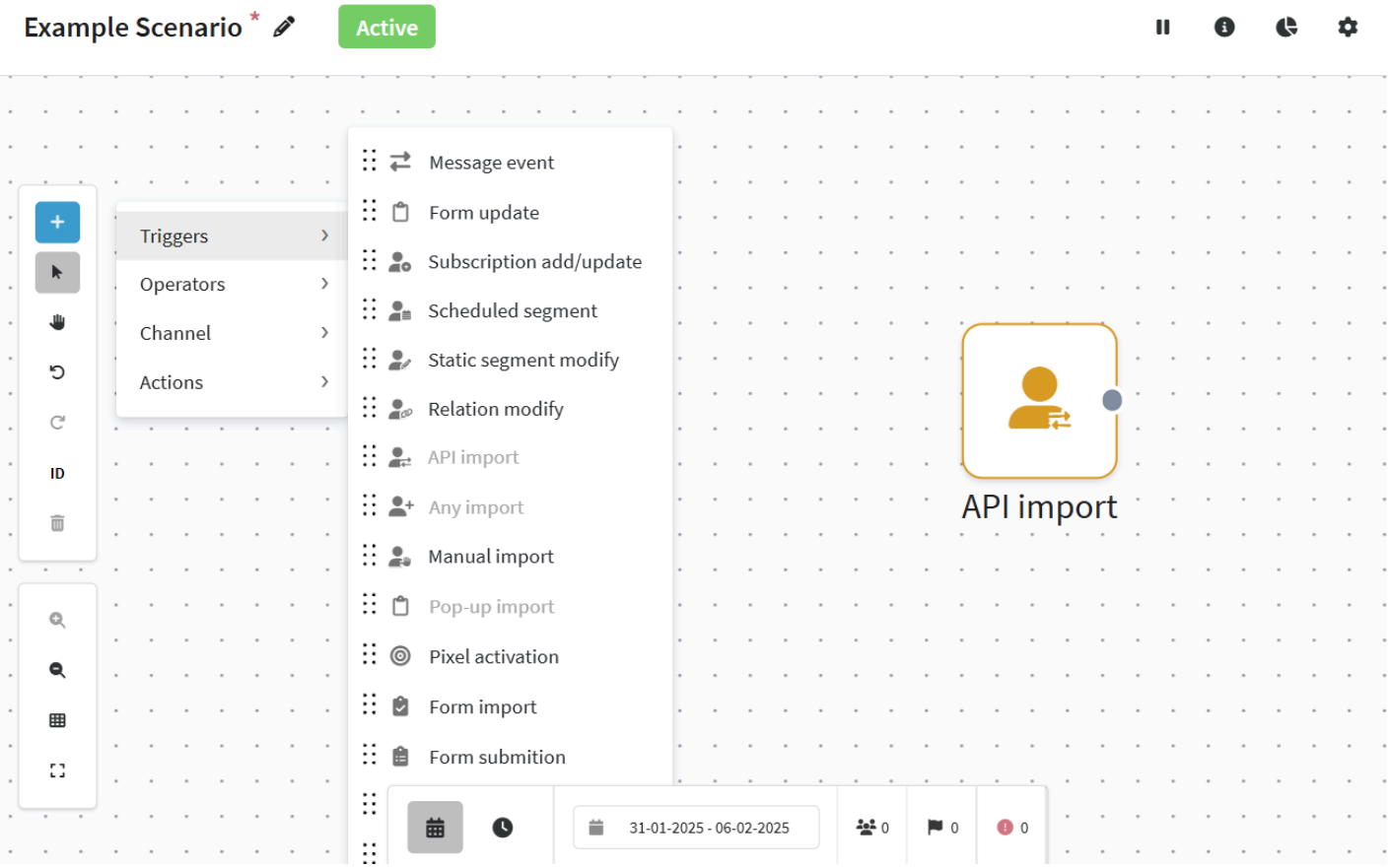
and edit the settings. The Clear all profiles option is also available in the same window - it allows you to remove the profiles that are currently there from the scenario.

If any required attribute is enabled for scenarios, you must select its value right away.

Configuring the capture of profiles to the scenario

The conditions for a profile to get into the scenario are determined by a trigger. A Trigger is a start element that starts the scenario. You can configure up to 10 triggers in one scenario.

To add a trigger, click + Element and select the desired item from the Triggers category:



TIP

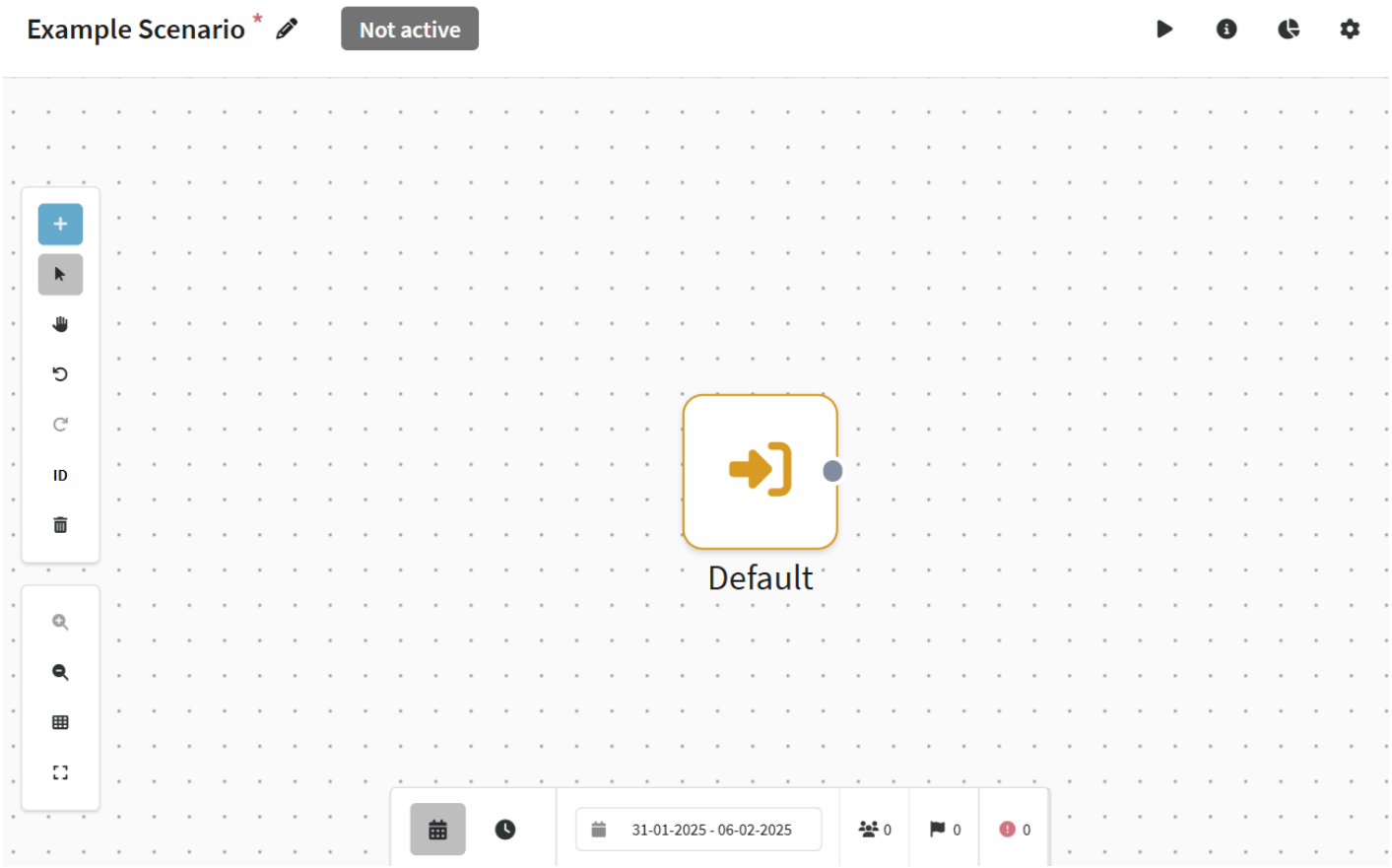
Detailed information on each trigger can be found in this article.

CAUTION

In order to save your scenario, you need to associate all start elements with any element.

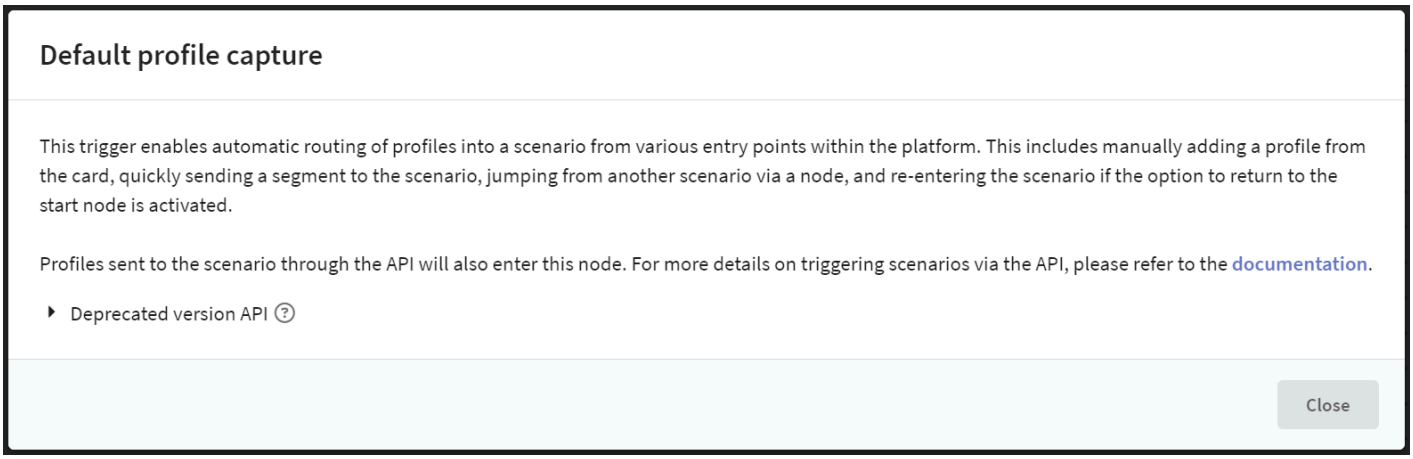
Default profile capture

Any scenario has an element Default profile capture. This element is a built-in feature of the scenario and can't be removed. Double click on it to read the description.

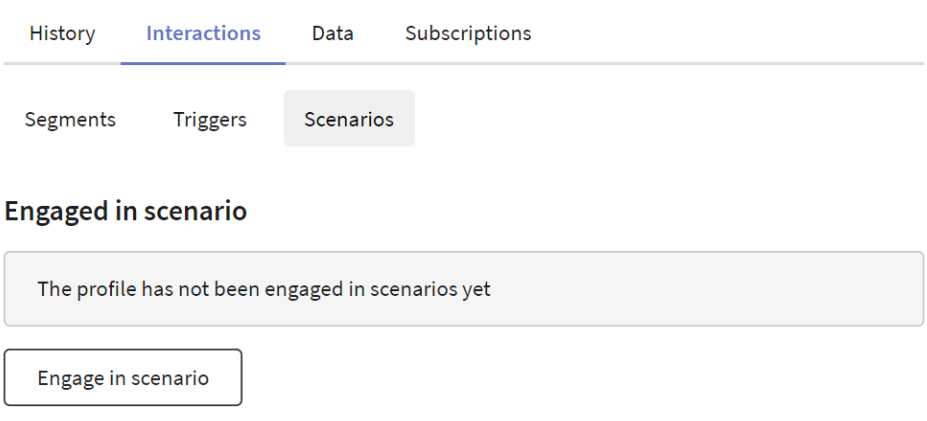


With the “Default” element, profiles can enter the scenario from different points in the platform:

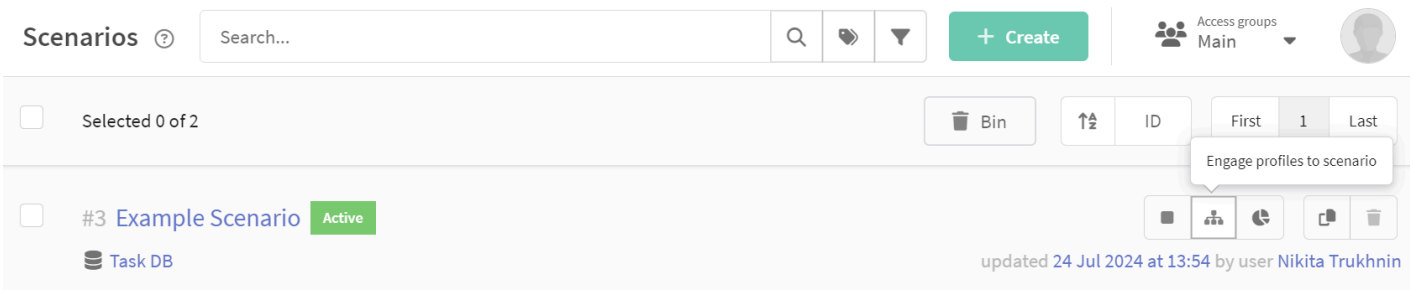
1. When sending a profile to the scenario via API. When the scenario is saved, the item window will display a URL that can be used to run it. However, it is recommended to use a newer version of the API. See documentation for more information.



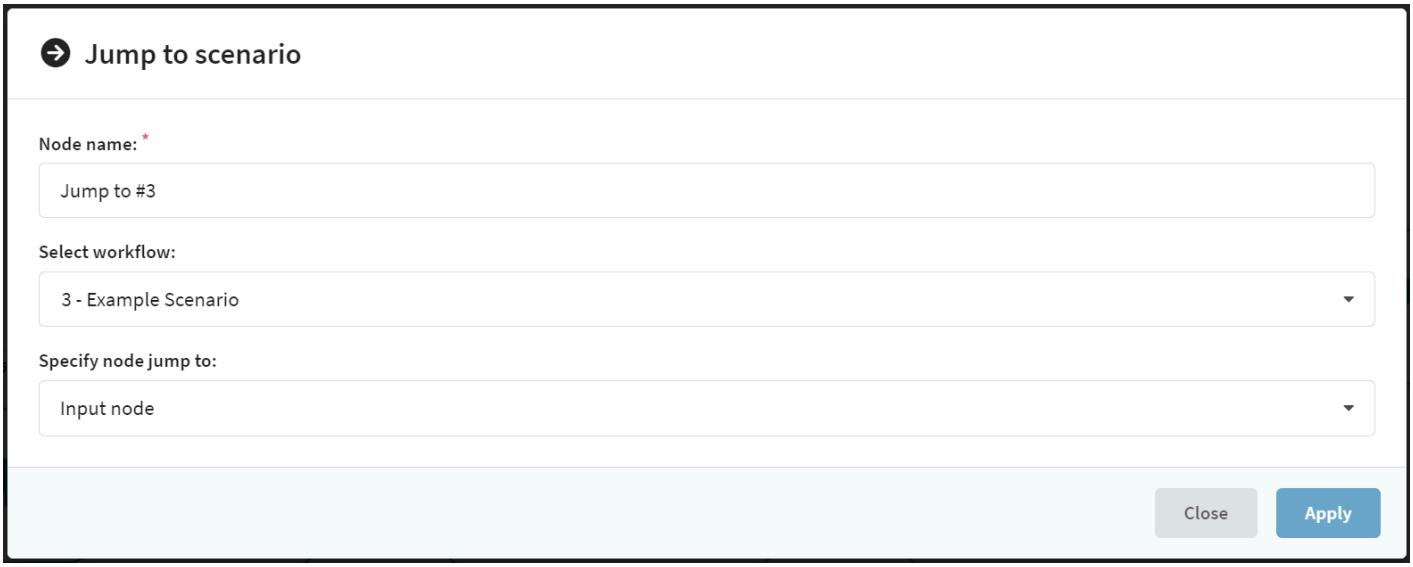
2. When adding a profile to a scenario manually:



3. When quickly sending a segment to a scenario:



4. Via the [“jump to scenario” node]() from another scenario:

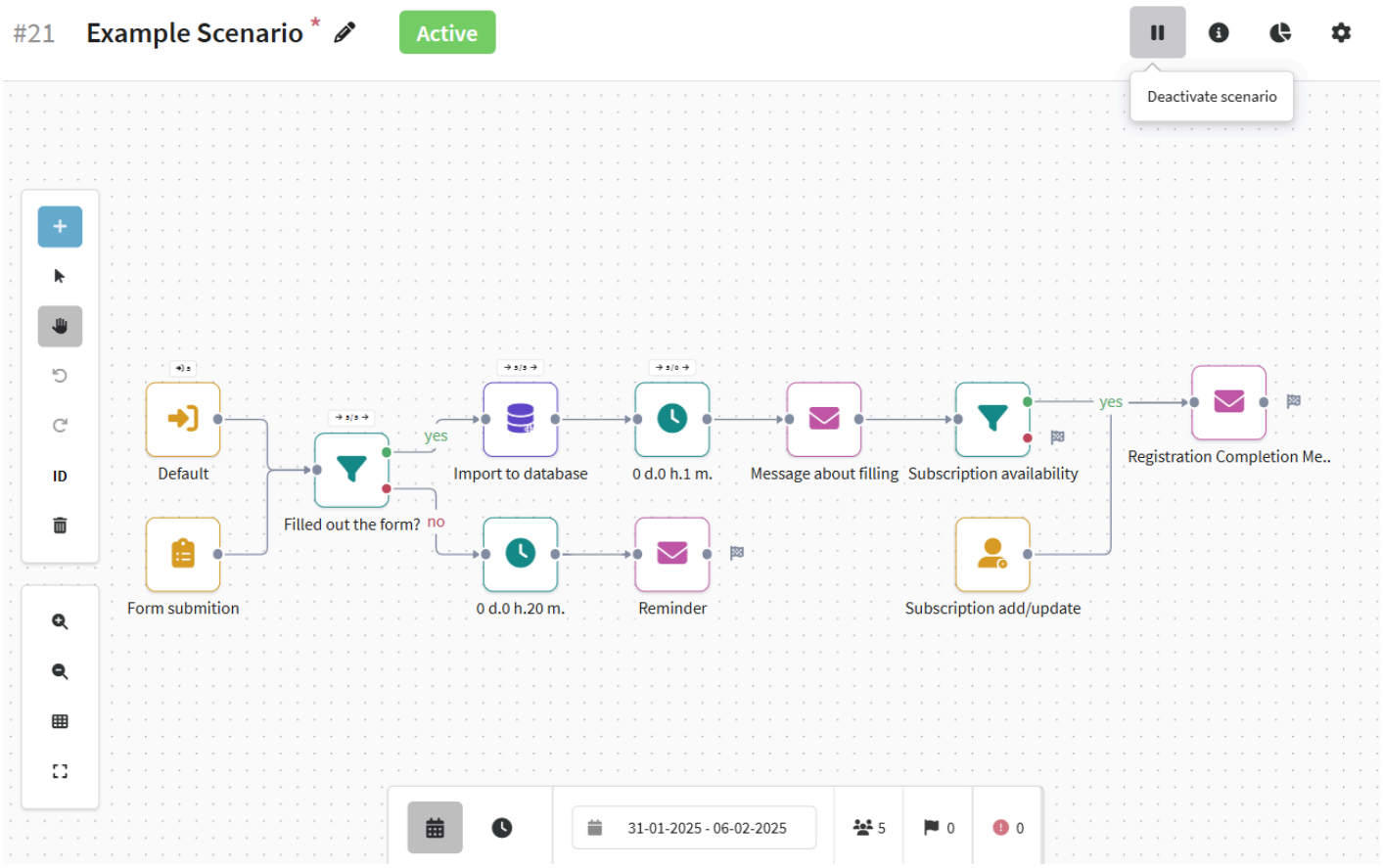


Activating and deactivating a scenario

To activate the scenario, click

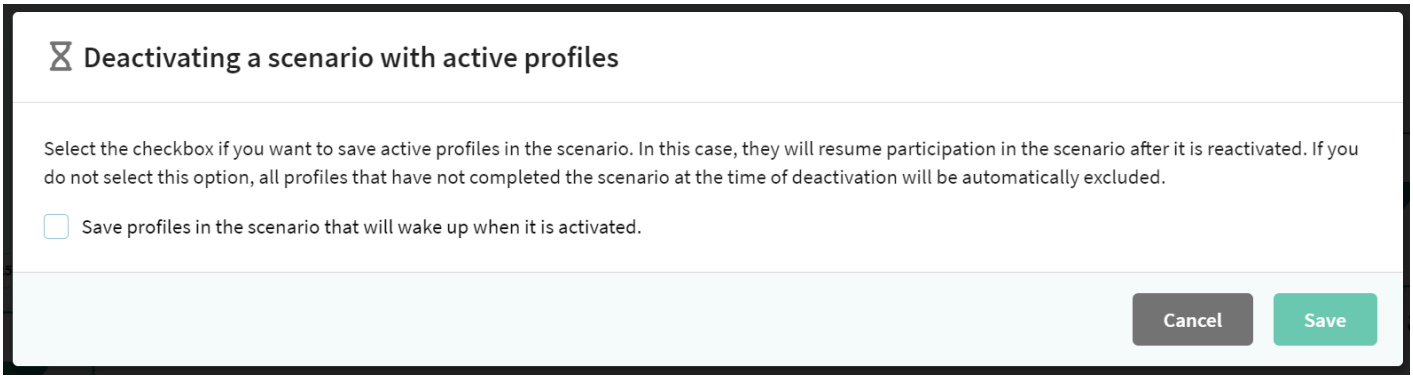
▶

and then save it. Once activated, the scenario will start capturing profiles according to the specified conditions.



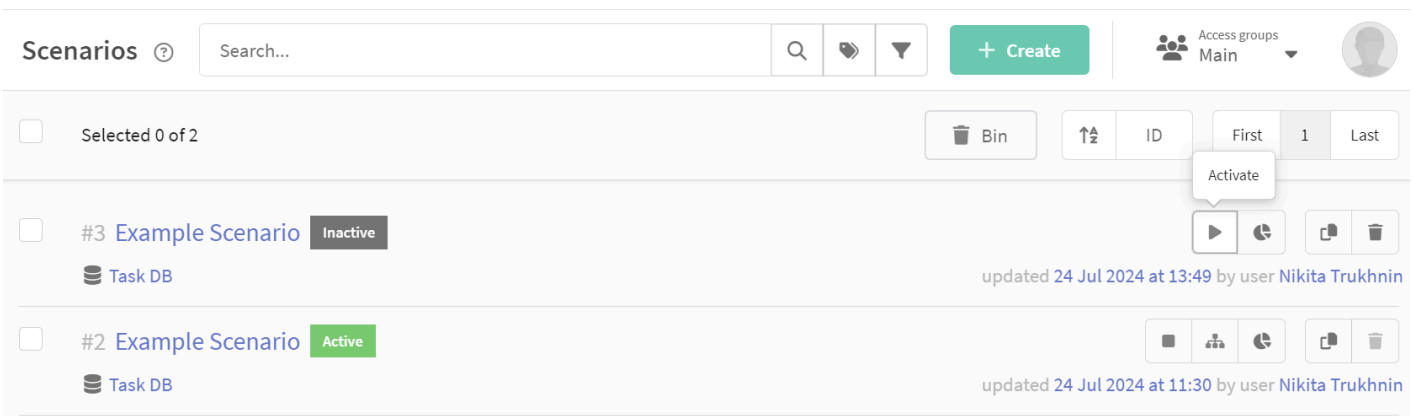
To stop the scenario, click

and save it. If there are profiles in the scenario at the moment of deactivation, you will see a warning window:



By default, after deactivating a scenario, all profiles will exit it. To avoid this, enable the setting Keep profiles in the scenario that will wake up when the scenario is activated. In this case all profiles in the scenario will be frozen, and after reactivation of the scenario they will continue to follow it.

You can also activate and deactivate a scenario from the general list:

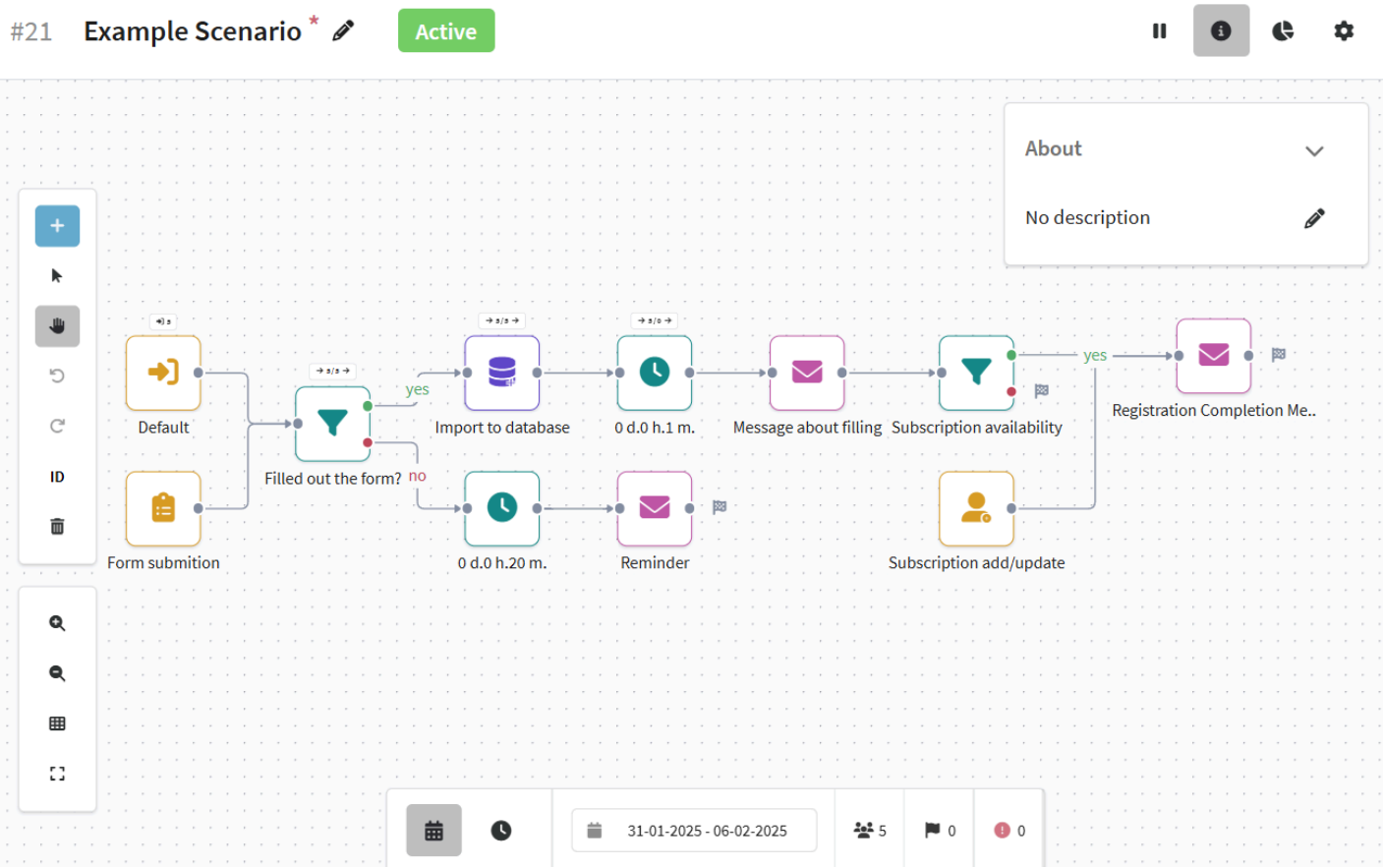


Scenario analytics

You can analyze the effectiveness of your scenarios directly in the scenario editor, and you can see detailed statistics on the mailings used in the chain in the channel report.

Client Scenario Completion Statistics

The statistics panel is located at the bottom of the editor. This panel contains information about completion of the scenario by profiles.



Viewing statistics is available in two modes:

— period statistics

— real time statistics

period statistics

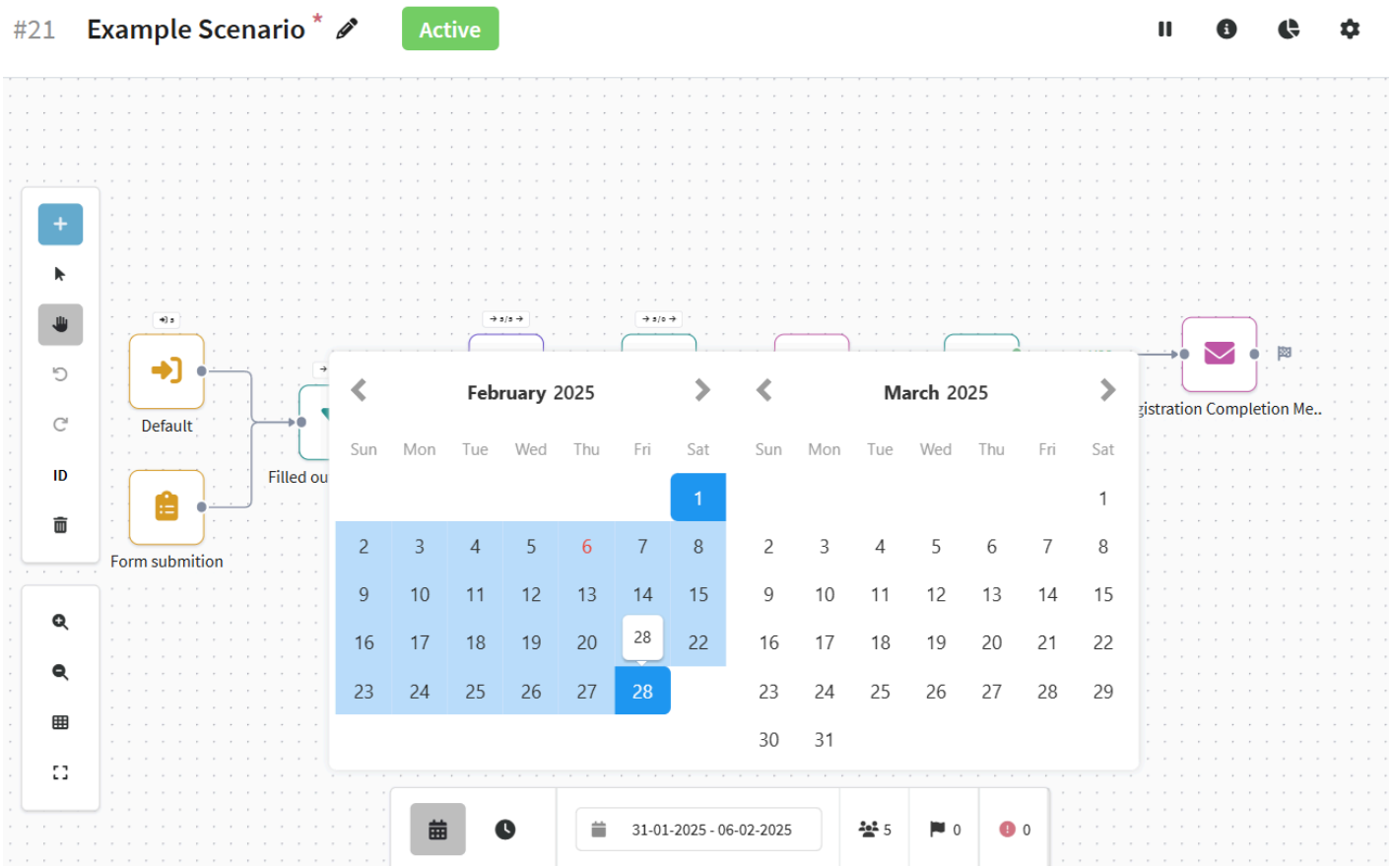
Here you can view scenario data for a specific time period:

— total number of profiles included in the scenario.

— number of profiles that have completed the scenario.

— errors that occurred during the scenario process.

By default, the panel displays data for the last 7 days. To change the period, enter the desired dates manually or select them from the calendar:



Also above each scenario element it is shown how many profiles entered the element and how many exited.

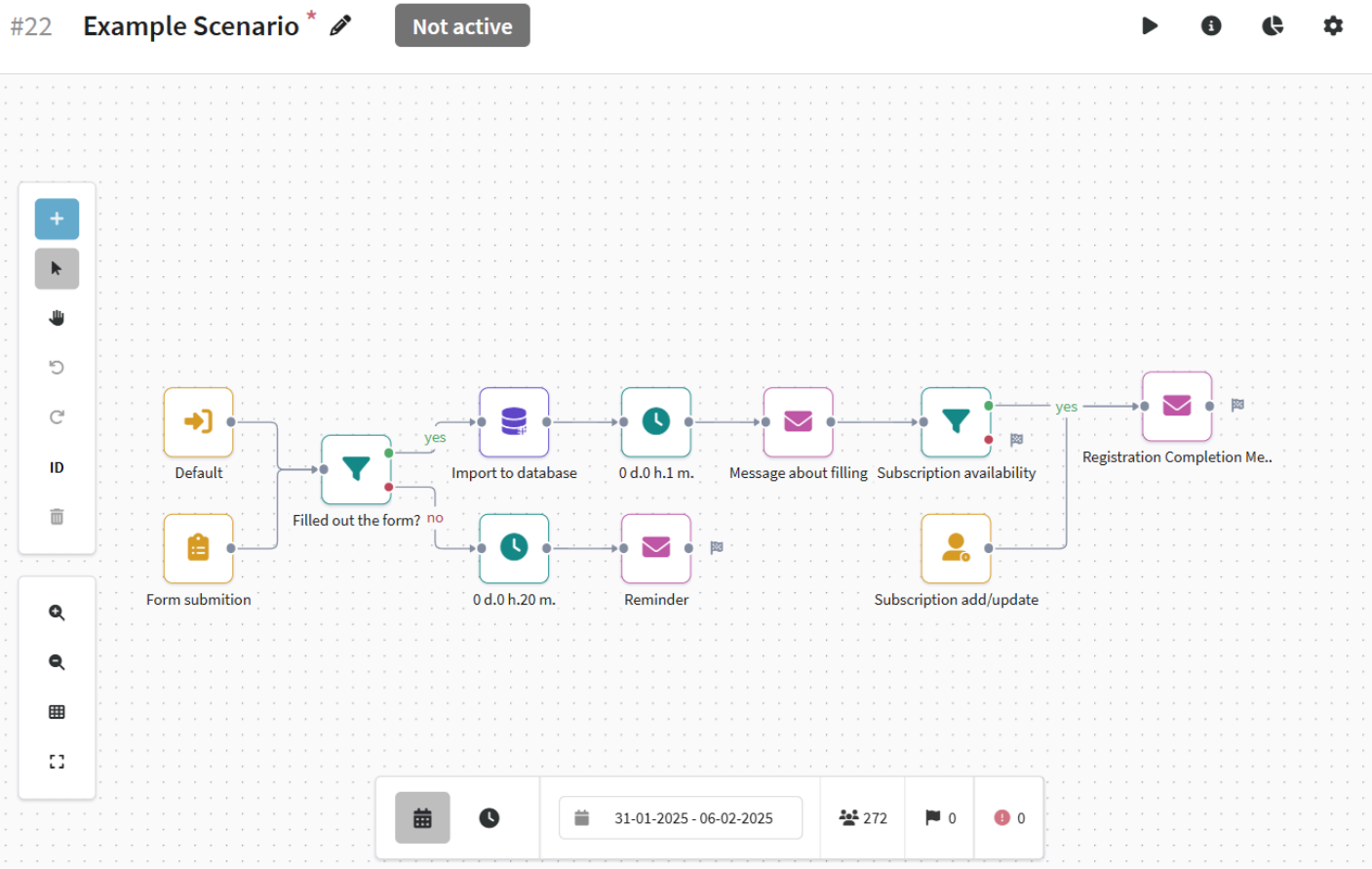
TIP

To find out on which element the profile left the scenario, open the profile card, go to the Interactions —> Scenarios tab and hover over the "Exit" stat. The item ID will be displayed:

Real-time statistics

Here you can get information about the profiles that are currently in the scenario.

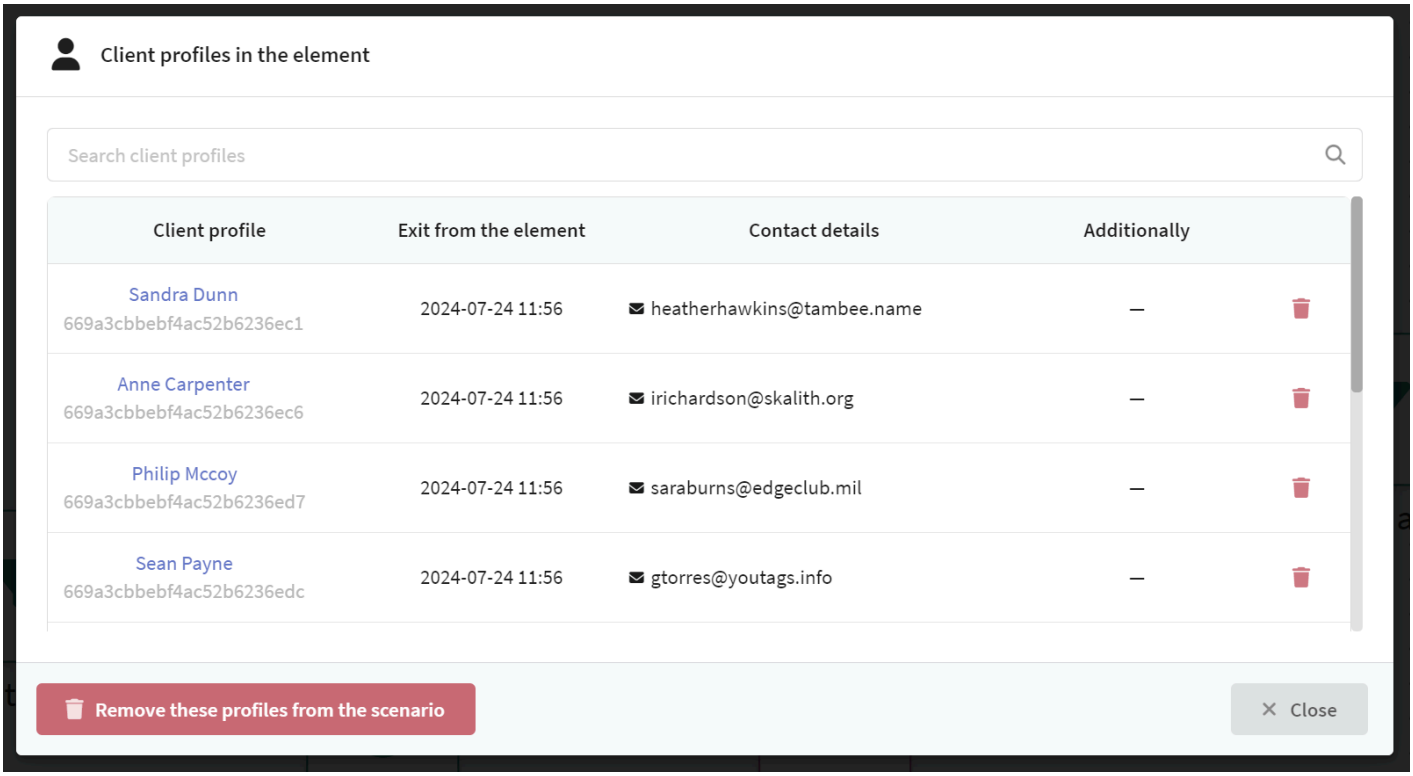
— the number of profiles that have entered the scenario but have not yet completed it.



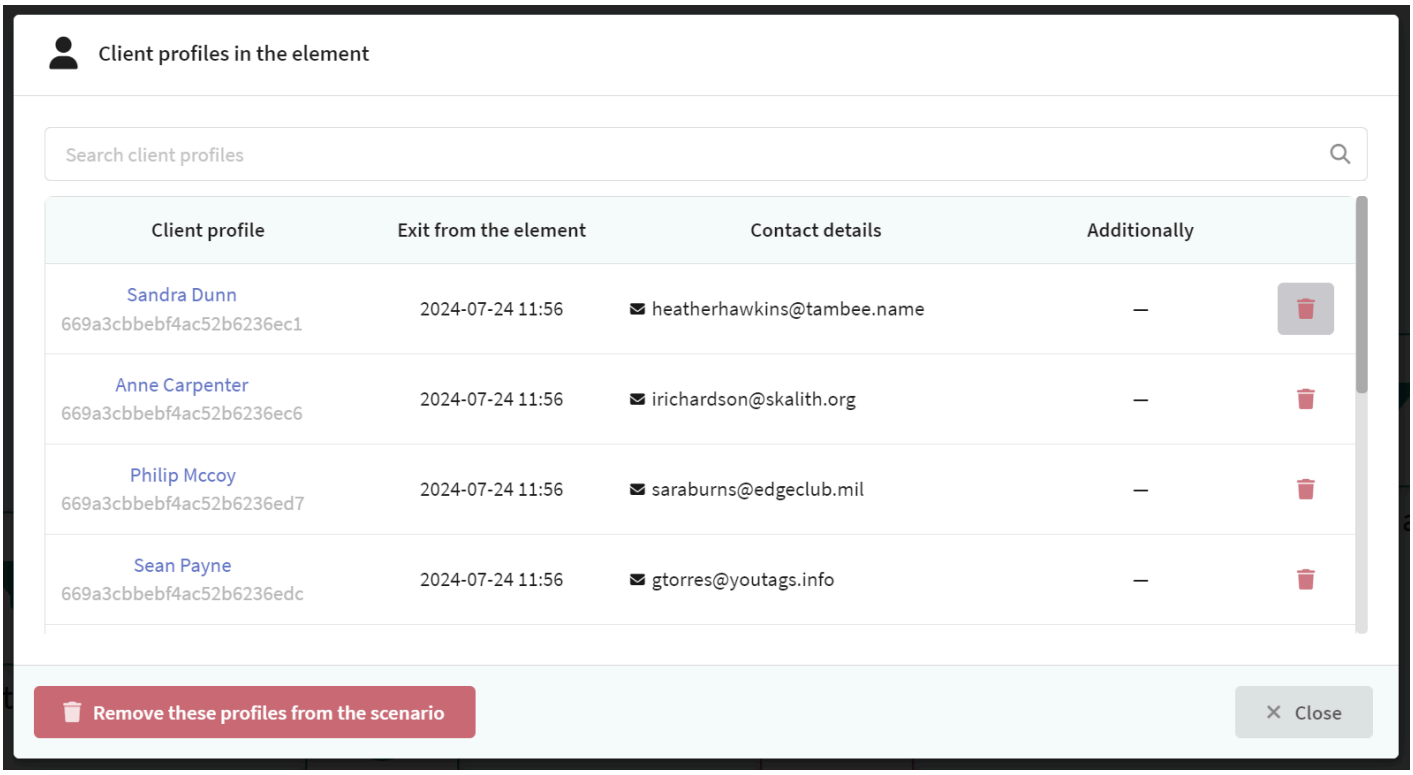
If there are profiles in the scenario at the moment, you will see a number above the element that contains them. The number shows the number of profiles in the element. There can be several such elements. In most cases, they are waiting elements (Pause, Best Delivery Time or Delayed Condition).

To find out information about the profiles in the element, click on this number. A modal window opens where you can:

- **go to the client's profile in the database;**
- **find out the exact time when the profile will leave the element;**
- **download a file with API content, if the corresponding parameter was passed in the API-request.**



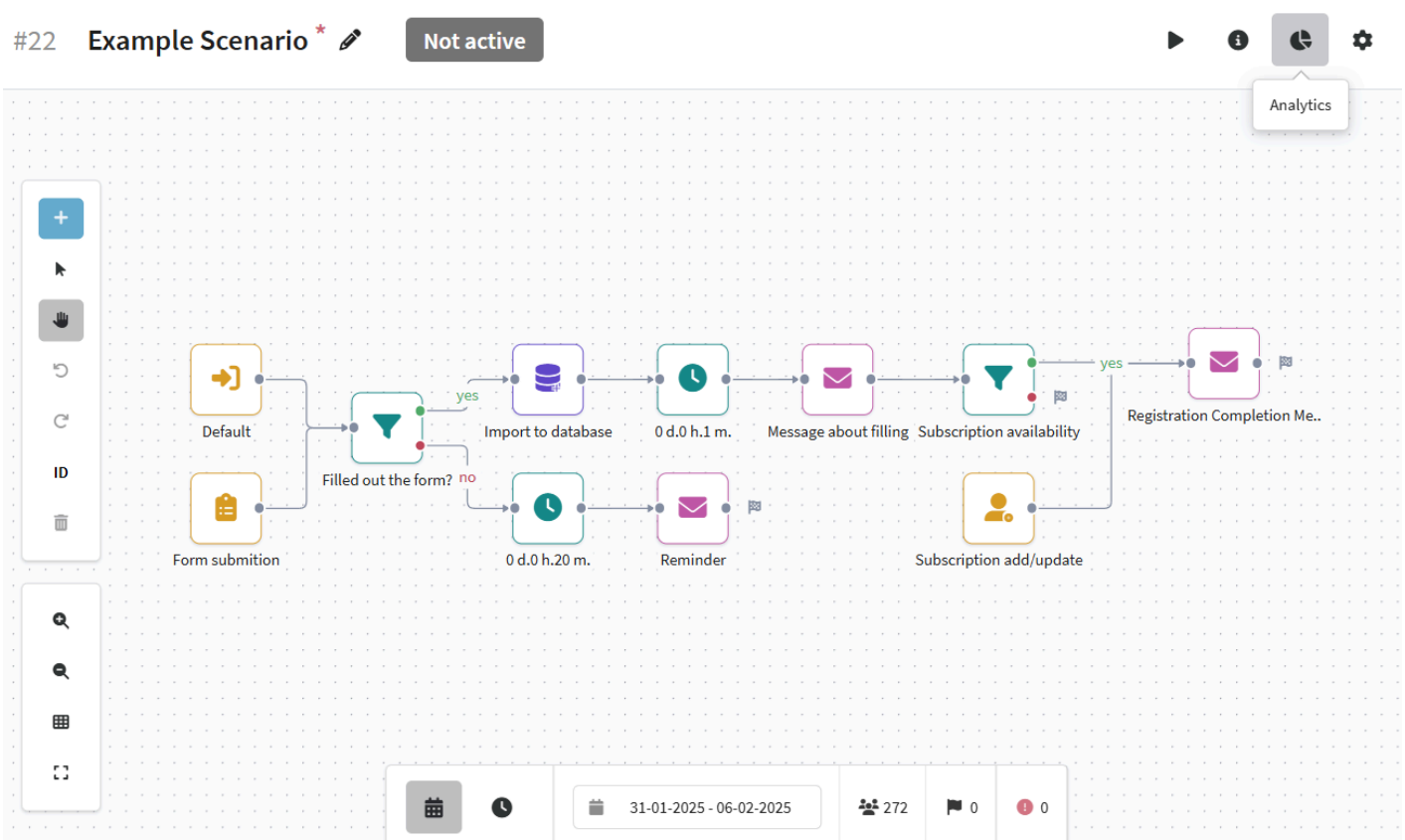
It is also available to delete profiles from the scenario. If you want to delete a specific profile from the scenario, click on the trash can icon next to its name:



To remove from the scenario all profiles that are currently in this element, click Remove these profiles from the scenario.

Mailing analytics in the scenario

To view a report of the mailings that the scenario runs, open [Analytics]:



Use groupings and filters to display the data you need. For example, if you want to see the statistics of the mailings involved in a scenario, you can use this setting:

Channel

Access groups
Main

All channels *

Save

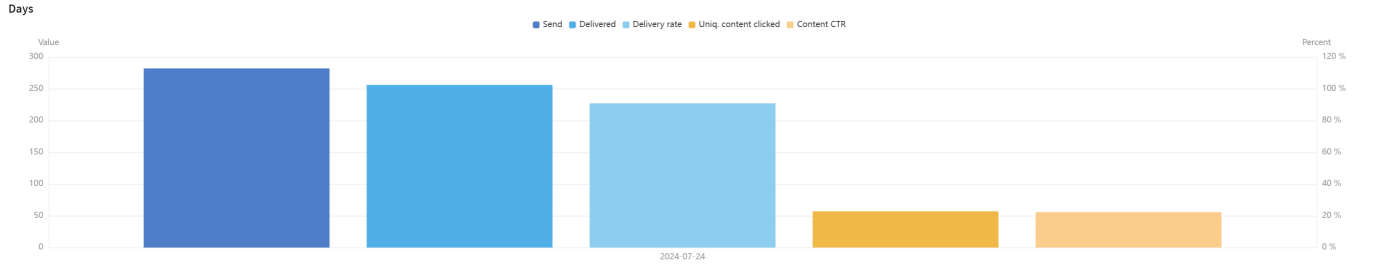
Report on: **All channels** For a period: Last 7 days

Group by Days in which to show:

Hide filter Hide diagram

Include workflow #4 Example scenario

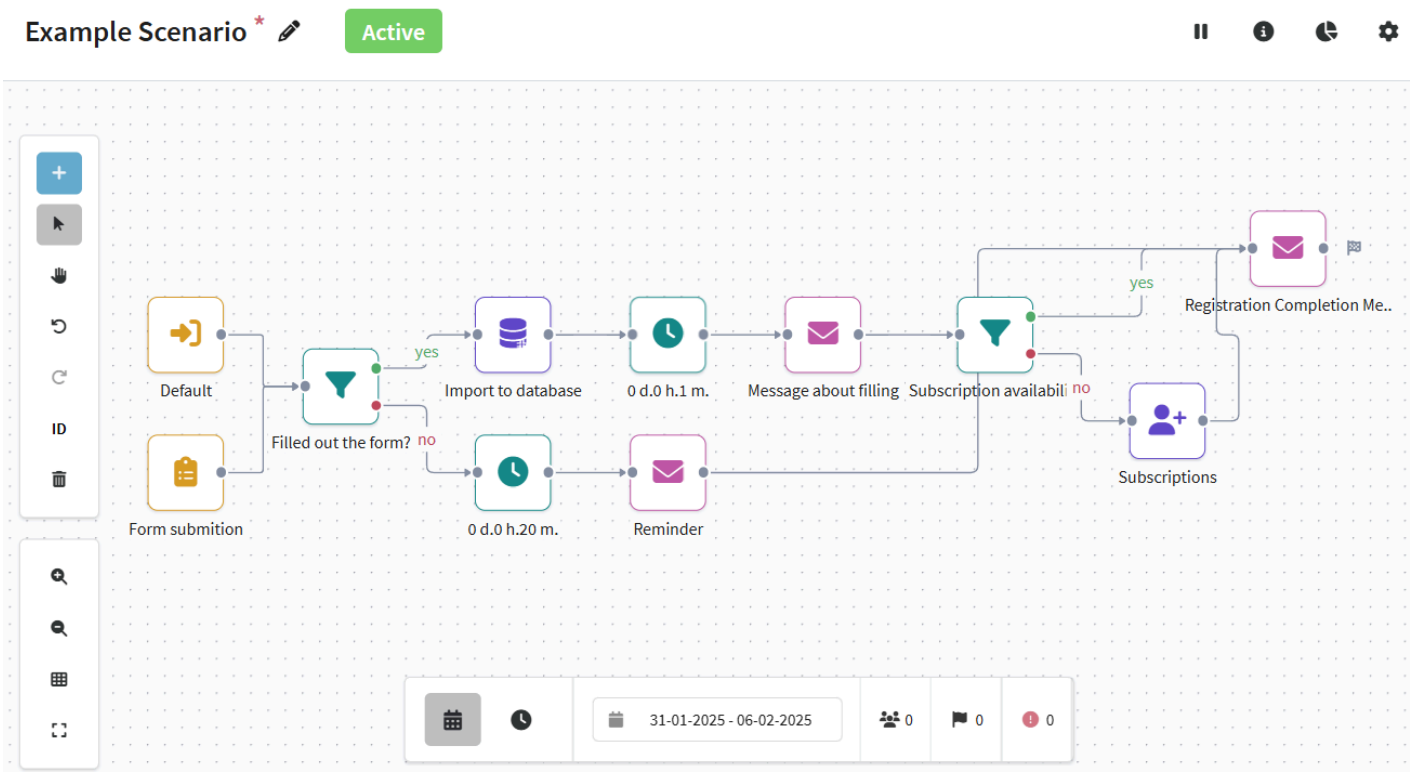
Days



Name	Send	Camp. errors	Deliv. errors	Suppressed	Policy offenced	Undelivered	Undelivery rate	Delivered	Delivery rate	Content clicked	Unsubscribed	Confirmed	Subscribed
Total	282	0	0	0	0	26	9,22 %	256	90,78 %	57	6	0	0
2024-07-24	282	0	0	0	0	26	9,22 %	256	90,78 %	57	6	0	0

Nodes of the scenario

A scenario consists of elements which are called nodes. Each node performs one action and then passes control to the next.

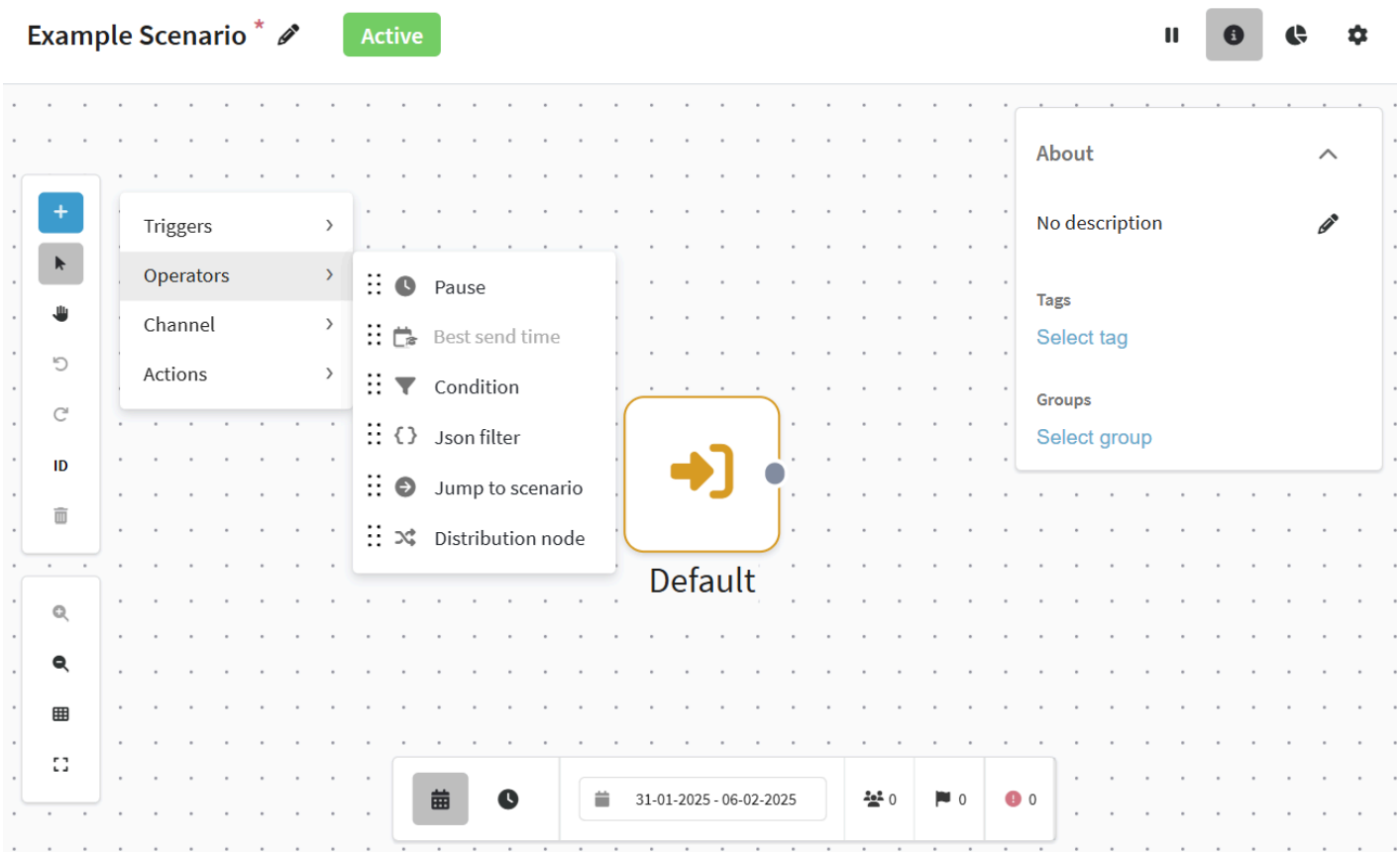


Any scenario must contain at least 2 nodes: a starting node and a connected node.

Node management

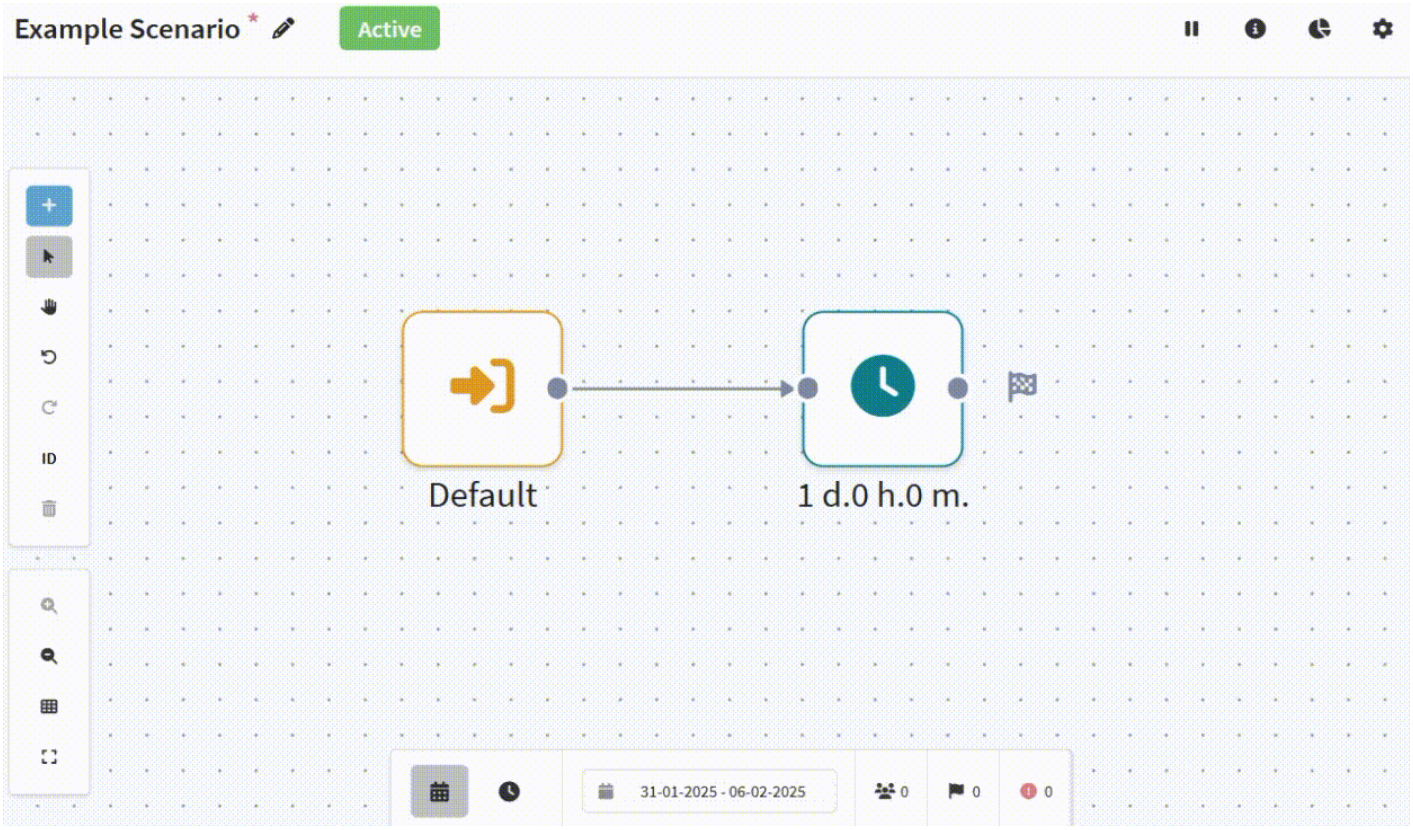
Adding a node

To add a node to the scenario, click the + Element button on the toolbar. In the opened list, click on the required element or drag it to the workspace holding the left mouse button.



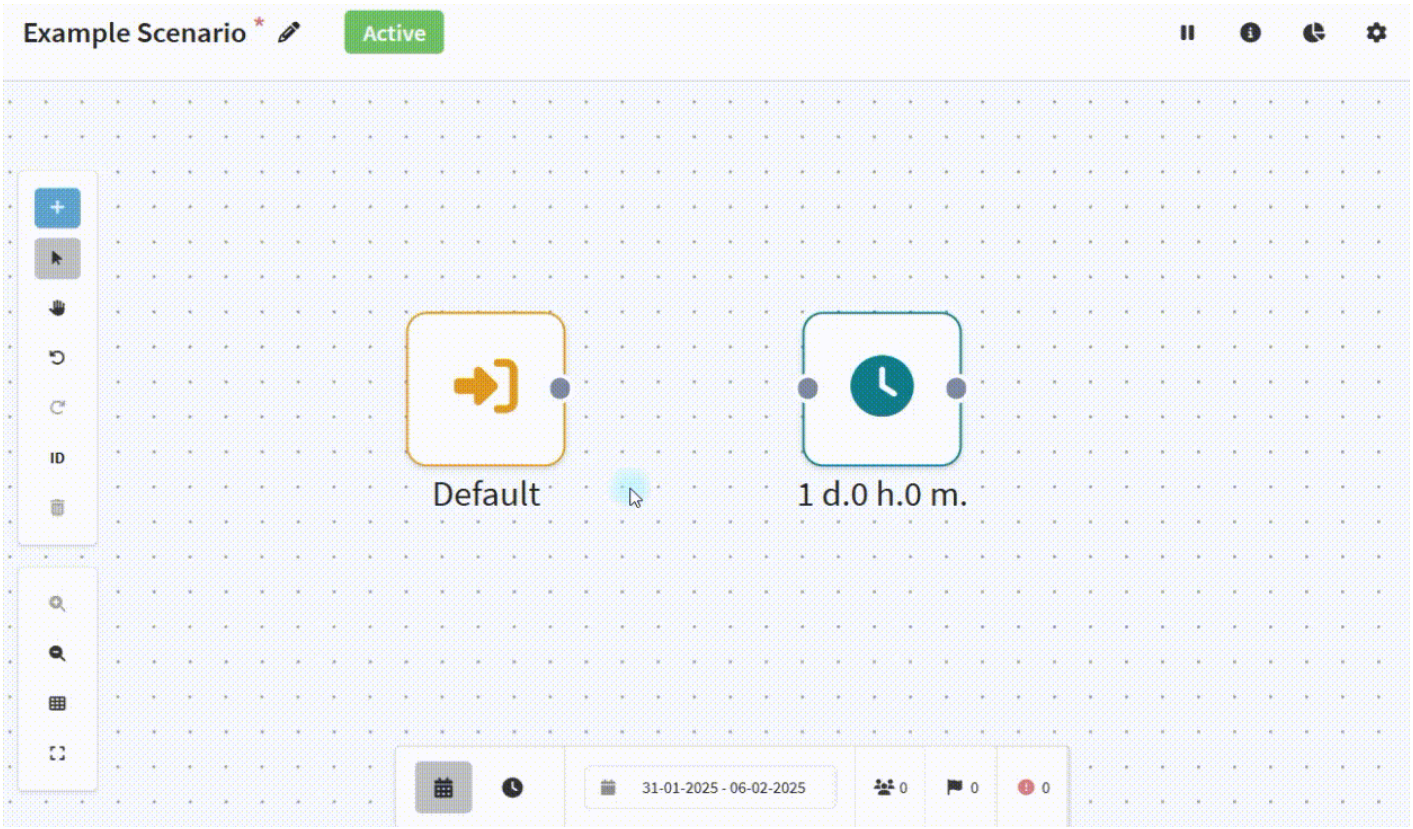
Setting up an node

Settings are set for each node of the scenario. To open the settings window, double-click on the desired node. To save the settings, click “Apply”.

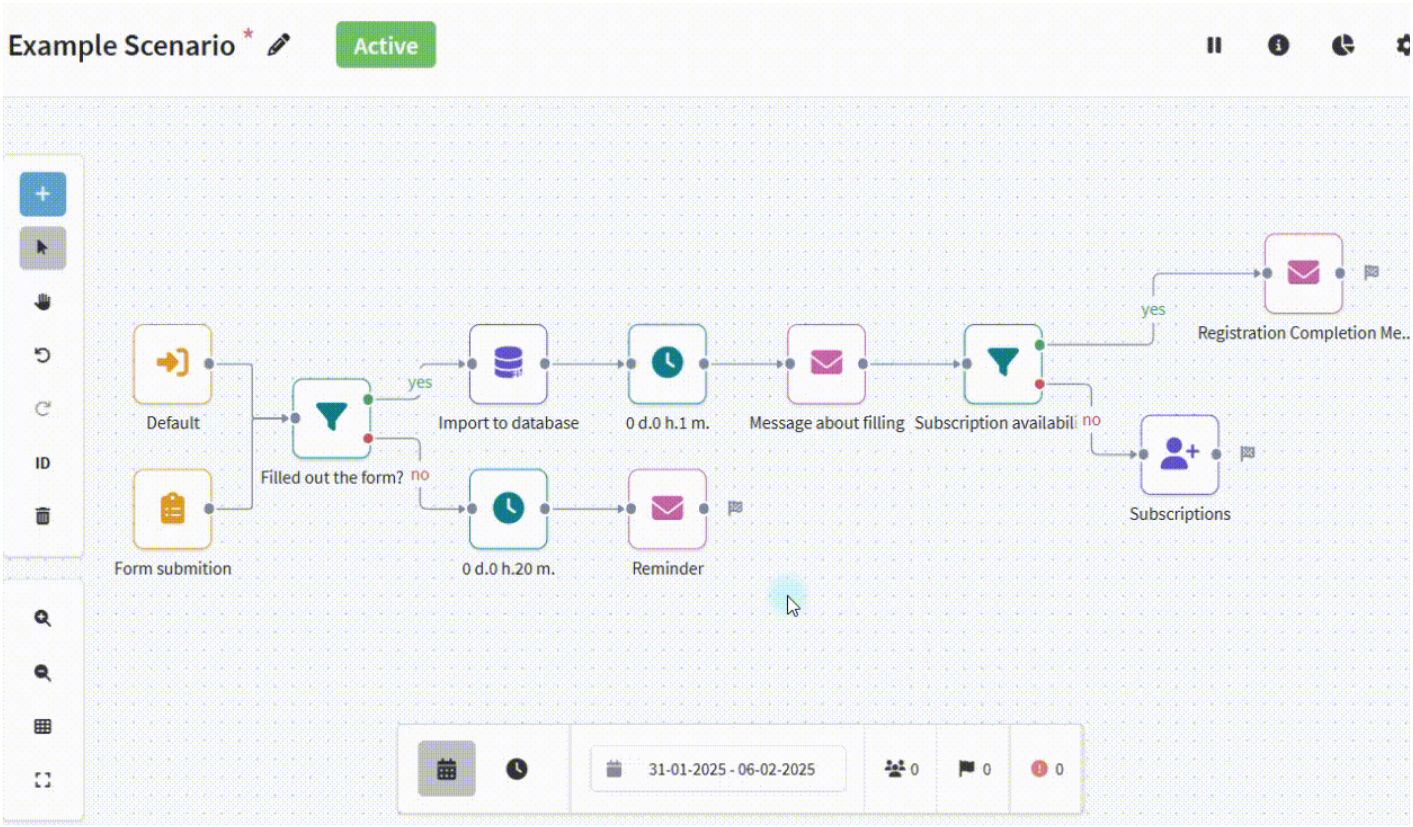


Adding a link between elements

Transition of a profile from one scenario node to another is defined using links. To link one node to another, draw a link between the elements:

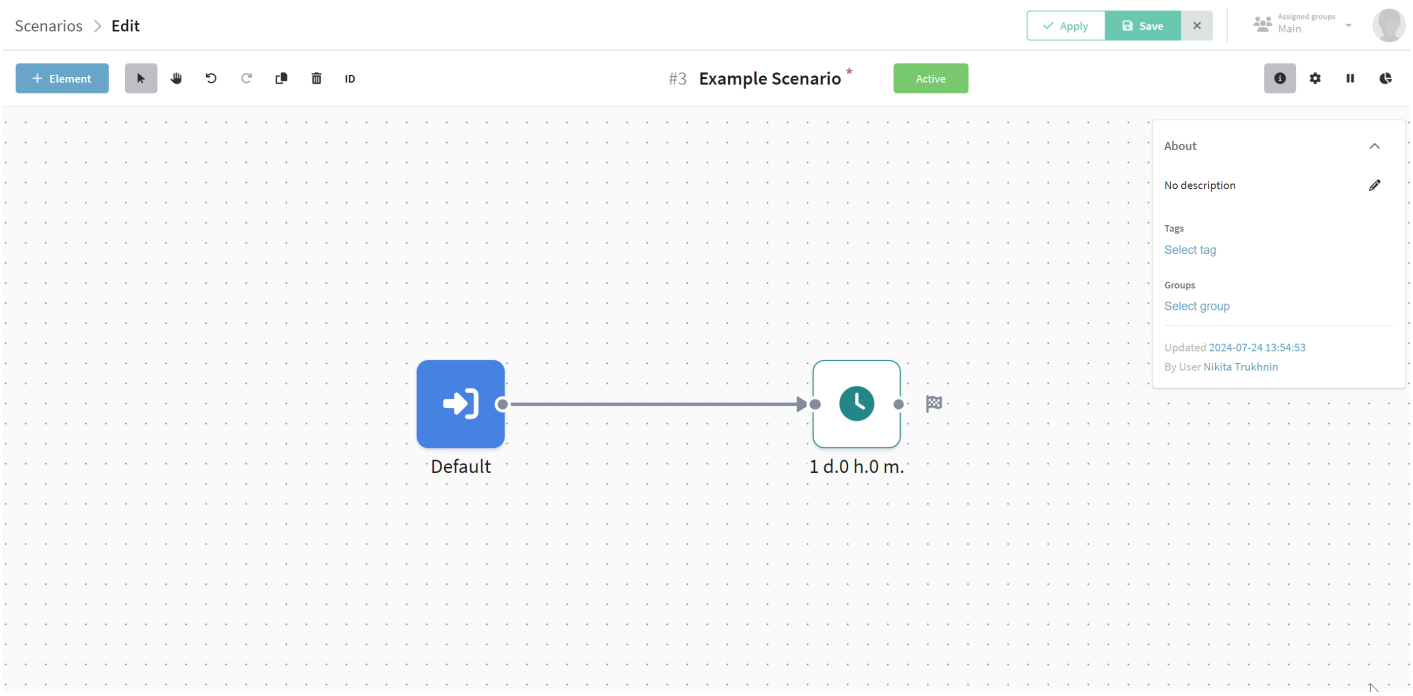


Elements can be linked not only sequentially. With the help of links it is possible to organize multiple passing of one and the same scenario section by a profile or switching to an alternative sequence of actions:

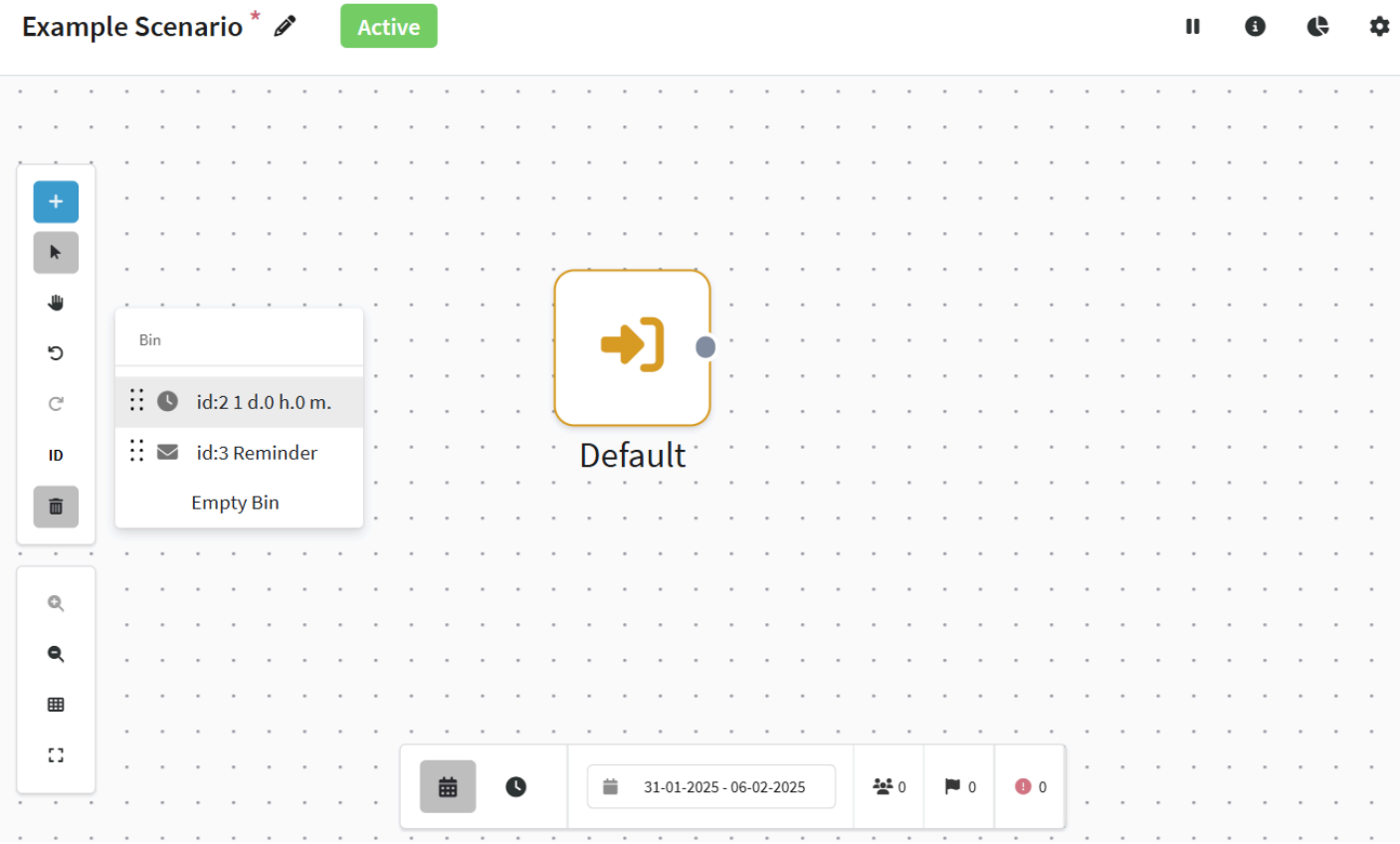


Deleting an node

To delete an node or a link between elements, select an object by clicking on it, and then click the trash can icon or `Delete/Backspace`. It is possible to delete multiple elements, including links between them.



If you have deleted an node from the scenario, you can restore it. To do this, click the Bin button in the lower left corner. A list with all deleted elements will appear. Select the node you want to return to the scenario. After restoring the node, it will retain its previous settings.



You can also irrevocably remove all elements from the bin by clicking the Empty bin button.

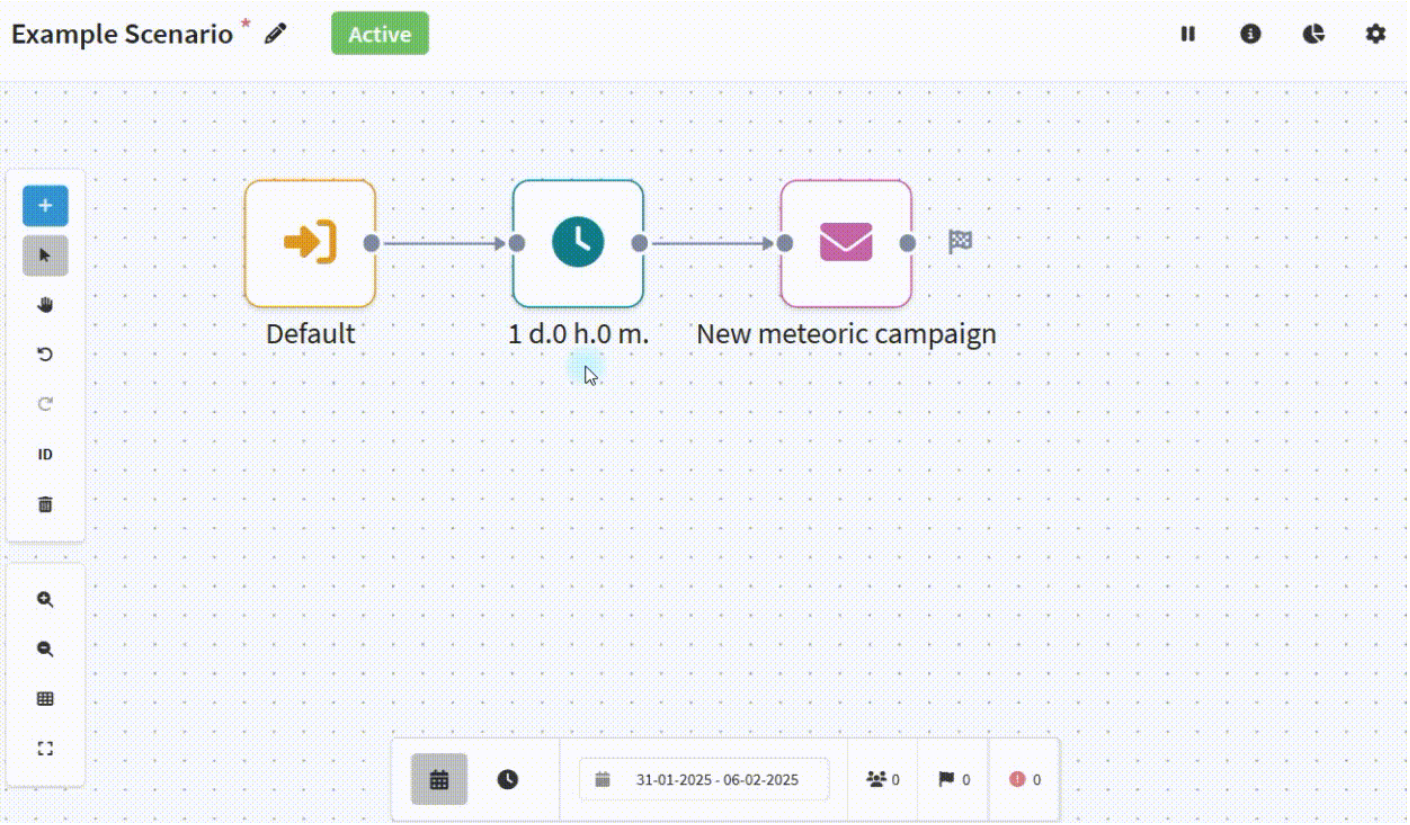
CAUTION

If you remove a node that has profiles, those profiles will be taken out of the scenario. If there are no profiles in the node, they will keep following the updated scenario.

Cloning of a node

To clone a node, select it and left-click

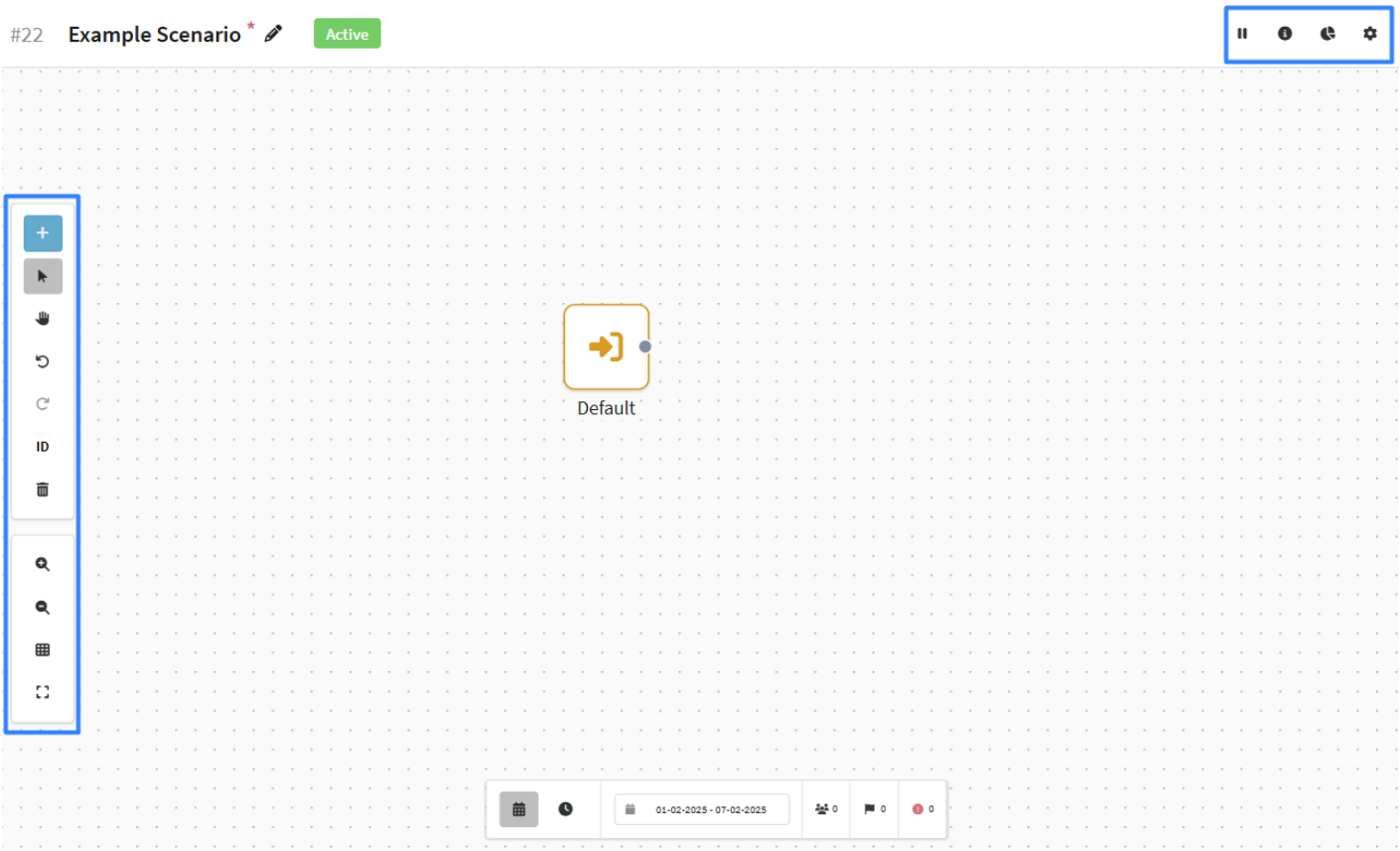
or use the `Ctrl + C` key combination. You can also clone multiple elements, including the links between them.



TIP

To select several elements of the scenario, activate the Select tool. Then hold down the left mouse button and select the area with the desired objects or hold down the `Crtl` key and successively select the desired objects with the left mouse button.

Toolbar



Tools for managing elements

— activate the “Select” tool. It is used to control elements of the scenario: move, connect, select, etc.

— activate the “Hand” tool (Space + drag). It is used to move around the scenario workspace.

— undo the last action (Ctrl + Z).

— revert the last canceled action (Ctrl + Shift + Z).

— open the “Bin”. The “Bin” contains all items deleted from the scenario.

ID

— display the ID of each node instead of the name.

Tools for setting up the scenario

— show scenario information (description, tags and groups for the scenario).

— show scenario settings (mode of profile re-entry into the scenario, etc.).

— to activate the scenario.

— to deactivate the scenario.

— view scenario analytics in a channel report.

Tools for workspace management

— zoom into the workspace (Ctrl +).

— downscale the workspace (Ctrl -).

— apply alignment. The elements are aligned relative to each other within the workspace.

— adjust the view. The scale changes to the optimal scale, allowing you to see all elements within the workspace.

List of elements

All elements of the scenario are divided into four categories:

- **Triggers - elements that define the way of capturing a profile into a scenario.**
- **Operators - elements affecting the profile movement through the scenario.**
- **Channels - elements, by means of which a message is sent to the profile.**
- **Actions - elements that trigger a certain action in the platform at the moment of profile entry.**

Triggers

A trigger is a node that starts a scenario. You can configure one or several triggers. The maximum number of triggers in one scenario is 10. One trigger type can be used multiple times in one scenario.

::: In this article, triggers are used in scenarios to start profiles within them. Do not confuse them with triggers in mailings. :::

Default profile capture

	This node is a built-in feature of the scenario and can't be removed. With the "Default" node, profiles can enter the scenario from different points in the platform.

Read More

This node captures profiles after following actions:

- **Sending a profile to the scenario via API. When the scenario is saved, the item window will display a URL that can be used to run it. However, it is recommended to use a newer version of the API. Go to the documentation for more information.**
- **Adding a profile to a scenario manually.**
- **Sending a segment to a scenario through quick actions panel.**
- **Jumping through the node from another scenario.**

Action execution

	Activated when the subscriber performs the desired action in the message.

Read More

From the drop-down list, select action in one of the submission channels, and specify the template in which you want to track this action:

Action	Description
Any link clicked in email	**The recipient clicked any link in the selected email template.**
Specific link clicked in email	**The recipient clicked a specific link in the selected email template.**
Email opened	**The tracking pixel recorded the opening of the selected email template.**
Email read	**The tracking pixel recorded that the selected email template was read.**
Email confirmed	**The recipient clicked the subscription confirmation link in the selected email template.**
Any link clicked in SMS	**The platform recorded a click on any link in the selected SMS template.**
Specific link clicked in SMS	**The platform recorded the first click on a specific link in the selected SMS template.**
Push opened	**The platform recorded the opening of the selected push notification. Push notifications are usually opened automatically if allowed by the browser or mobile OS.**
Any link clicked in push	**The recipient clicked any link in the selected push notification template.**
Pop-up opened	**The platform recorded the display of the selected pop-up template to the user.**
Click on pop-up	**The user clicked the selected pop-up template.**
Pop-up closed	**The platform recorded a click on the close button of the selected pop-up template.**
Pop-up custom event	**The platform recorded a custom event from the pop-up.**
Pop-up filling	**The platform recorded the user filling in one or more fields in the selected pop-up template.**
Notify click	**The platform recorded a click on any link in the selected Notify template. The trigger fires on every click and re-enters the profile into the flow unless otherwise configured.**
Notify unique click	**The platform recorded the first click on any link in the selected Notify template.**
Notify specific link click	**The platform recorded the first click on a specific link in the selected Notify template.**
Notify read	**The platform recorded the reading of the selected Notify template.**
MAX bot read	**The platform registered a read event for the MAX bot template.**
WhatsApp*/Viber™/Telegram bot/MAX bot click	**The platform recorded a click on any link in the selected WhatsApp*/Viber™/Telegram bot template.**
WhatsApp*/Viber™/Telegram bot/MAX bot unique click	**The platform recorded the first click on any link in the selected WhatsApp*/Viber™/Telegram bot template.**
WhatsApp*/Viber™/Telegram bot/MAX bot specific link click	**The platform recorded the first click on a specific link in the selected WhatsApp*/Viber™/Telegram bot template.**
WhatsApp*/Viber™/Telegram bot subscription confirmation	**The platform recorded a click on the subscription confirmation link in the selected WhatsApp*/Viber™/Telegram bot template.**

Form update

	The trigger captures the profiles whose data will be updated as a result of filling out the form into the scenario. It is triggered only when updating existing profiles.

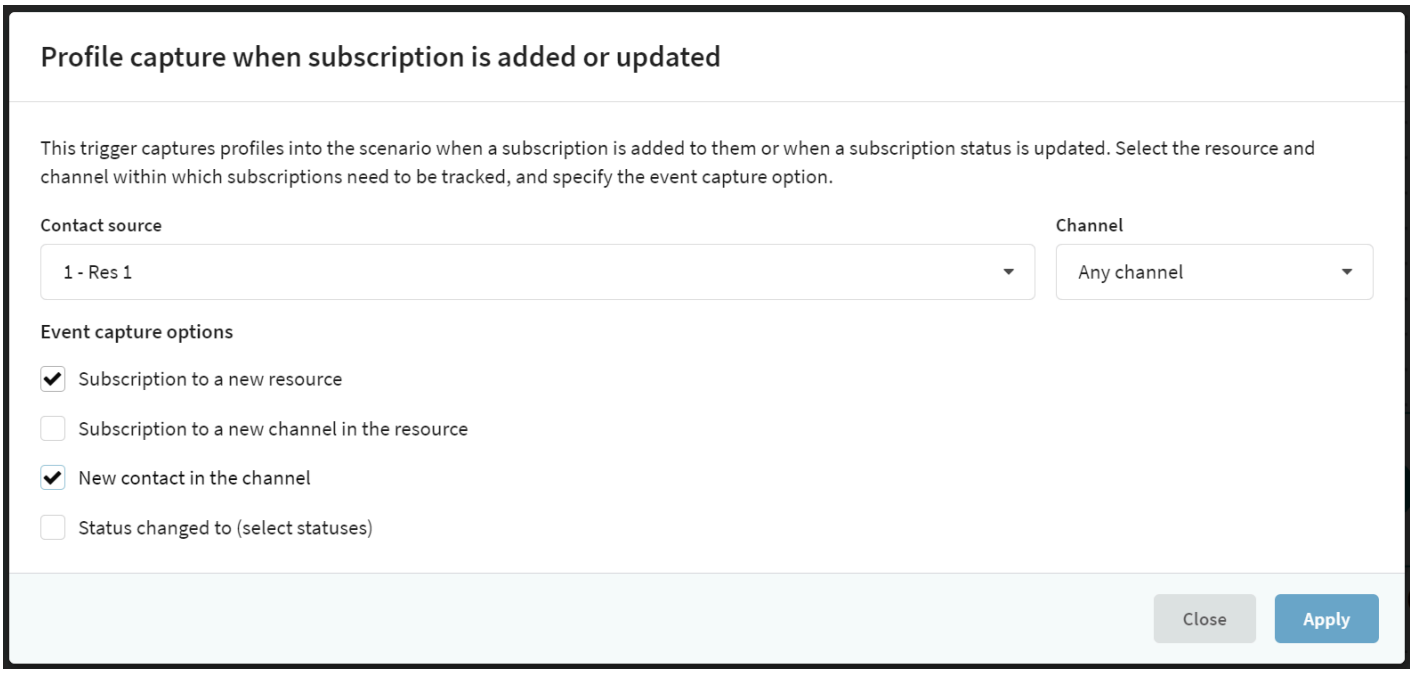
Read more

In the drop-down box, specify the form to which the trigger will be bound:

Subscription add/update

	Trigger is activated when a new subscription is added to a profile or the status of an existing subscription is changed. For the trigger, you can separately specify **Resource and **Channel** for which changes will be tracked. Otherwise, the mailing will expect changes for all channels of the selected resource, for all resources for the selected channel, or for any subscriptions of the profile.**

Read more



- **Subscription to a new resource — condition will work only for the first subscription to the resource specified in the mailing. Adding other subscriptions to the same resource will not activate the scenario.**
- **Subscription to a new channel in the resource — subscription to a new (for the profile) channel in the resource was added. In other words, there was a subscription to email newsletters, now there is also a subscription to SMS or Push.**
- **New contact in the channel — activates the trigger each time a new subscription appears. The first subscription to a channel is also considered a "new contact in the channel".**
- **Status changed to (select status) — will run when the subscription status changes to those selected from the drop-down menu.**
- **The first three items are selected — the trigger will fire with each new subscription.**
- **All the boxes are unchecked — nothing will happen.**

NOTE

Subscriptions that are added with a status other than "Subscribed" will not be activated by the trigger.

Scheduled segment

	A group of profiles that meet certain conditions is put into the scenario on a set schedule.

Read more

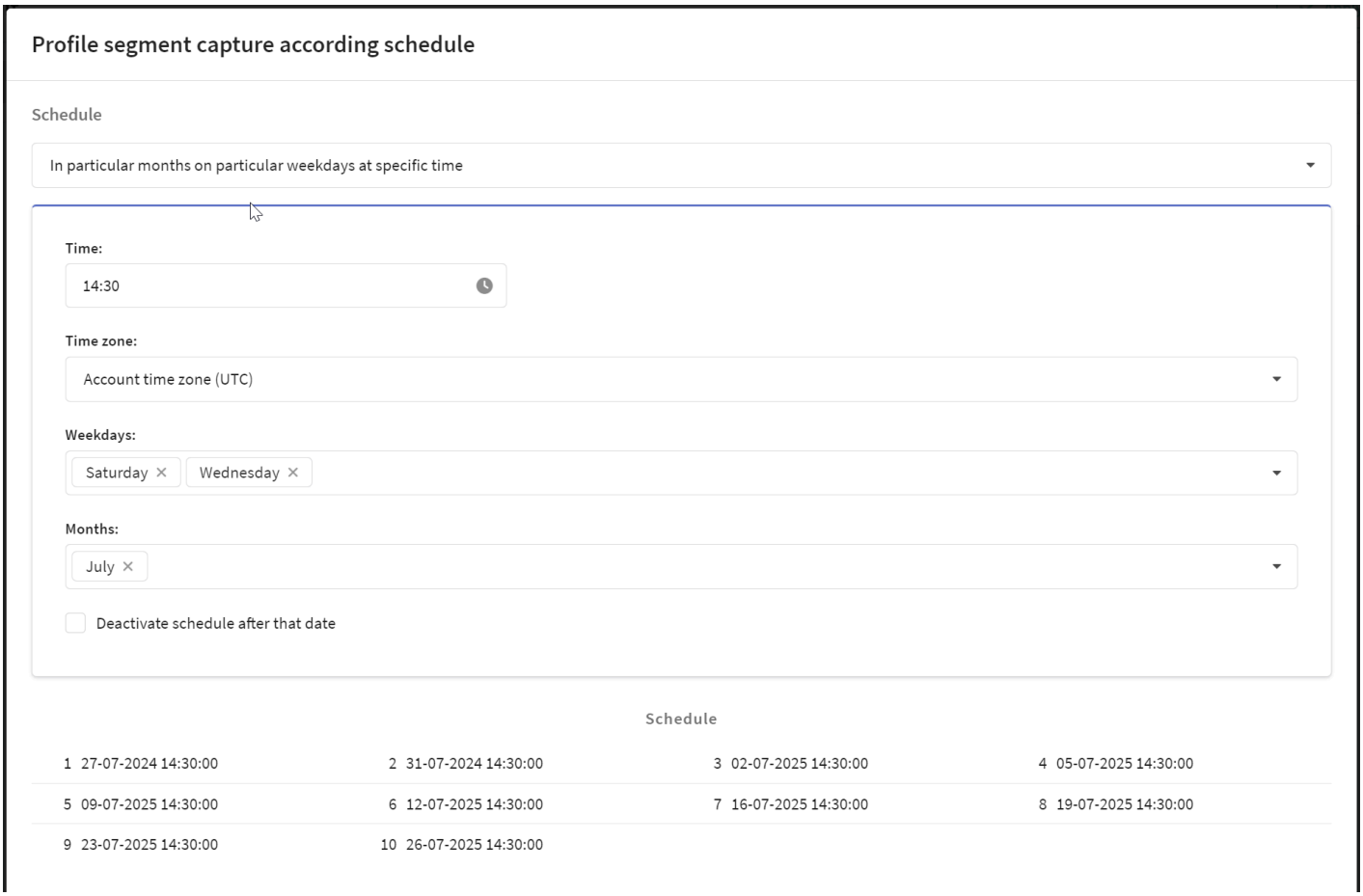
First, you need to configure a capture schedule. Select a schedule type and configure the time when the scenario will run for the segment.

Additionally, you can Deactivate the schedule after a certain date or change the time zone for the start time.

Once changes have been made, a table with the next runs of the scenario will be displayed below the schedule.

TIP

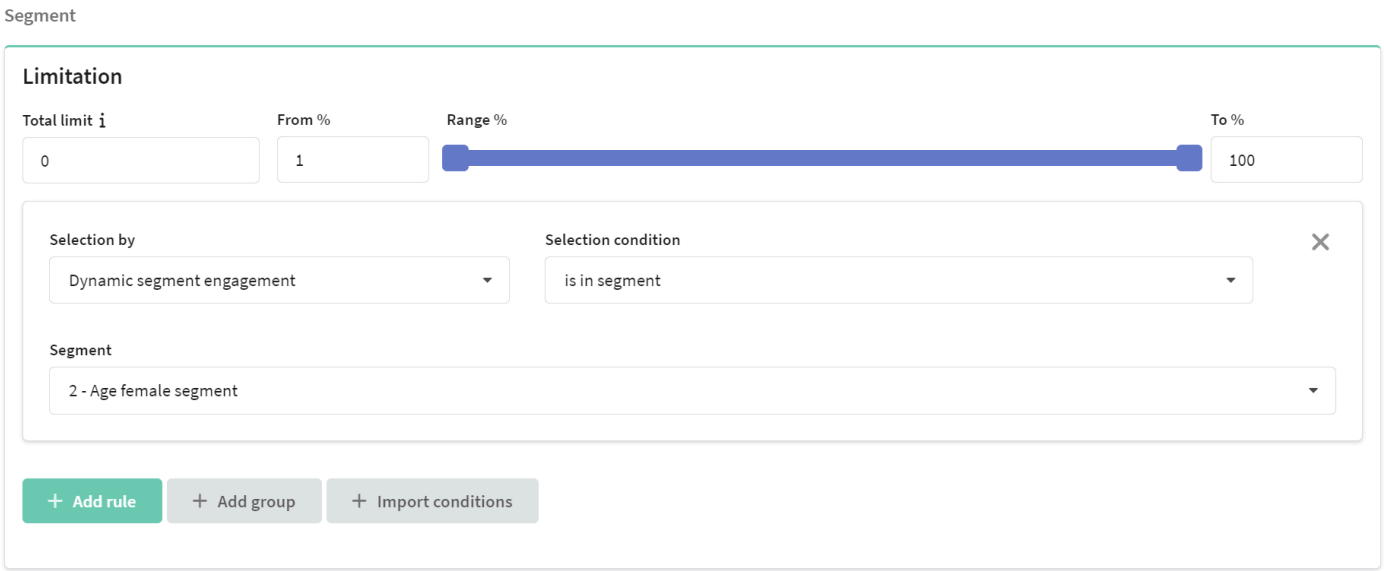
Setting up a schedule is described in detail in separate article.



Next, configure segmentation profiles. Build a segmentation query against the database by adding rules or groups of rules. Available segmentation by:

- **basic profile data**
- **activity history**
- **subscriptions**
- **and other parameters.**

Segmentation is described in more detail in relevant documentation section. You can optionally set the limit of the resulting sample as a percentage.



TIP

Scheduled scenario is launched in the same way as a regular mailing. The customer segment is calculated dynamically at each launch.

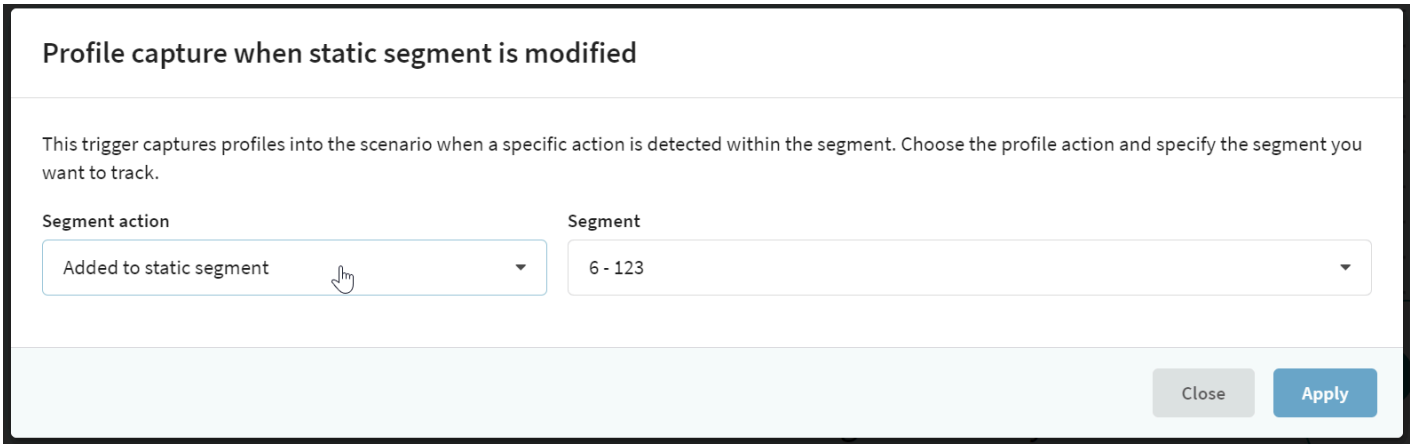
With the help of such scenarios, you can organize a sequence of communications for customers who are about to expire: the scenario by segment (in the example below) will select every day only those customers who have 1 month left before the expiration date.

Static segment modify

	Trigger is fired when there is a change in [static segment**](/user-guide/segmentation/9407279): a profile can both enter and exit the specified static segment.**

Read more

In the drop-down box, select the action (entering or leaving a static segment) and the segment you want to monitor:



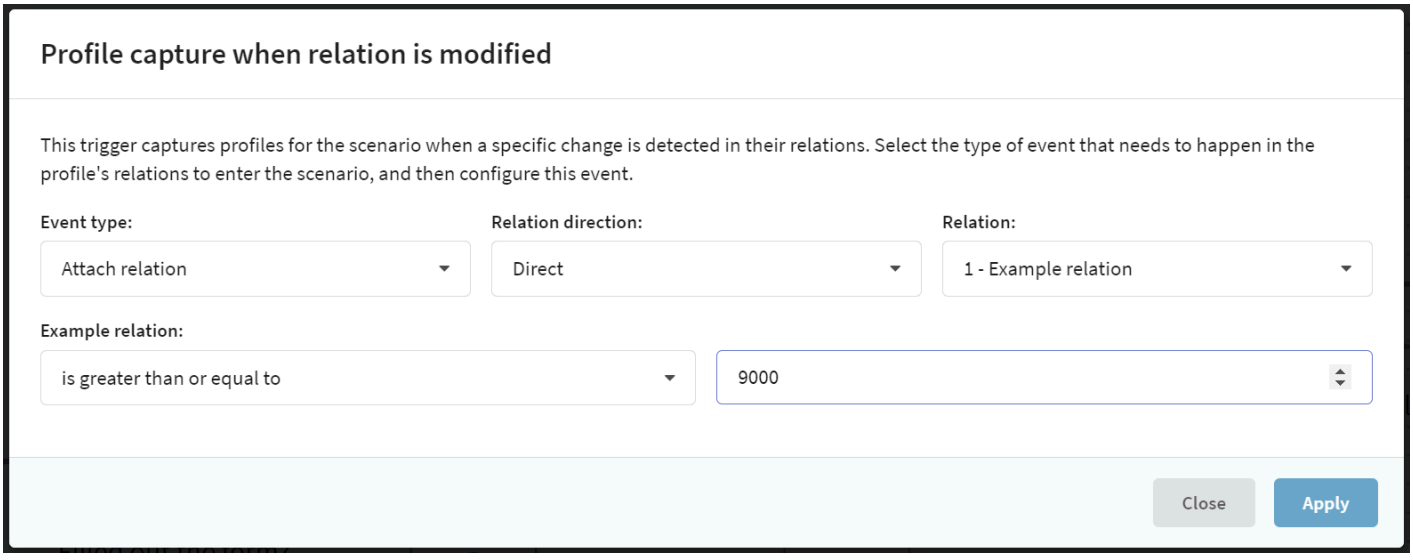
Relation modify

	Trigger is started when [link**](/user-guide/profiles-and-databases/20480122) between profiles changes: both forward and backward. The occurrence, change or deletion of a link can be taken into account, as well as the link property reaching the value set in the trigger.**`Delete/Backspace`

Read more

To configure a trigger, specify:

- **relation,**
- **the type of event attached to it,**
- **relation direction,**
- **link property,**
- **property value.**



API import

	The trigger is activated when a new profile is added to the database via [API query**](/developer-guide/profiles/2195477). The profile must add to the database selected for the scenario.**

Read more

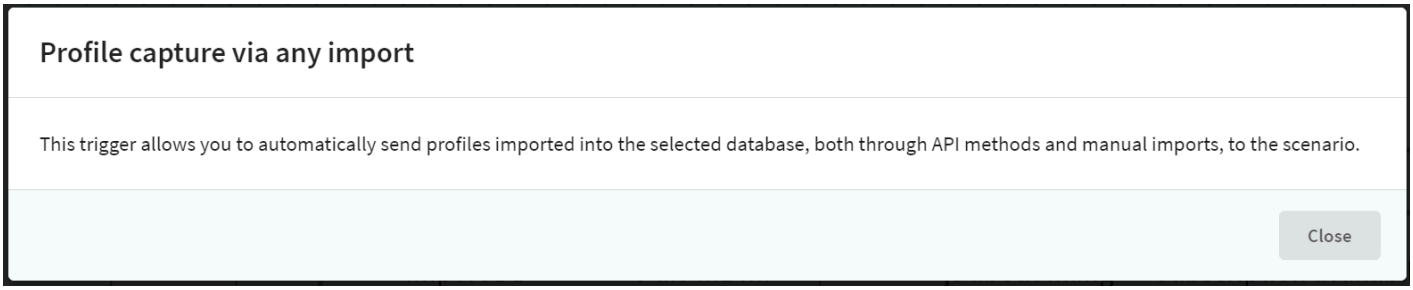
CAUTION

If you select the Any Import method, this trigger becomes unavailable.

Any import

	The trigger is activated when a new profile is added to the database in any way: via [API request**](/developer-guide/profiles/2195477) or [manually](/user-guide/profiles-and-databases/9405665#Subscriberprofile-Scenarios), from the profile card.**

Read more



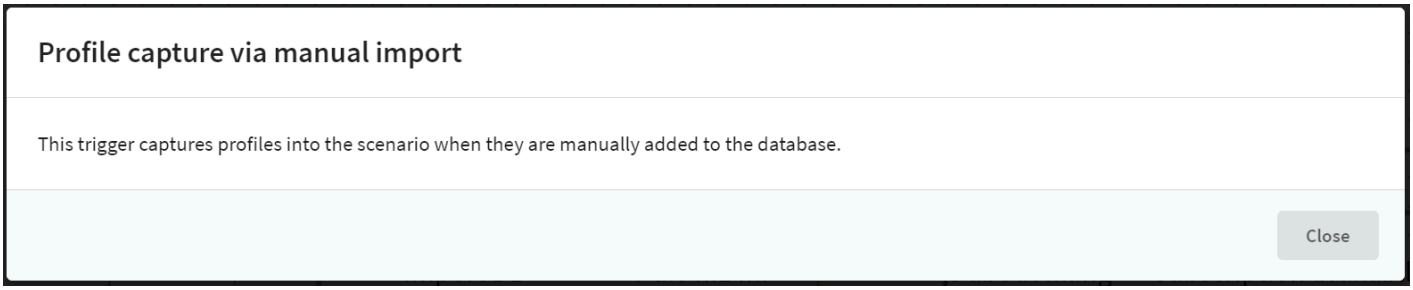
CAUTION

If you select the **Any Import ** method, the other import triggers become unavailable.

Manual import

	The trigger is activated when a new customer profile is [manually added**](/user-guide/profiles-and-databases/9405665) to the database.**

Read more



CAUTION

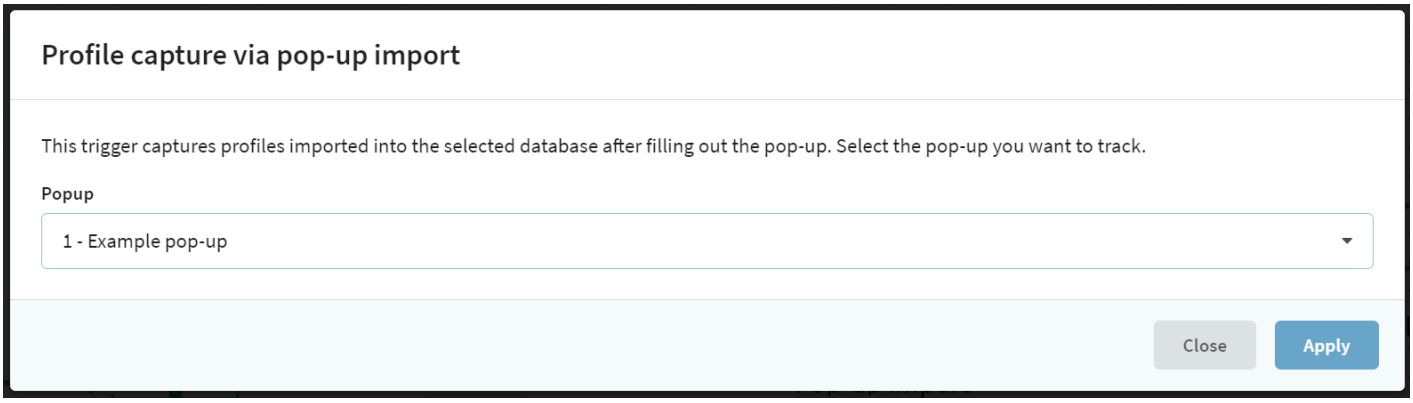
If you select the Any Import method, this trigger becomes unavailable.

Pop-up import

	The trigger will work if a new profile is created in the database as a result of a user filling out and submitting a form in a pop-up window.

Read more

Select the pop-up you need to track the completion of. Pop-up window can be created in Weblayer —> Popups, and you can customize importing profiles to the database in the pop-up editor in the Actions.



CAUTION

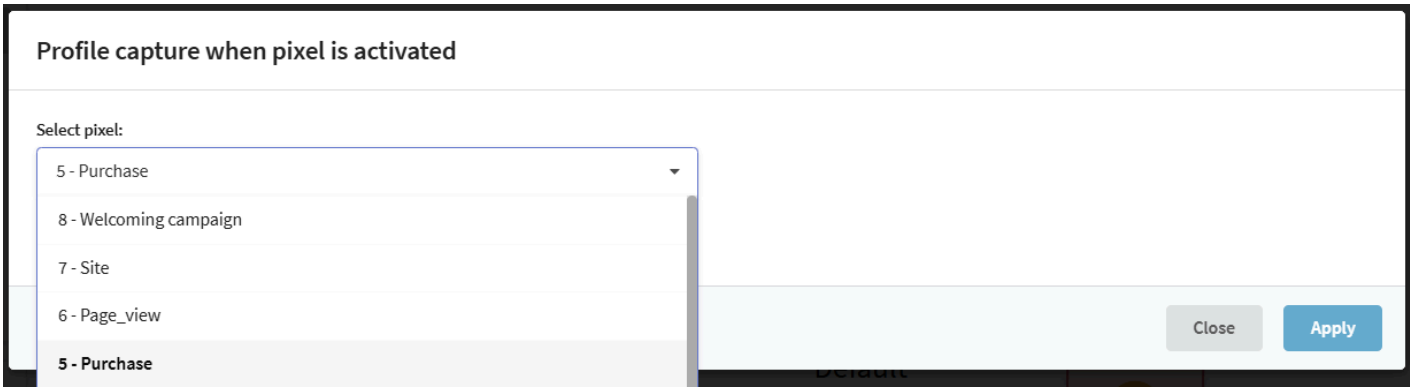
If you select the Any Import method, this trigger becomes unavailable.

Pixel activation

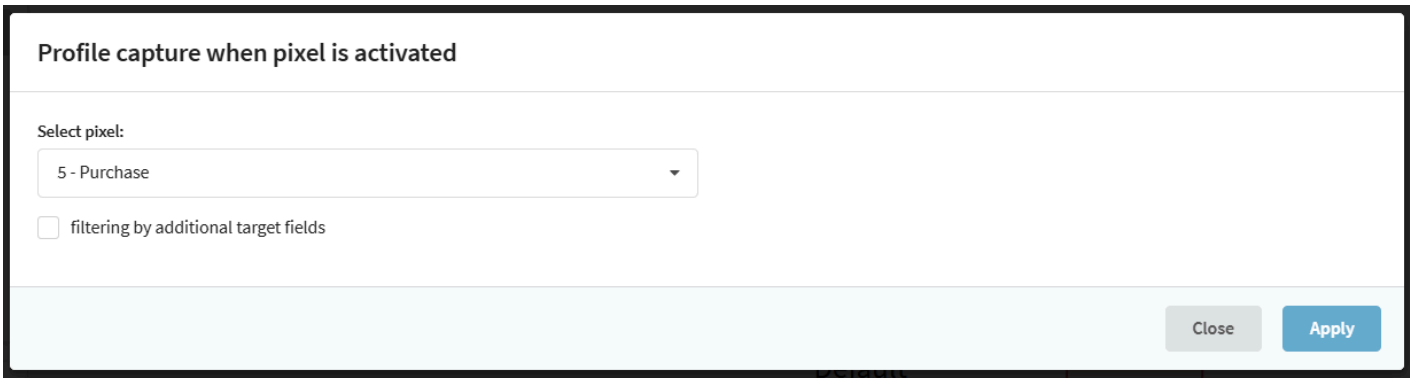
	The profile enters the scenario when the pixel is activated.

Read more

Select the required pixel in the drop-down box:

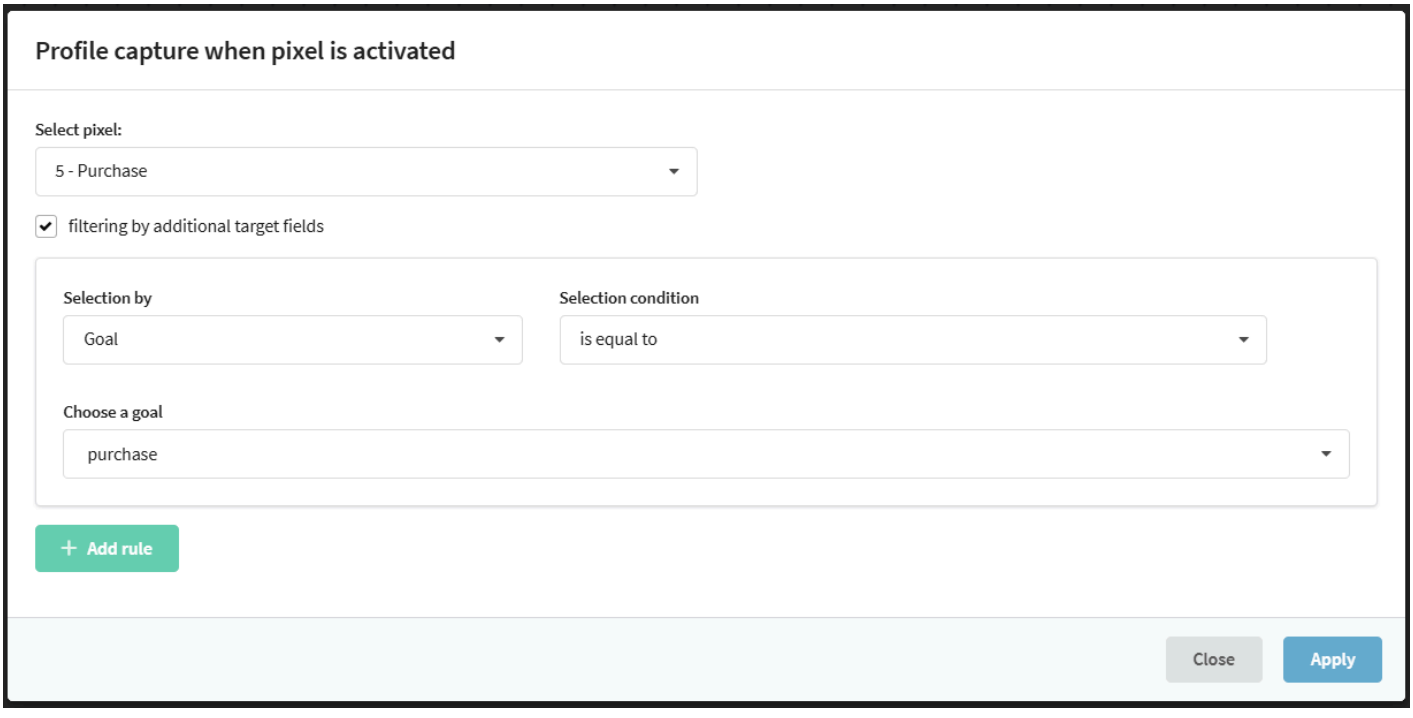


This trigger can keep track of the default pixel triggering:



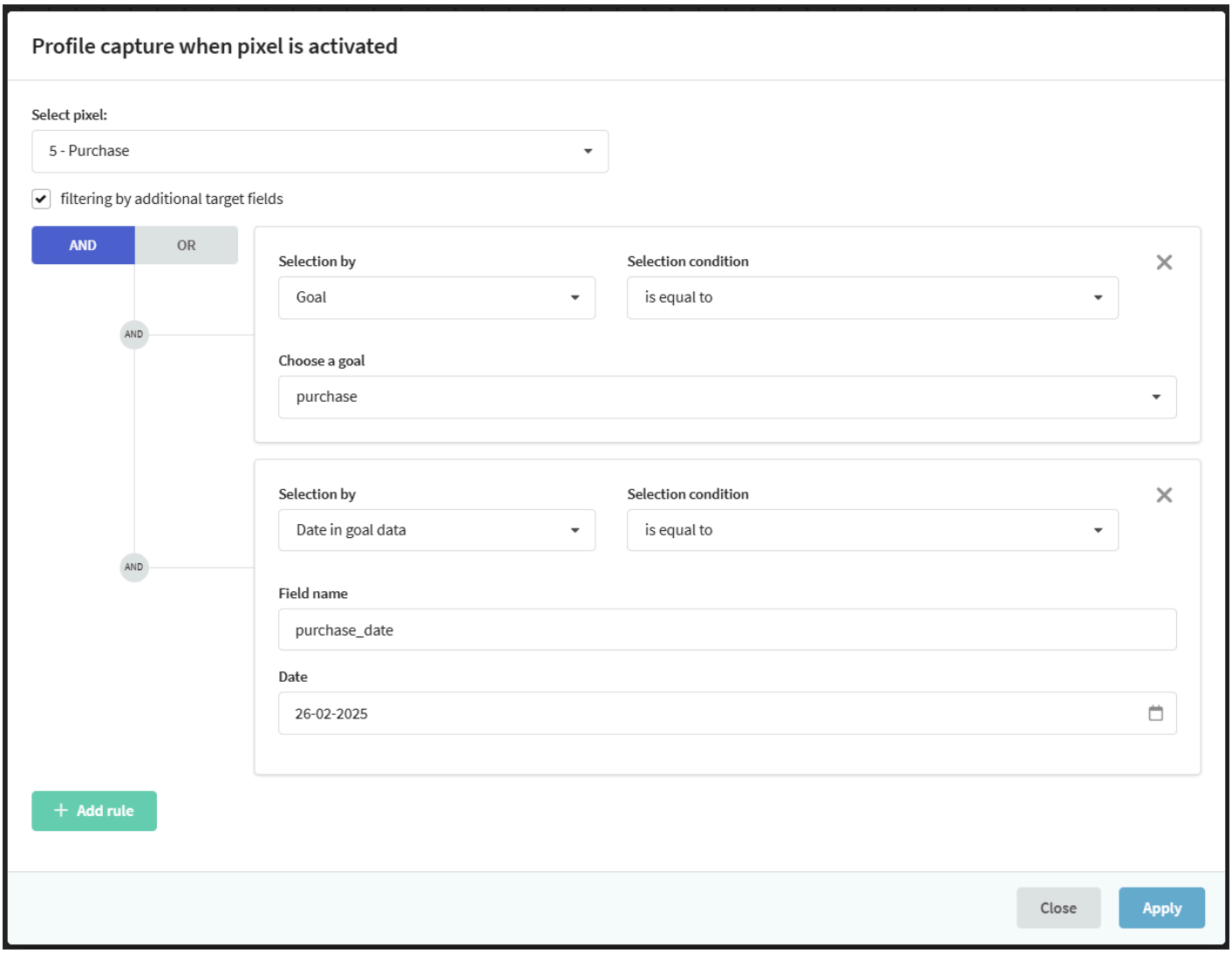
If you are required to track only certain events, you can set up a trigger to register a specific target. To do this, check the Filteing by additional target fields box and configure the required selection parameters.

With the help of such a trigger, the pixel will record only those events that are related to a specific target. For example you can set up filtering so that only profile registrations, purchases or other key actions are tracked.



For more precise customization, you can use a combination of several selection rules. This is required by checking the Filtering by additional target fields box, clicking Add rule and configuring two or more selection rules.

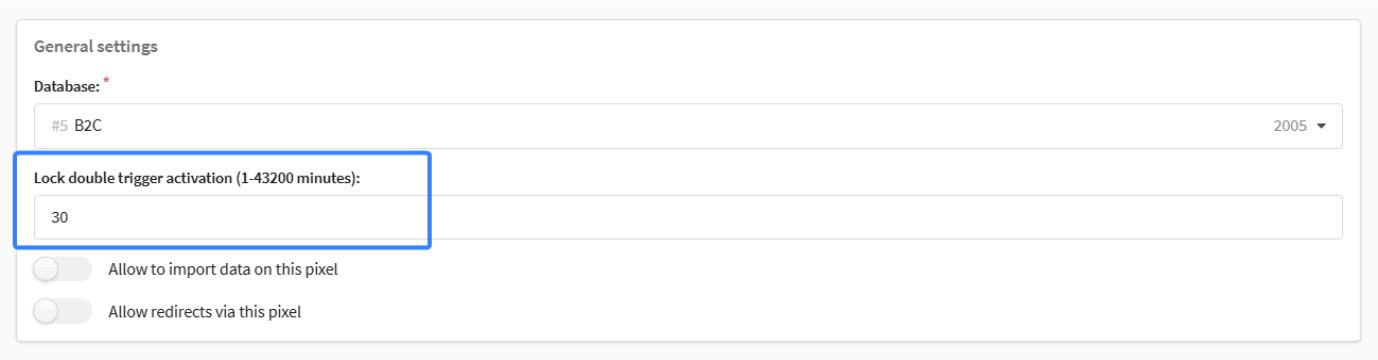
Use AND when you want an event to strictly fit both conditions. Use OR when you want the event to fit at least one of the conditions.



You can also filter profiles by additional fields. Values of additional fields transferred via API request.

TIP

If the trigger does not activate in mailings or scenarios, although the trigger conditions are met, it is recommended to check pixel trigger interval.

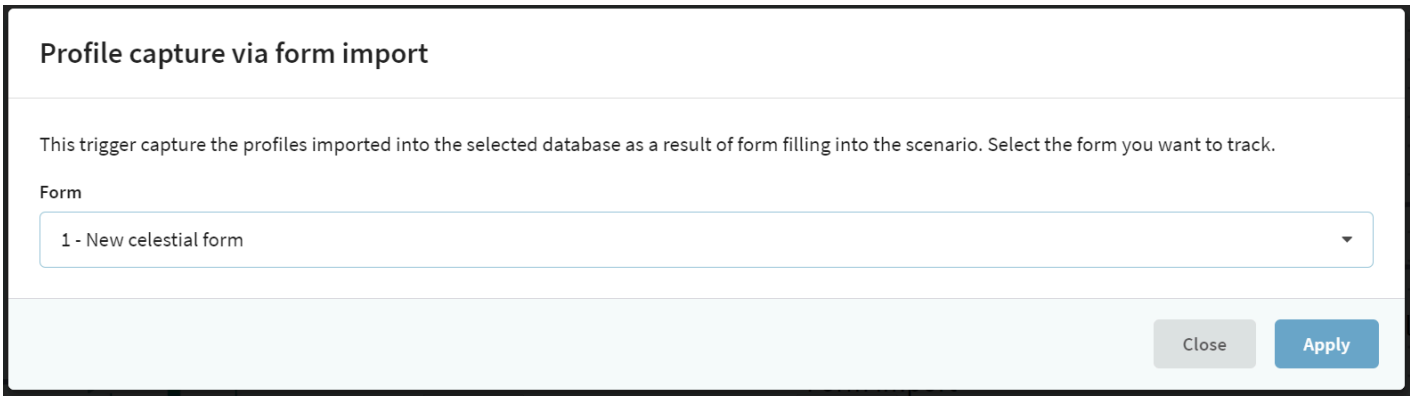


Form import

	The trigger will work if a new profile is created in the database as a result of a user filling out and submitting a form.

Read more

To use this trigger, you need to activate the Profiles import option on the Actions tab when configuring the form. The import must take place in the same database that is used in the scenario.



CAUTION

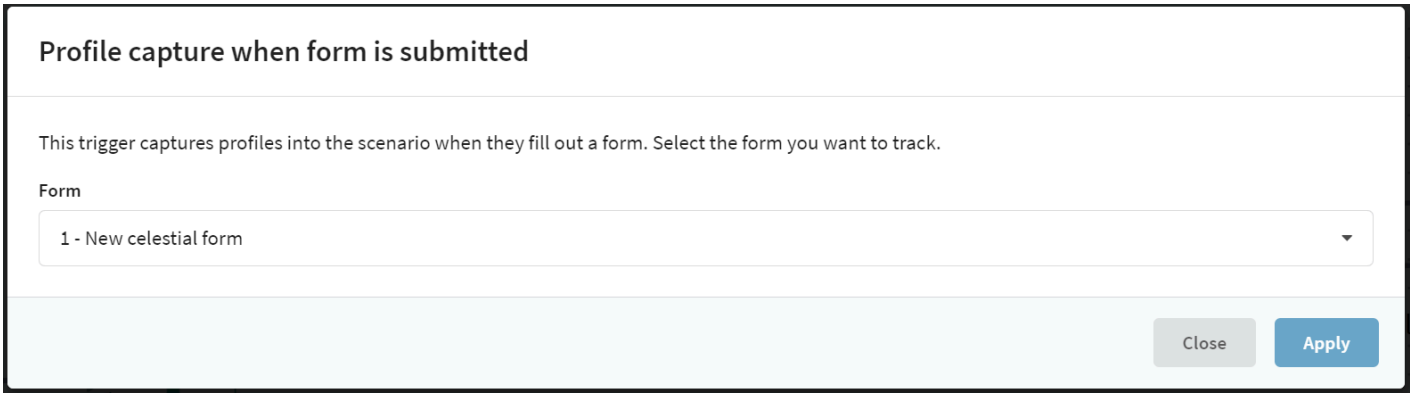
If you select the Any Import method, this trigger becomes unavailable.

Form submition

	Trigger will be activated only if the form was filled out on the transition from the email.

Read more

In the drop-down list, specify the form whose completion you want to track:



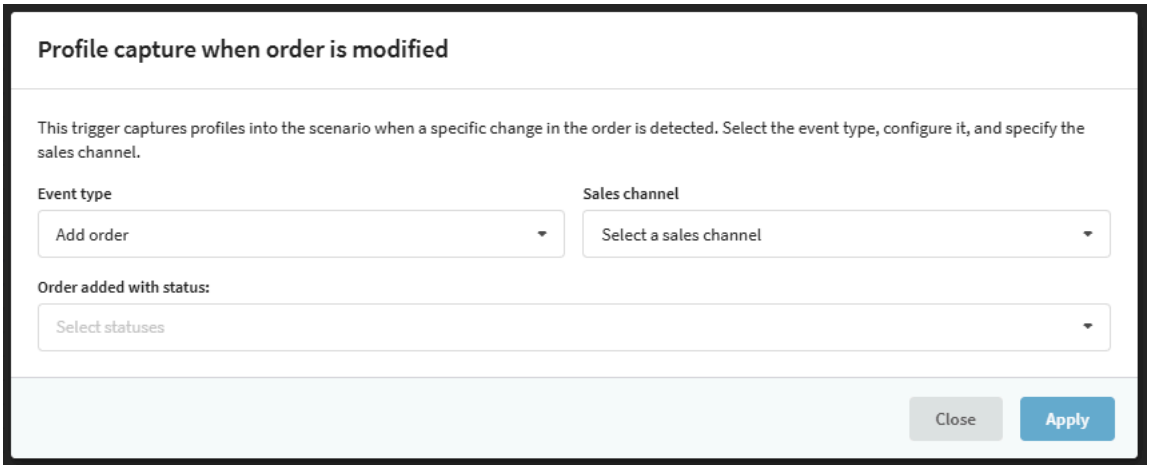
TIP

Read more about how to send a profile a link to the form in this article.

Order modify

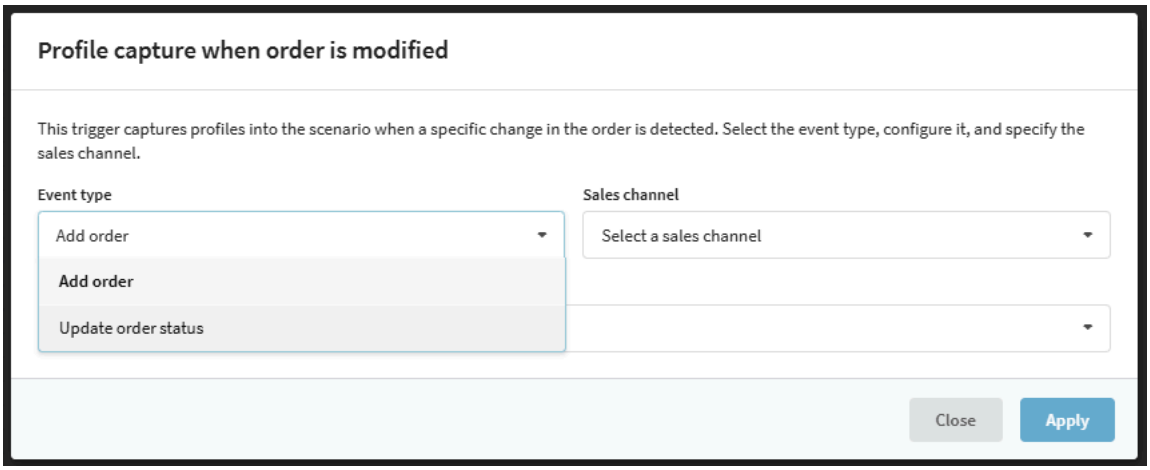
	This trigger captures profiles into the scenario when a specific change in the order is detected. The trigger is activated whena sales channel linked to the database selected for this scenario is configured.

Read more



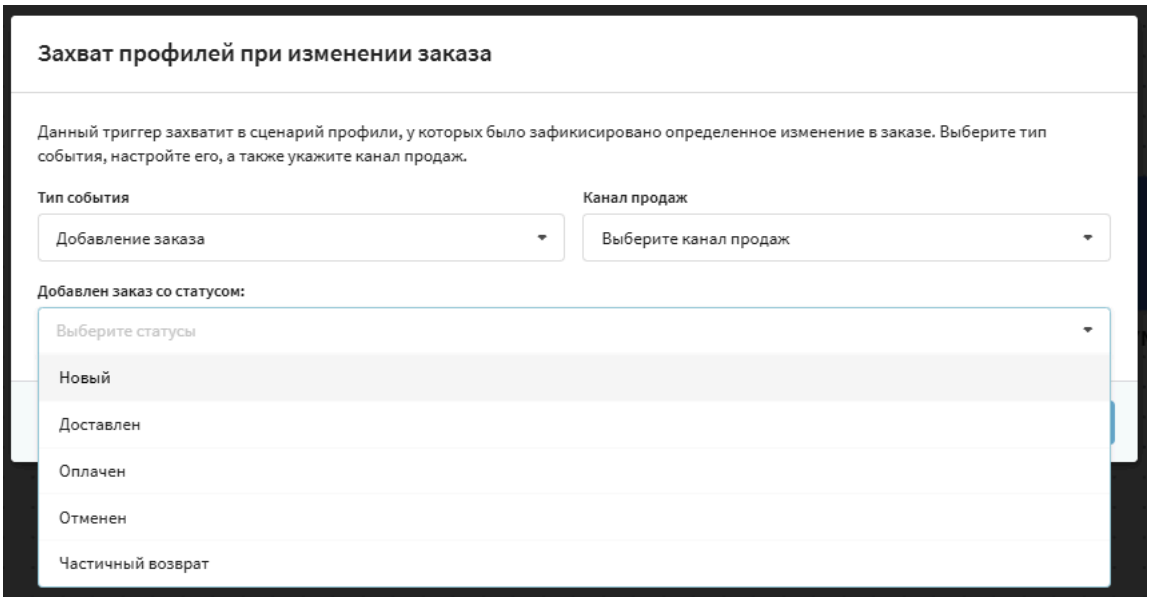
In the drop-down list slect the event type, sales channel and order status.

The following variants of event types are available:



- **Add order — the trigger will activate when an order with one of the specified statuses is added to the market.**
- **Update order status — the trigger will activate if the status of an existing order changes to one of the selected values.**

You can select one or more order statuses that result in a trigger:



TIP

Read more about how to customize orders in [this article](#).

Field value update

	The trigger is activated when the specified [profile field**](/user-guide/profiles-and-databases/66420#default-fields) takes the desired value.**

Read more

Select the profile field, condition, and expected value:

Conditions vary depending on the type of field:

Condition	Field type	Description
changed	**all**	**Any change in the profile field**

Condition	Field type	Description
equal	**all, except *tags*, *object*, *array of objects***	**The new value is equal to the expected value**
not equal	**all, except *tags*, *object*, *array of oblects***	**The new value is not equal to the expected value**
not empty	**all, except *tags***	**The field received any non-empty value**
empty	**all, except *tags***	**All values are removed for the profile field**
added any tag from the list	***tags***	**The list is defined in the trigger**
removed any tag from the list	***tags***	**The list is defined in the trigger**
property changed	***object***	**The path to the object key is specified. If the value of the key changes, the trigger will be activated.**
property is equal	***object***	**The object key path and expected value are specified. If the value of this key takes the expected value, the trigger will be activated.**
property is not equal	***object***	**The object key path and expected value are specified. If the value of this key does not take the expected value, a trigger will occur.**
element property has changed	***array of oblects***	**Specifies the path to an array element. If the property of this element changes, the trigger will fire.** **When selecting the path to the element property, you also select the data type to which the compared value should be converted. The following data types are available for comparison: string, int, float, and bool. Comparison of complex data types (object, array) is not possible. In the example below, you will not be able to track changes to the properties of the `key_1` and `key_4` elements.** `[ { "key_1": [1,2,3,4,5], "key_2": 2 }, { "key_3":"value", "key_4": [1,2,3,4,5] } ]` **Note that the drop-down list contains available paths to the field property, but the index of any element is 0 by default - `$[0].key_3`. To set the path to the desired element, enter the index manually in jsonpath format - `$[1].key_3`.**
element property is equal	***array of oblects***	**The path to the array element and the expected value are specified. If the property of this element takes the expected value, the trigger will be activated.**
element property is not equal	***array of oblects***	**The path to the array element and the expected value are specified. If the property of this element does not take the expected value, a trigger is activated.**
element added	***array of oblects***	**A new element is added to the array.**
element removed	***array of oblects***	**An element is removed from the array.**

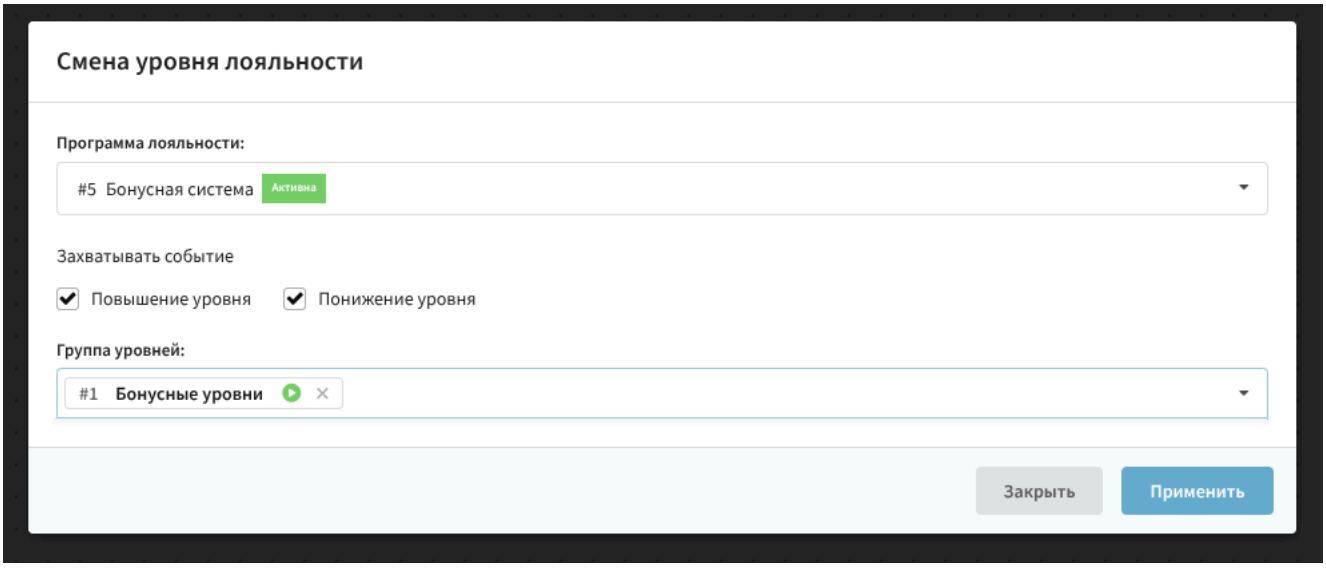
CAUTION

This type of trigger can be used with links in templates that are set to Update profile fields. Note that if there are other triggers attached to link jumps in addition to the field update, they will also be triggered.

Loyalty level change

	The trigger is activated when a profile moves to a different level in a loyalty program.

Read more

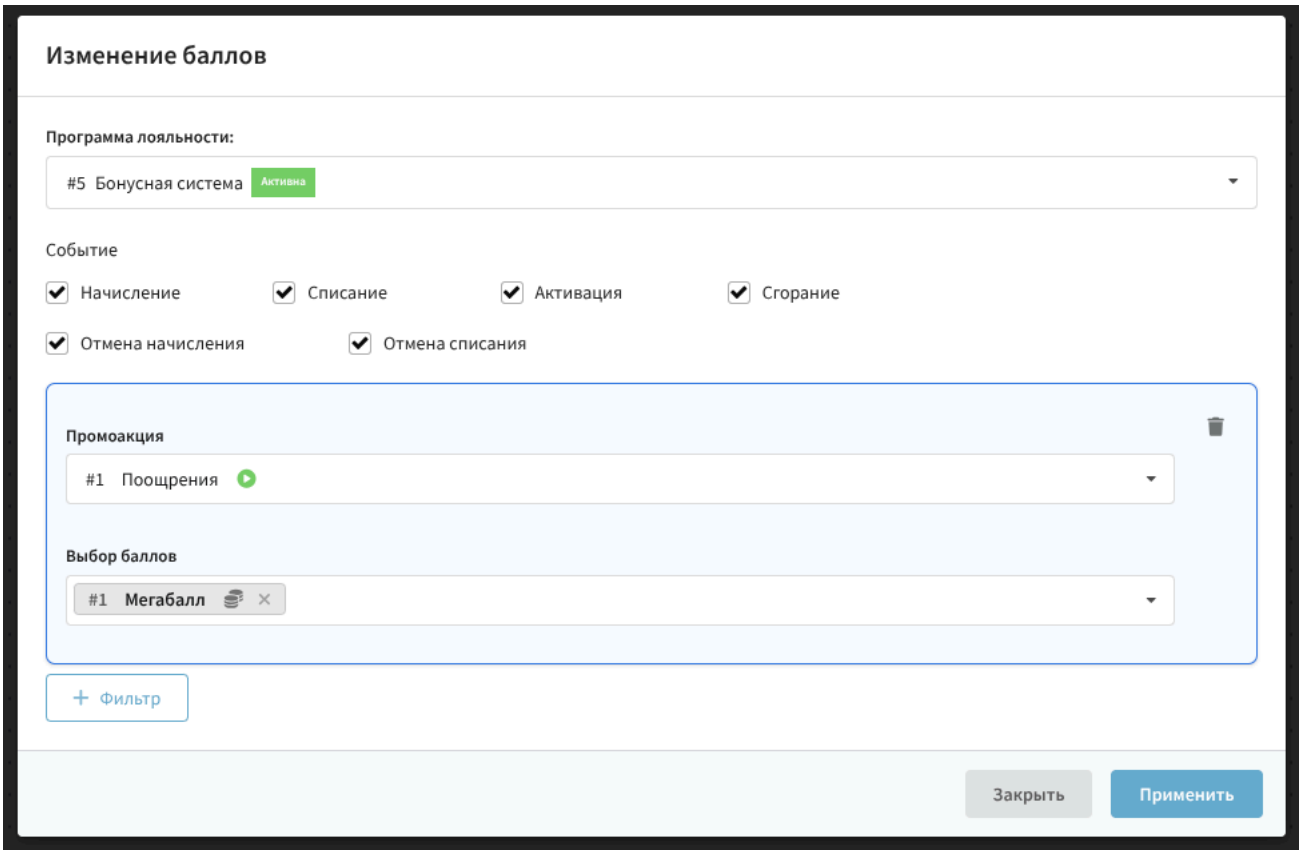


To configure the trigger, select a pre-created loyalty program and tiers group. You can configure the trigger to fire on a level upgrade, downgrade, or any level change.

Points change

	The trigger is activated when the number of points on a profile changes in a loyalty program.

Read more



To configure the trigger, select a pre-created loyalty program and the actions that will activate the trigger:

- **Accrual**
- **Redemption**
- **Activation**
- **Expiration**
- **Accrual cancellation**
- **Redemption cancellation**

You can also add a filter to select the promotion and points type for which this trigger will fire.

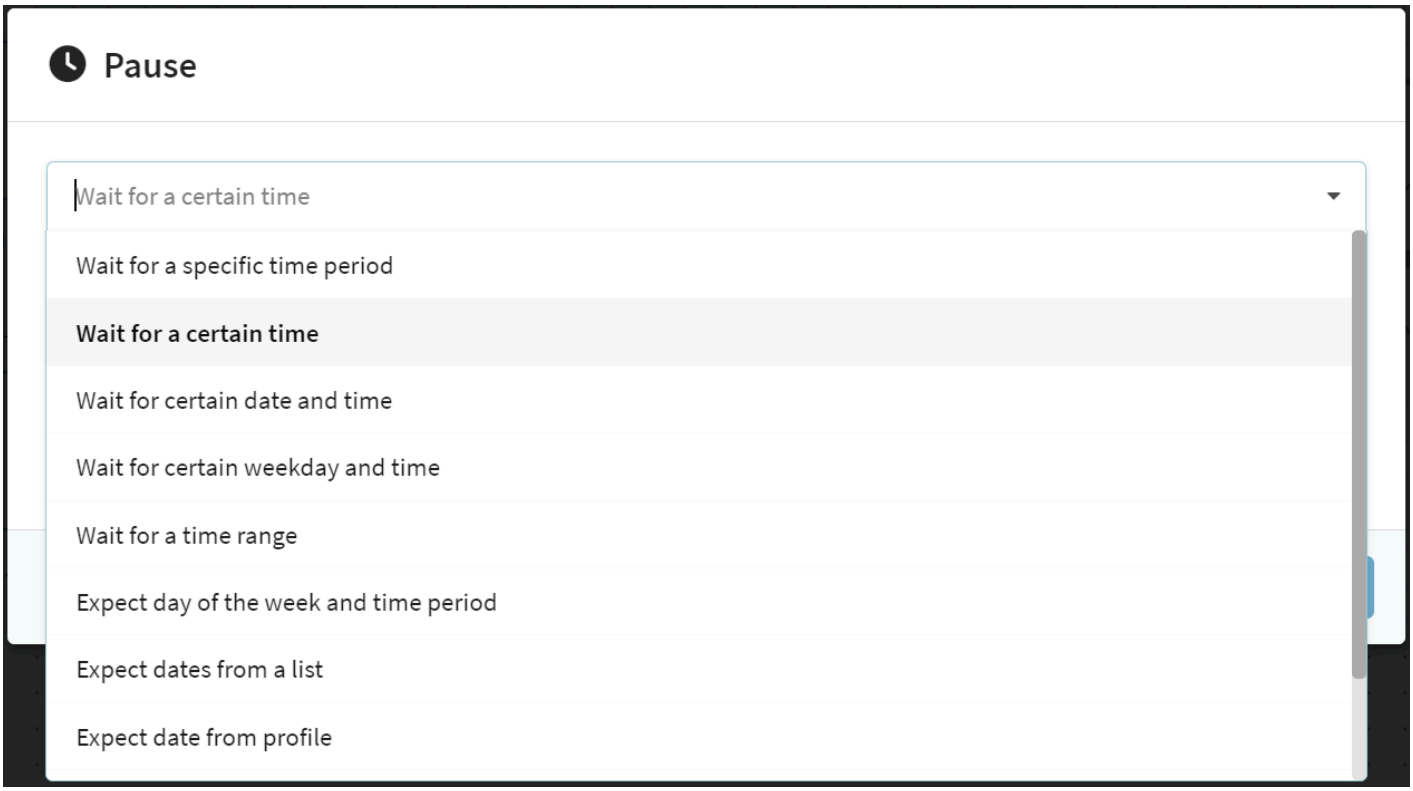
Operators

Pause

	Sets the intervals between scenario actions. With this node you can configure the start time of the node that follows. This allows, first of all, a more delicate approach to the issue of communication time with the client. The node can also wait for certain events: the start of a promotion, a holiday, etc.

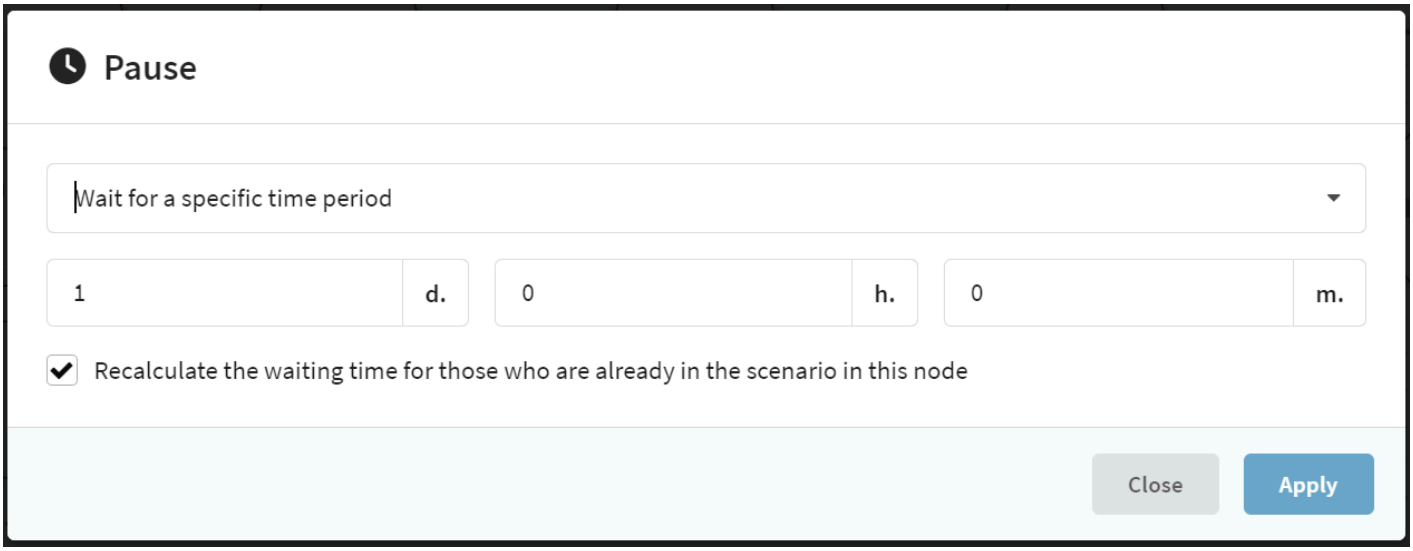
Read more

The following options are available:



Pause relative to the previous action

To wait a specific amount of time after a scenario has captured a customer profile or after a previous action has been performed, select the "Wait for a specific time period" option:



TIP: Recalculate waiting time

When using the waiting time recalculation feature, the platform will recalculate the pause duration for those who are already in this Scenario node. For example, you initially set a pause with a duration of 10 days and then increase the waiting time to 20 days. In that case, a profile that is captured in the scenario after you make the change will wait 20 days. As for a profile that was already in the node for 8 days before the change was made, the wait time would be another 12 days, not 20.

If the feature is disabled, changing the "Pause" node will not recalculate the waiting time.

Pause until a certain time

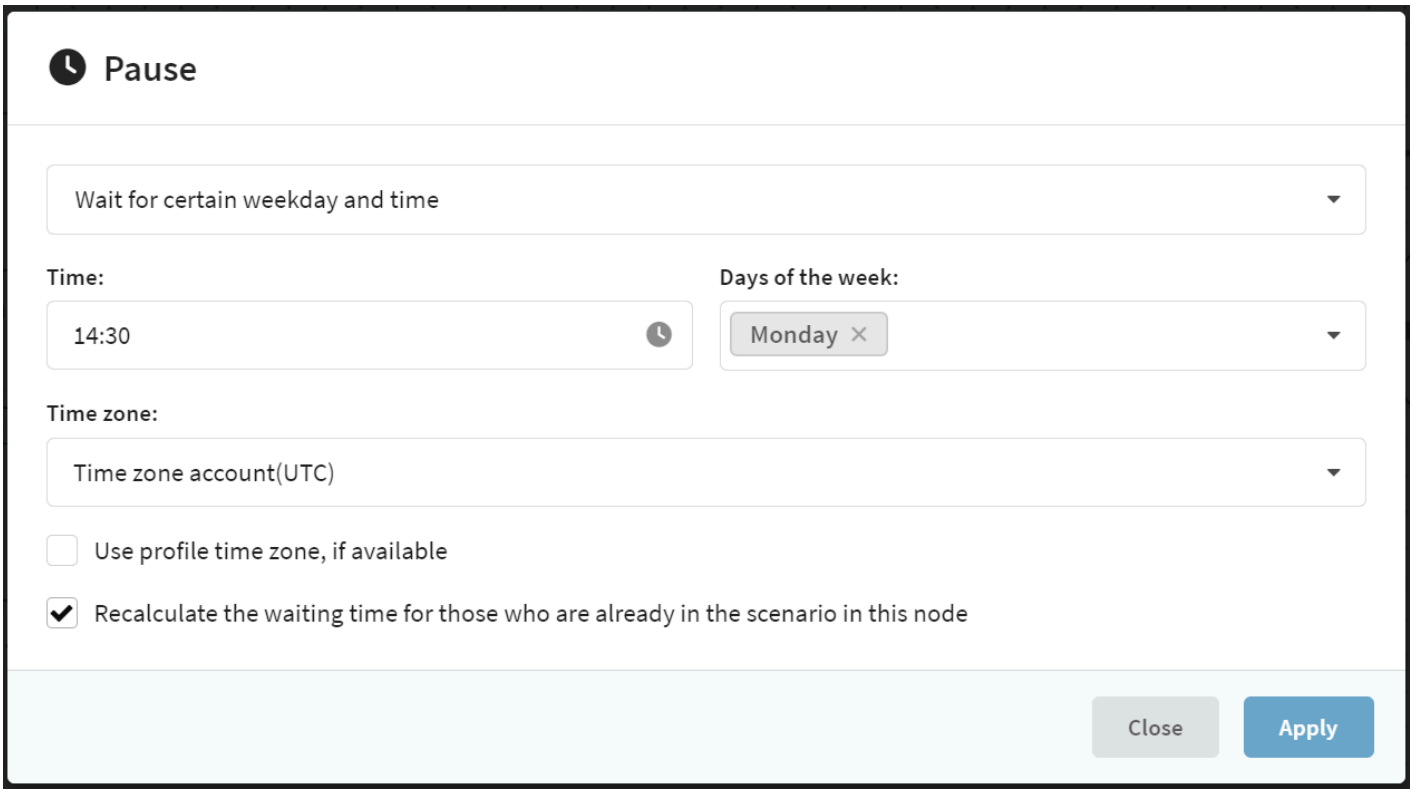
These options pause the scenario until a certain time of day - at the nearest selected time, on a specific date, on the nearest day of the week. And if the time zone of the profile is known, it can also be taken into account.

- **Wait for a certain time — the exact time of profile transition to the next node is specified.**
- **Wait for a certain date and time — indicates the exact time and date of the transition to the next node.**
- **Wait for certain weekday and time — the exact time and one or more days of the week are set up**
- **Expect days from a list — selects the exact time and multiple dates when the profile moves to the next node. Each profile, upon entering, "stops" until the nearest date from the list.**

TIP

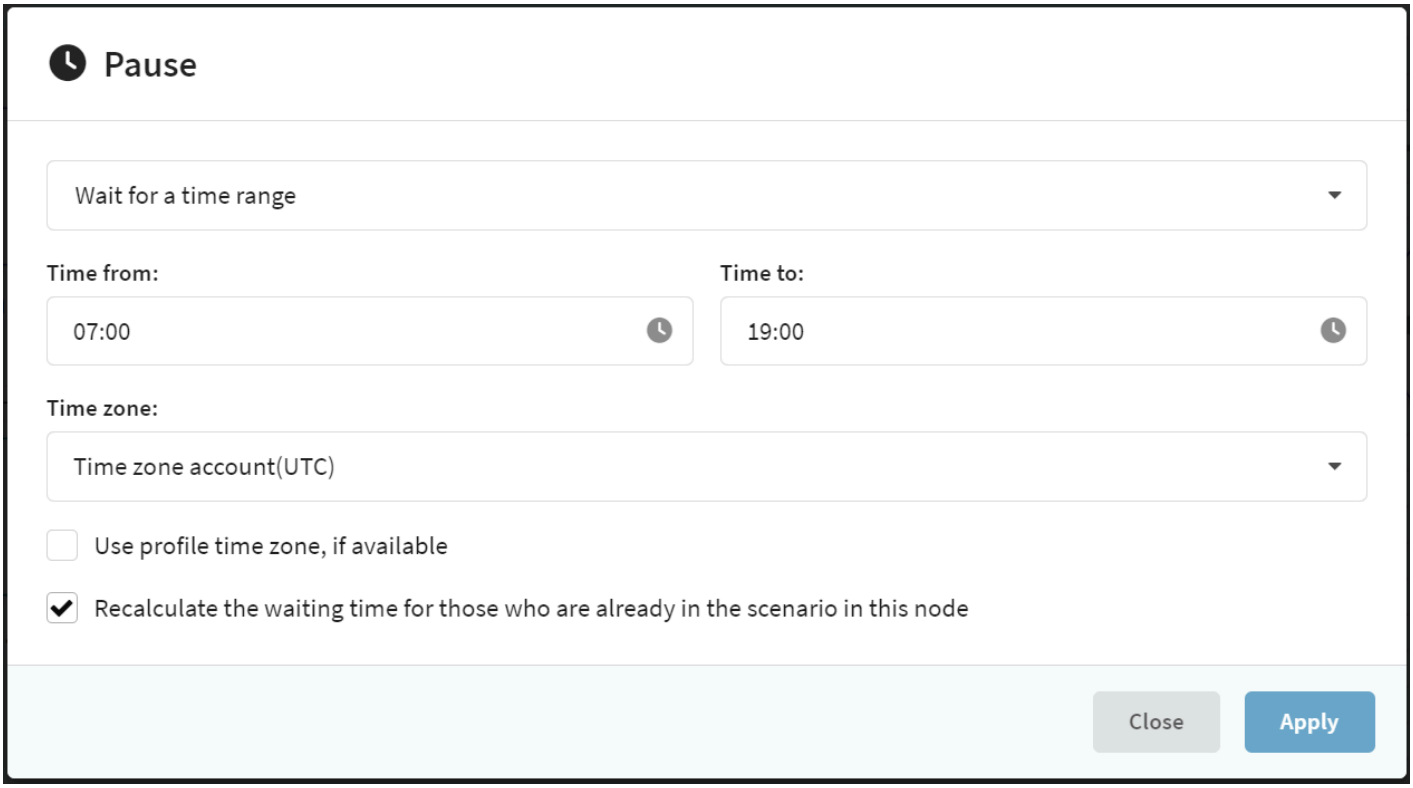
Here's an example of how the date list works in the "Pause" node.

Suppose you set the following dates: "2023-02-01", "2023-07-01", and "2023-10-01". If a profile reaches the node on January 1, 2023, it will continue through the Scenario on February 1. If a profile reaches the node on March 1, 2023, it will proceed on July 1. If a profile reaches the node in August, it will wait until October 1. Each profile waits only once until the nearest date relative to the moment it enters the node.

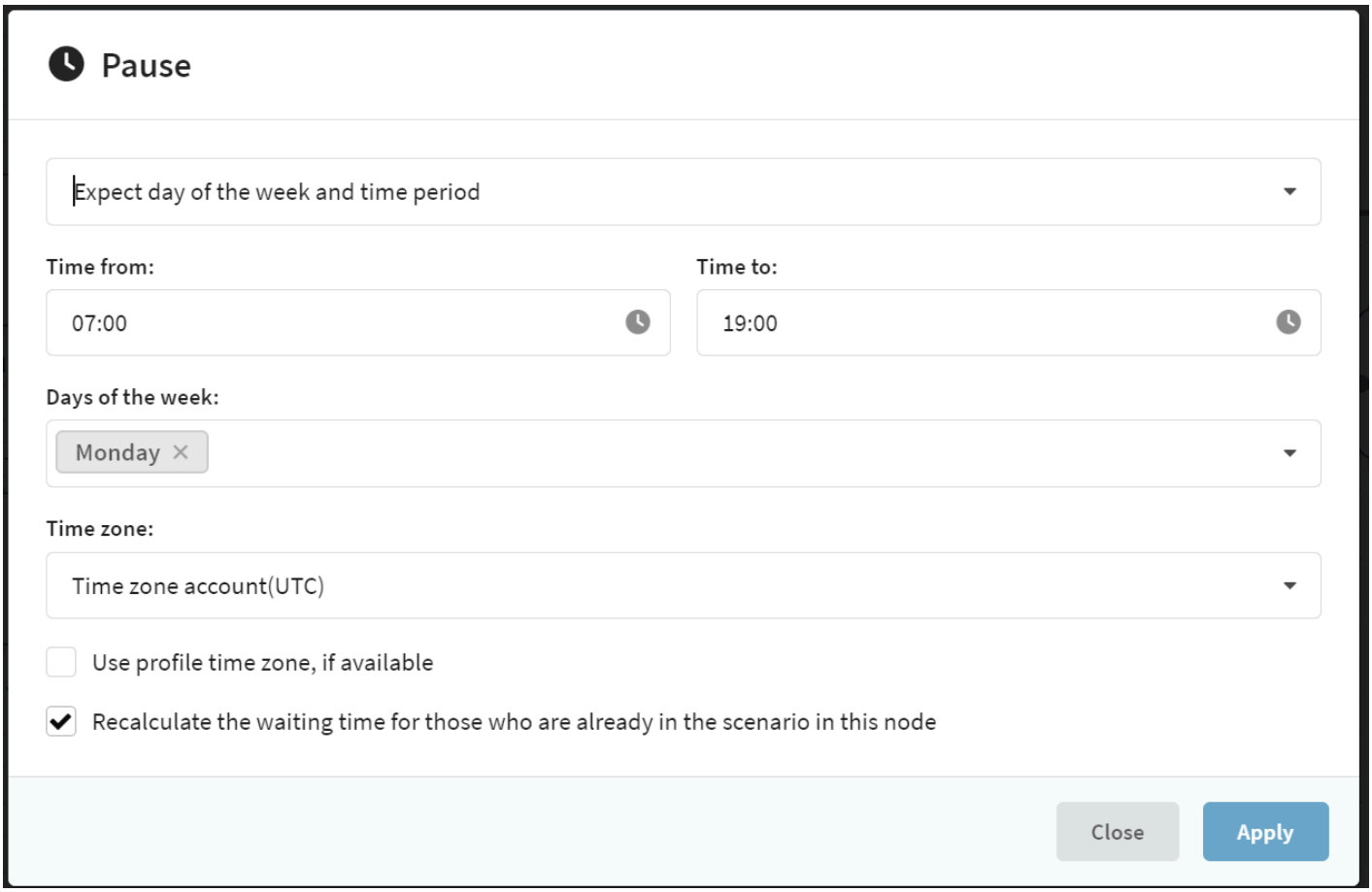


Pause until a certain period of time has elapsed

The Wait for a time range option sets the activity hours for the next node of the scenario. For example, if you want a certain mailing to run from 9am to 9pm:



The Expect day of the week and time period option sets the days and hours of activity for the next node of the scenario. For example, if you want a certain mailing to run only on Saturdays and Sundays from 10 to 12.



Expect date and time from profile

These settings will pause the scenario until the date or time specified in one of the profile fields:

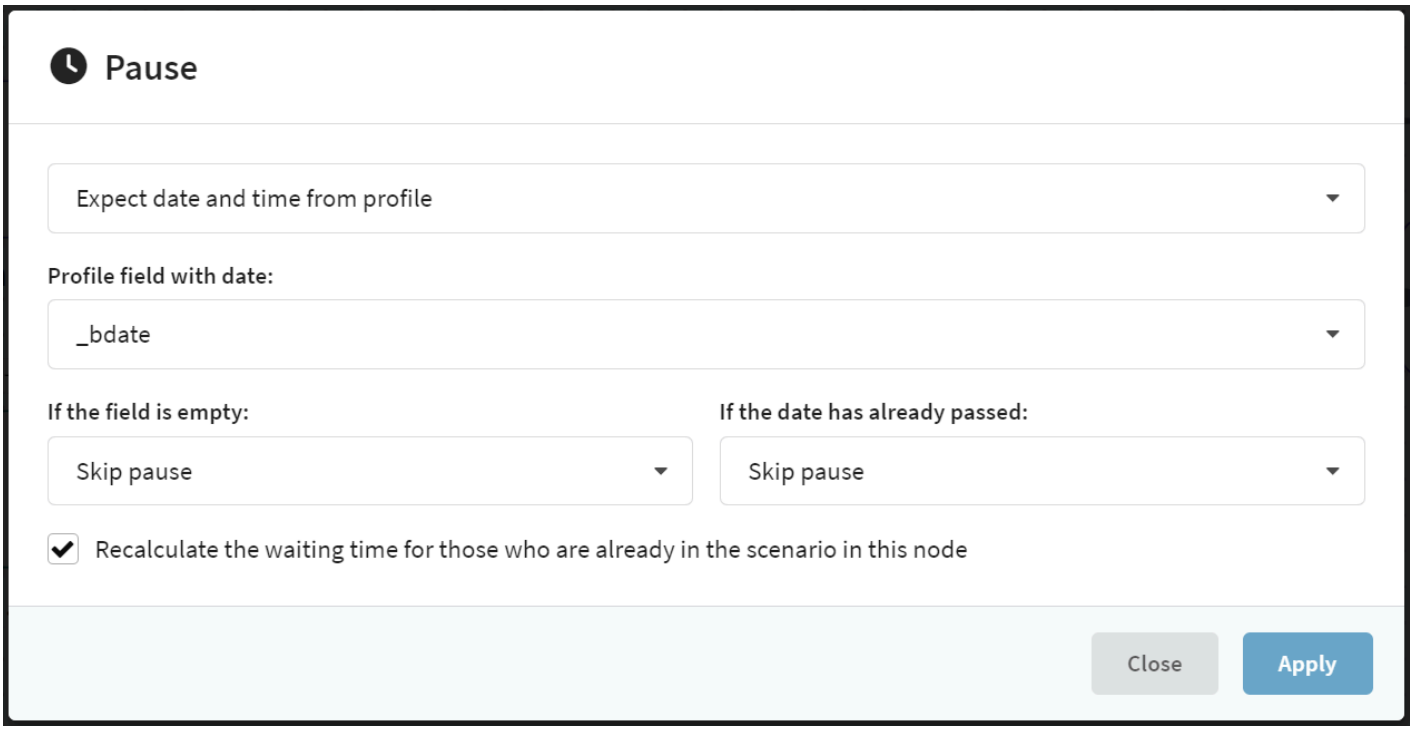
- **Expect date from profile — the profile will move to the next node when the date specified in the selected field and the time specified in the "Pause" node settings occur.**
- **Expect time from profile (date-independent) — profile will move to the next node when the time specified in the selected field occurs. The date may not be the same as the current date.**
- **Expect date and time from profile — the profile will move to the next node when the dates and times specified in the selected field occur.**

Select a field in the database with the required data. It can be date of birth (`_bdate`), date of registration in the system (`_regdate`) or any custom field with the "Date" type (for example, date of a scheduled event).

Next, configure the scenario to work if the selected field is not filled in for the profile:

- **Skip pause — the profile will immediately move to the next node without waiting for a certain time.**
- **Abort a script for a profile — the profile will leave the scenario.**

If the date specified in the profile field has already passed, you can either drop the profile immediately to the next node of the scenario, ignoring the pause, interrupt the scenario for the profile, or move the date to the next year.



Best send time

	Sets a pause before the next node of the scenario, calculated on the basis of ML service data. The profile will not move to the next node until the optimal time to send it a message. The optimal way to work is to add a node from the "Channels" category after the "Waiting for best send time" node. The mailing will then send the profile a message at a time that is most convenient for them. </td>

Read more

INFO

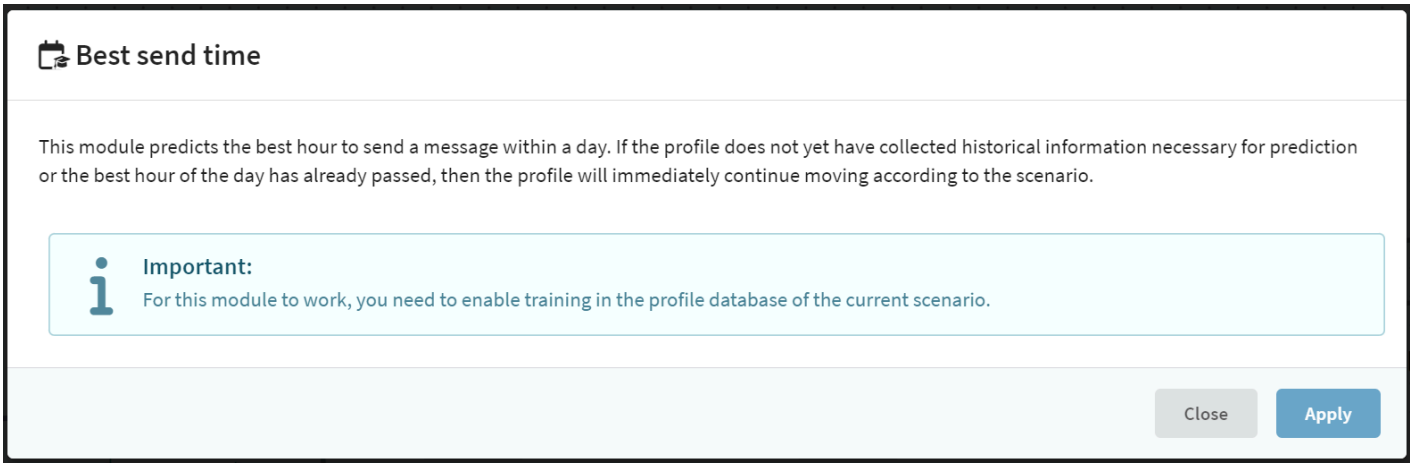
The calculation of the best time is done through machine learning algorithms. ML service determines the best hour when the probability of a profile opening a message is the highest, and selects an array with several good sending hours.

The node is available if the profiles captured in the scenario come from a database for which machine learning is enabled.

The profile will move to the next node when its best sending hour arrives.

If the best hour has already passed when the profile hits the node, the node will wait until the next good sending hour. If the good hour has also passed, the profile will immediately move to the next node.

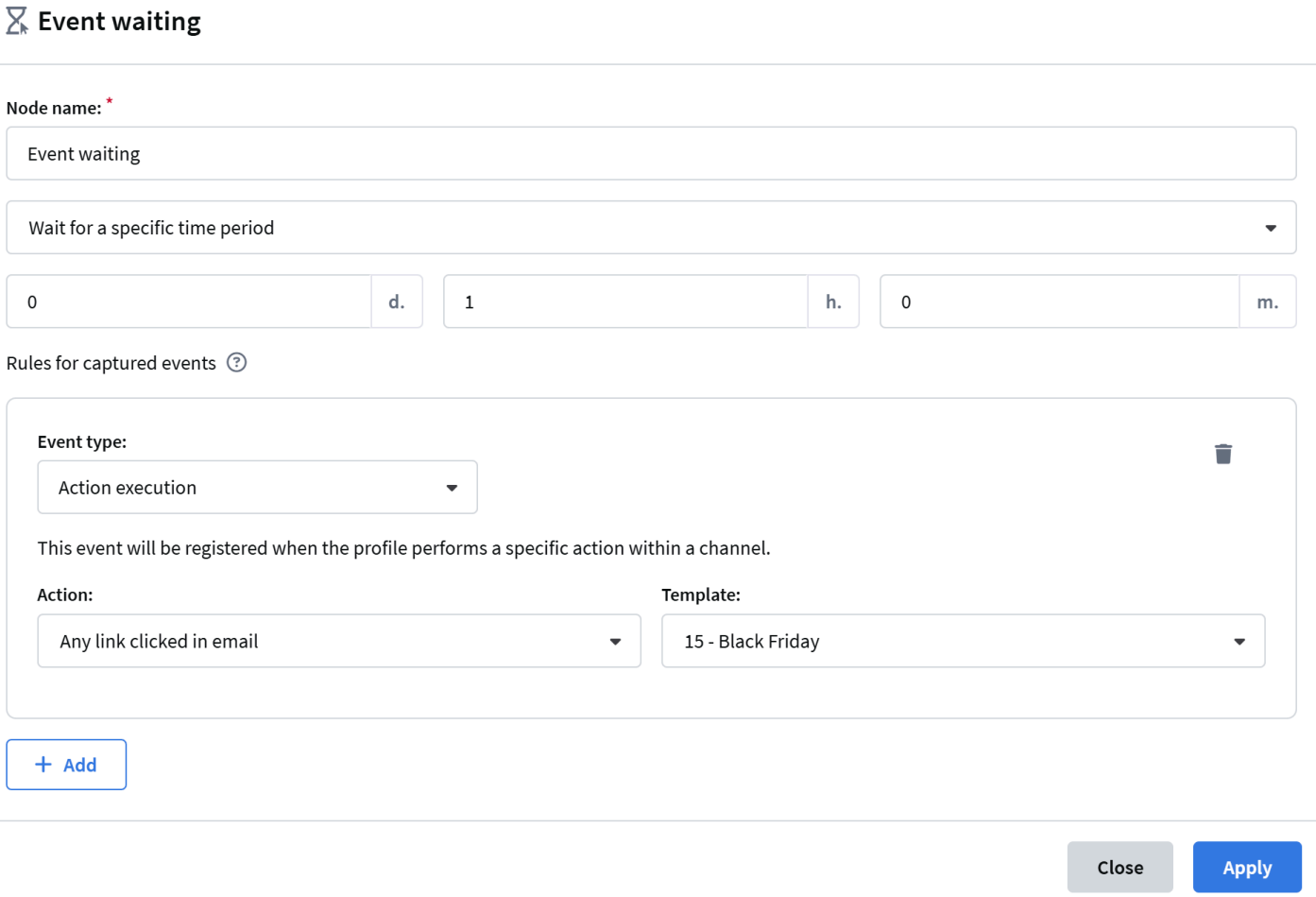
If a profile that has no data on the best sending hour enters the node, it will also be immediately forwarded to the next node.



Event waiting

	Stops a profile's progress along the scenario **until a specific action is performed by the profile. Once **any** of the specified events is registered by the platform, the profile will proceed to the next node along the upper (green) branch. If the waiting time exceeds the defined value, the profile will move along the lower (red) branch.**

Read more



Enter the node name for display in the scenario. Then configure the event waiting time. The profile will remain at the node for the specified period. If none of the specified events occurs during this time, the profile will proceed along the scenario's lower branch.

Wait time settings:

- **Wait for a specific time period — sets a fixed duration for waiting for the event. The profile will move along the lower branch if the event does not occur within the specified time.**

  ```
  ![](/img/attachments/9407058/en/days-hours-minutes.png)
  ```

- **Wait for a certain time — the profile will move along the lower branch if the event does not occur by the specified time of day.**

  ```
  ![](/img/attachments/9407058/en/certain-time.png)
  ```

- **Wait for a certain date and time — the profile will move along the lower branch if the event does not occur by the specific date and time.**

  ```
  ![](/img/attachments/9407058/en/date-and-time.png)
  ```

- **Wait for a certain weekday and time — one or more days of the week and an exact time are configured. The profile will move along the lower branch if the event does not occur by the nearest selected day and time.**

  ```
  ![](/img/attachments/9407058/en/day-and-time.png)
  ```

- **Expect dates from a list — several dates and a time are selected. The profile will move along the lower branch if the event does not occur by the nearest date from the list. Each profile waits until the date closest to the moment it entered the node.**

  ```
  ![](/img/attachments/9407058/en/date-list.png)
  ```

- **Expect date from profile — the profile will move along the lower branch if the event does not occur by the date specified in the selected profile field.**

  ```
  ![](/img/attachments/9407058/en/from-profile.png)
  ```

- **Expect time from profile (date-independent) — the profile will move along the lower branch if the event does not occur by the time specified in the selected profile field. The date may differ from the current date.**

  ```
  ![](/img/attachments/9407058/en/from-profile-witohut-date.png)
  ```

- **Expect date and time from profile — the profile will move along the lower branch if the event does not occur by the date and time specified in the selected profile field.**

```
![](/img/attachments/9407058/en/from-profile-date-and-time.png)
```

Condition

	Realizes logical selection of the next step of the scenario for customer profiles. The selection condition is defined by Segmentation rules. If the condition is met, the client profile will continue to move to the right in the scenario. If the condition is not met, the profile will move down the scenario and an alternative sequence of actions can be configured for it. a node can also wait for the condition to be met for a certain amount of time.

TIP

If you link several Condition elements vertically, you can create several alternative chains in one - depending on the results of the checks.

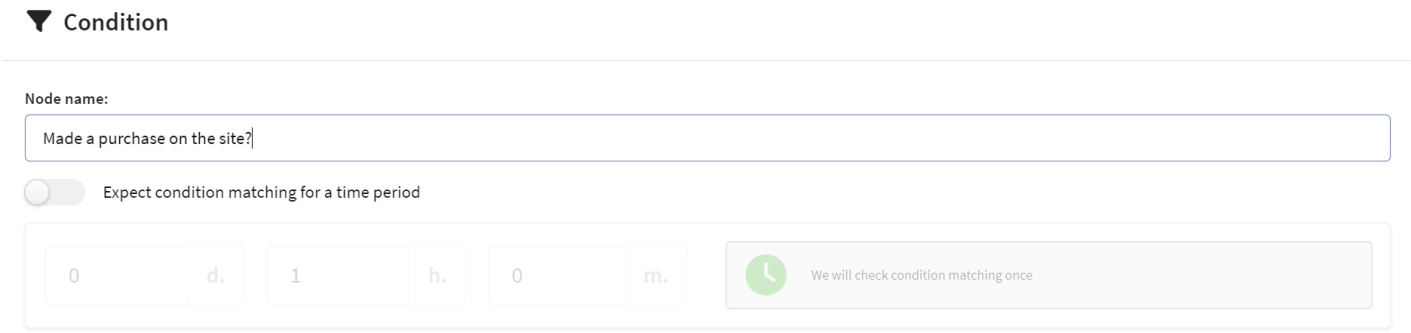
Read more

General settings

Enter the name of the node to be displayed in the scenario.

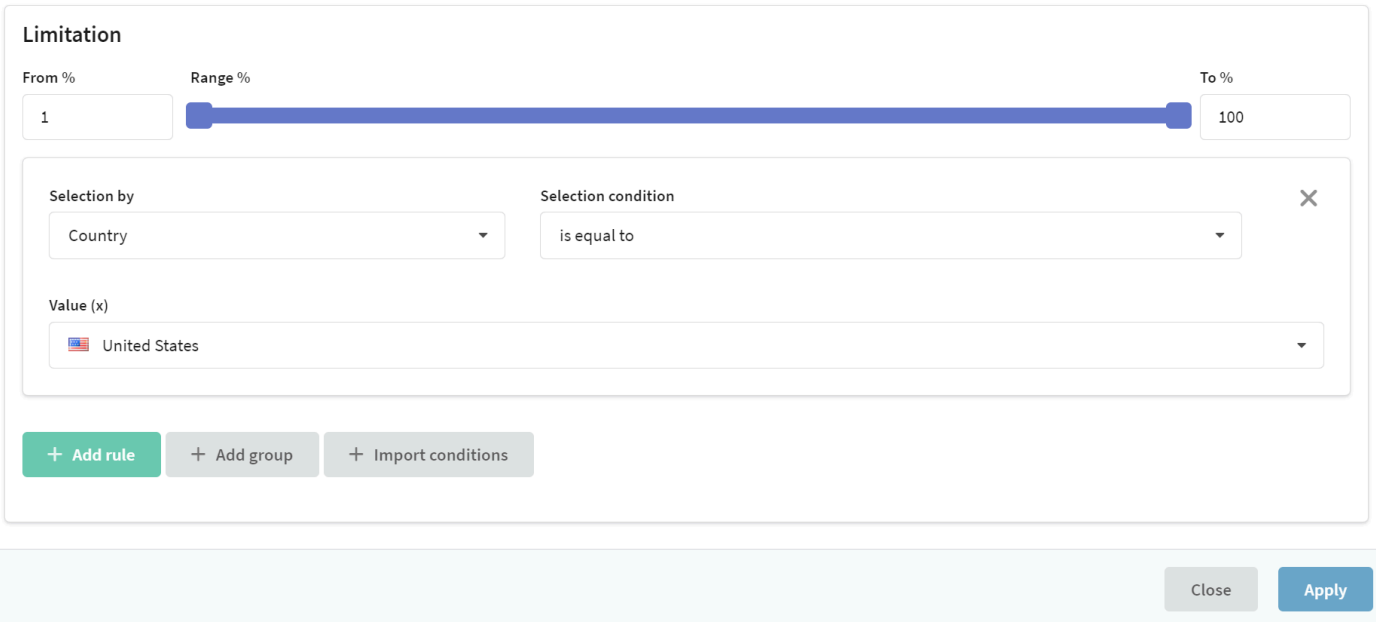
TIP

If you want to dynamically check if the condition is met, enable the “Expect condition matching for a time period option. In this way the check will be performed several times. If the waiting time is fixed, do not enable this option, but instead add the Pause” node before the condition.



Rules for fulfilling the condition

The condition is composed by rules and rule groups similar to Segmentation rules. Subscribers can be categorized by personal data, subscription data, action history:



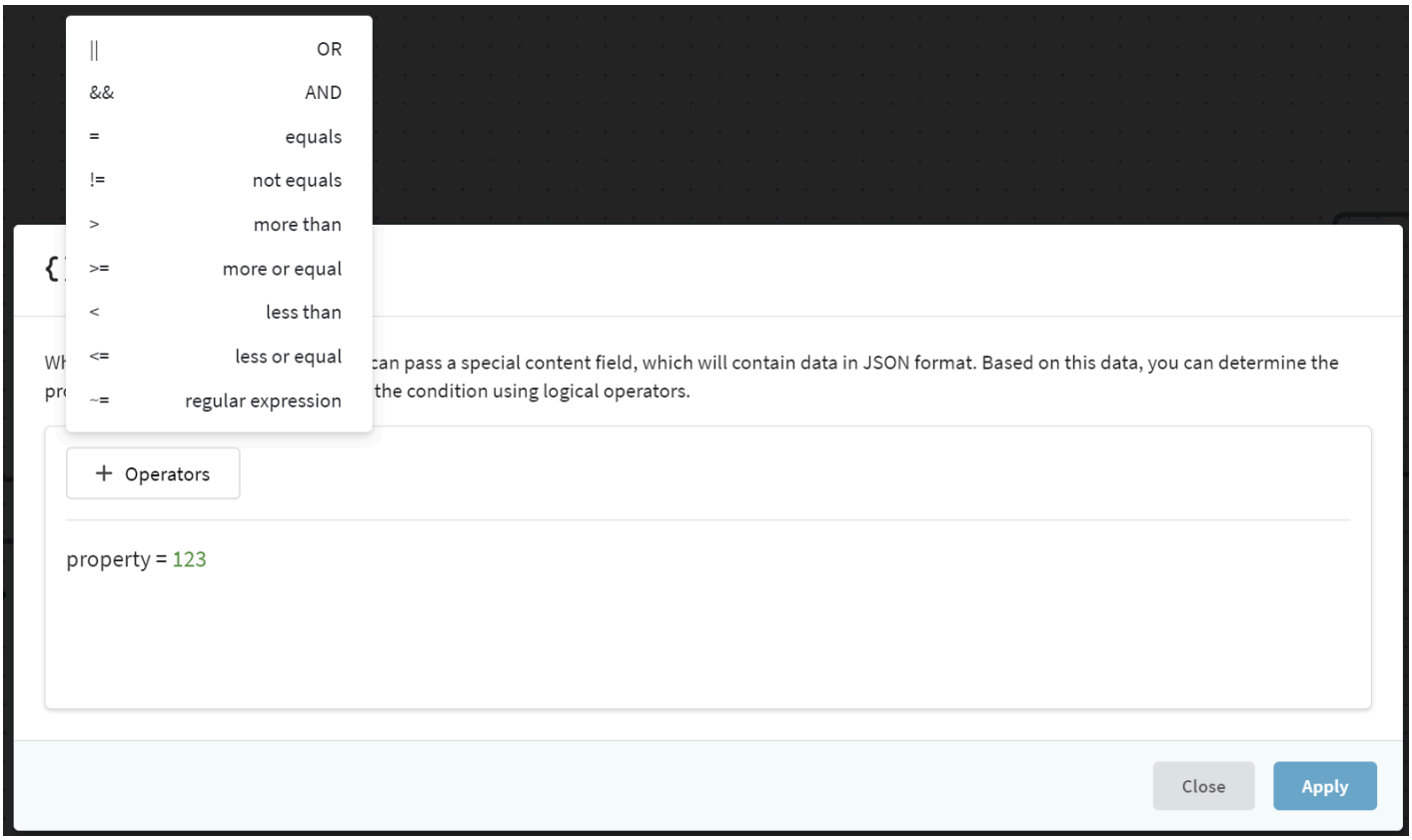
JSON filter

	If you call a scenario via API, you can pass a special `content` field with an undefined structure in JSON format (you define it). This data can be used to insert into messages, return to an Action Hook, or to build conditions in a scenario. For example, when a user takes an action, the website (or application) activates a scenario and passes all data related to the action to it. The "JSON filter" node determines further actions for the profile based on the validation of this data.

Read more

JSON filter works in the same way as a normal condition. You can repeatedly use this node to build complex schemes.

You can set conditions in the code editor using logical operators and expressions.



An example of a `"content"` field in an API request:

```
{
  "content": {
    "customer_id": "12345",
    "total_purchases": 5,
    "last_purchase_amount": 250,
    "average_purchase_amount": 100,
    "preferred_category": "electronics",
    "loyalty_points": 300,
    "membership_status": "premium",
    "discount_coupons_used": 2,
    "active_coupon": "win0513",
    "promo": null
  }
}
```

Note, the JSON object key is specified without quotes (e.g., `membership_status="premium"`).

Operator	Example	Description
`||` **Or**	`(preferred_category = "electronics") || (preferred_category = "computers")`	**The condition will be fulfilled if the customer is interested in the "Electronics" or "Computers" categories.**
`&&` **And**	`(total_purchases >= 5) && (average_purchase_amount >= 100)`	**The condition will be fulfilled if the customer has made 5 or more purchases and the average check exceeds 100.**
`=` **Equal**	`membership_status = 'premium'`	**The condition will be fulfilled if the client has premium-status.**
`!=` **Not equal**	`membership_status != 'standard'`	**The condition will be fulfilled if the client does not have a standard-status.**
`>` **More than**	`last_purchase_amount > (2 * average_purchase_amount)`	**The condition will be fulfilled if the customer's last purchase amount exceeds twice the average purchase amount.**
`>=` **More than or equal**	`last_purchase_amount >= 250`	**The condition will be fulfilled if the amount of the last purchase equals or exceeds 250.**
`<` **Less than**	`loyalty_points < 500`	**The condition will be fulfilled if the number of bonus points is less than 500.**
`<=` **Less than or equal**	`loyalty_points <= 500`	**The condition will be fulfilled if the number of bonus points is 500 or less.**
`~=` **Regular value**	`active_coupon ~= '^win'`	**The condition will be fulfilled if the promo code starts with "win".**
`!~=` **Opposite regular value**	`active_coupon !~= '^win'`	**The condition will be fulfilled if the promo code does NOT start with "win".**
`+` **Addition**	`(loyalty_points + bonus_points) < 200`	**Arithmetic addition operator.**

Operator	Example	Description
`-` **Subtraction**	`(loyalty_points - penalty_points) > 100`	**Arithmetic subtraction operator.**
`*` **Multiplication**	`(bonus_points * years) > 360`	**Arithmetic multiplication operator.**
`/` **Division**	`(total_purchase / 365) > 5`	**Arithmetic division operator.**
`%` **Modulo division**	`(total_sum % 15) = 5`	**Operator for calculating the remainder of division.**
`^` **Exponentiation**	`(promotion_points ^ 3) > 360'`	**Operator for exponention a number to a degree.**
`is defined` **Checking the existence of a field**	`promo is defined`	**The condition will be fulfilled if the “promo” field exists, regardless of whether a value is specified for it or not.**
`isnot defined` **Field absence check**	`promo isnot defined`	**The condition will be fulfilled if the “promo” field is not present in the code.**
`is null` **Checking empty field value**	`promo is null`	**The condition will be fulfilled if the “promo” field has no value (its value is `null`).**
`isnot null` **Checking the field value**	`promo isnot null`	**The condition will be fulfilled if the “promo” field has any value (equal to anything other than `null`).**
`contains` **Checking if the value on the right contains the value on the left**	**String:** `"hello" contains "he"` **Array:** `[1, 2, 3] contains 2`	**The condition will be fulfilled if the field “hello” contains the value “he”.**
`in` **Checking if the value on the left is contained in the field or array on the right**	**String:** `"he" in "hello"` **Array:** `2 in [1, 2, 3]`	**The condition will be fulfilled if the value 2 is contained in the array [1, 2, 3].**
`notin` **Checking if the string or array does NOT contain the value of the field on the left**	`"bye" notin "hello"`	**The condition will be fulfilled if the “hello” field does NOT contain the value “bye”.**

To access a nested field of an object, use the following syntax: `object.field`.

To access elements of an array by index, use the following syntax: `array.[0].field`.

Examples of usage:

```
// Object:

{
  "content": {
    "customer_id": "12345",
    "total_purchases": 5,
    "promo_campaign": {
      "tickets": 115
    },
    "tickets_info": [
      {
        "ticket_id": "T001",
        "price": 50,
        "date": "2024-08-30"
      },
      {
        "ticket_id": "T002",
        "price": 75,
        "date": "2024-08-31"
      }
    ]
  }
}

// Accessing a nested field of the object:

promo_campaign.tickets >= 100

// Accessing the first element of the array:

tickets_info.[0].ticket_id
```

Jump to scenario

	Sends the profile to the beginning of another scenario or to a certain node of another scenario. In this case, the current scenario and the scenario to which the profile moves must use the same database.If the chain in the scenario continues after the transition node, the profile moves to another scenario and at the same time continues to move through the current scenario.

Read more

Enter the name of the node. Select the scenario to which you want the profile to be sent. If you want the profile to jump to the beginning of another scenario, select input node as the node to jump to. If you want the profile to go immediately to a specific node in another scenario, select that node from the list.

Distribution node

	Creates multiple elements that separate scenario options for customer profiles regardless of any conditions. Both brute force and static distribution are available.

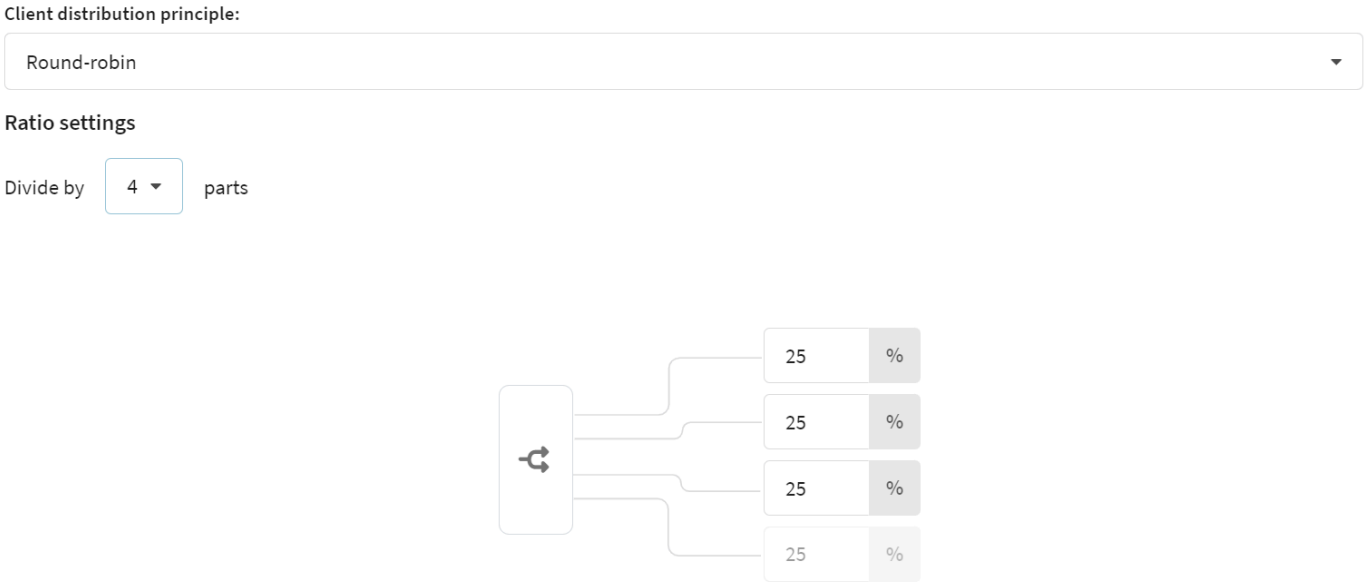
Read more

General settings

Select the distribution principle and the number of parts into which the audience will be distributed. You can change the percentage by editing the distribution elements.

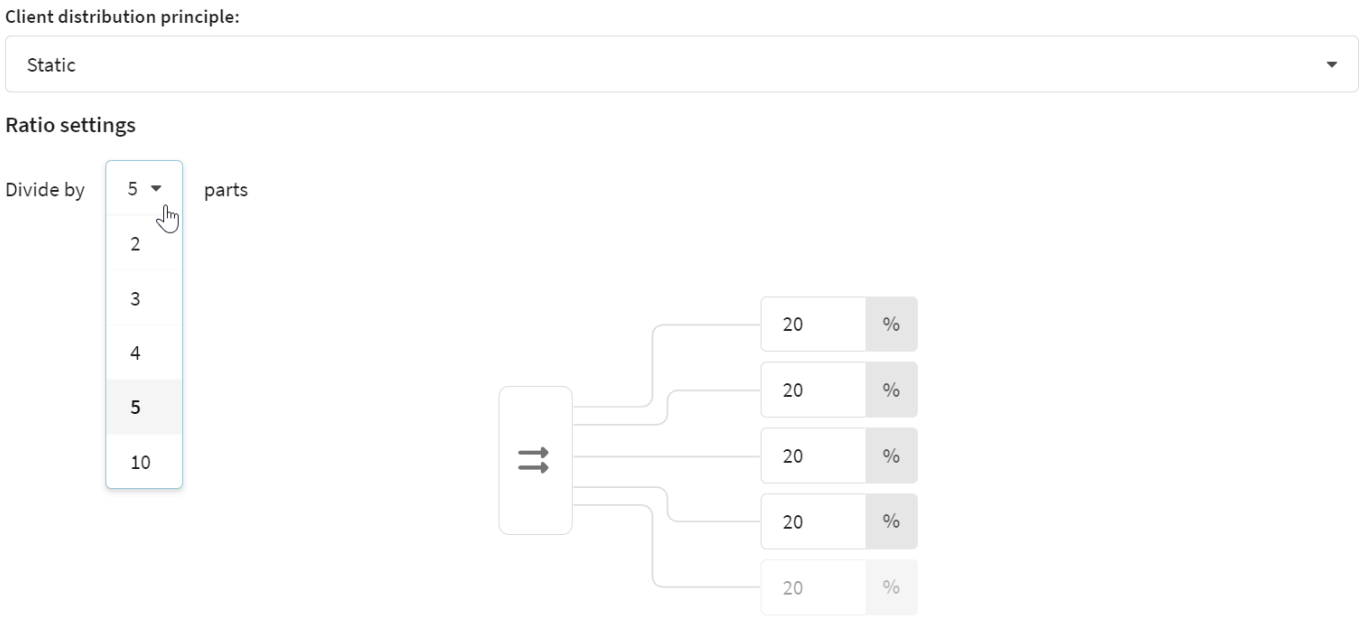
Distribution by round-robin

Profile distribution takes place at the moment of hitting the scenario node according to the specified proportions. It is based on the mechanism of Smooth Weighted round-robin balancing.



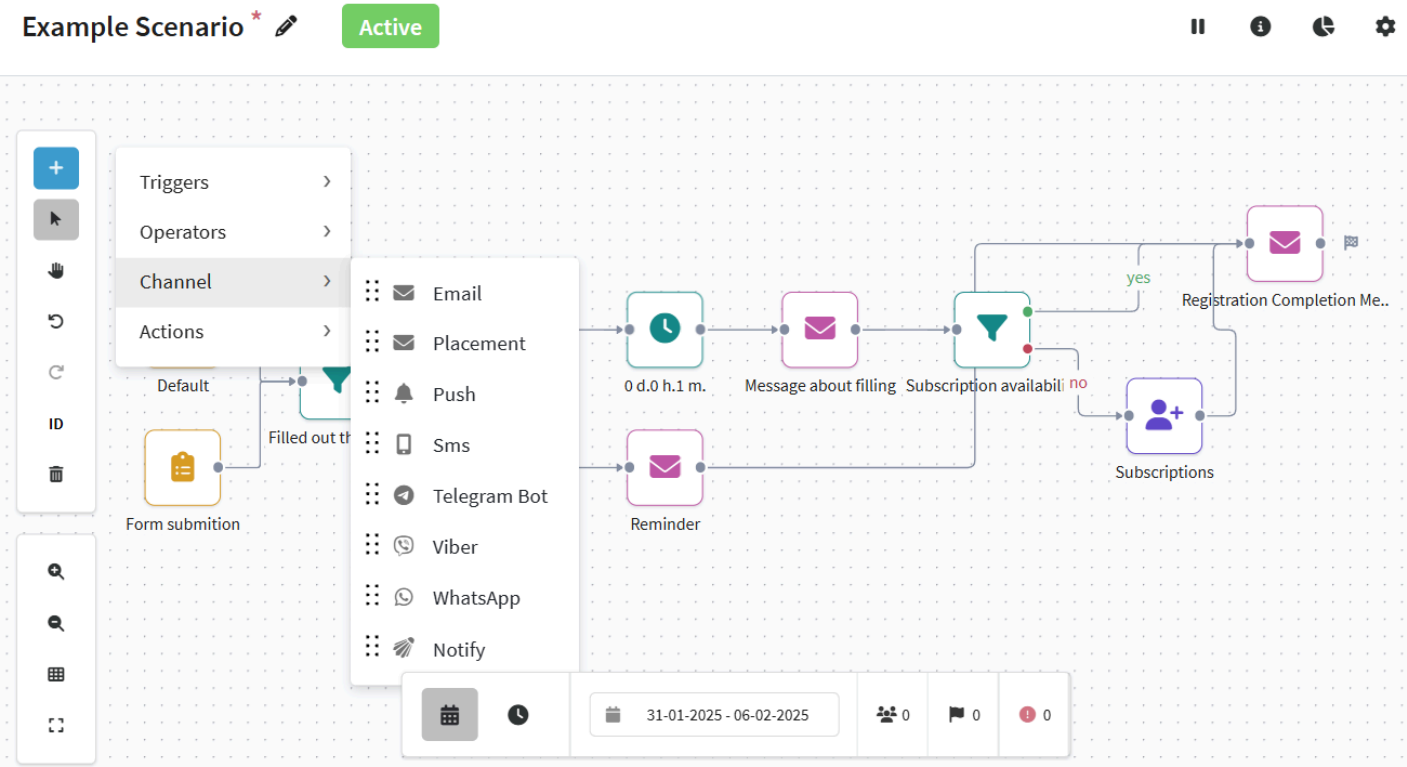
Static distribution

When using static distribution, the profile will always fall into the same distribution percentage, which cannot be changed. Static distribution is not recommended for a small number of profiles (up to 10,000).



Channels

Elements in this category launch a trigger mailing through a specific channel. A new mailing is created directly in the node editor.



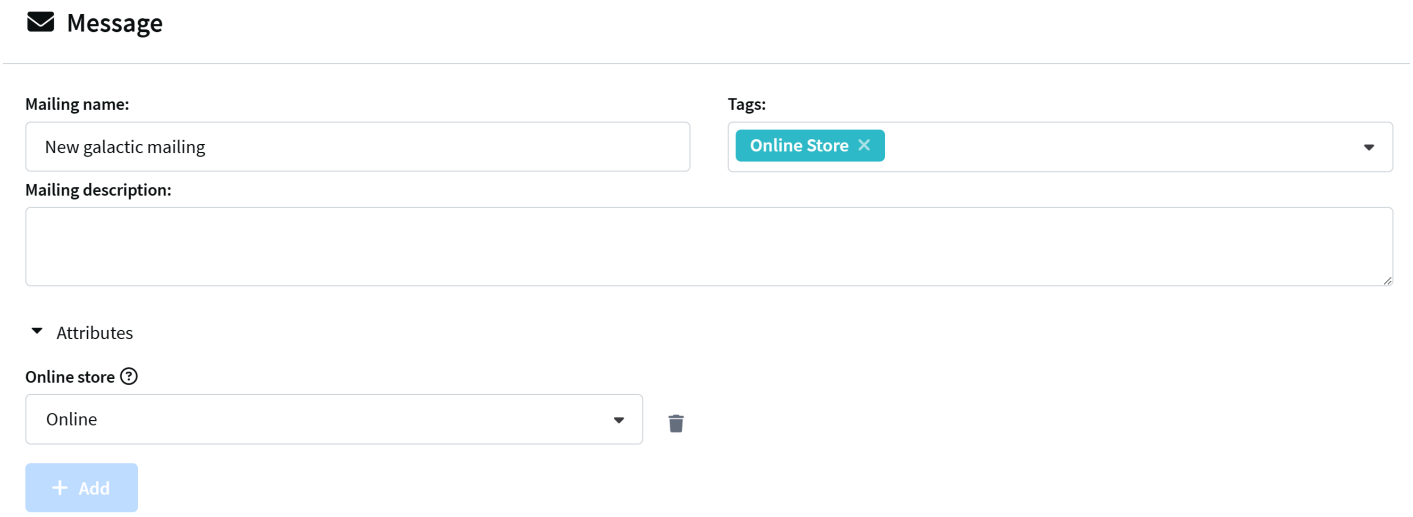
TIP

The Placement node allows you to launch a placement mailing in a scenario. Setting up such a mailing is described in [this article](#).

Read more

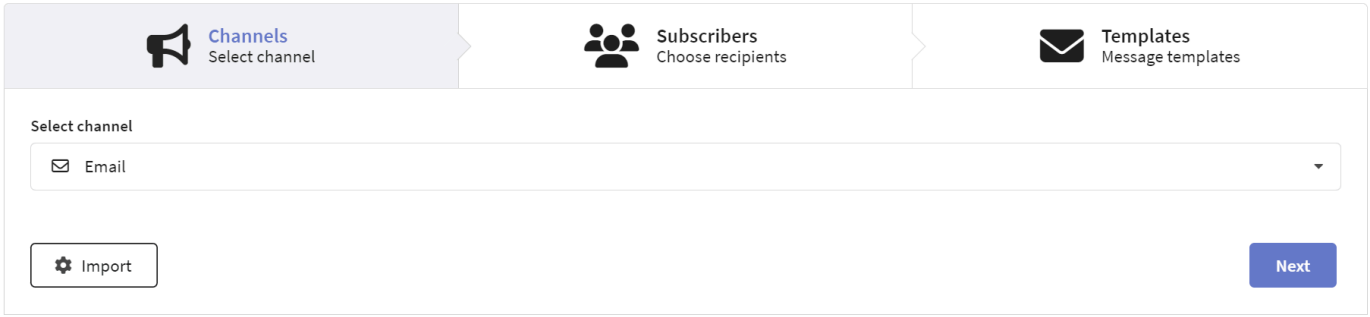
General settings

Enter a name for the new mailing, add a description if necessary. Here you can also set custom tags and set highest priority sending:

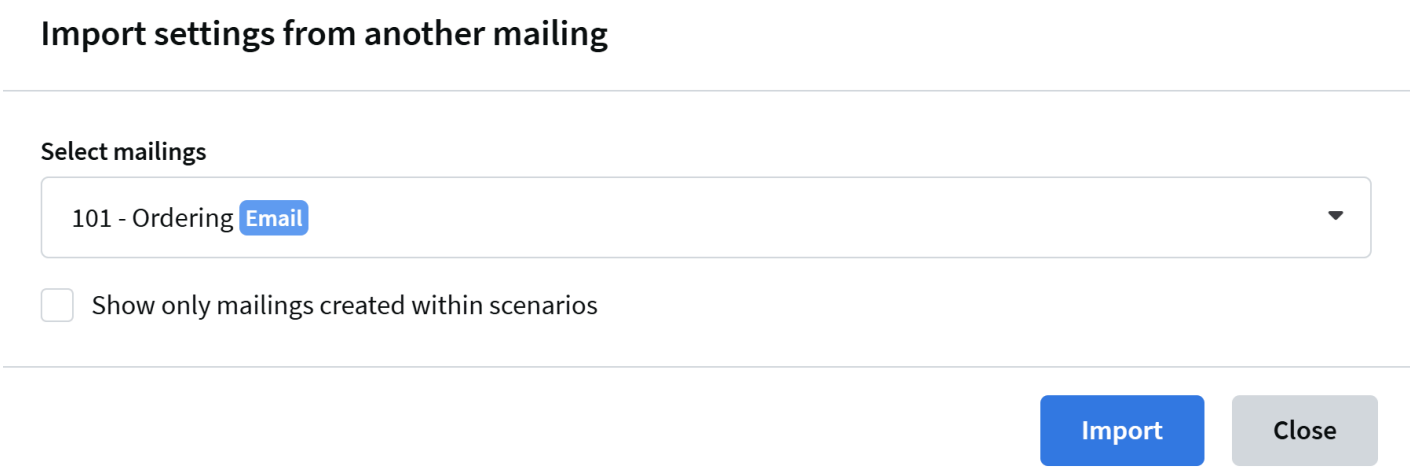


Mailing settings

On the first settings tab, select a communication channel: each mailing can only be sent via one channel.

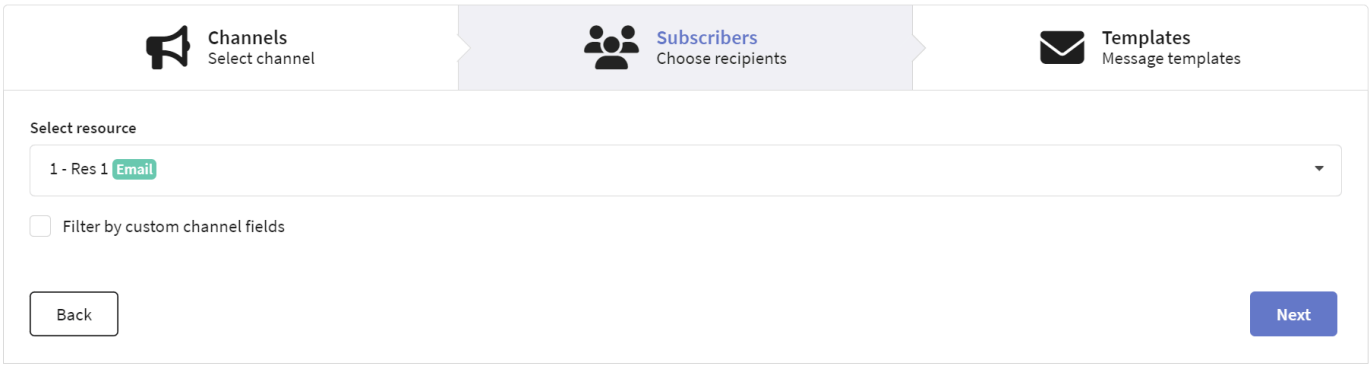


On the same tab you can import an existing mailing into the node. This can be a single mailing or a mailing included in other scenarios:



Subscribers

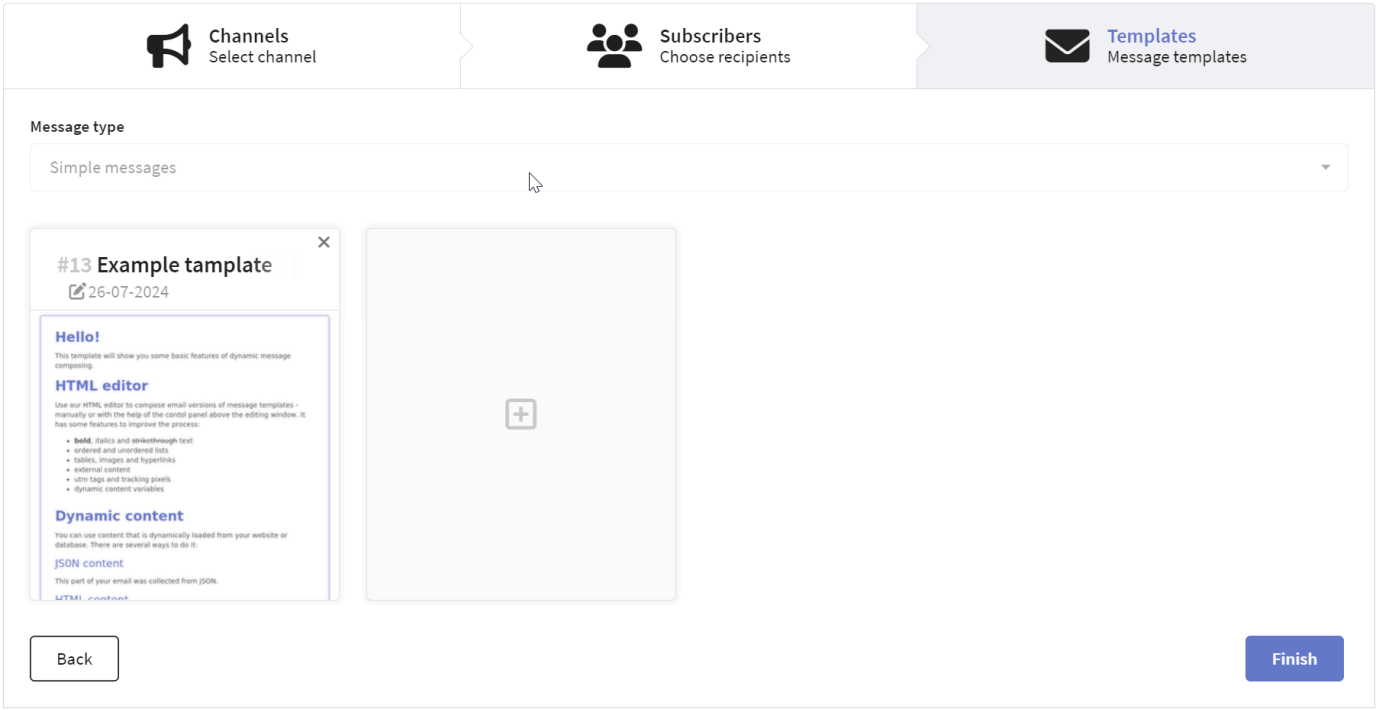
The trigger in the chain will use the contact details of subscribers of one of your resources. Choose a resource that your target audience subscribes to and proceed to the next step:



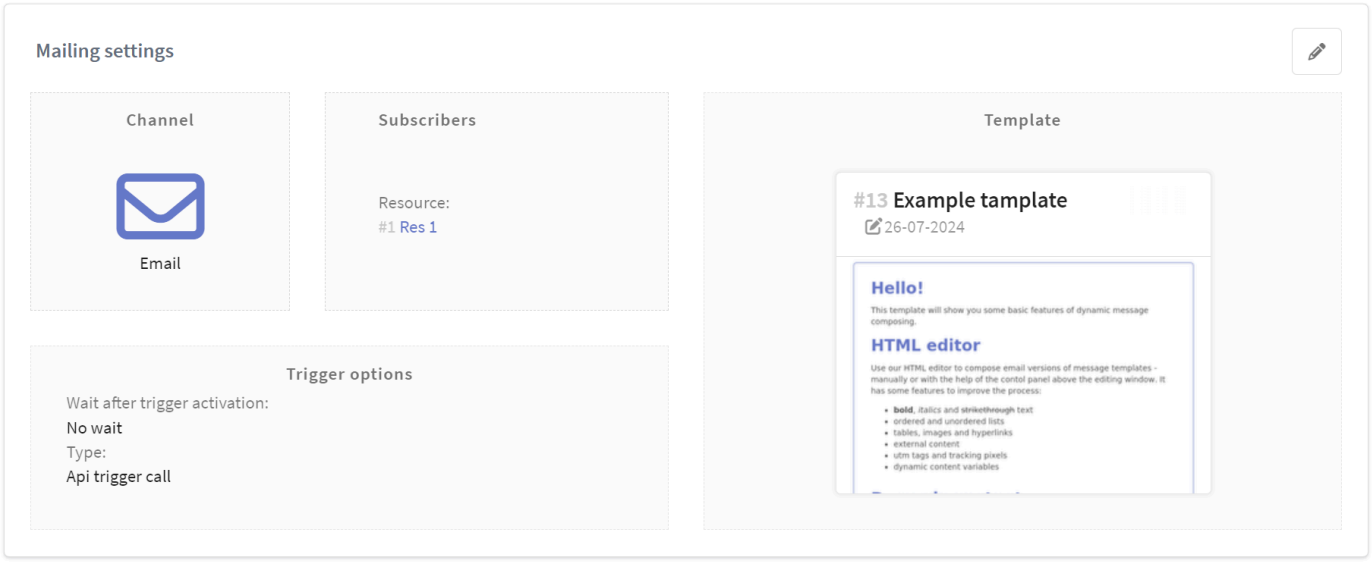
Additionally, you can filter by custom channel fields. To create a filter by fields, select the standard or optional subscription field. In the "Selection condition" field, select a logical operator and, if necessary, specify a value.

Content

On the third tab, select Message Template that you want to send to subscribers. It is possible to select several templates at once (up to 16 templates in one mailing). In this case they will be sent in random order. Just like when creating a mailing, you can open a preview of the desired template and edit it, if necessary:

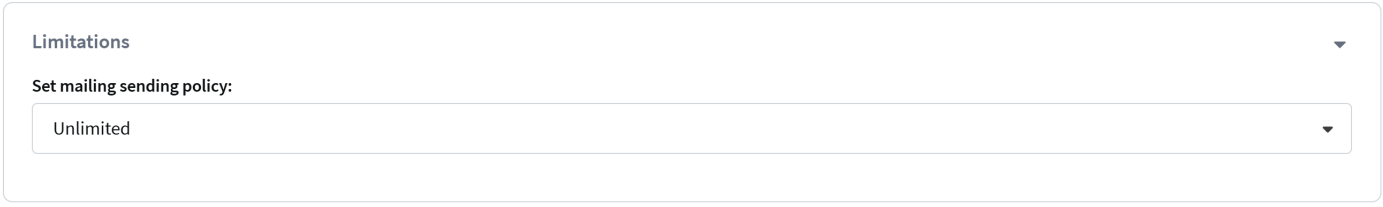


After selecting the template, click the “Finish” button. The mailing settings will be displayed in one tab, and you can make the necessary changes at any time by clicking the edit button:



Mailing Limitations

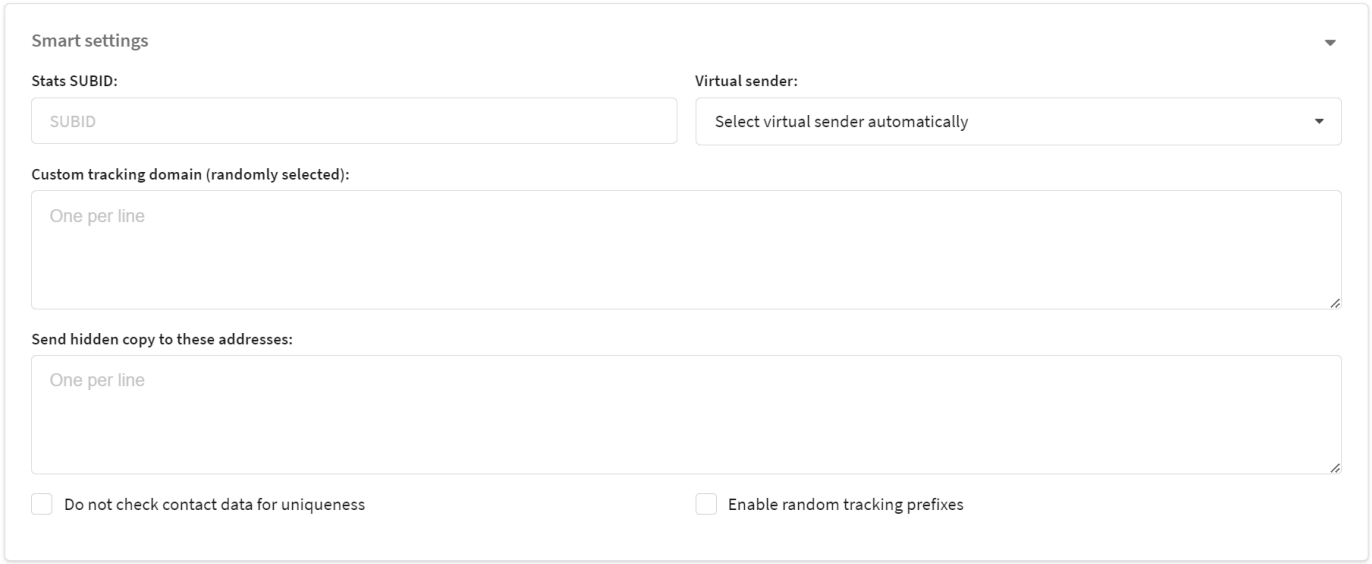
In this opening block, you can set send policy of the mailing. By default, the policy of the selected resource or the global account policy is applied:



Smart settings

In smart settings, you can additionally customize the mailing:

- **SUBID statistics**
- **Selecting a virtual sender**
- **Custom domains for event tracking (in addition to the one configured in the resource)**
- **List of addresses for hidden copy to sent e-mails**
- **Other options depending on the platform configuration**



CAUTION

If there is an error when sending a mailing, the profile will continue to follow the scenario.

Actions

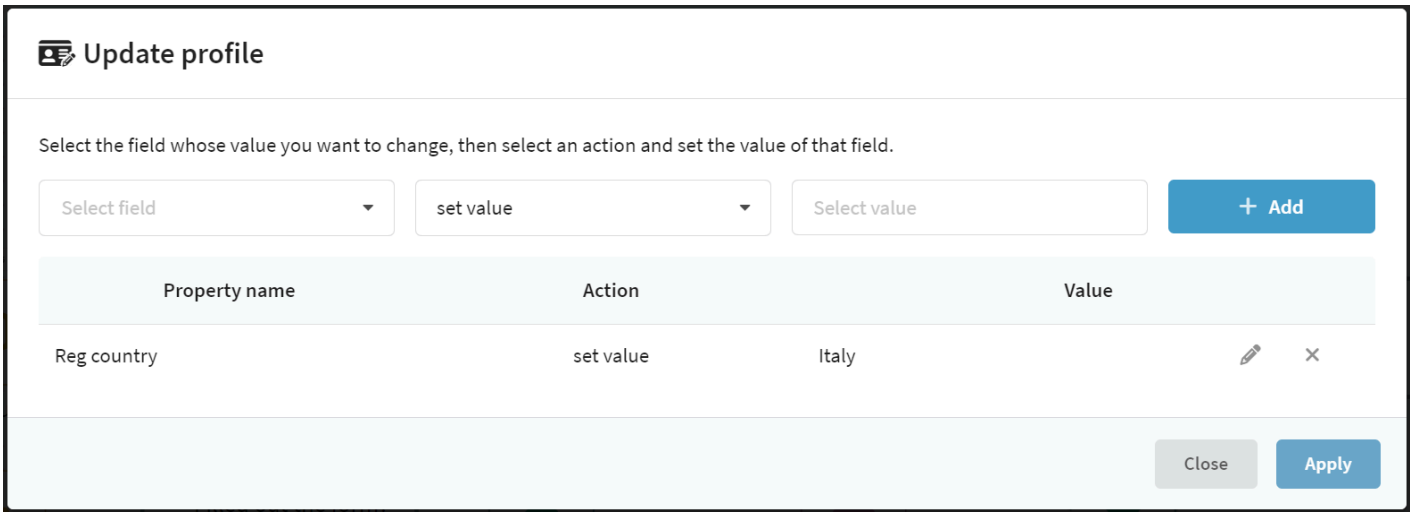
Update profile

	Updates the data of selected fields of the subscriber profile or deletes values in these fields. **Related actions are available for different field types. The node can add tags, change field values with selectable options and numeric fields.**

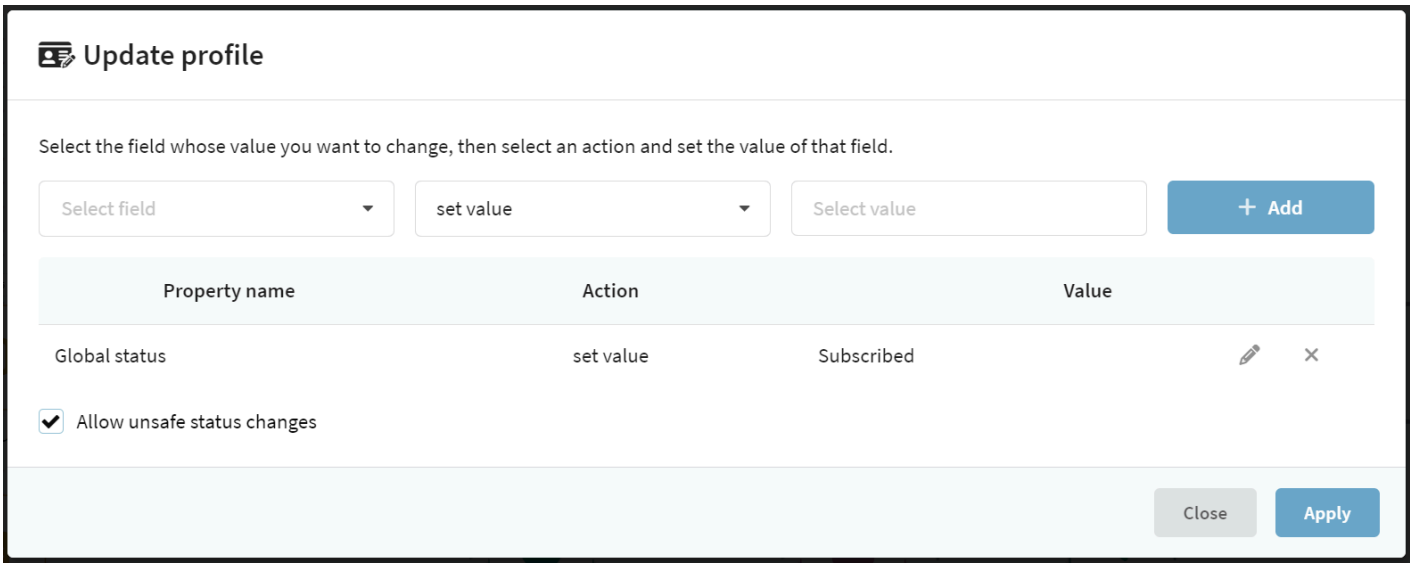
Read more

From the drop-down list, select the field you want to update. Then select the action and set the value of that field. Press + Add. The added field can be edited by clicking on the pencil icon, as well as deleted from the node.

Action	Field type	Description
Set value	**All**	**The new value of the field is specified manually. For fields of the List (Enum) type, the value is changed to one of those specified in the list, for example, the selected service package. For fields of Boolean type the value is changed to Yes or No.**
Remove value	**All**	**Deletes the value from the selected field.**
Take the value from the field	**All**	**The value is taken from the `apicontent` field. It is used together with "API call" node, which is able to save the request result into `apicontent` object.**
Increase by	**Integer**	**Increases or decreases the value. To decrease a value, you must enter a negative number.**
Add element	**Tags**	**Write another tag in the field.**
Remove element	**Tags**	**Remove one of the existing tags from the field.**



With this node you can also change the global status of a profile. If you allow unsafe status changes, the node will update profiles with statuses other than "Subscribed".



INFO

When referring to the sequence number of an array element from `apicontent`, the % sign must be used. Counting in such arrays starts from zero. For example, we have the following object with an array:

```
{
    "results": [
        {
            "objectId": "12asd34",
            "name": "Company 1"
        }
    ]
}
```

In order to get the name of the first organization, we need to use the following query `res.results.%0.name`, where `res` is the field to store the response from the API Request node. Please note that queries do not support JSONPath.

Example of correct query structure: `resp.data.data.%0.profile._region` Example of incorrect query structure: `{jsonpath(resp.data "$.data[0].profile._region" "item")}`

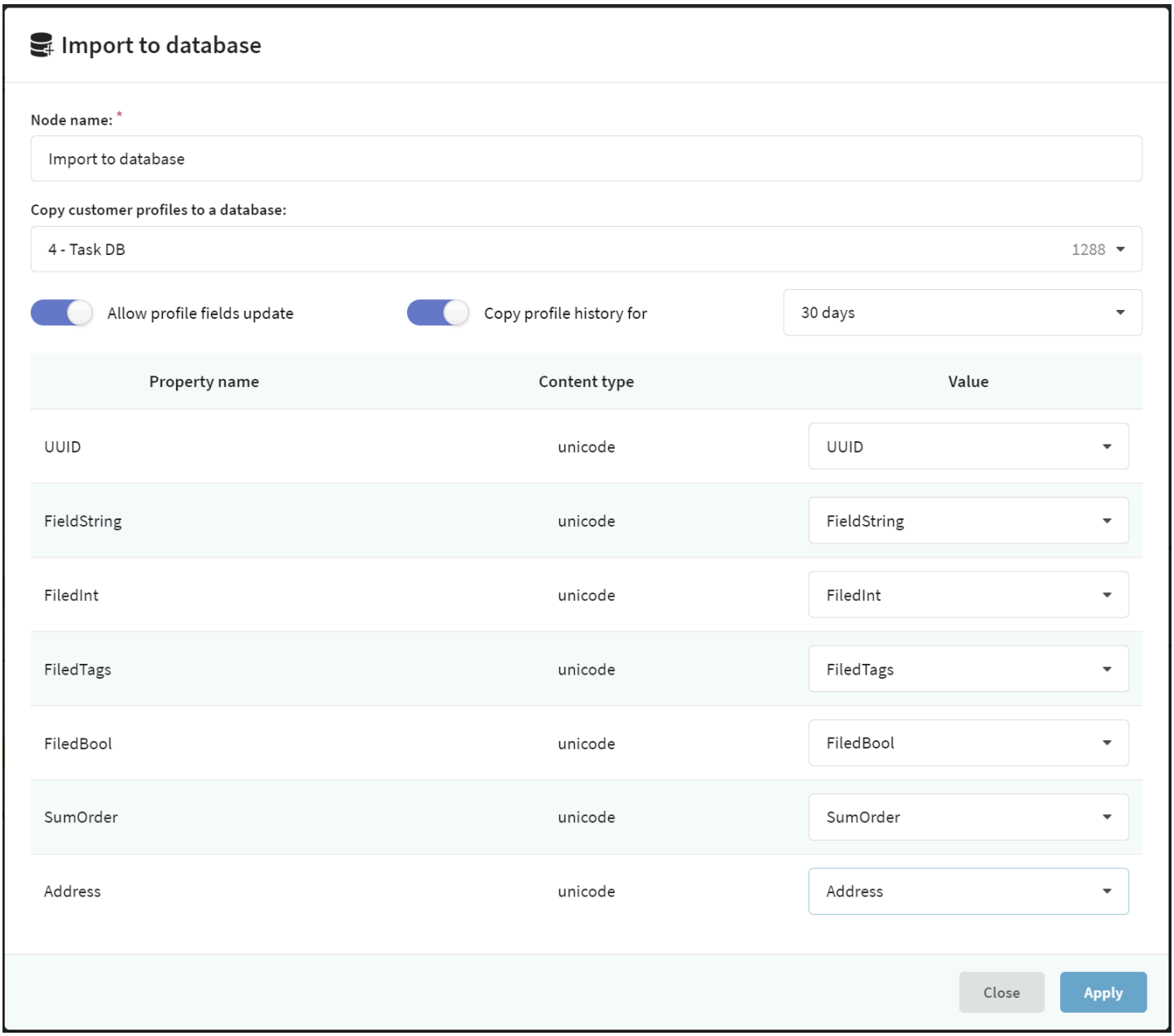
Import to database

	Copies a subscriber's profile from the profile database for which the scenario was created to another one. You can also copy profile history for a certain period and transfer custom profile fields.

Read more

Enter the name of the scenario node and select the profile database to which you want to copy the customer profile. If you want to update the data of a profile that already exists in the target database - activate the "Update subscriber fields if it already exists option. If there are additional fields in the source database, their data can be written to fields of the appropriate type of the target database. To do this, customize the field matching.

You can also transfer the profile history from the source database to the target database. To do this, use the Copy profile history for" option and select the period of data you want to transfer. Profile history includes mailing actions (sends, opens, clicks, etc.), subscription events, imports, pixels, promotion code assignments, and participation in static segments.



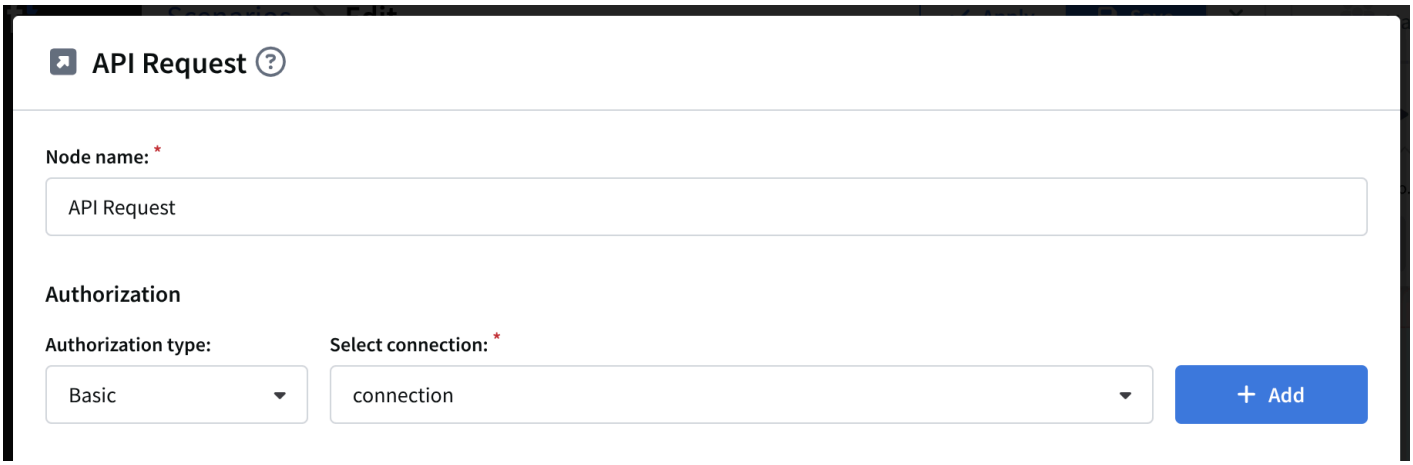
API request

	Send a GET or POST request to another system. You can add parameters to the request with static data or taken from a subscriber. Used for integration with other systems.

Read more

Enter the name of the node.

Select the required authorization type. This is necessary in order to transfer profile data to external services that use protected authorization. You can first link the external service to the Altcraft platform via the connection settings, or proceed to create it directly from this menu:



Select type of request to the external system. In the URL field enter the address of the service receiving the data:

The request URL also uses variables. Click `</>` or enter a value manually. Example of using a promo code in the request URL:

```
http://www.example.org/api/?Promocode={loyalty.black_friday.promocode}
```

The user can limit the number of profiles passing through this node within a specific time period. This is necessary to ensure the stable operation of the scenario without overloads. To do this, enable the corresponding toggle, specify the number of profiles, and set the time interval (minute, second, or hour):

▸ ****Rate Limiter Mechanics****

Static segment update

	Adds a customer profile to the static segment of the audience or removes it from the segment.

Read more

In the node settings, select add or delete and the available static segment:

Subscriptions

	Subscribes a client to a specific resource or updates existing subscription data. Profile data or subscription data for another resource can be used as a subscription source.

Read more

The system uses two types of subscription channels:

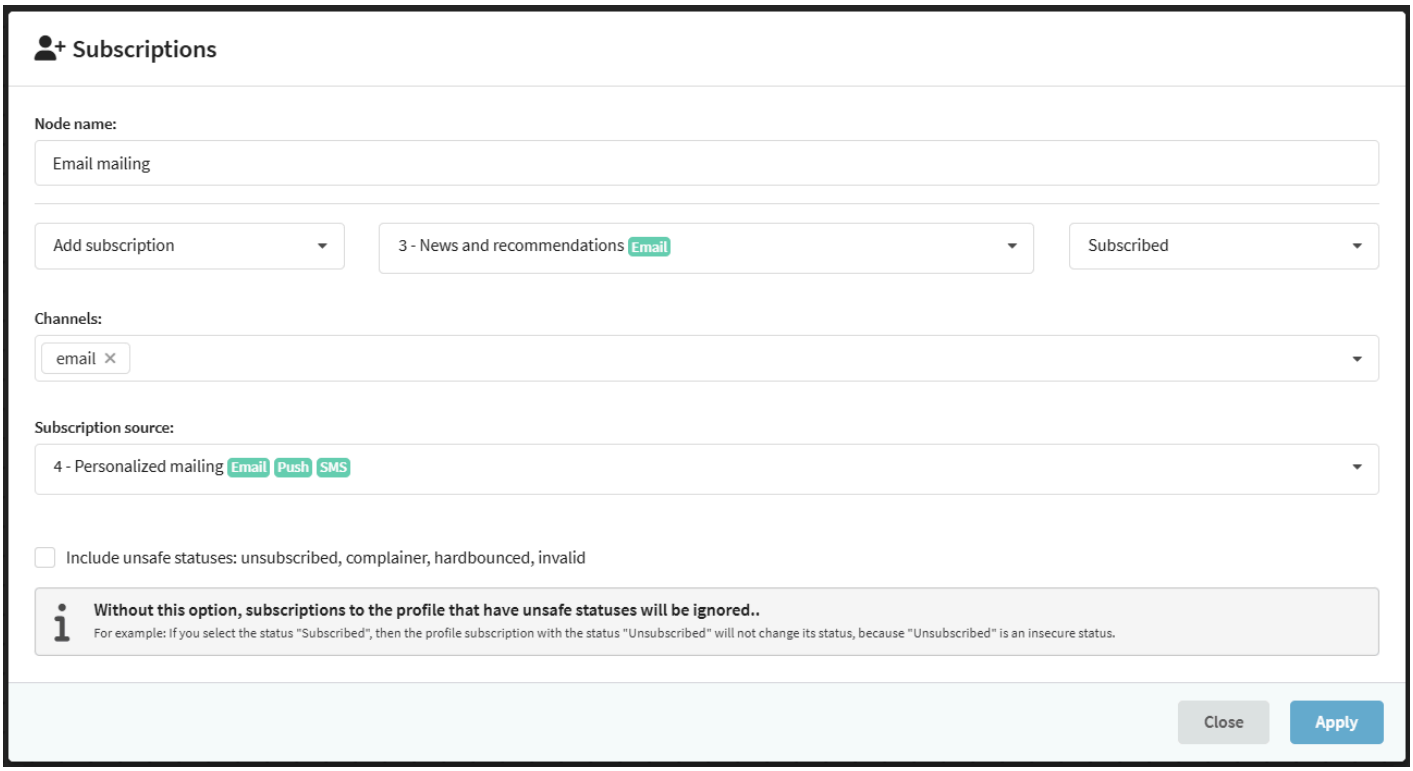
- ****Indirect channel** — sends a message to an unspecified number of users. It is used only for sending to custom channels and Telegram Group-channel.**
- ****Direct channel** — sends a personalized message to a specific recipient.**

Add and update subscription

- **A subscription can be **added** if the client is not already subscribed to the resource.**
- **You can **update subscription**. For example, when you are confirming a subscription or while using custom unsubscribe scenarios.**

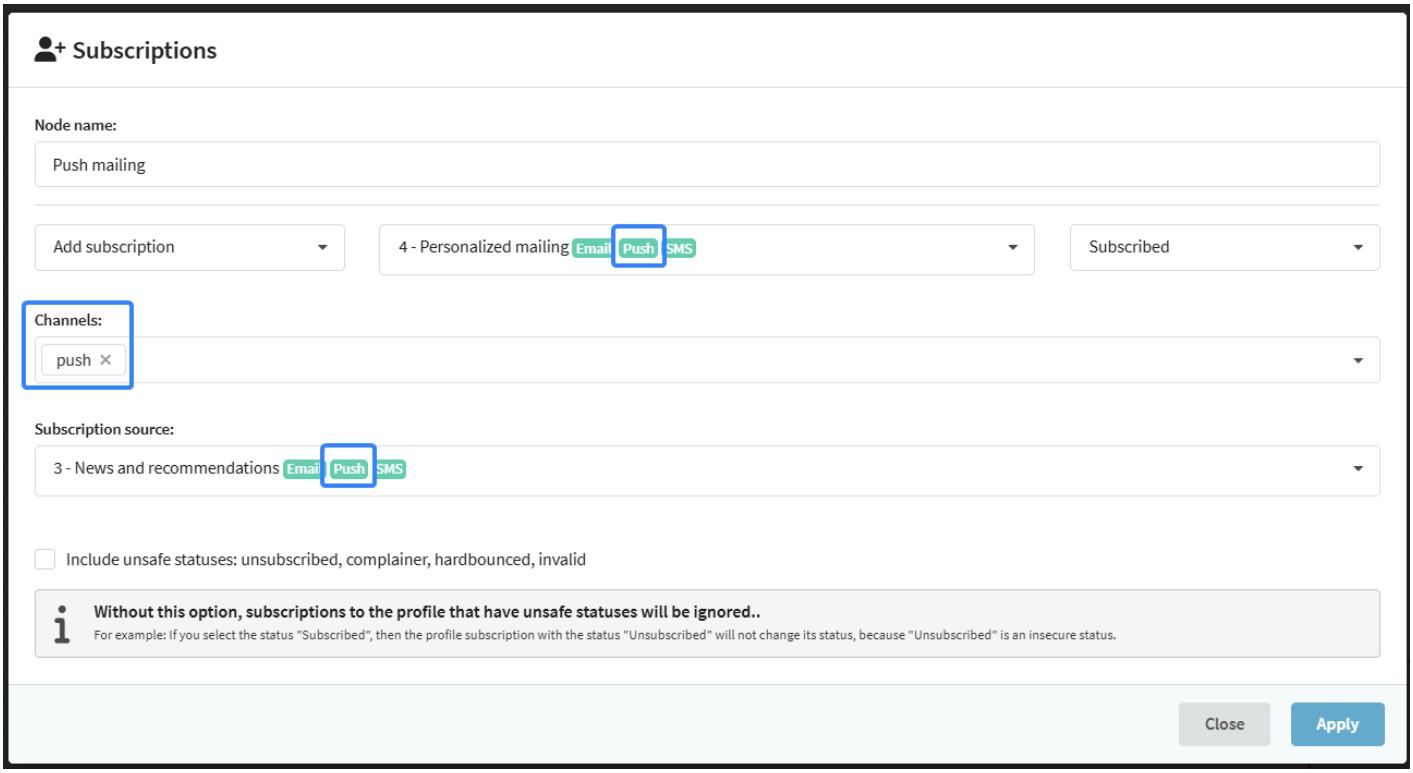
How to add or update subscription to a resource:

1. **Enter node name.**
2. **Click Add subscription or Update.**
3. **Specify the resource from the drop-down list.**
4. **Change the subscription status if it is required.**
5. **Select the available channels.**
6. **Use profile data (e.g. Email, Phone) or subscription data from another resource as the subscription source.**



TIP

You can only subscribe to Push notifications if you select a resource for which you have already configured Push as the subscription source.



Promocode

	Assigns a promocode from the selected loyalty program to the customer profile. The promocode can then be sent to the customer via one of the elements in the "Channels" category or used to calculate personalized offers.

Read more

Enter the name of the scenario node and select the required loyalty program. The program must generate a sufficient number of promocodes or the option of creating promocodes by scenario must be active.

CAUTION

If the profile already has a promocode for this loyalty program, the scenario will be fulfilled.

If a promocode cannot be assigned for one of the reasons below, the scenario will end on that node:

- **promocode limit has been reached**
- **loyalty program is not active**
- **the loyalty program has no free promocodes, and its option "The scenario can create promocodes if there are not enough of them" is disabled**

Goal registration

	Records in the client profile the achievement of a certain goal and the associated cost of the action. Can be used for scoring subscribers, LTV and RFM segmentation, in reports on conversion and profit.

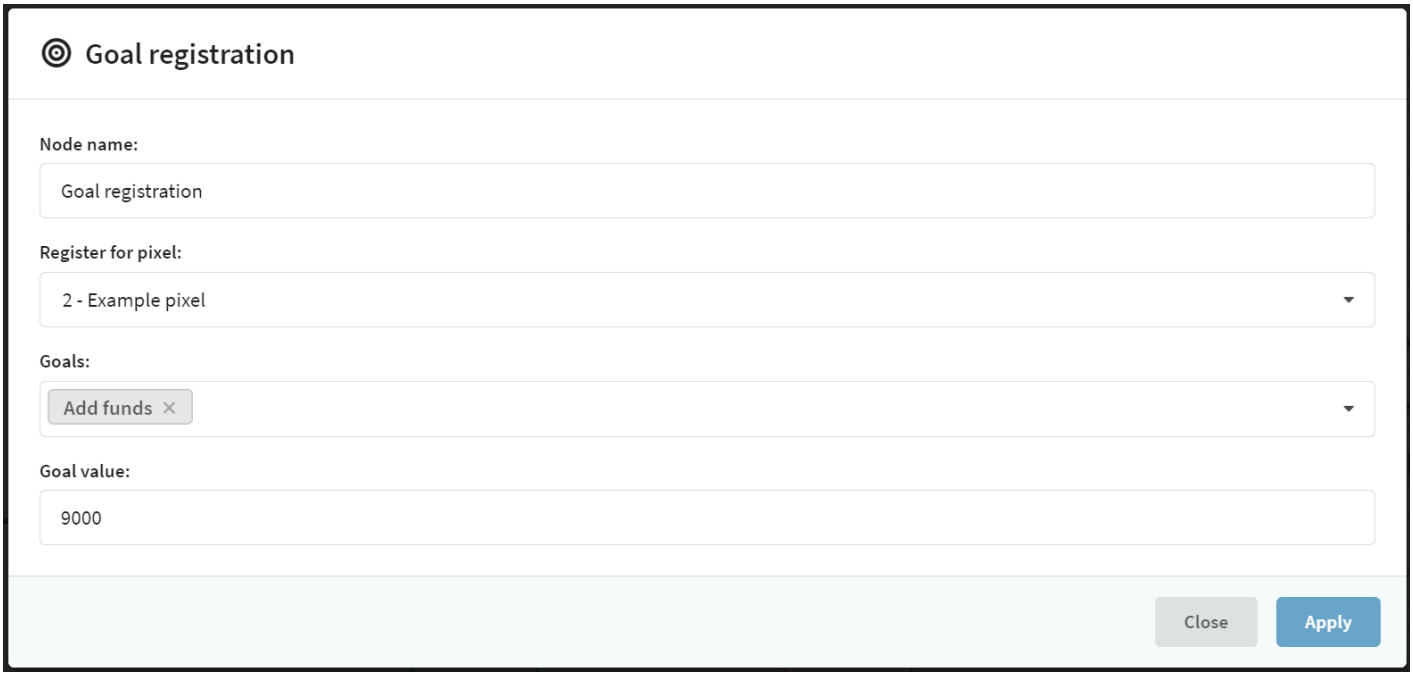
Read more

Enter name of node and pixel to which the target will be bound. Registration of identical targets for different pixels can be used in affiliate programs or for A/B testing of the interface of your site or other resources.

Below, select existing targets or add new ones. Specify the value to be transferred for the targets.

TIP

The value for ALL targets will be written as a single value. If multiple targets with different values are to be passed, add several such elements - one after the other, with different targets.



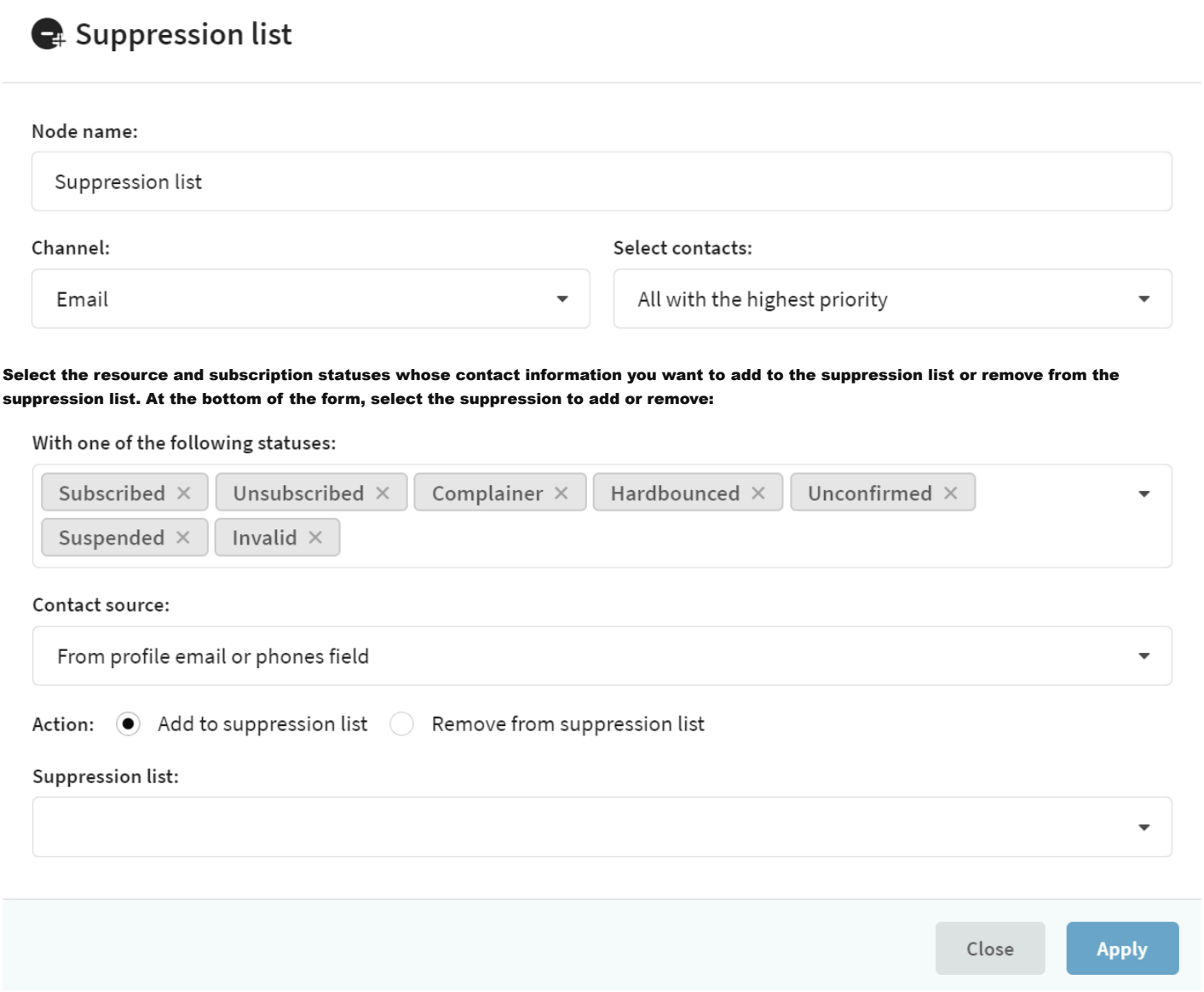
If necessary, you can pass variables related to pixel trigger data inside templates. You can learn more about these variables here.

Suppression list

	Adds certain client contact details (email address and phone number) to the stop list of the platform. Communication with the customer using this data will be blocked. Or vice versa, removes contact details from the stop list, restoring the ability to communicate with the client.

Read more

Enter the name of the node to be displayed in the scenario. Next, select which subscriptions contact information you want to block or restore. Select the communication channel and priority of subscriptions to change:

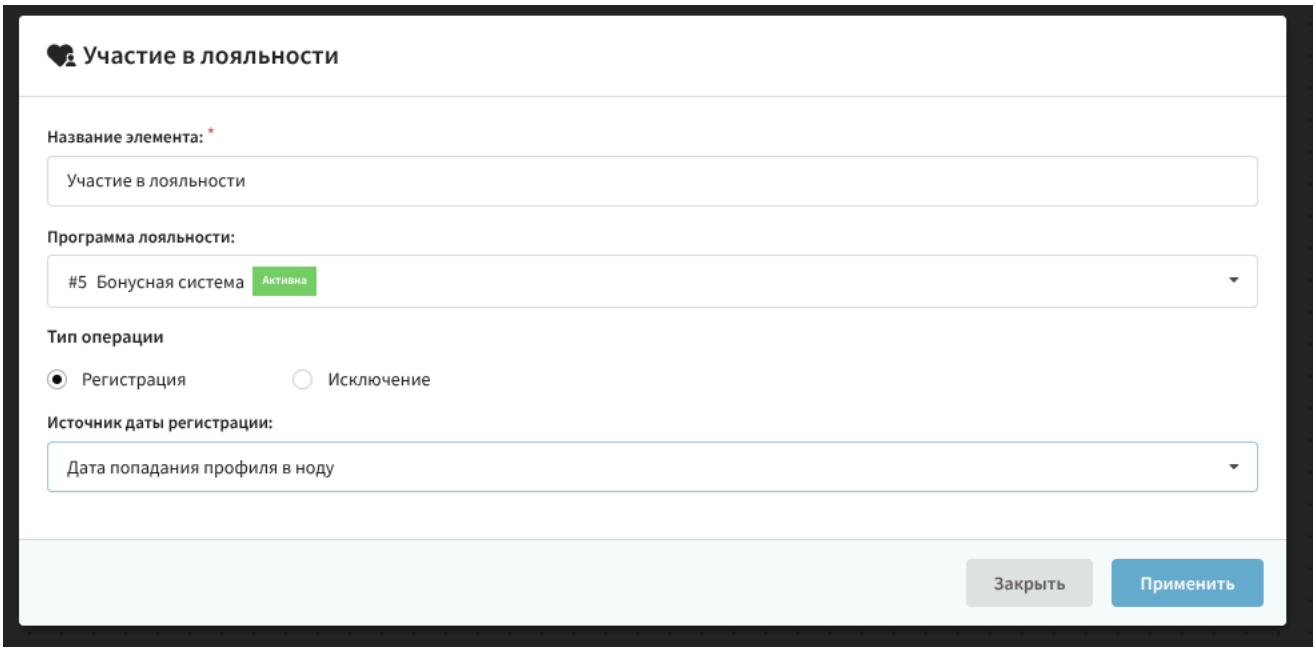


Select the resource and subscription statuses whose contact information you want to add to the suppression list or remove from the suppression list. At the bottom of the form, select the suppression to add or remove:

Loyalty participation

	Registers or removes a user from a loyalty program.

Read more



To configure the node, select a pre-created loyalty program and the operation to be performed on the profile — registration or removal from the loyalty program.

When registering a profile, you can specify the source of the registration date in the loyalty program:

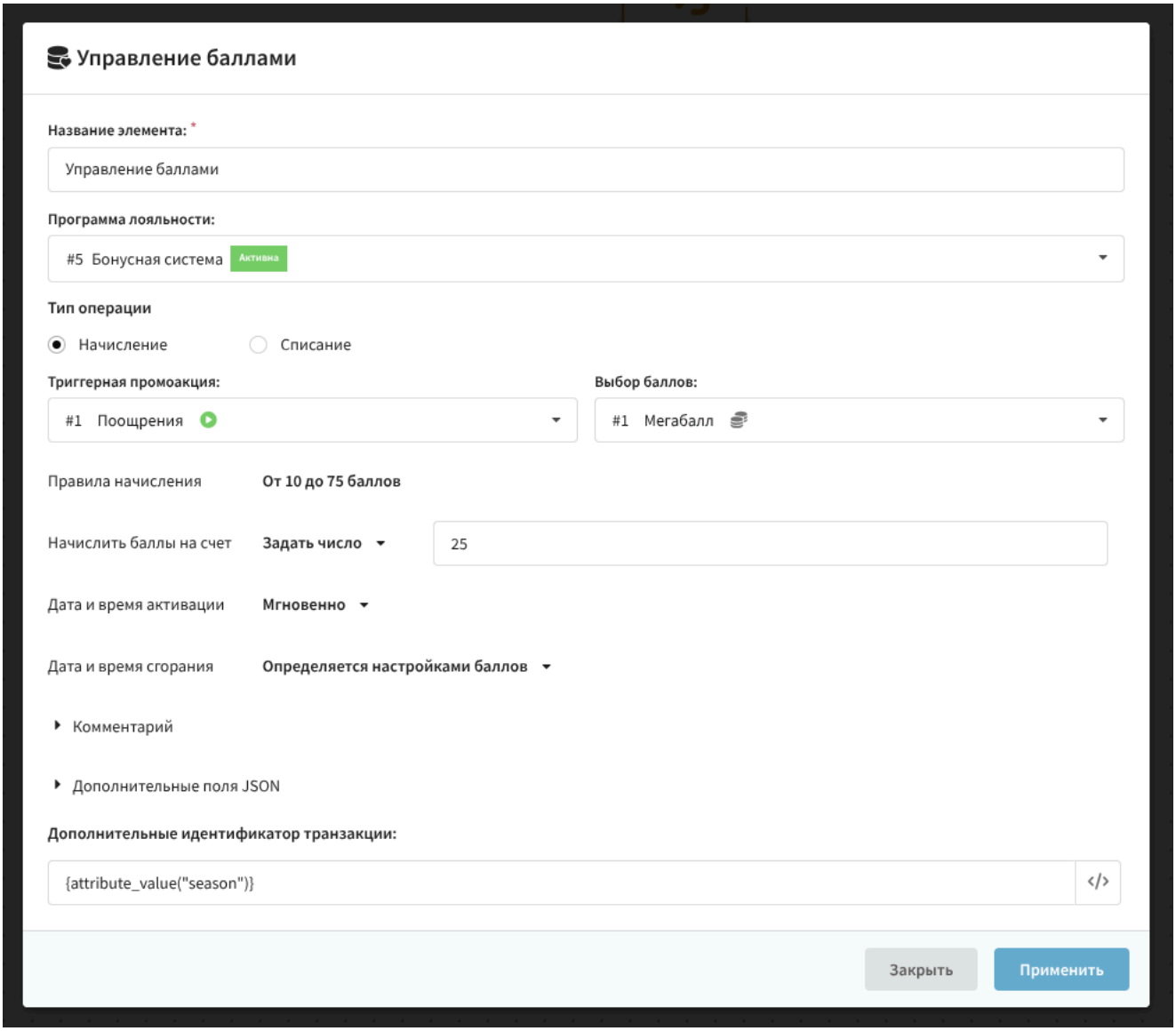
- **Profile entry into the node date — the time when the profile entered this scenario node;**
- **Date from profile field — the registration time is taken from the selected profile field with a date value;**
- **API content — the registration time is taken from the specified API field.**

For the last two options, two ways of handling invalid data are available: using the profile entry into the node date or registering an error.

Points management

	Changes the number of points on a profile in a loyalty program.

Read more



To configure the node, select a pre-created loyalty program and the operation type — points accrual or points redemption.

Then specify the promotion under which the accrual is performed and the points type. The number of points can be set as a fixed value, taken from a profile field, or passed via API content.

Additionally, you can specify:

- **Activation date and time (immediately, from a profile field, or from API content);**
- **Expiration date and time (according to points settings, from a profile field, or from API content);**
- **Transaction comment;**
- **Additional JSON fields;**
- **Additional transaction identifier.**

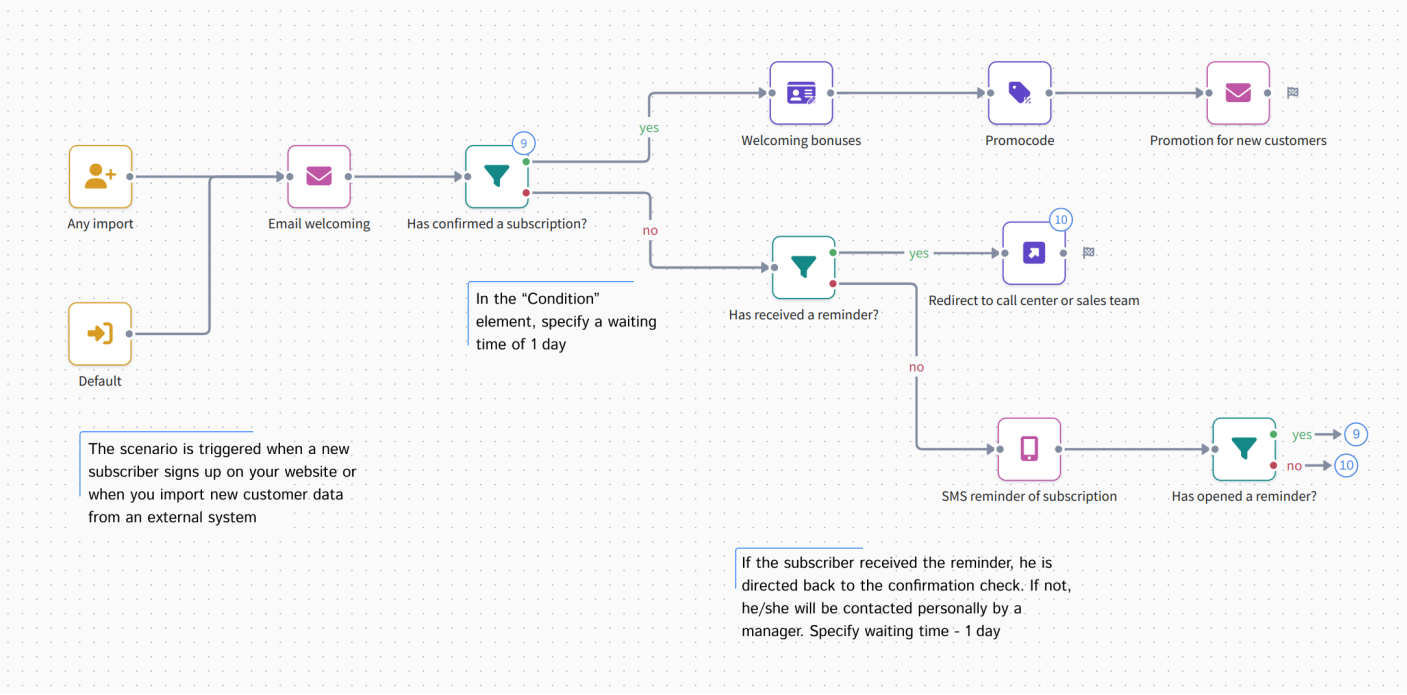
The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia. ** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.

Classic marketing scenarios

Welcome workflow

Welcome workflows can start engaging your customers right after registration. A s displayed below will help you:

- **automate opt-in process**
- **remind visitors of a dropped registration**
- **assign welcome bonuses and promotion codes**
- **redirect customers to your call centre or sales team for a direct call**



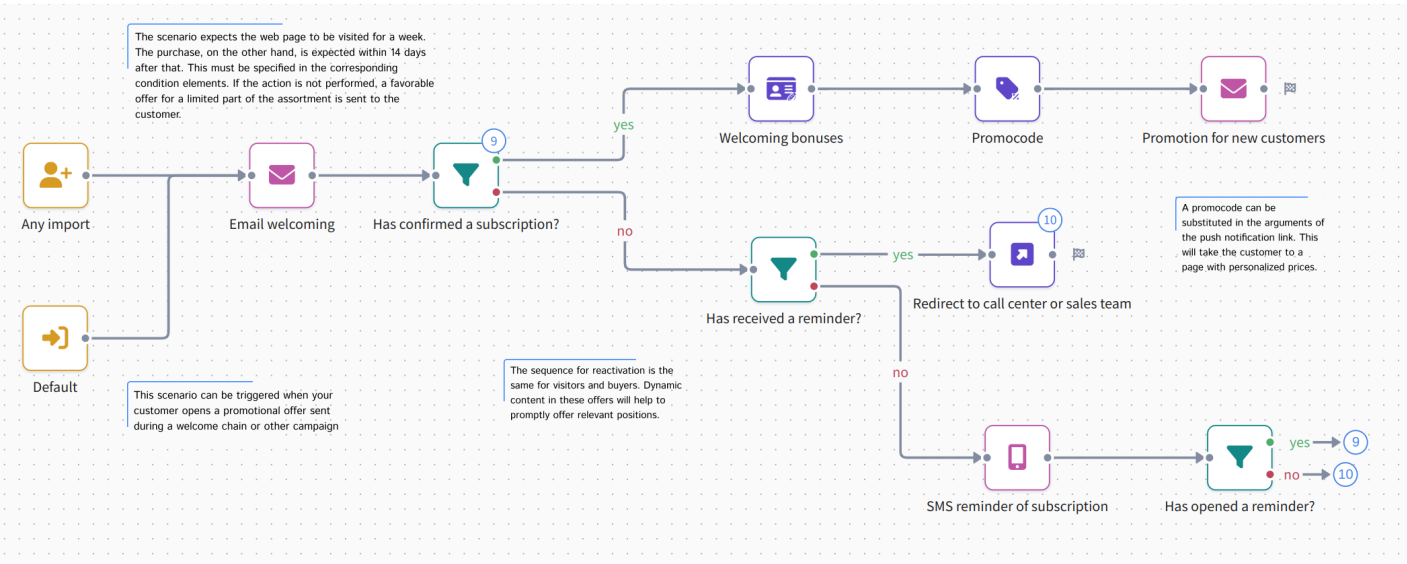
TIP

You will find a guide to creating a welcome scenario on this page.

On-boarding: first purchase

On-boarding workflows will help you engage customers from the moment of the first promotion they receive. For example, after a mass mailing about new product line you can encourage customer activity by a scenario similar to this one:

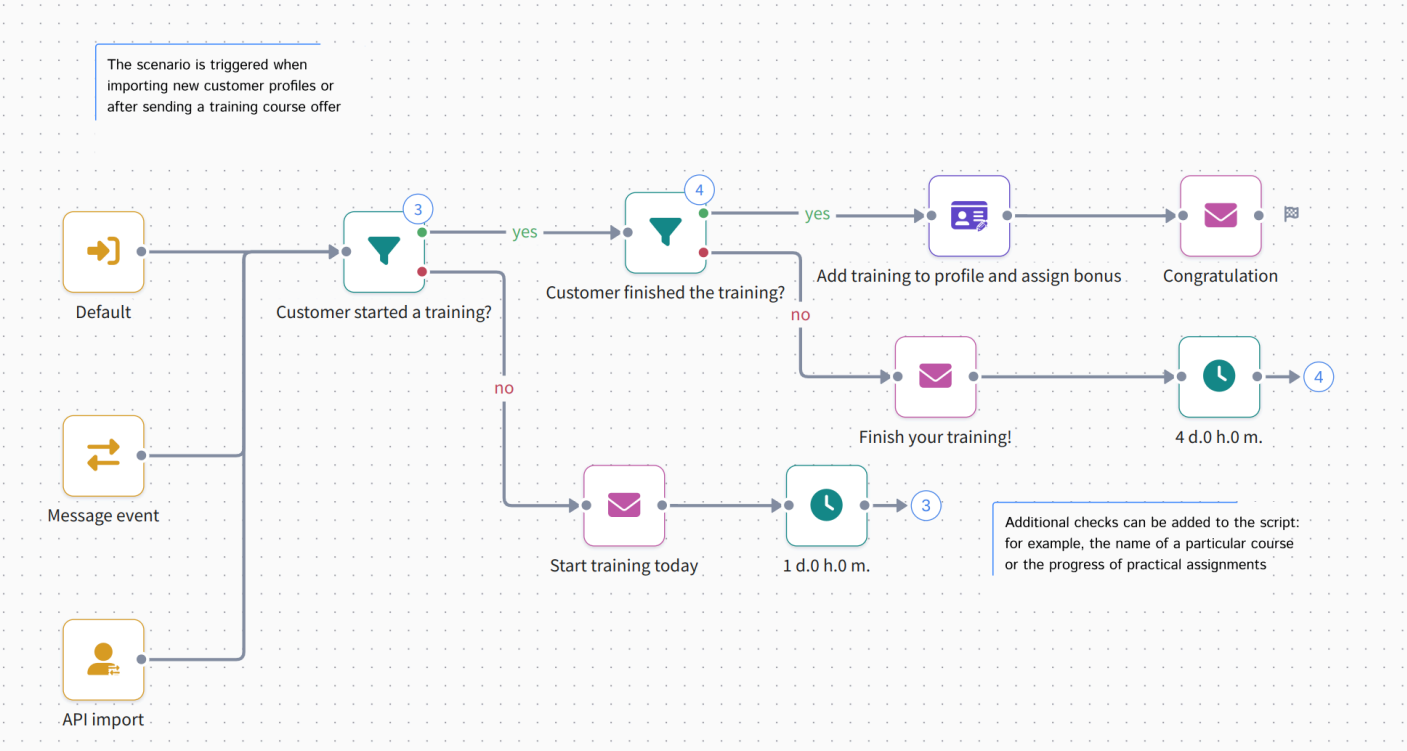
- **scenario launches when a customer interacts with your email**
- **reactivates customers who do not pay attention at first**
- **reactivates those who open store page and leave with no purchases**
- **assigns limited stock [promotion codes](/en/user-guide/loyalty-programs/) (for tripwires e.g.)**



On-boarding: customer education

One of the best ways to gather a highly loyal audience is to make them educated in your field of business. This becomes crucial when speaking of financial sector where customers' trust is the key to success. And here you come as a tutor figure — not mere seller.

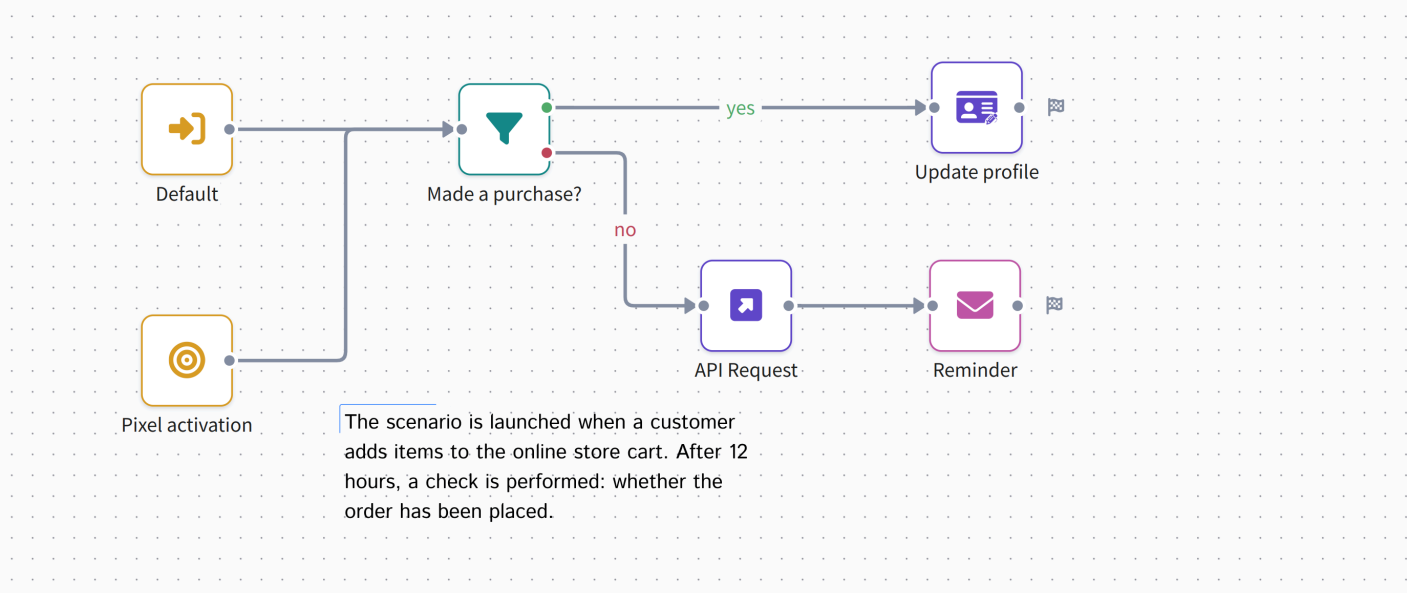
Customer learning can be automated and made interactive with Altcraft Marketing scenarios, that will check customers' progress and assign relevant trainings:



Abandoned shopping cart

If your potential customer is interested in your offer as much as adding products to a shopping cart — though not proceeding with checkout — you can get a second chance by reminding of the abandoned cart. If you can sacrifice a discount in exchange for customer retention — a scenario can automatically assign promotion codes for a limited stock of products.

API request is used to launch this scenario:

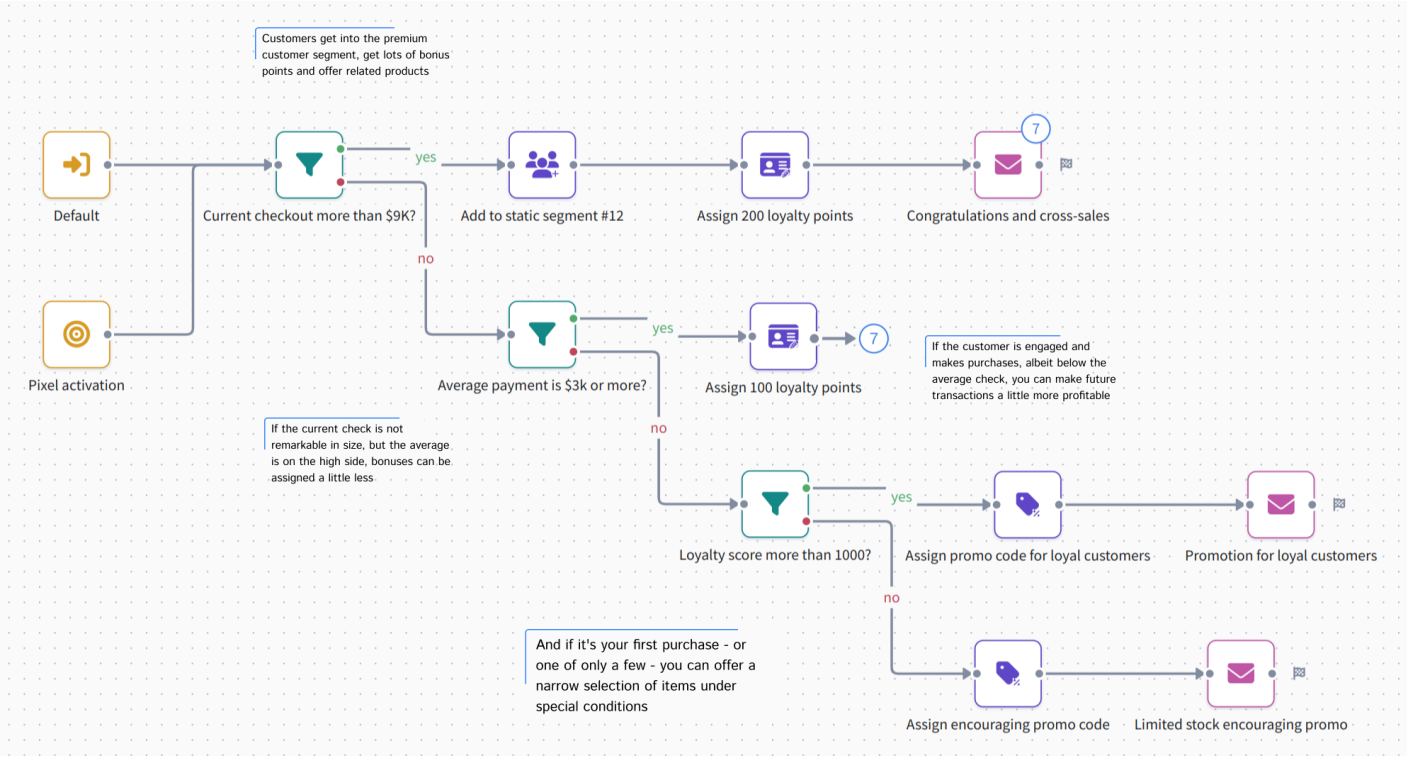


TIP

You can find an example of how to implement a shopping cart abandonment scenario in this article.

Customer loyalty management

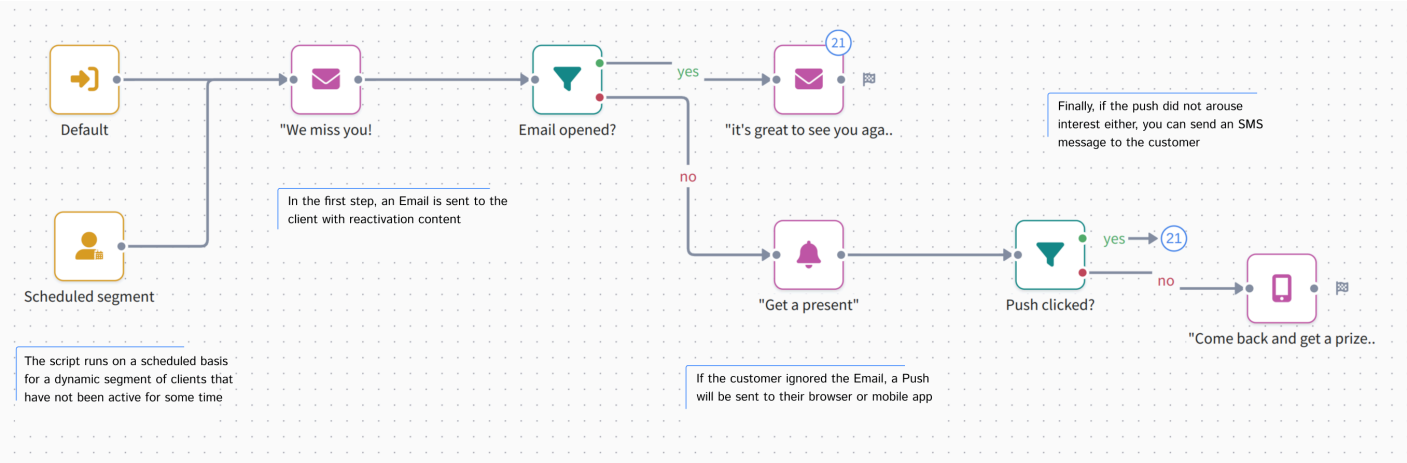
Customer loyalty is one of the key aspects of the modern market. It is a good way to implement gamification of your marketing processes. Compose a scenario that will ispect customers' loyalty score, RFM metrics (average payment) and order details. It will definitely nourish your customers' loyalty:



Omnichannel customer reactivation

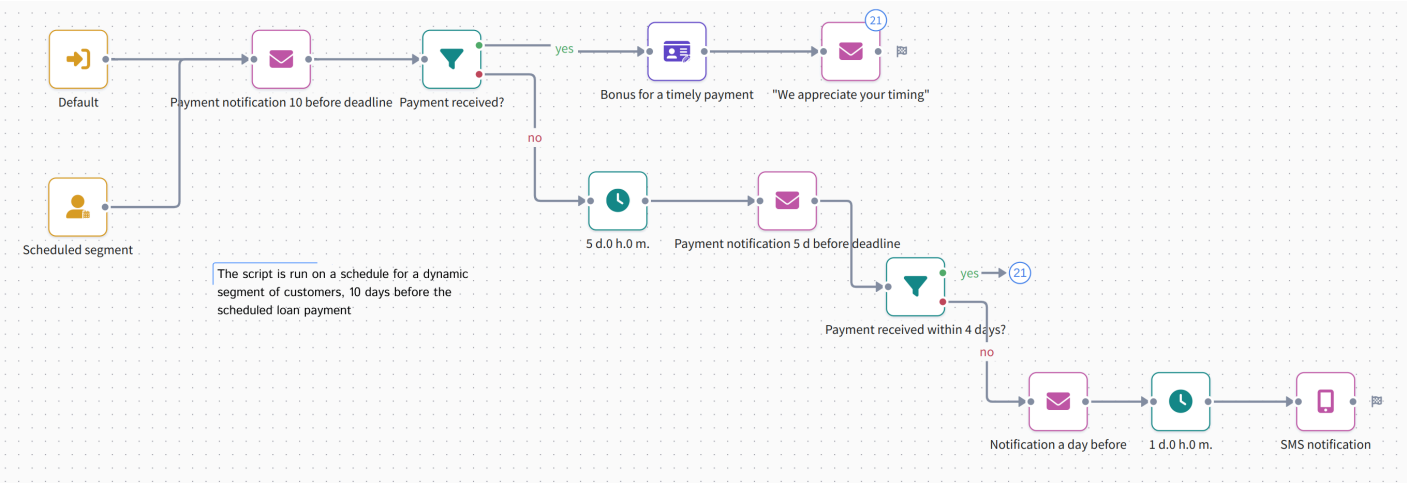
You can reactivate inactive customers automatically with the help of the scenario in this example. Scenario is launched periodically - same as a regular mailing. Each launch a dynamic segment will provide customers who have ignored your mailings and offers for a specific period of time.

The sequence itself is what is called Cascade mailing — messages are sent consequently until a customer reacts to one of them. You can use this type of omni-channel communication in a wide variety of cases:



Regular payment notification

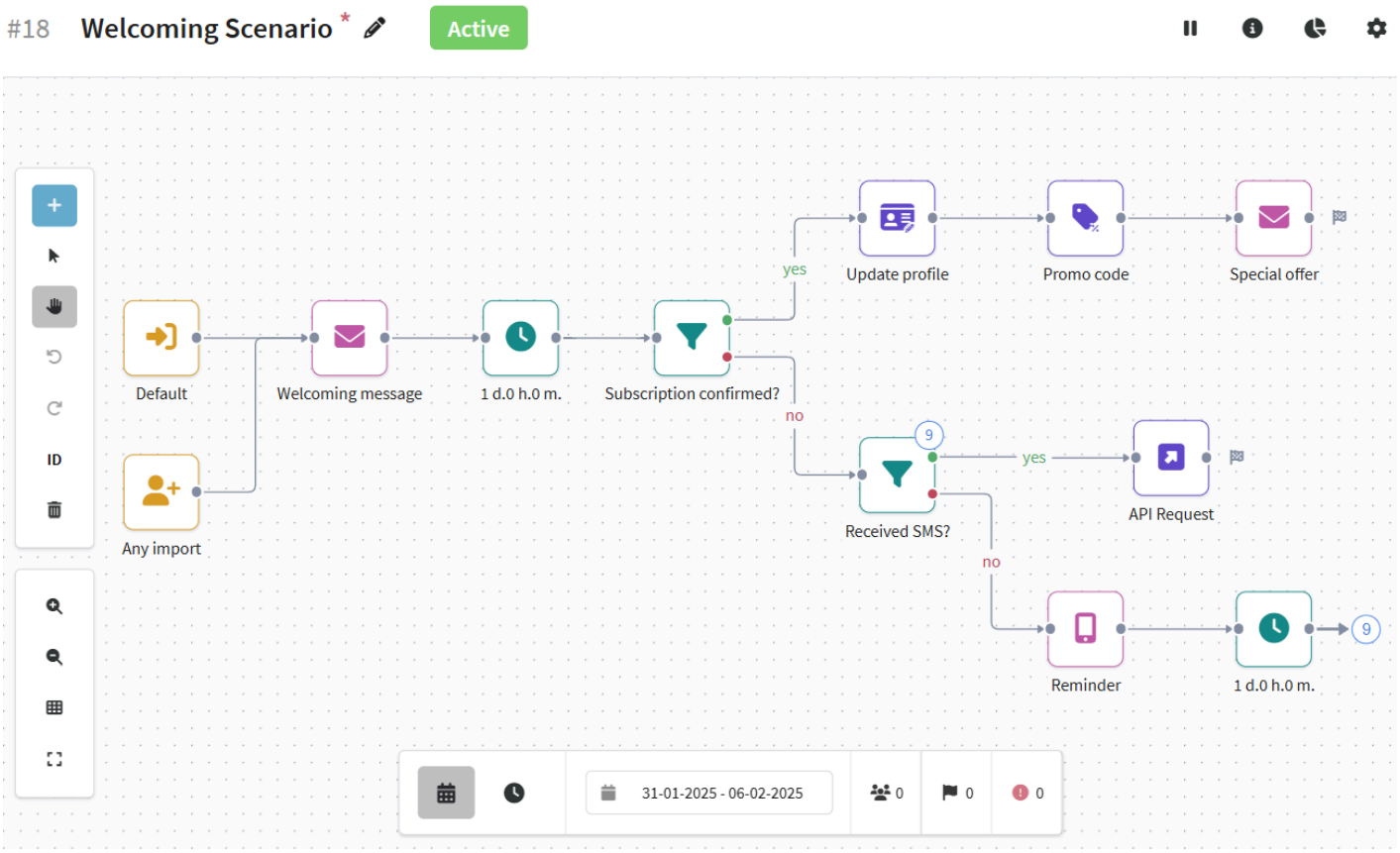
This scenario will help you automate omnichannel notification about regular payments, service period ending and similar workflows. A dynamic segment and schedule are used to launch a sequence of notification — via one or several channels. Before sending another notification scenario will check if the payment has been made:


Default
Payment notification 10 before deadline
Payment received?
yes
Bonus for a timely payment
21
"We appreciate your timing"
Scheduled segment
no
5 d.0 h.0 m.
Payment notification 5 d before deadline
The script is run on a schedule for a dynamic segment of customers, 10 days before the scheduled loan payment
Payment received within 4 days?
yes
21
no
Notification a day before
1 d.0 h.0 m.
SMS notification

Step-by-step welcome scenario guide

This guide will help you set up an automation scenario, managing interactions with new customers - from registration to first promotional mailings and purchases. We give you some basics on how to choose scenario nodes based on the situation at hand or resources used.

Welcome workflow is one of the classic marketing automation scenarios. You can implement it effectively with Altcraft Marketing.



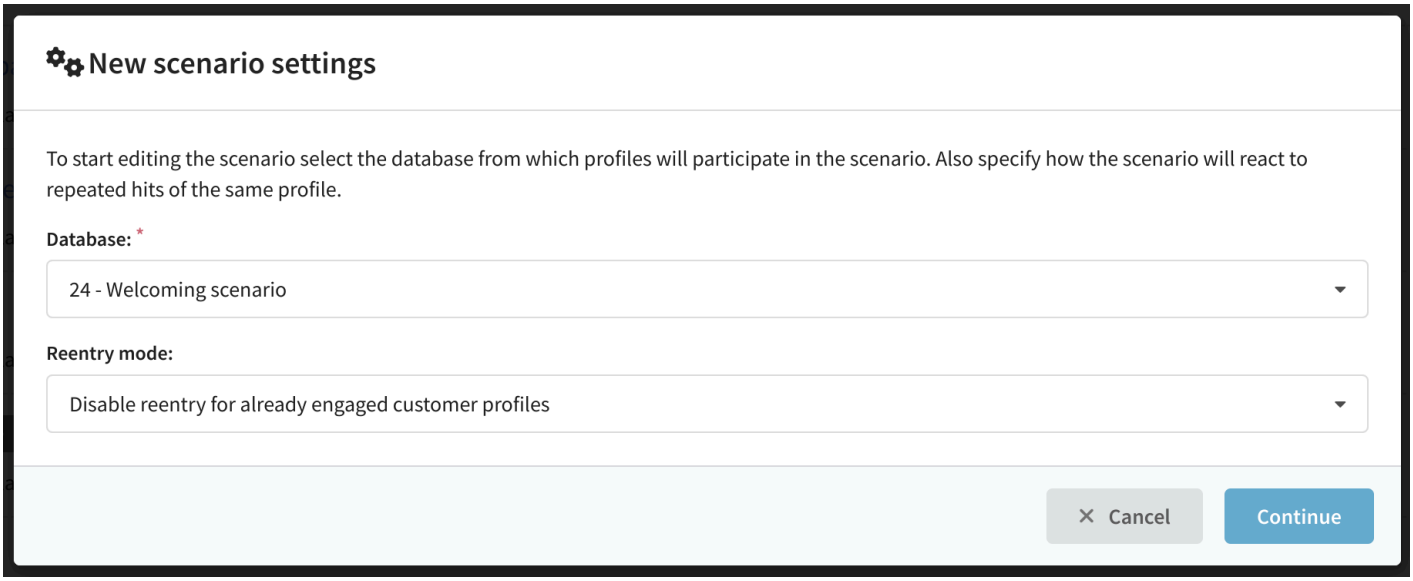
TIP

Automation scenario basics are described in this article.

This page will provide you with information about scenario steps - nodes.

Step 0: Create a new scenario

In the Scenarios section, create a new scenario by clicking the appropriate button at the top of the screen. Specify the profile database and disable re-entry for the profiles involved in the scenario, as the greeting is a one-time event:



Step 1: Configure customer capturing mode

Customers will get engaged in the scenario when their subscriber profiles are created within the platform. Thus, capturing event depends on how you import your customers' data into the platform:

- **Any profile import - launches any time a new customer profile is created.**
- **Manual profile import - launches when you create a customer profile manually**
- **API profile import - launches when you import a profile via platforms' web API from your website or integration middleware (ETL)**
- **Form import - launches when filling in form fields. To learn how to create a form in the platform, read in this article.**
- **Popup import - launches when filling out a form in a popup. You can learn more about creating and customizing popups in this article.**

There is also an element capture default profiles in any scenario. It is required to capture profiles added manually, via API and chained from another script.

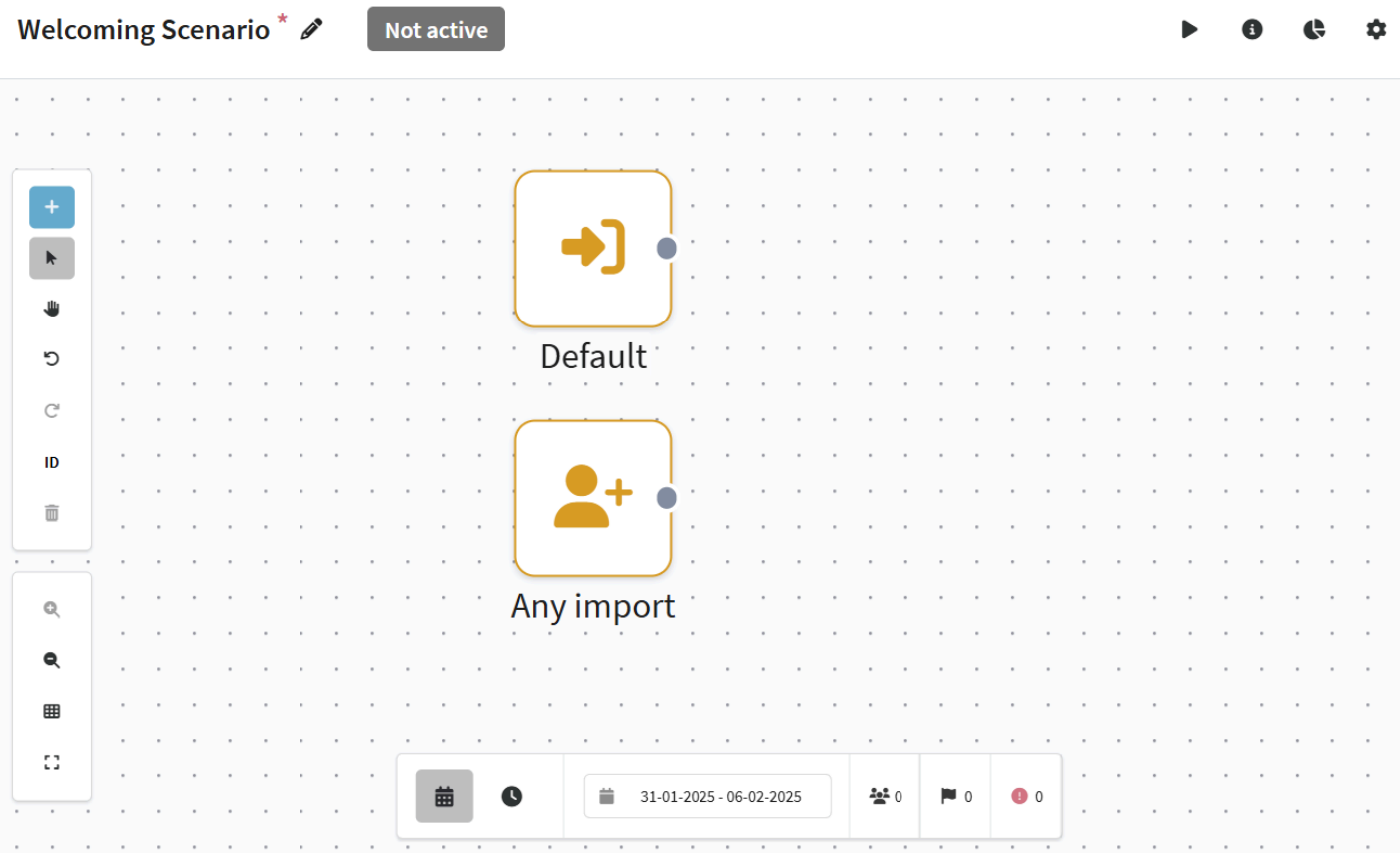
TIP

Various customer data import and update options are described in this article.

This article describes scheduled customer profile import from an SQL storage.

API requests considering customer profiles are described in this section of API docs.

In order for all new profiles from the database to get into the script, we use the trigger Any profile import:



Step 2: Create an Email welcoming message

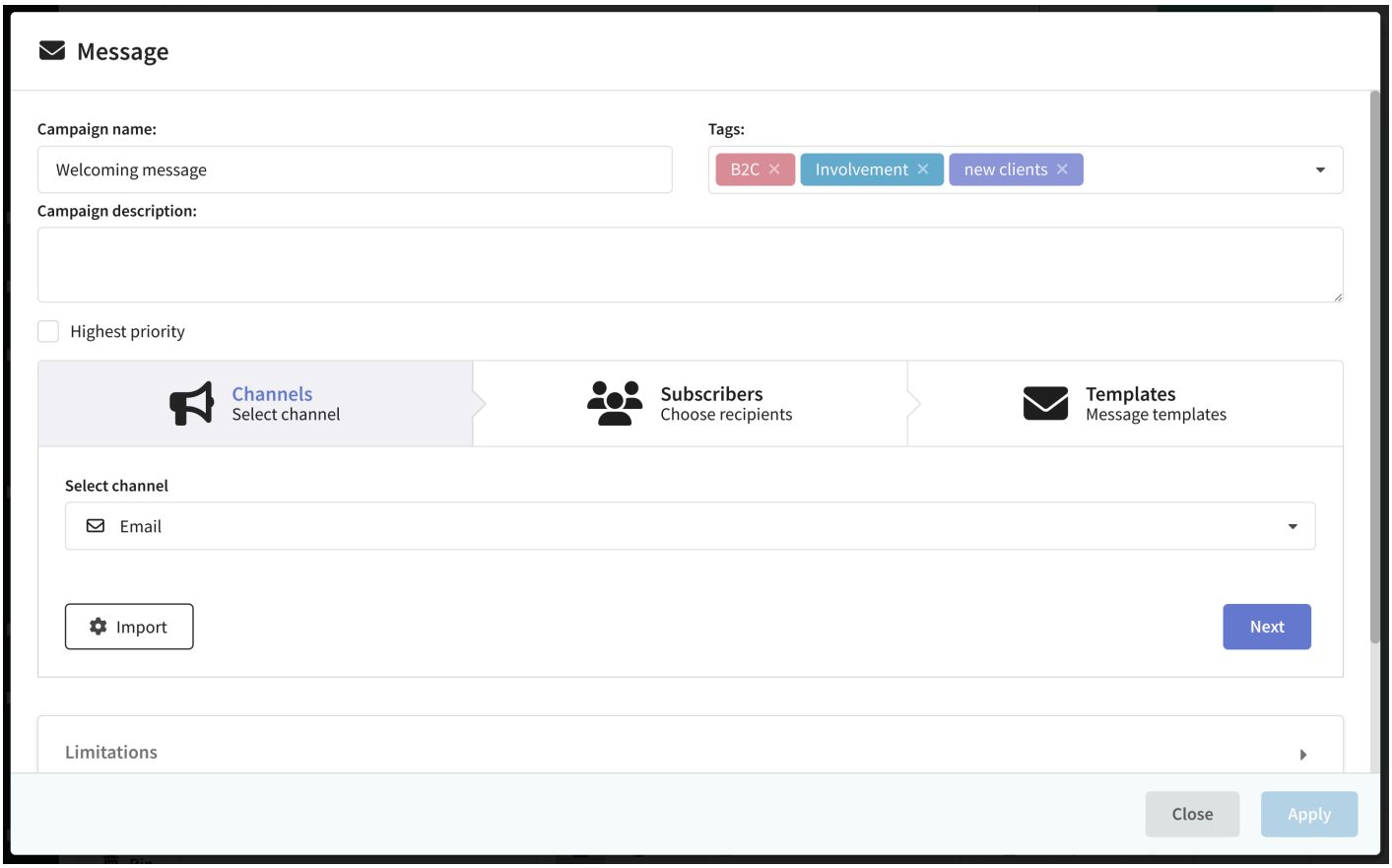
Add the following element to the script - email distribution. To do this, click on + Element - Channels - Email. This will open the mailing customization window. Here you can specify channel, resource and message template.

TIP

Subscription resources for your audience are described in this article.

This section will tell you all you should know about message templates.

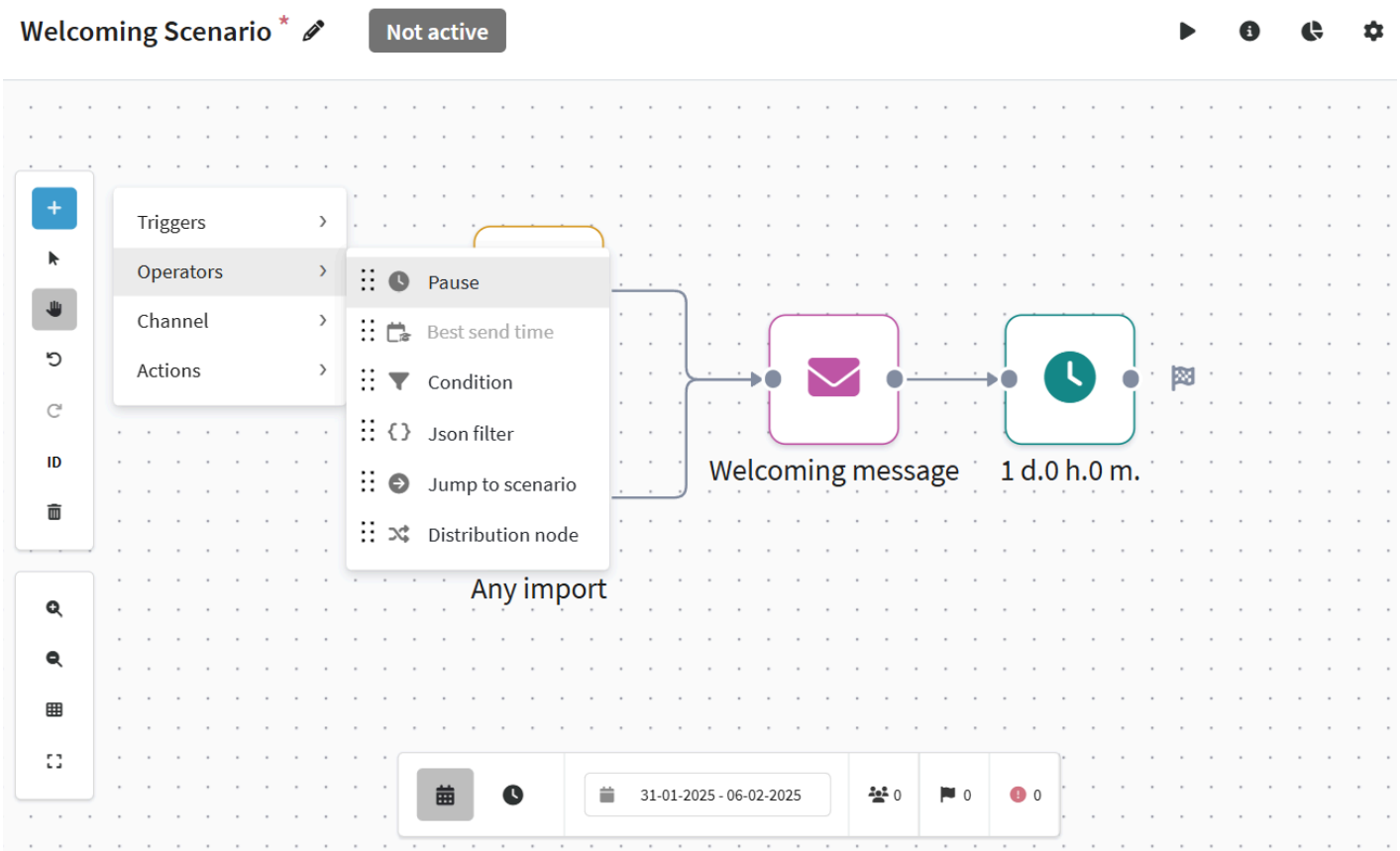
You can find a step-by-step Email mailing guide here.



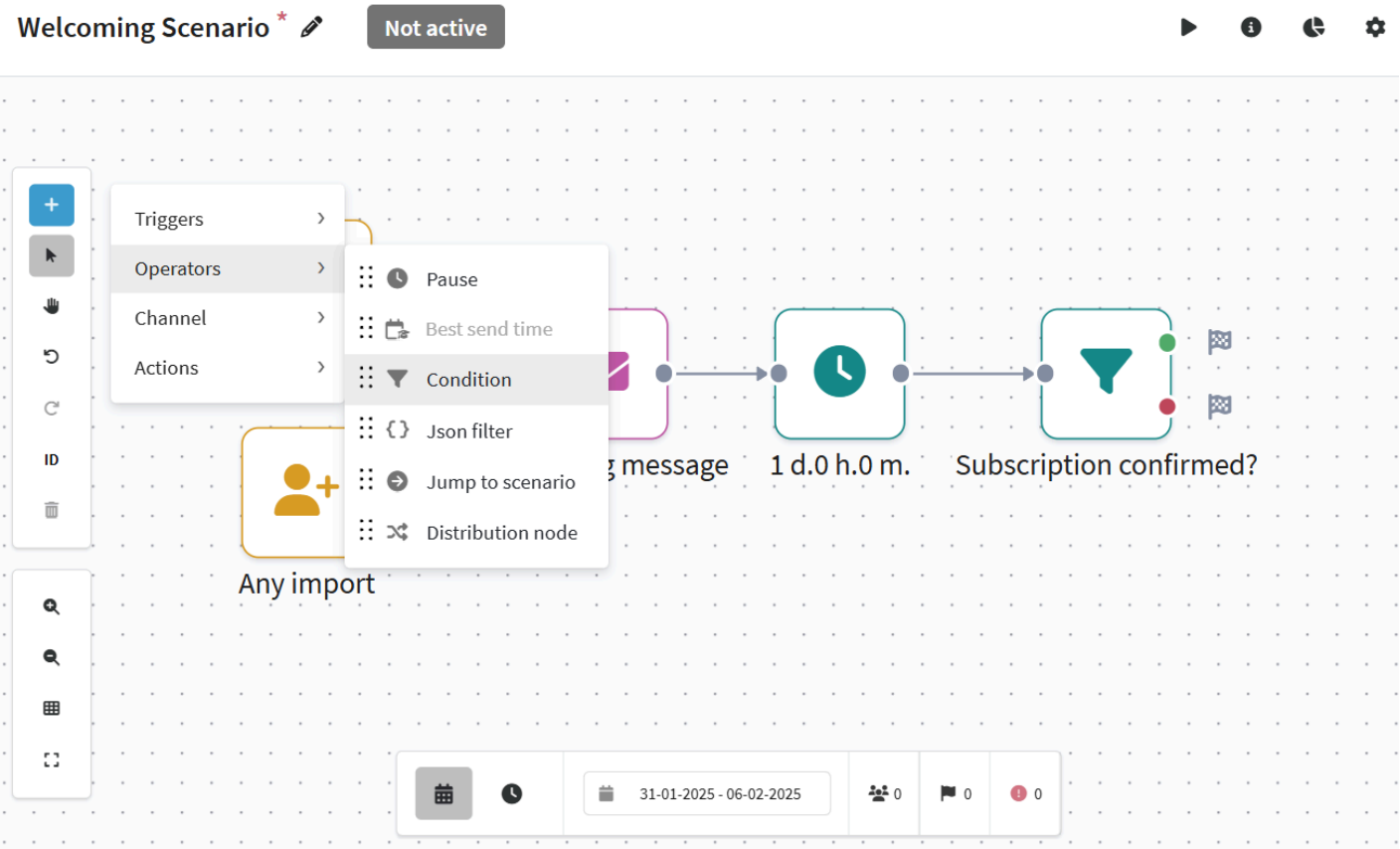
Step 3: Set up a sequence for active recipients

After you send an Email welcome, a part of your customers will open, read it, click a link to the purchase page and make a deal. The percent of those who do is called your marketing conversion. Now, to increase welcome workflow conversion let us compose a Customer journey sequence. Various scenario nodes will help you a lot.

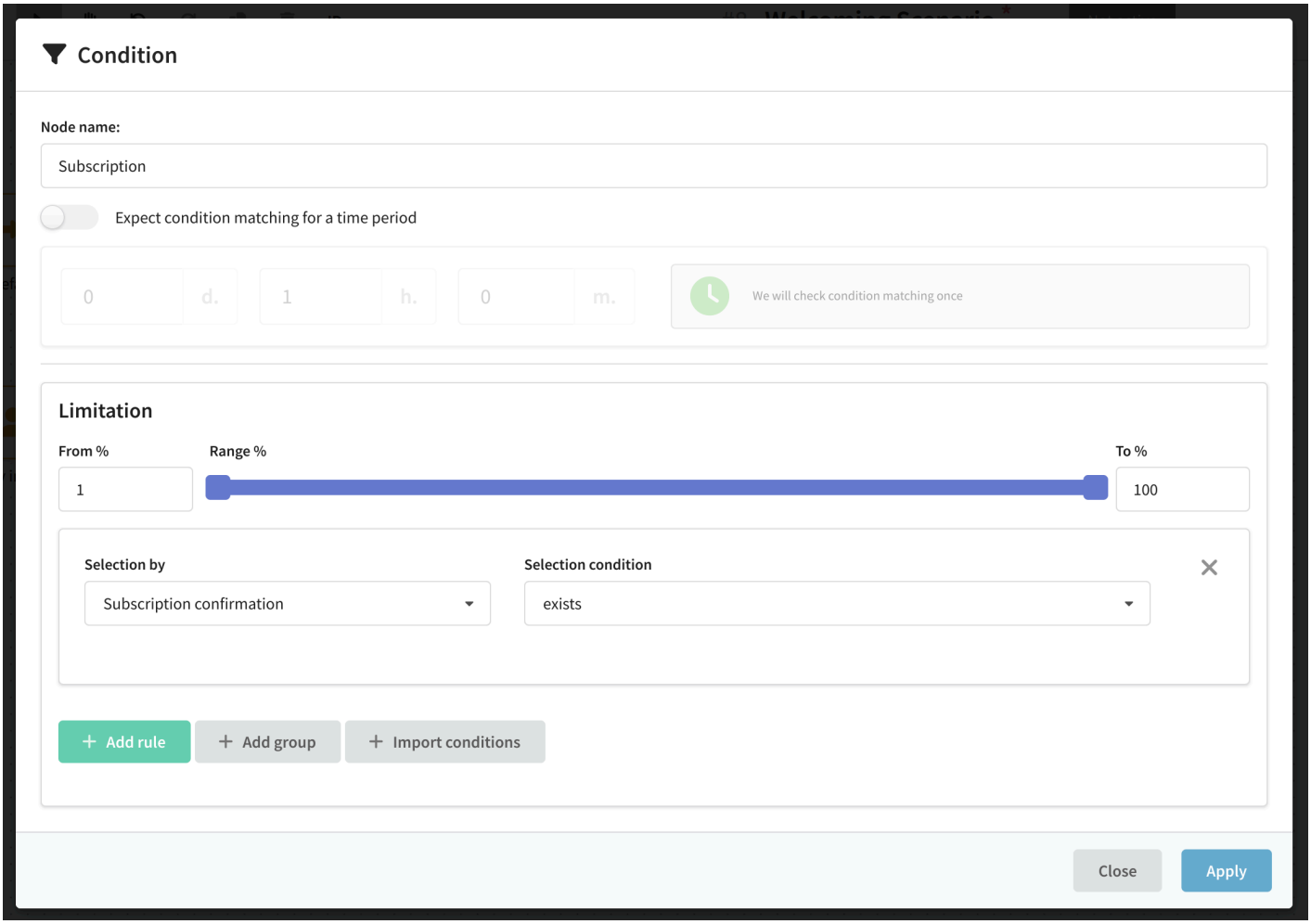
The next element will be the pause. With its help we will give customers time to read the email. The Pause element is located in the Operators category:



Next, using the Condition element, let's separate out those subscribers who opened the welcome message and confirmed their registration on the first day:

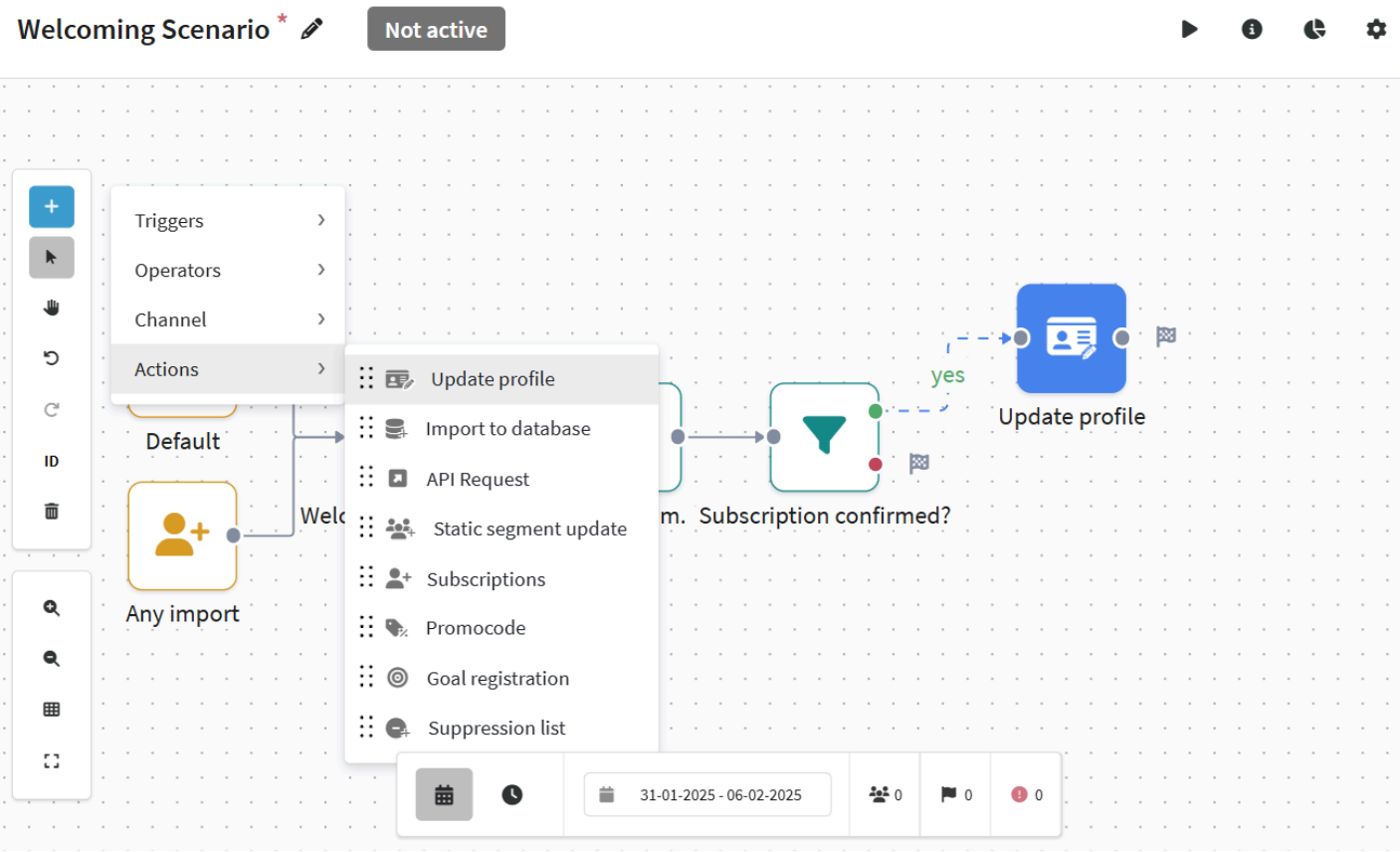


You can use the "Email opened" or "Click on link" rule instead of the registration confirmation. The example uses the DOI (double opt-in) strategy subscription confirmation:

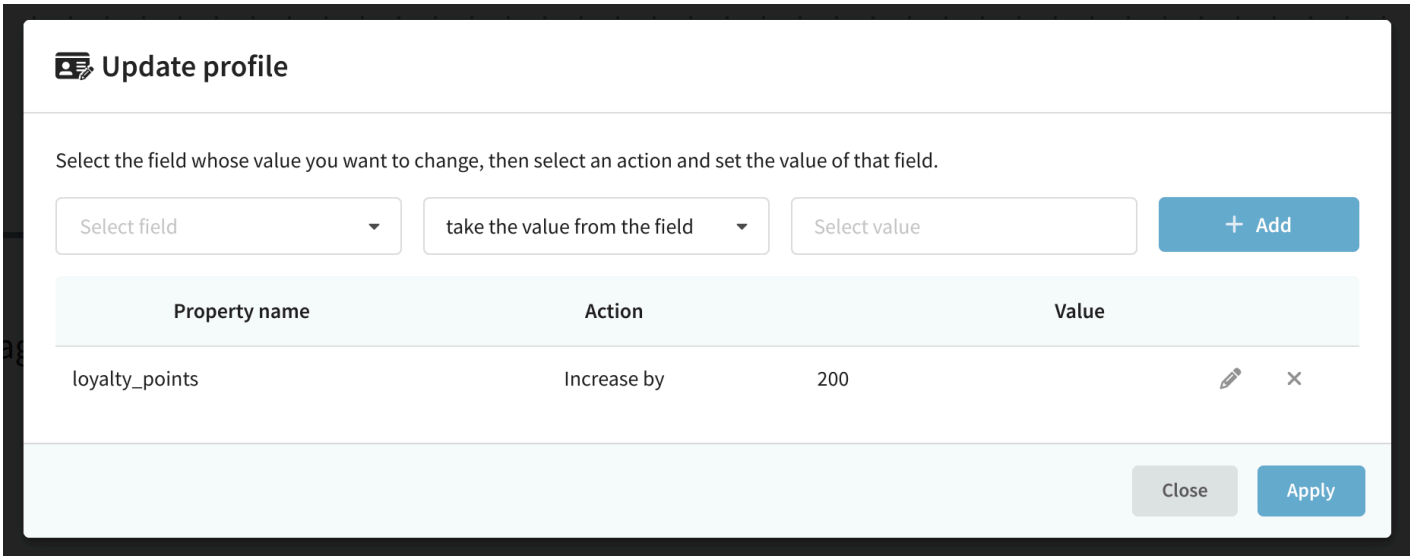


These customers are likely to be highly motivated to buy. It's a good idea to reward these customers, for example, you can give them bonus points right away. These can then be used to assign promo codes, discounts or gifts.

Such actions in marketing are called scoring. Scoring is one of the elements of sales gamification that can significantly increase conversion rates. To implement it in the platform, you must first create a numeric field in the profile database for the value of loyalty points. You can update these points in the script using the Update Profile element:



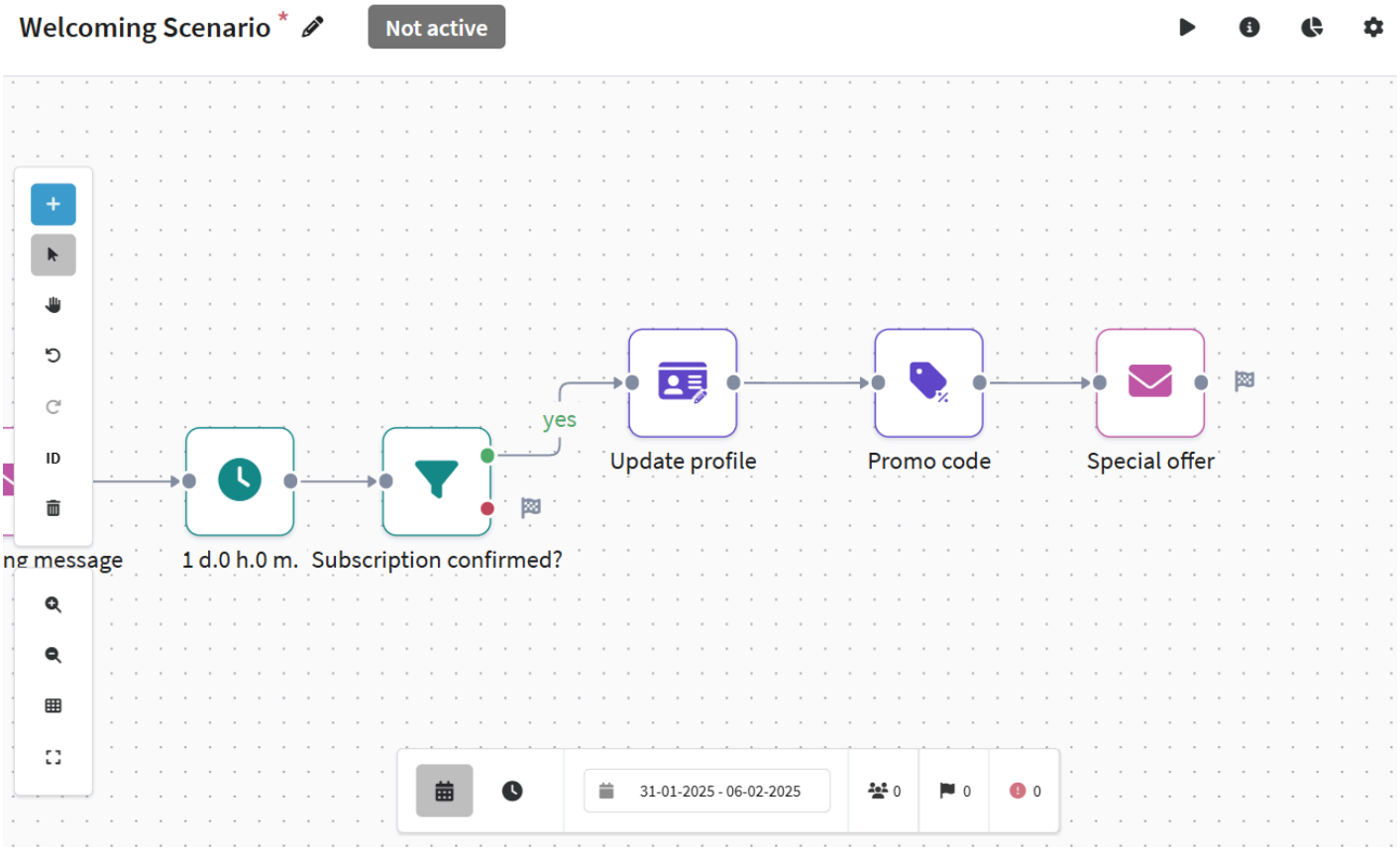
Add the element to the top (green) branch coming from the condition. In the element settings, specify the field that will contain loyalty points and increase its value by the required amount:



Next, you can assign promo codes to new customers (for example - for a part of a product serving specifically to warm up sales) and send promo offers to customers. To do this, add a Promo Code element and an email channel message.

TIP

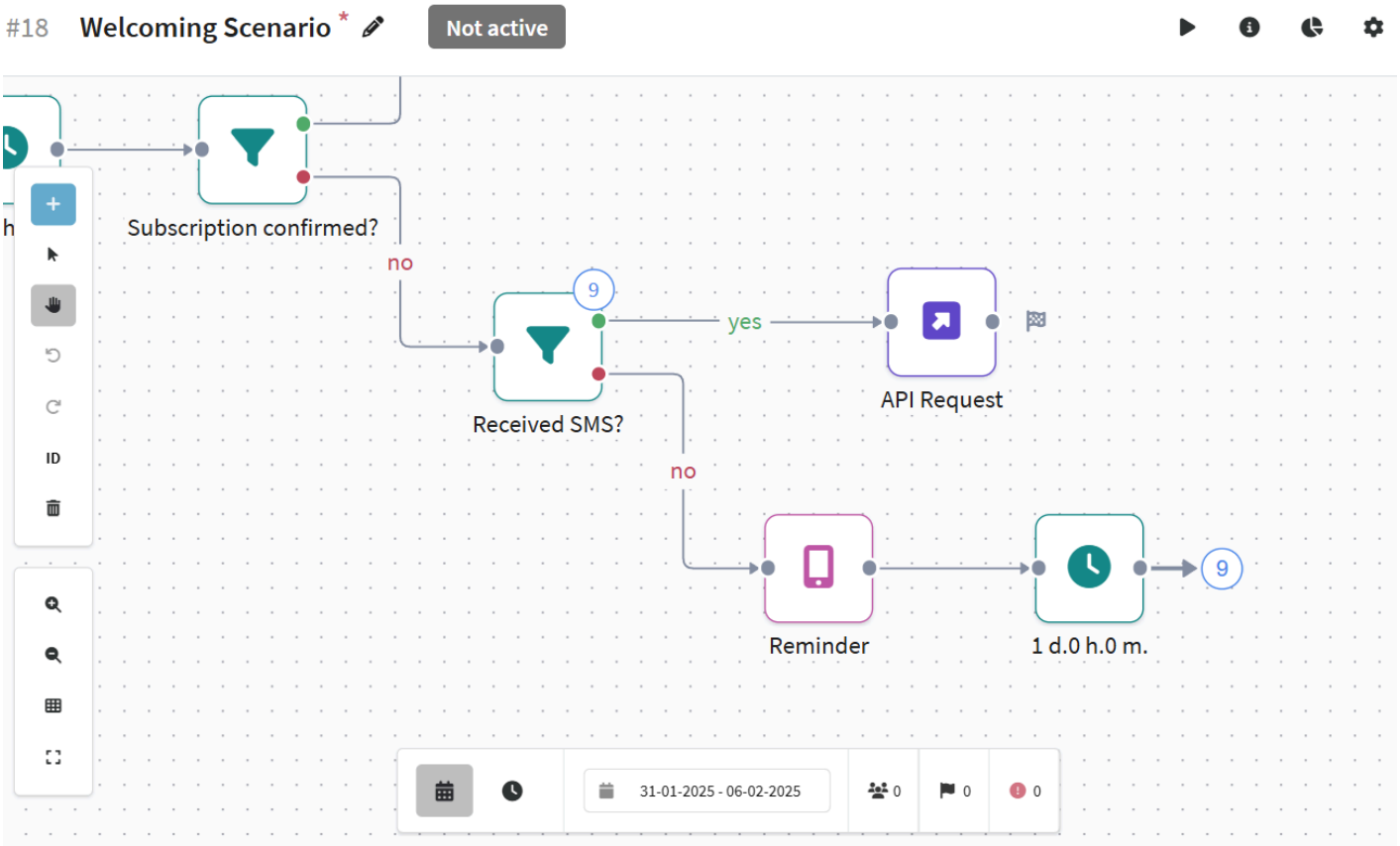
Promo codes are created by Altcraft Marketing's loyalty programs. You can read more in this article.



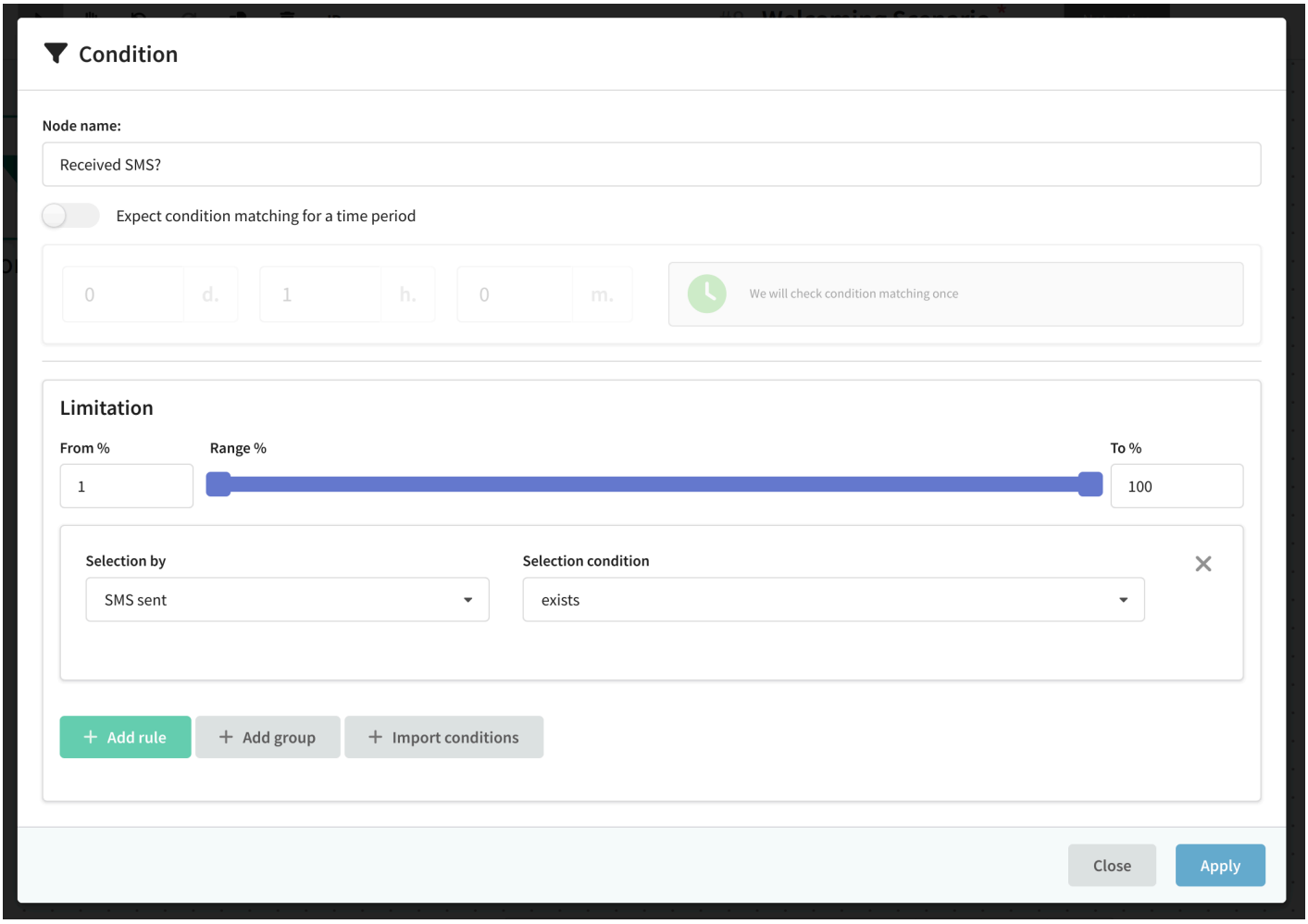
Step 4: Set up a sequence for not-so-active recipients

Some of the customers to whom you sent a welcome message may have forgotten about it, not received it, postponed it, or simply not interested in the offer. In this case you can remind them about the incomplete registration. It's best to use the SMS channel: everyone has a phone and it's always at hand.

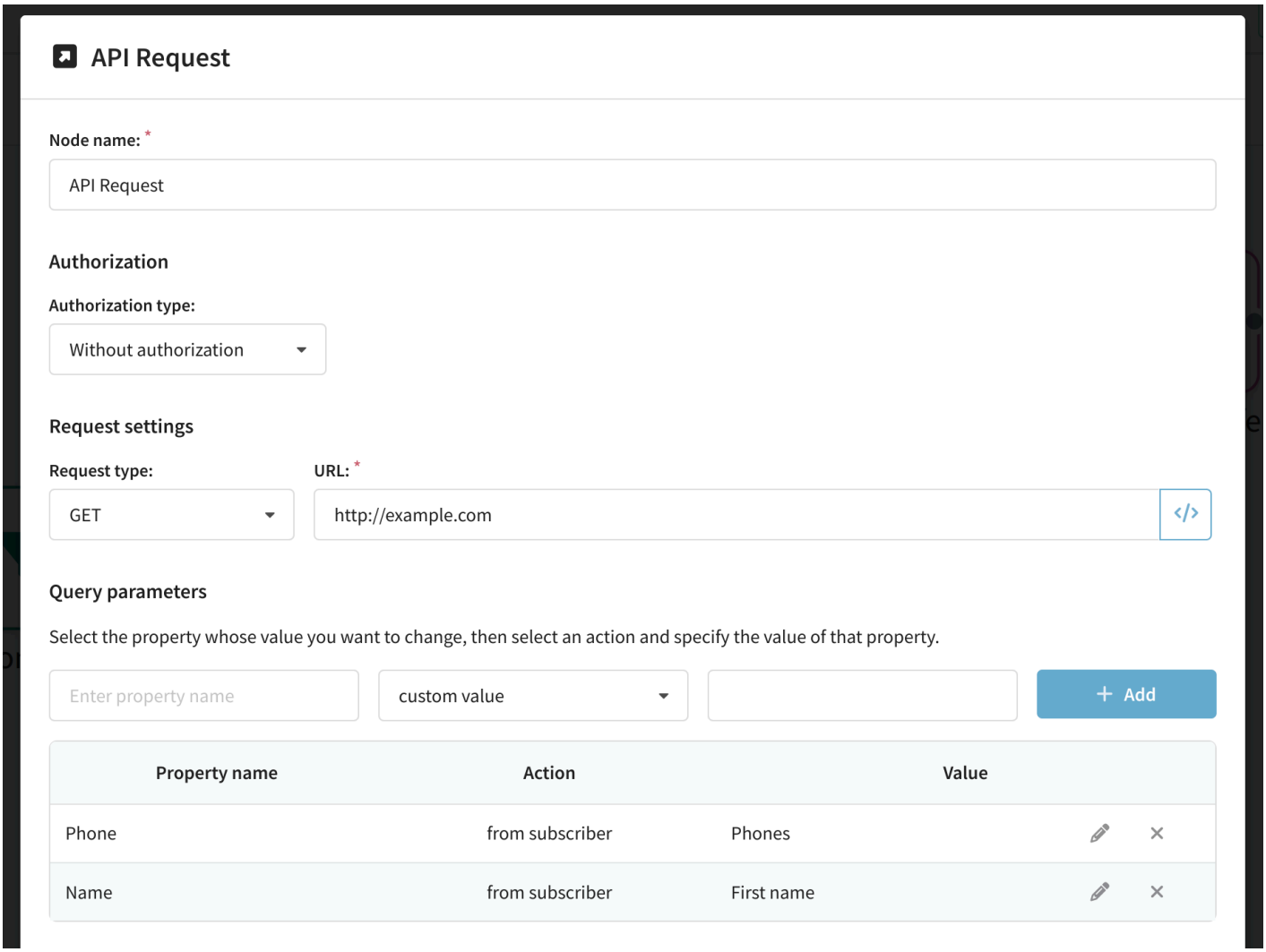
However, it is usually not always advisable to send the same reminder more than once. It is much more effective to use different reactivation tactics. In the example, let's use two: first, a SMS is sent, and if there is no response to it, a notification with contact information from the profile is sent to managers or call center operators so that an employee can personally communicate with the customer. Our "branch" for inactive customers would look like this:



To apply two tactics at the same time, add a Condition element for those who have been sent an SMS reminder at least once. Select the rule There is an existing SMS sending:



From the top (along the green branch), add the API Request element. In it, specify the network address, the type of request, and the data you want to pass to the call center. Contact your company's system administrators for details:

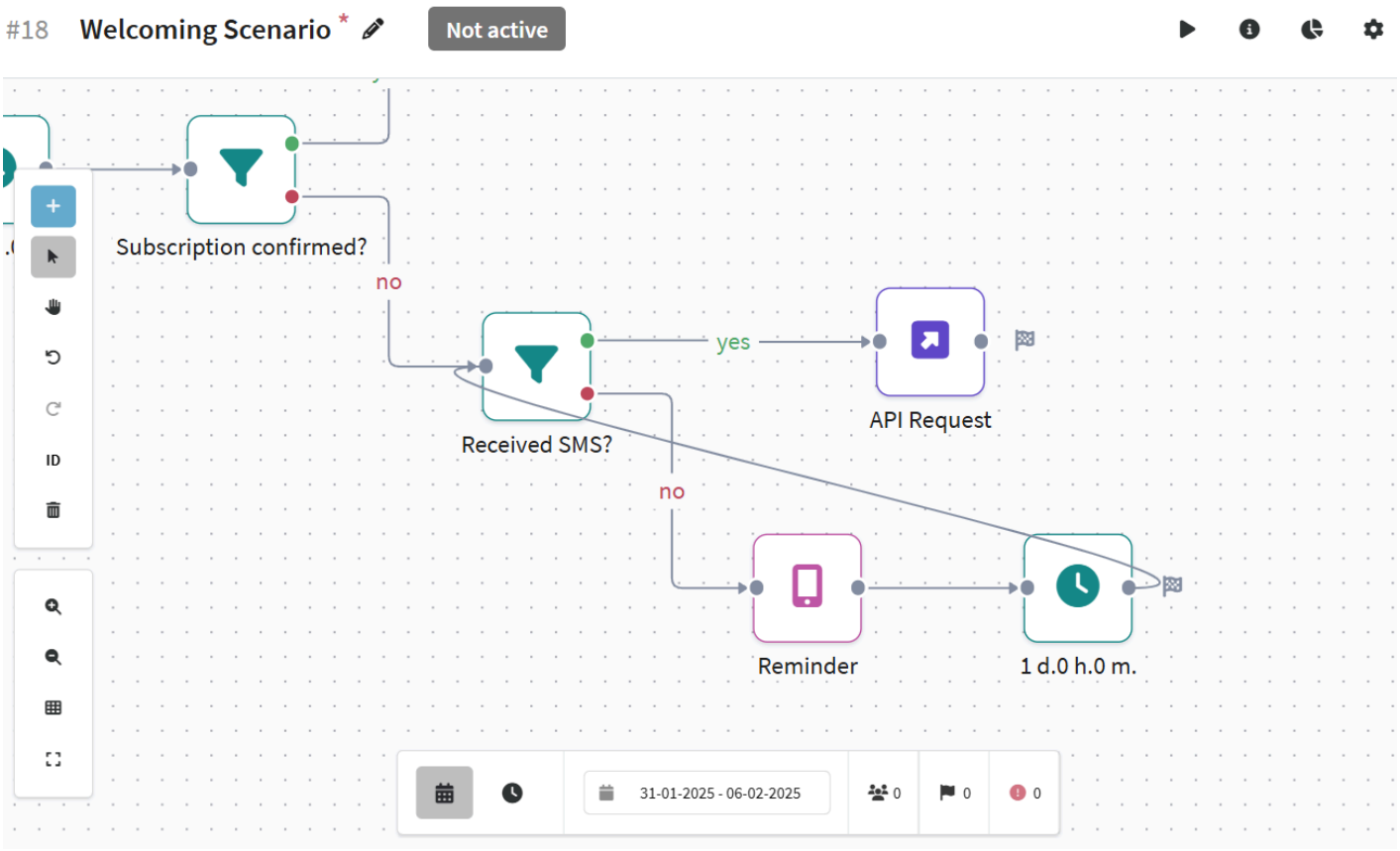


Down from the condition (along the red branch) add an SMS element where you create a new SMS mailing or import an existing one.

TIP

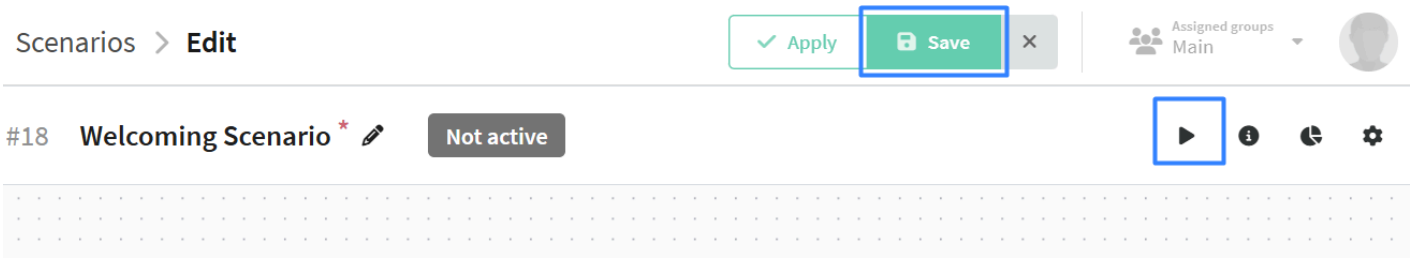
You can find a step-by-step guide on how to create an SMS newsletter right there.

After the message, add a Pause - just like you would after an Email Welcoming. When time runs out after sending the SMS, the customer should be sent back to the beginning of the script, to recheck the subscription confirmation in the Condition element. To do this, drag the cursor from the right point of the Pause element to the SMS sending condition:

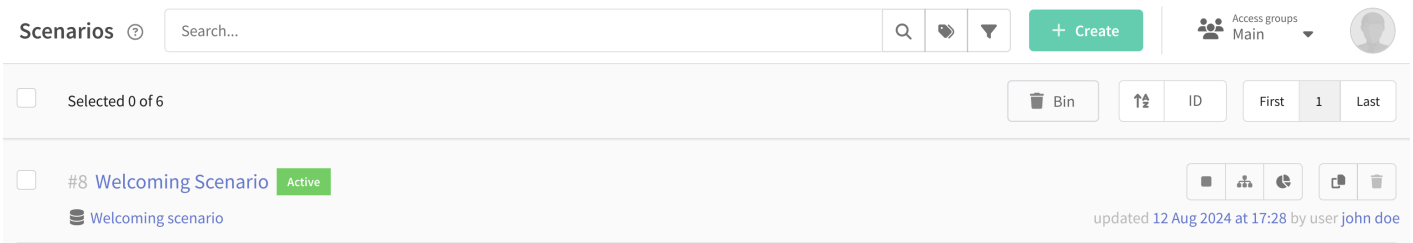


Step 5: Save and activate the scenario

Once you have made the necessary settings and configured the script steps, save and activate it by clicking on the appropriate buttons at the top of the screen:



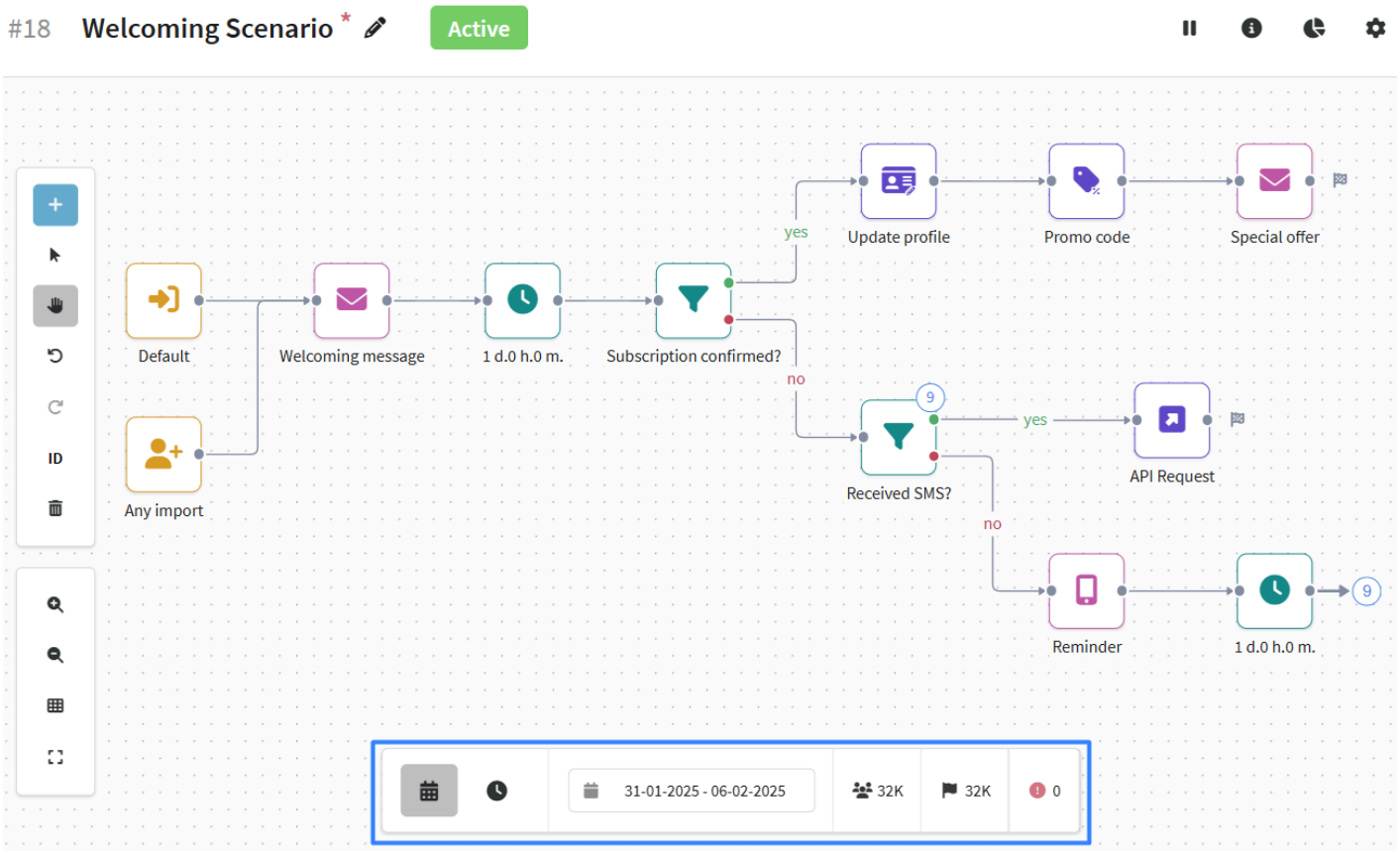
The welcome sequence will be a great start to effective long-term communication with your customers. You can open the script for viewing and editing by clicking on its title:



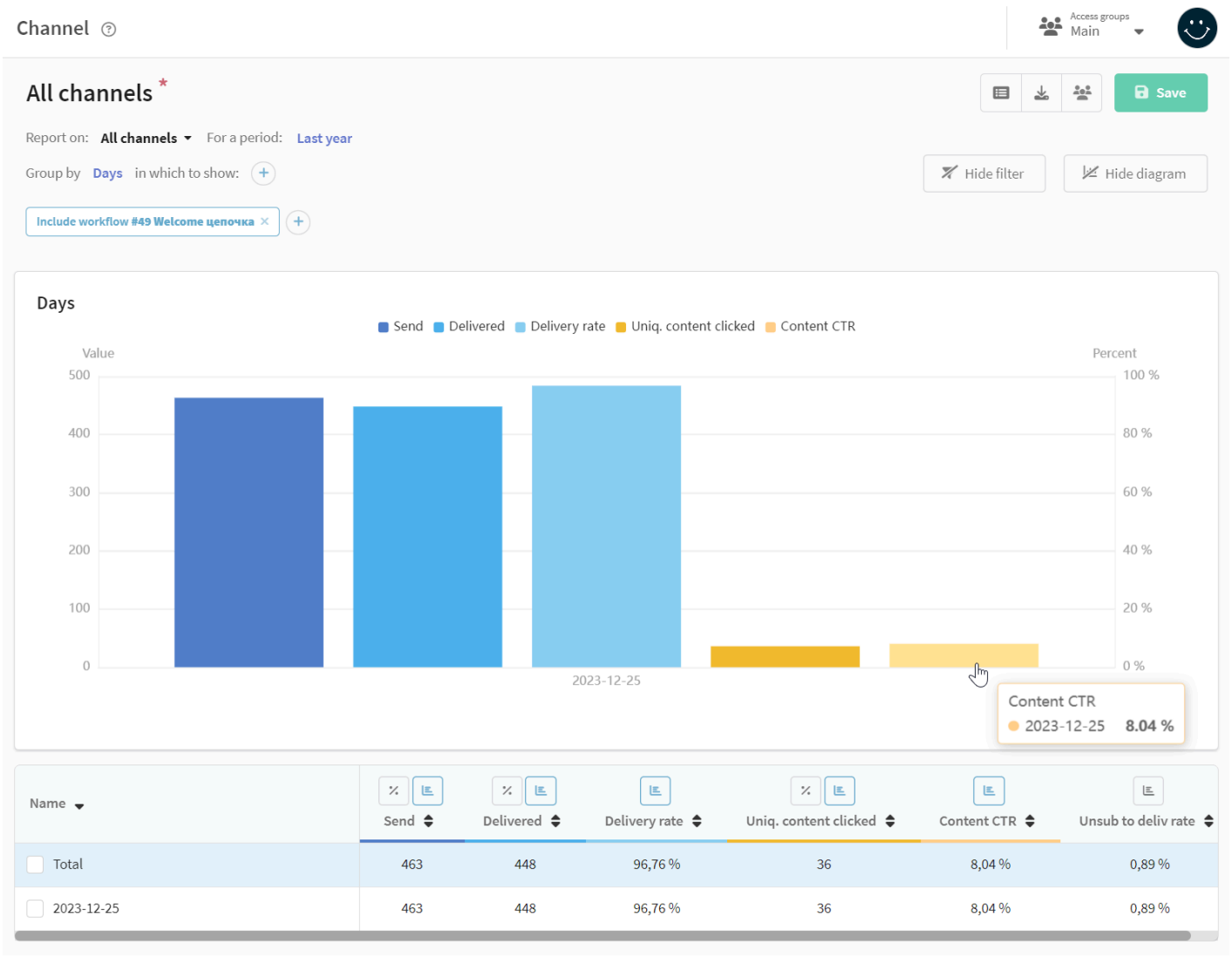
Step 6: Analyze performance

You can evaluate the effectiveness of a scenario using several analytical tools.

The first is statistics within the scenario. Select the required period and view the pass rates:

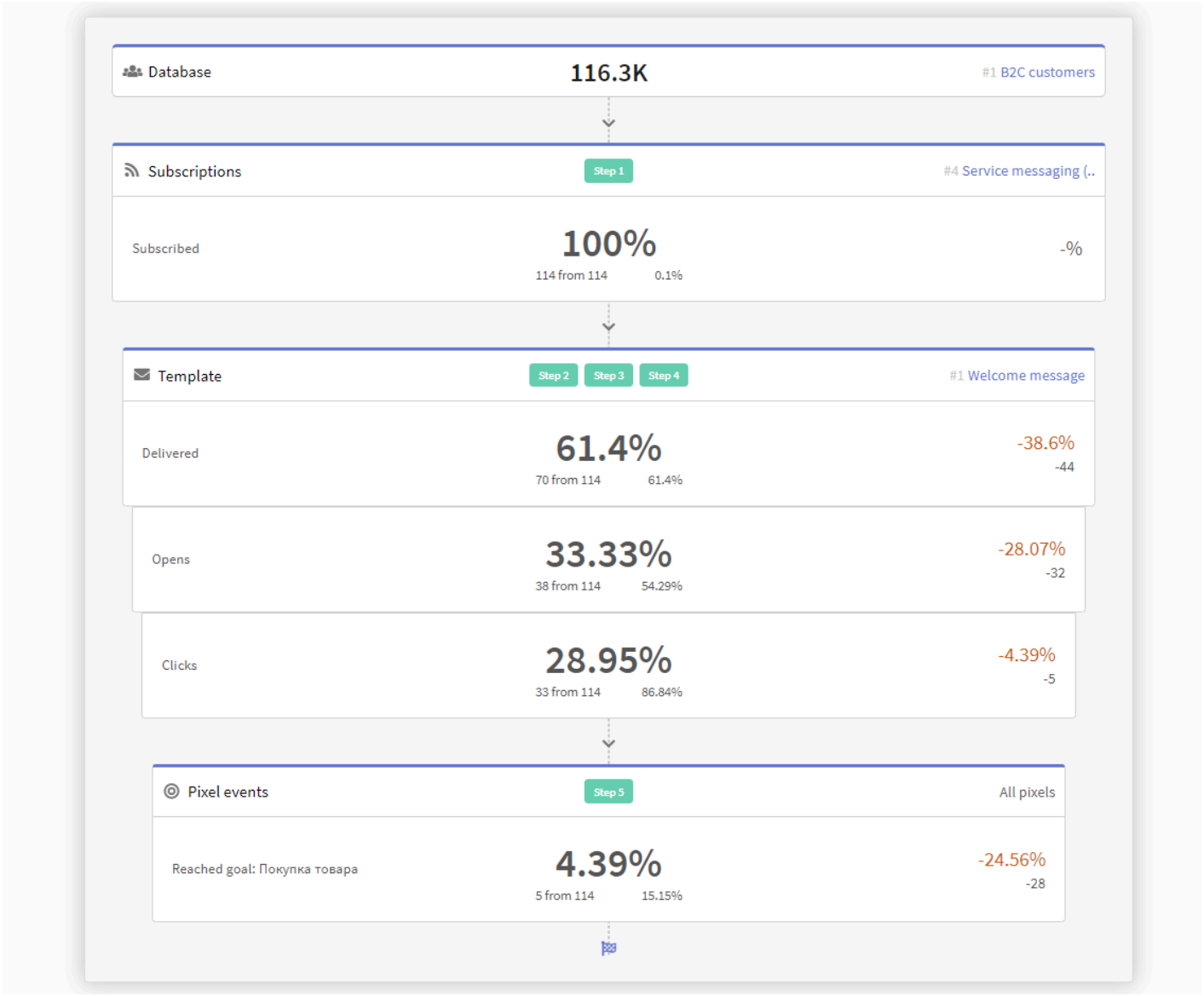


The second is [channel report](#) of the scenario. It will provide detailed information on the mailings within the scenario:



Name	Send	Delivered	Delivery rate	Uniq. content clicked	Content CTR	Unsub to deliv rate
Total	463	448	96,76 %	36	8,04 %	0,89 %
2023-12-25	463	448	96,76 %	36	8,04 %	0,89 %

Finally, you can add mailings and other scenario events to [funnel-report] to monitor conversion rates even more confidently. A step in the funnel can be whether a customer falls within the sample conditions or reaches a certain goal (Purchase, Contribution).



Scenario for automatic notification of the manager

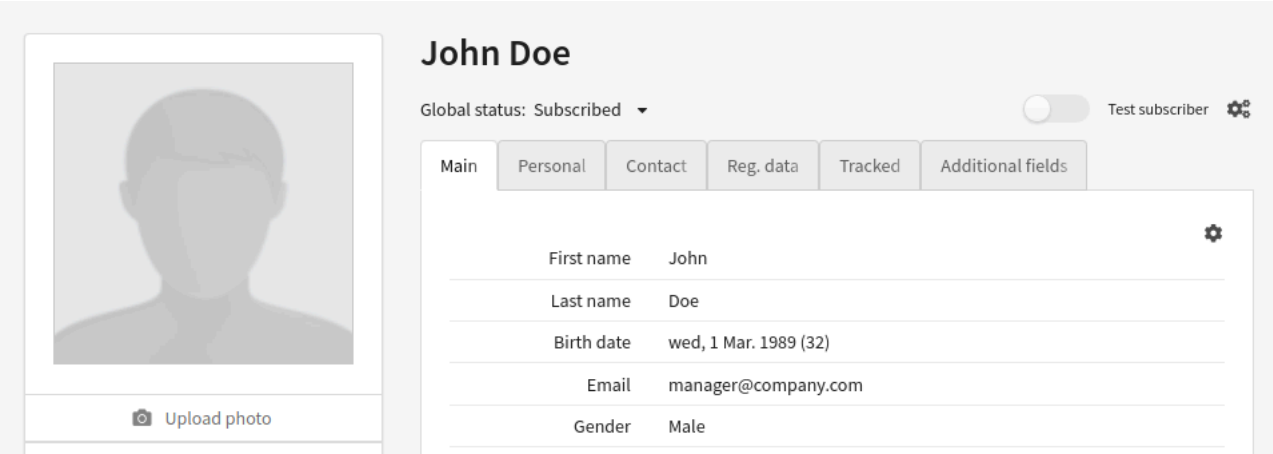
Let's look at a typical case that can be implemented using scenarios in the Altcraft Marketing platform.

Here is the task. Every time the user performs any target action, the manager must receive a letter with information about the client.

Let's consider the implementation of this functionality through an API call from a scenario. The lead database is supposed to be created and filled.

Step 1. Create a manager database

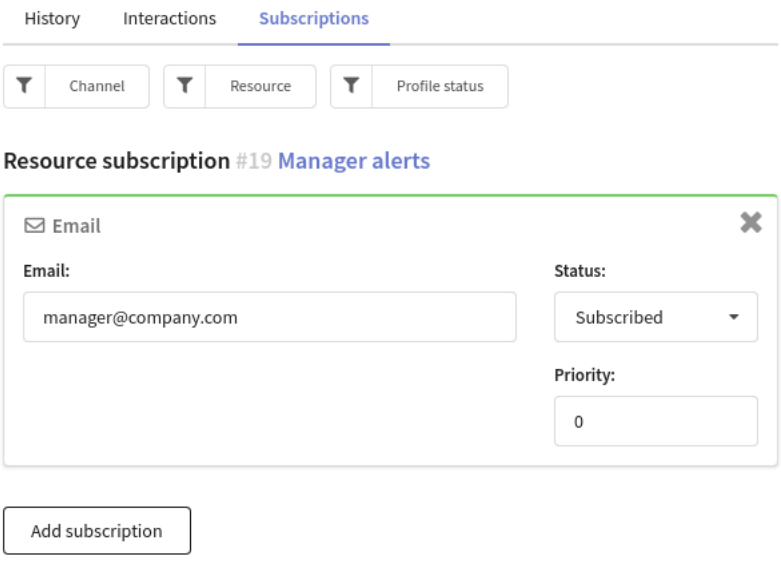
First of all, it's necessary to create a database. Let's call it "Managers". You need to set up the database and specify its field structure. Then you need to create a new profile.



You can learn more about profile bases right there.

Step 2. Create a resource and a subscription

Only when the manager database is created and the profiles are added to it, it is necessary to create a resource (let's call it "Manager alerts"). When you create a resource, specify the Managers database in its settings. After the resource has been created, you need to subscribe a manager profile to it. In the example, let's add a subscription manually:



You can learn more about resources in this article.

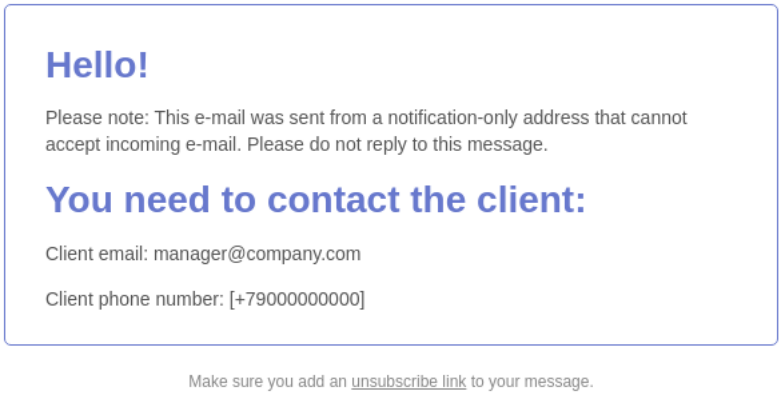
Step 3. Create a template

Now it's time to create the template of the message that will be sent to a manager. Let's call it "Alerts". We're going to use variables apicontent to personalize messages.

Client email: {apicontent.mail}

Client phone number: {apicontent.ph_number}

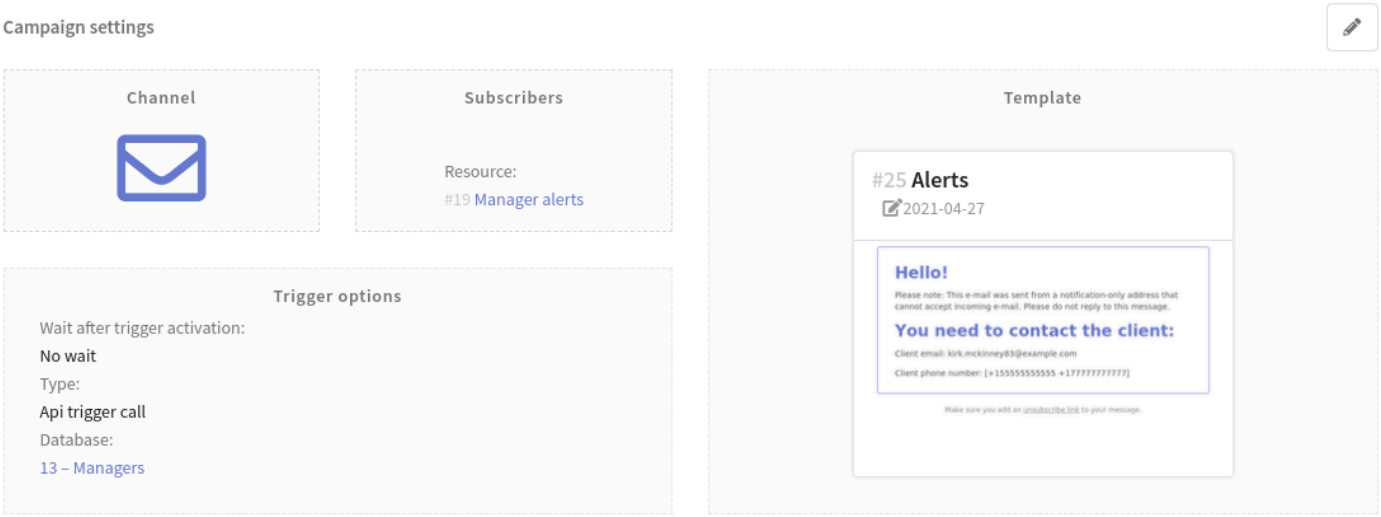
The example of the template for notification of a manager:



You can learn more about templates in this article.

Step 4. Create a trigger

We're going to use a trigger mailing as a tool of notifying a manager. When creating it, you need to select the resource "Manager alerts", the template "Alerts", the condition for triggering - API trigger call and the database "Managers".



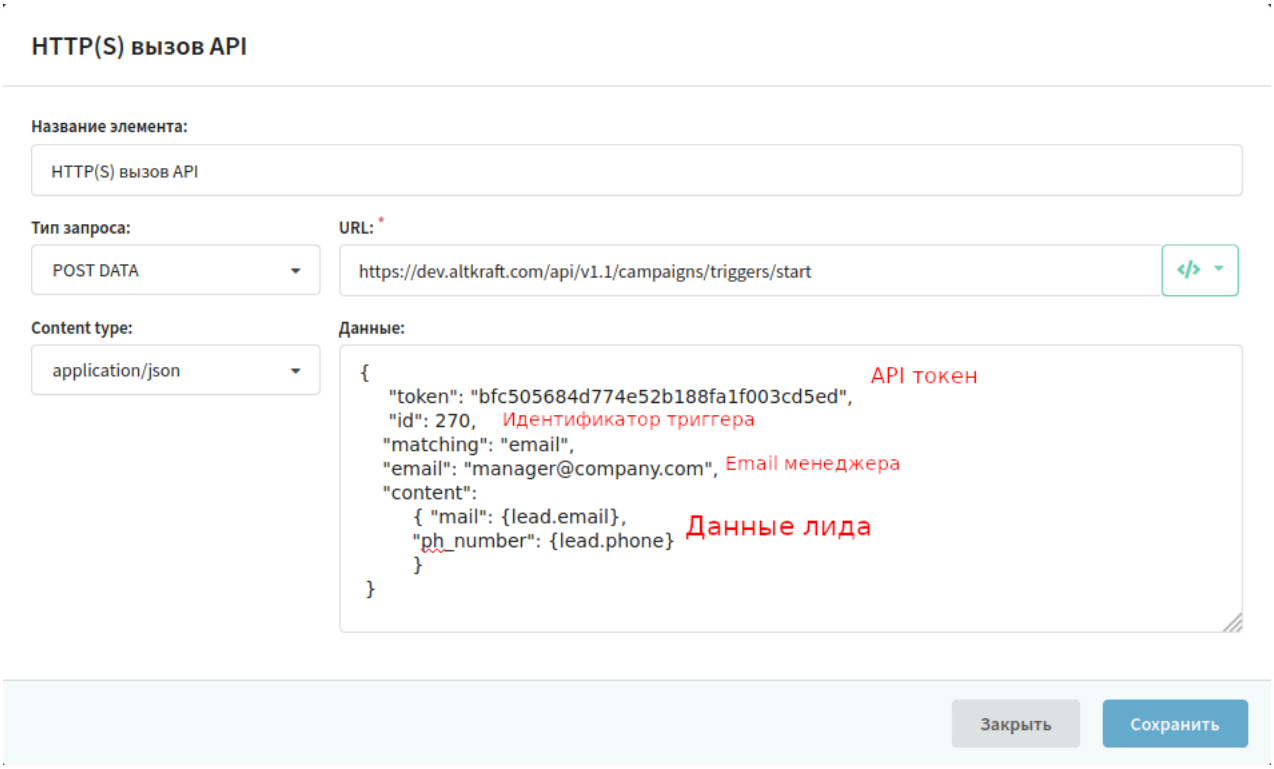
You can learn more about trigger mailings right there.

Step 5. Create a scenario

The key element that will launch a trigger call and then send a notification to a manager is the HTTP(S) API Call node. Select POST DATA as the request type, Content type - JSON or XML (in our example, it's JSON). The URL contains the address at which the trigger will be called (`api/v1.1/triggers/start`). The body of the API request is placed in the Data section, according to which the "Trigger - notification" will be called (you can learn more about the API right there).

```
{
  "token": "bfc505684d774e52b188fa1f003cd5ed",
  "id": 270,
  "matching": "email",
  "email": "manager@company.com",
  "content": {
    "mail": "{lead.email}",
    "phones": "{lead.phones}"
  }
}
```

This request specifies the API token for authorization, the trigger identifier, the email of the manager, as well as the email and phone number of the lead, about which the manager will be notified.



INFO

When sending a request, an API token is required for authorization. The token must be created in the platform. More details can be found in our article.

Well, you've just completed the configuration of the manager's notification automation scenario!

Abandoned cart scenario

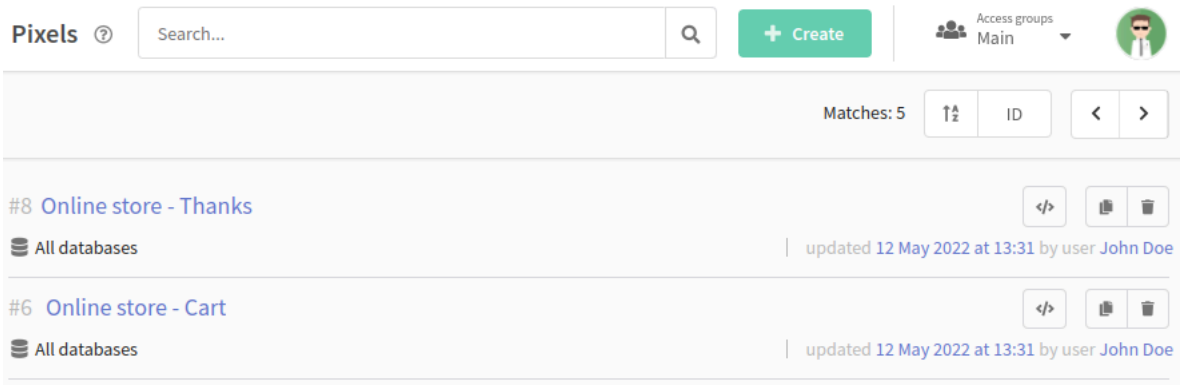
Visitors to an online store often add items to their carts but do not place an order. However, if you remind them about the abandoned items after a while, the chances of completing the purchase will increase. This guide is here to help you set up a short scenario that will automatically send reminder emails to your customers.

Step 1: Place tracking pixels on website pages

Let's say that a user adds items to the cart, goes to the cart, pays for the purchase and then is redirected to a thank-you page. A subscriber will receive a reminder about the abandoned cart if they visit the cart page, but do not get to a thank-you page afterwards, i.e. do not complete the checkout. Pixels will help you track visits to certain website pages.

In the Pixels section, create 2 pixels:

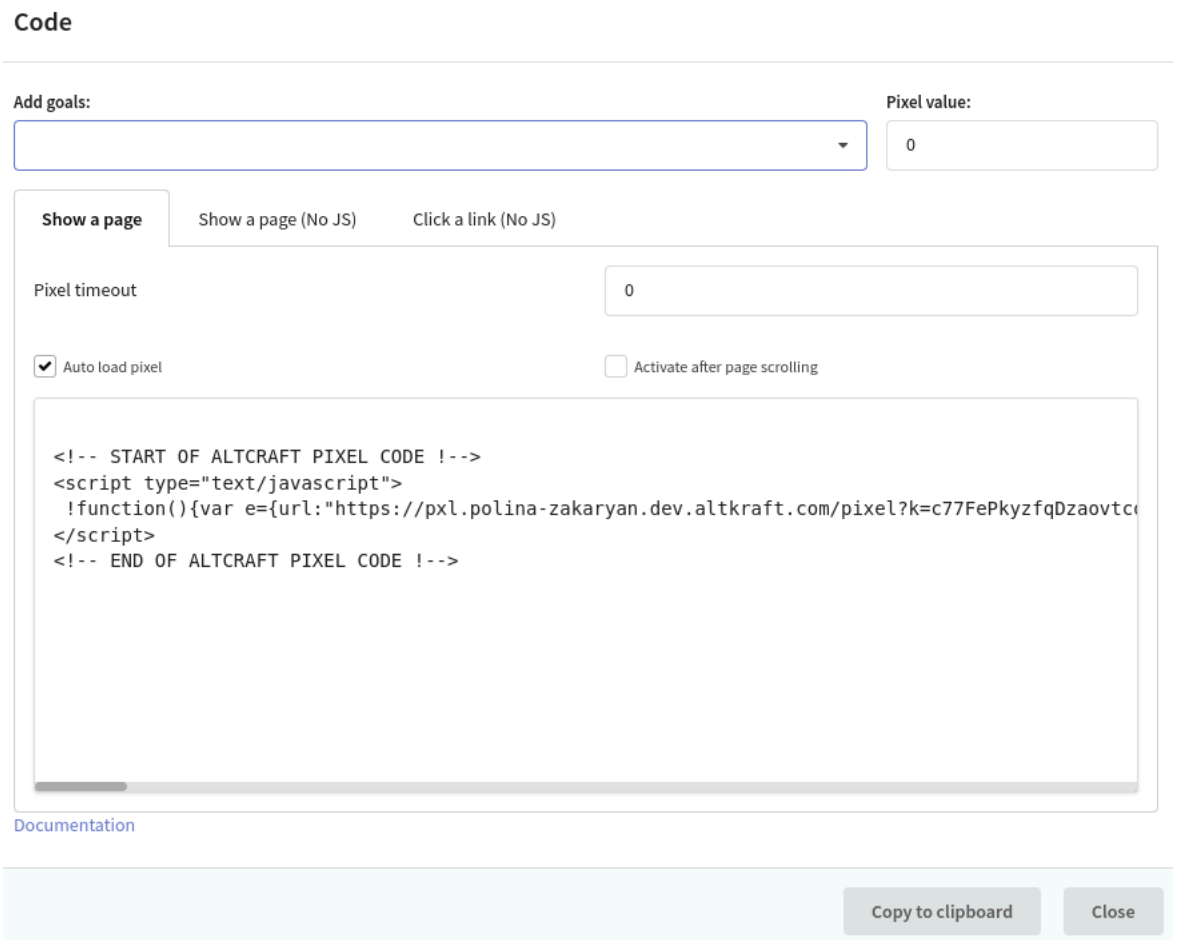
- **Online store - Cart: it tracks visits to the cart page**
- **Online store - Thanks: it tracks visits to a thank-you page**



TIP

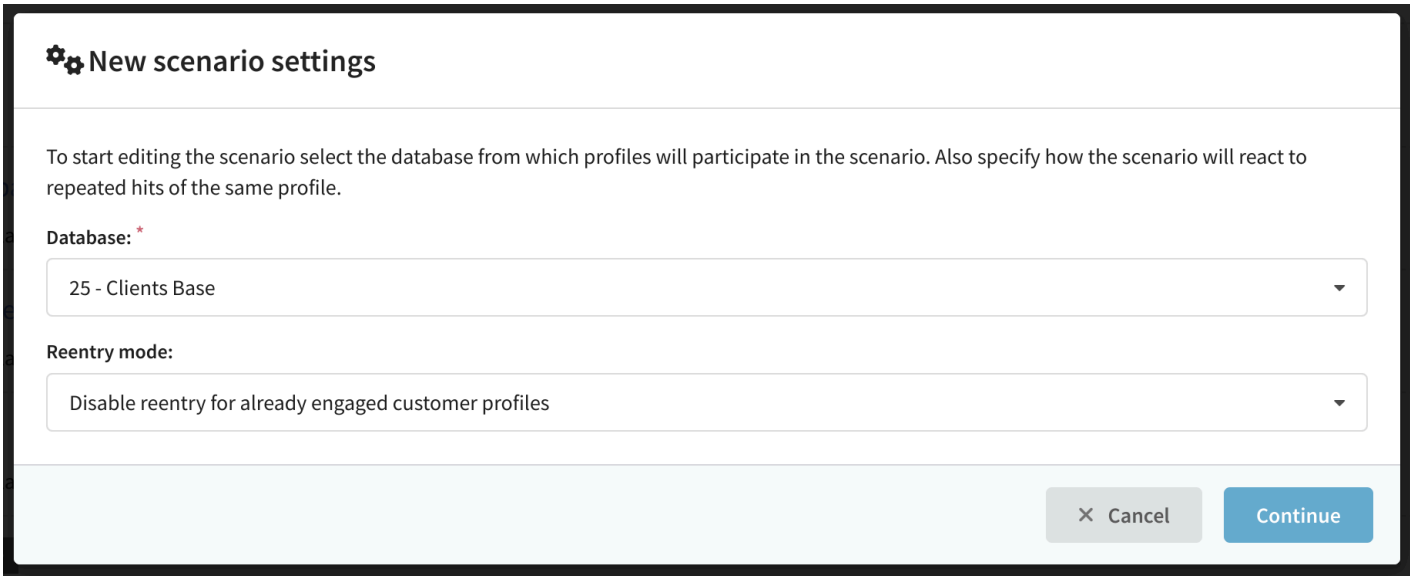
In some situations, a website visitor may open the cart several times. If you don't want to spam your client's email by sending several reminders in one day, block reactivation of the pixel for the cart for 7200 minutes (5 days).

To view the pixel code, click "Get pixel code". Copy the code from the window and place it on the website pages:

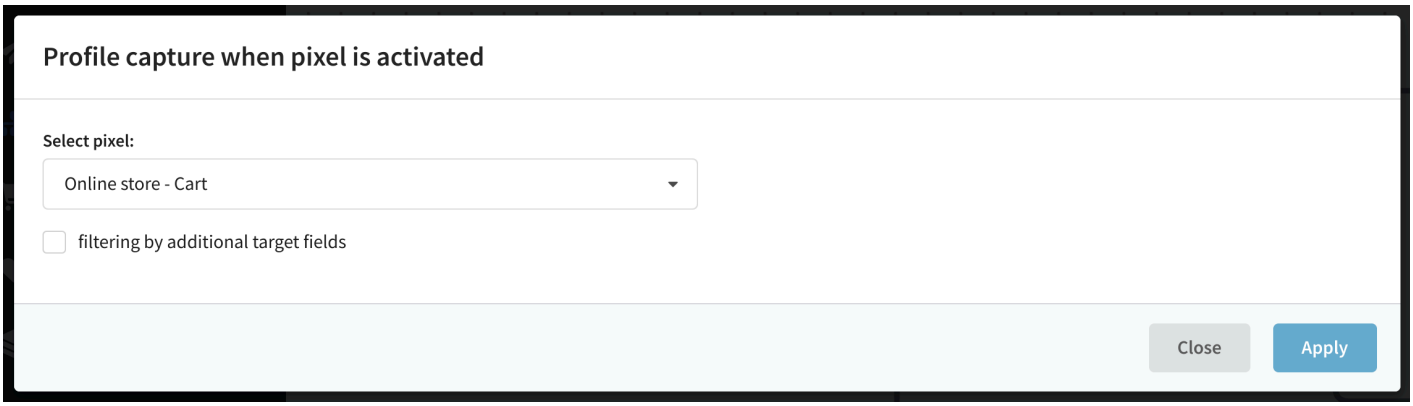


Step 2: Configure customer profile capture

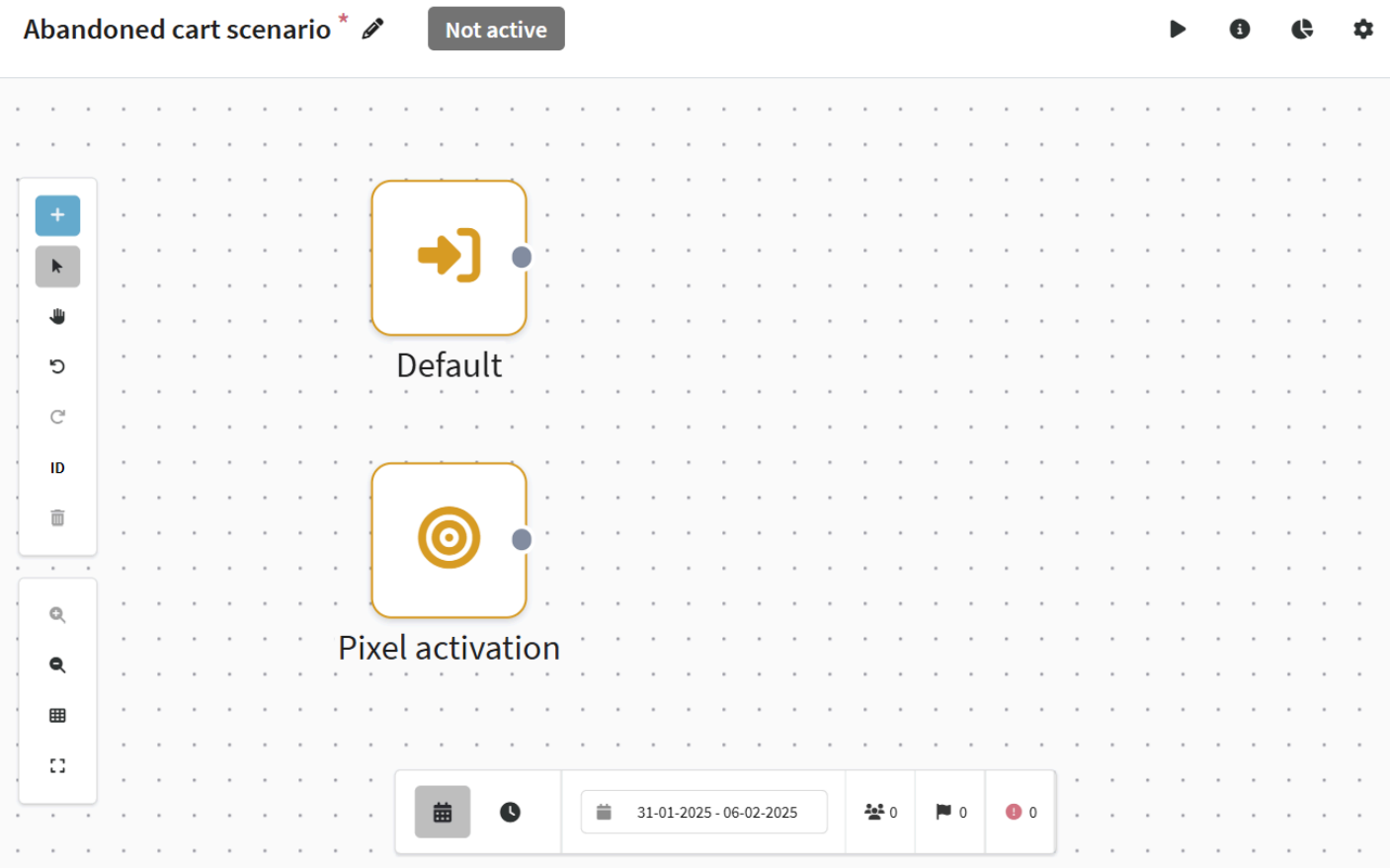
Create a new scenario. Specify the base - the profiles of these clients will be hit in the scenario if the capture condition is met. Select what to do with the profile in case of repeated hits. The option Disable reentry for already engaged customer profiles will work for us. This way the customer will be able to get into it again, but only when he is going to make the next purchase after some time.



To get users into the scenario, add a trigger element When a pixel is activated. The pixel we placed on the page with the shopping cart (Online store - Cart) should be activated.

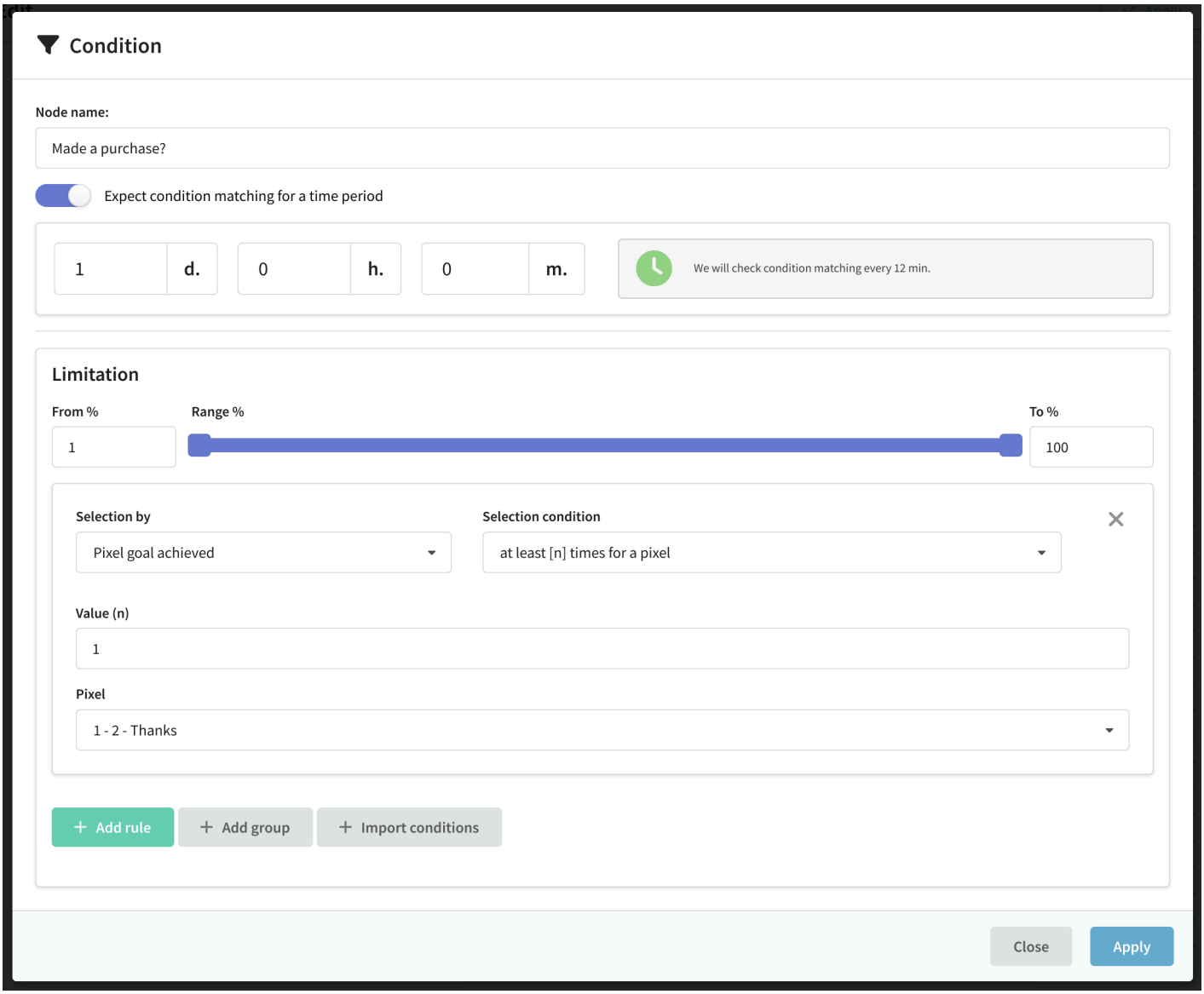


In any script, an Default Import element is automatically created. It is required for importing profiles manually, via API and when switching from another script. This is a system element, it is impossible to delete it:

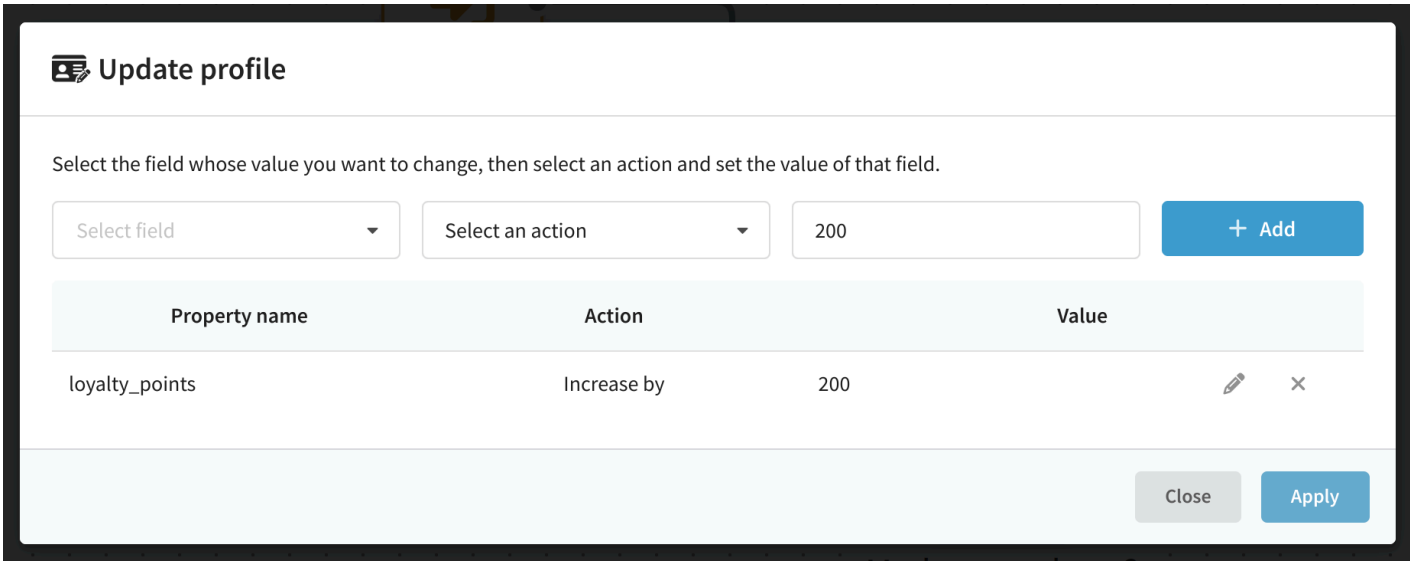


Step 3. Set up a workflow for the profiles that have completed their orders

Next, add the Condition element to check whether the user has made a purchase or not. The condition will be fulfilled if a person went to the thank you page, i.e. the Online store - Thanks pixel was activated. When customizing the element, toggle the slider Wait for condition for a specified time and specify the waiting time - 1 day. The platform will check if the customer has made a purchase during this time:



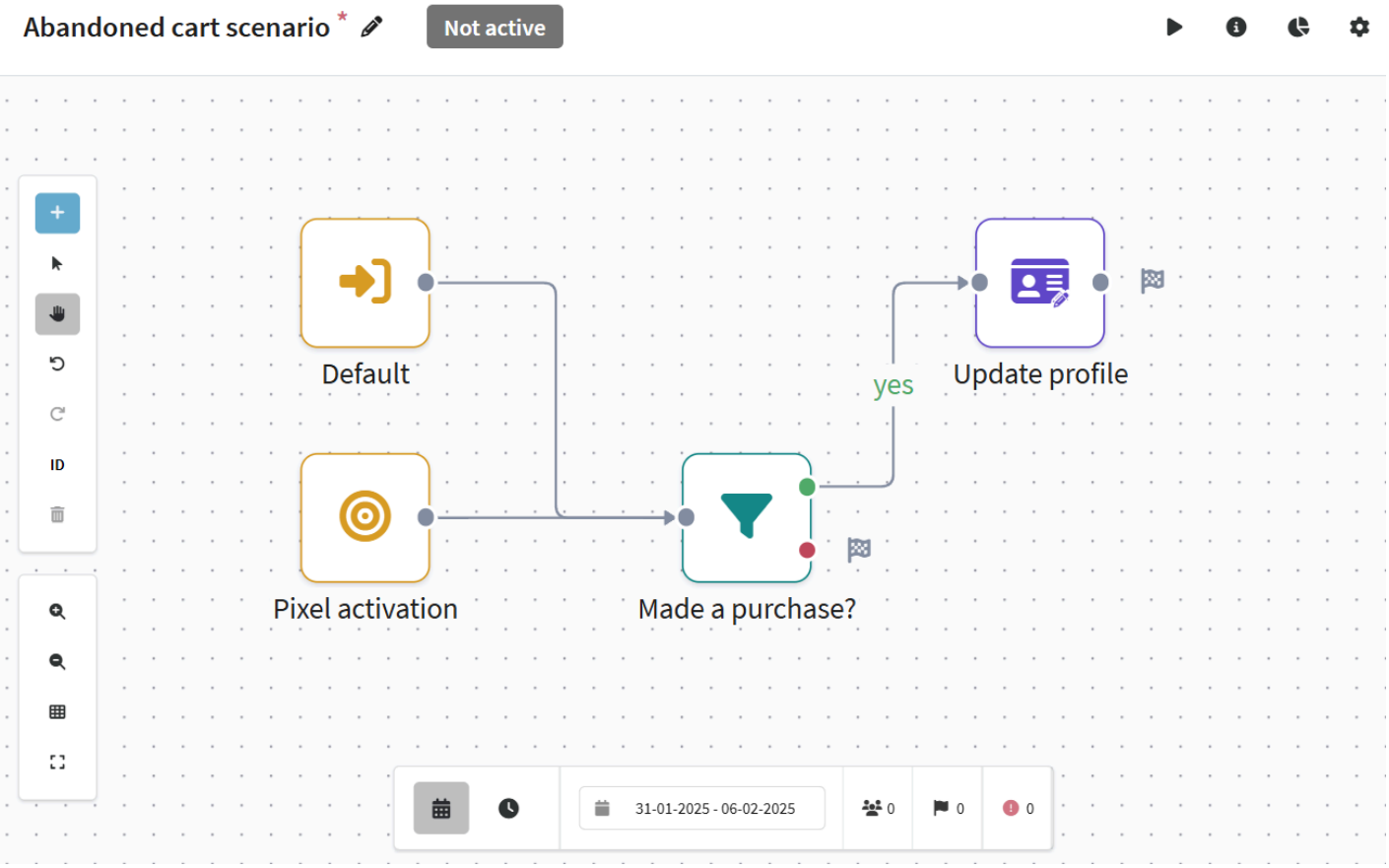
If the customer has made a purchase, award him bonus points. To do this, there should be an additional numeric field in the profile database, such as loyalty_points. Add a Update Data element to the scenario and increment the field value:



TIP

If you want to change the number of bonus points depending on the purchase amount, use the "Take value from `{apicontent}` field‘ action instead of “Increase by”. To do this, you need to add the API Request element in the previous step and save the http response with the required data.

So, the positive branch of scenario is ready:

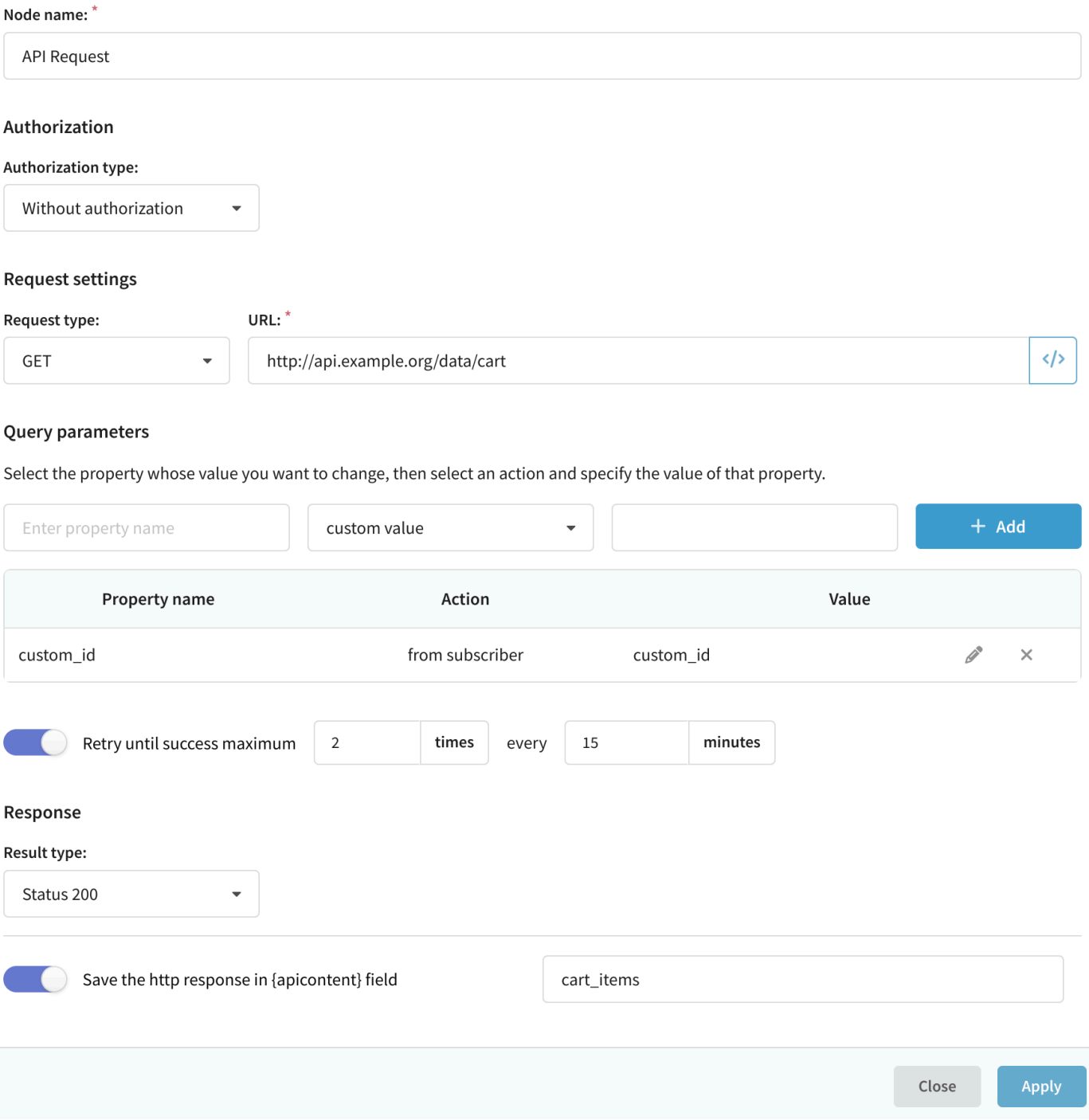


Step 4. Set up a workflow for the profiles that have left the website

Let's go back to the customers who left the site without completing the checkout. After the condition element, the scenarion will redirect them to the red branch.

Using the API call, we need to pass the data about the products abandoned in the cart to the platform, and then substitute this data into the email template. Add an API Request element. Specify the URL where the data about the items in the cart is stored. The request type is GET.

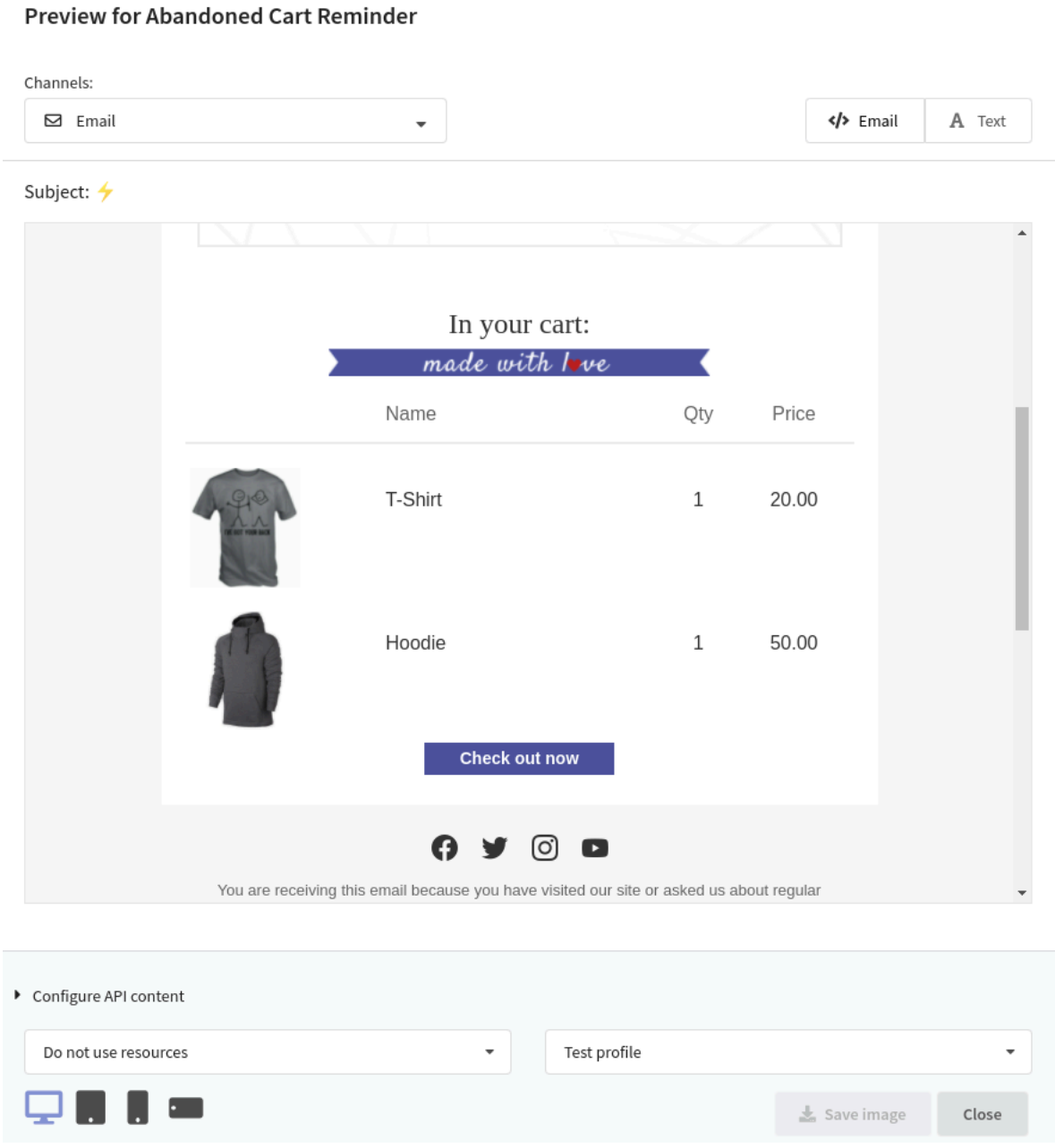
Next, activate the field "Save the http response in {apicontent} field" and specify the name of the field where the response will be saved. The name of the field can be arbitrary, for example, `cart_items`. It is this field that will later be used when substituting the variable into the template. Add the necessary query parameters, such as id or profile email, which will be used to find the desired cart:



Save the scenario at this point. All that remains is to create a message template and add the distribution element to the script.

Step 5: Create an email template

Go to the Templates section and create a new template for the email sending channel:



The content of the email should be dynamically updated depending on the recipient. The `{apicontent}` variable is used for this. It stores the data that you receive using an API call node. For example, a JSON object with data about products in the purchaser's cart.

Example of a JSON cart

```
[
  {
    "product_id": 1,
    "product_name": "T-Shirt",
    "product_price": "20.00",
    "product_quantity": 1,
    "product_image_url": "https://example.com/pic/jersey.png"
  },
  {
    "product_id": 2,
    "product_name": "Hoodie",
    "product_price": "50.00",
    "product_quantity": 1,
    "product_image_url": "https://example.com/pic/hoodie.png"
  }
]
```

Once you've saved a JSON object in the apicontent field, you'll have access to the elements of this object by key. For example, `{apicontent.cart_items.field_name1}`. To add a variable,select from the list (`</>`) Services —> API content.

If the product data is in an array, use loops:

```
In your cart:

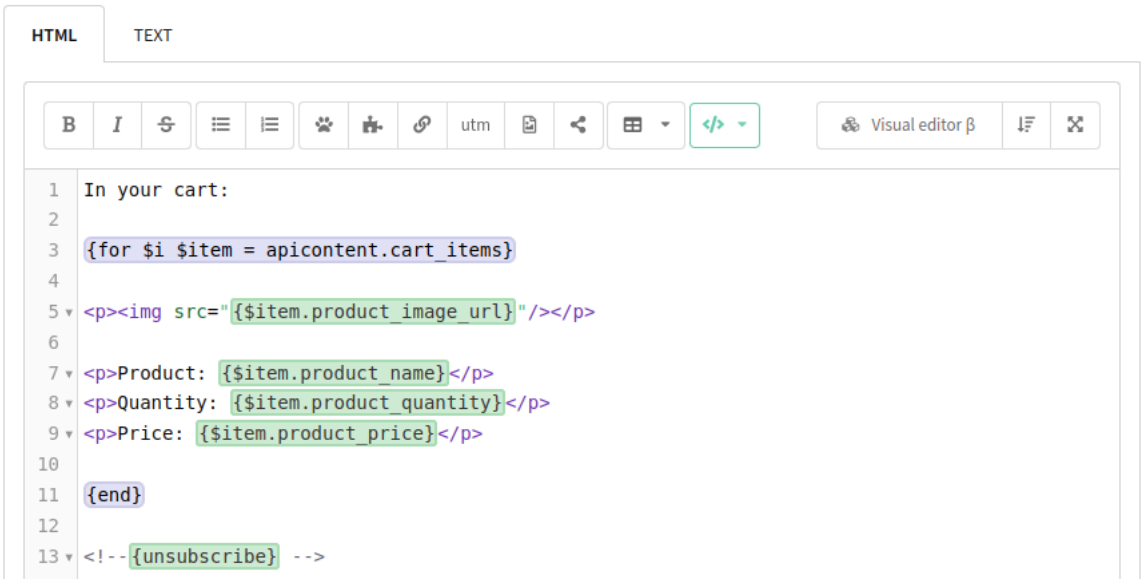
{for $i $item = apicontent.cart_items}

<p><img class="img" src="{$item.product_image_url}"/></p>

<p>Name: {$item.product_name}</p>
<p>Quantity: {$item.product_quantity}</p>
<p>Price: {$item.product_price}</p>

{end}

<!--{unsubscribe} -->
```



Step 6: Check the preview of the apicontent variable

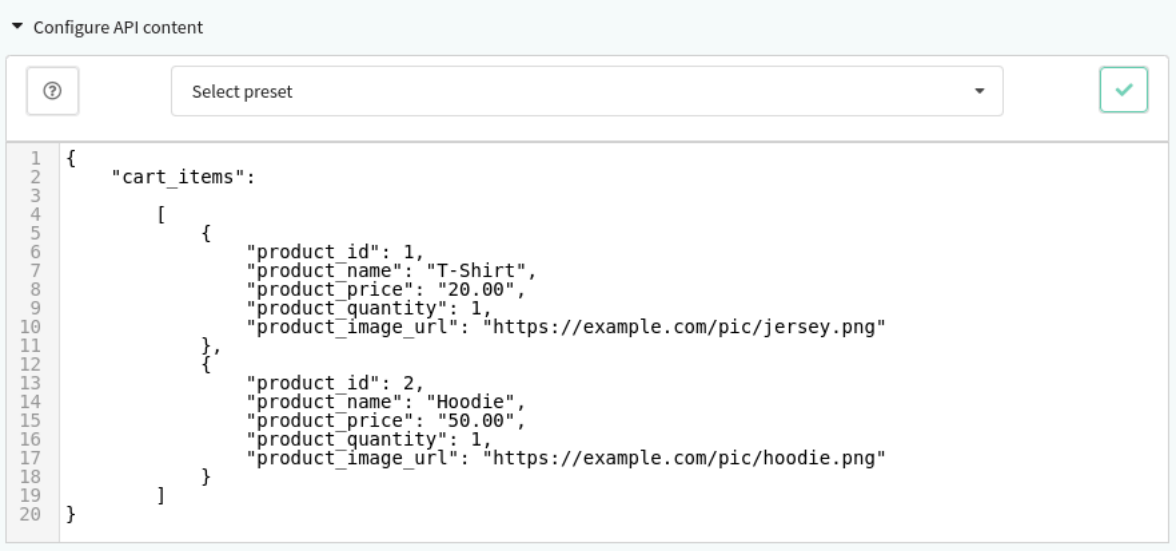
When adding the apicontent variable, the message preview will not work because the variable doesn't contain any information. If you want to know what message will look like, set up the preview window.

Open the preview mode

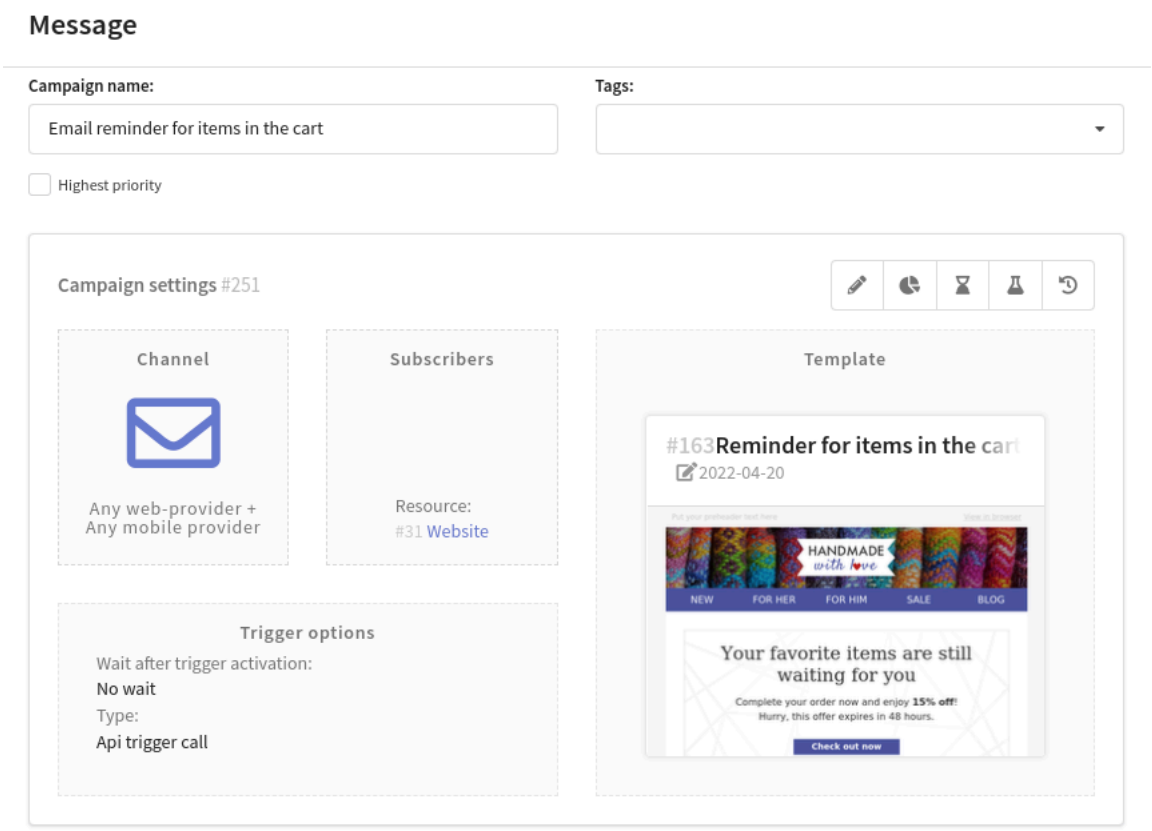
and find "Configure API content" below. Add a JSON object similar to what you'll receive after the API request in the scenario.

Please note that the test array of products must be stored in the `cart_items` field.

Click on the check mark. The values of the field will be added to the template:



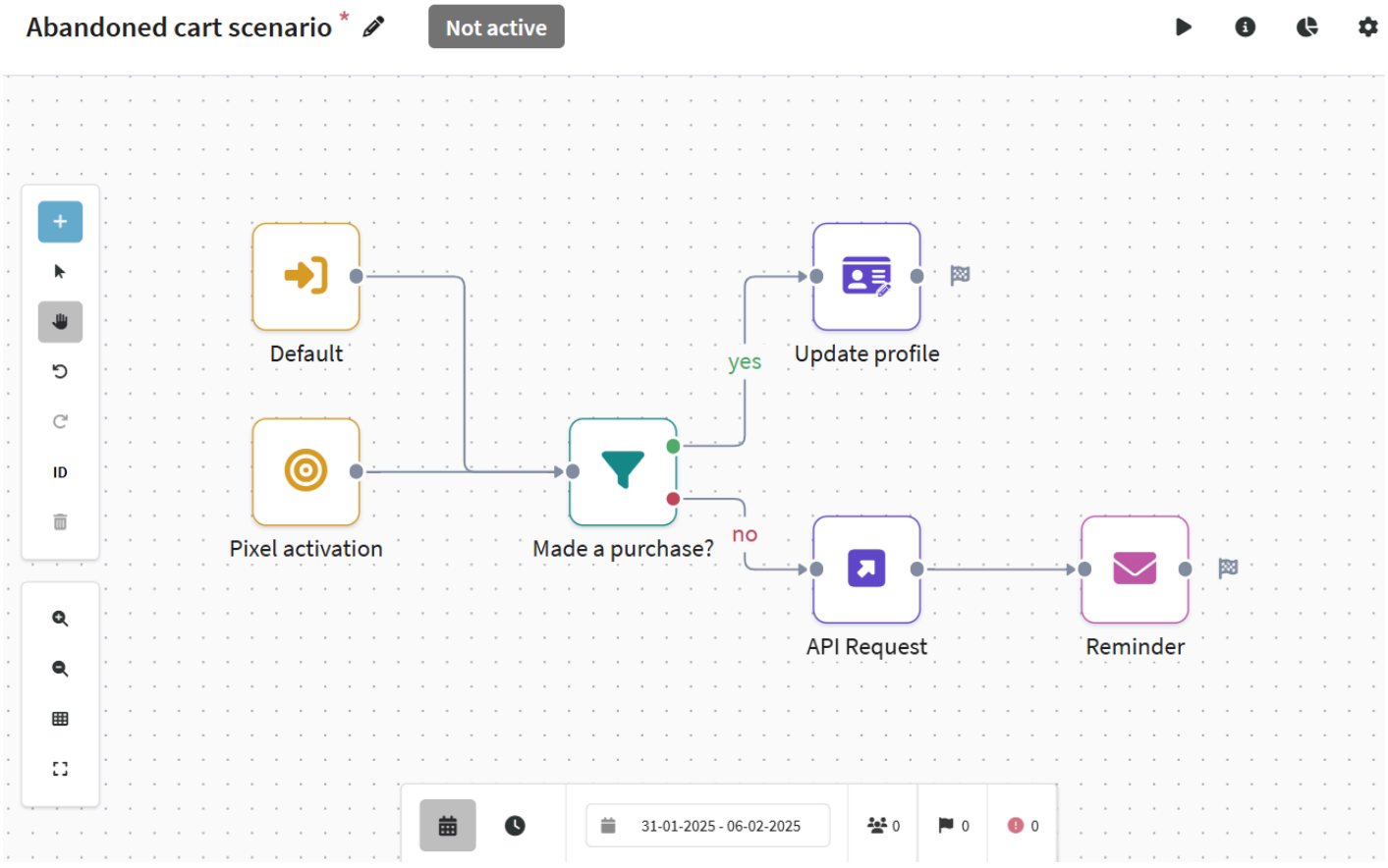
Go back to the scenario and add an Email element from Channels category. Select email channel, the resource of subscribers and created template in the settings:



Save the scenario by clicking the appropriate button at the top of the screen.

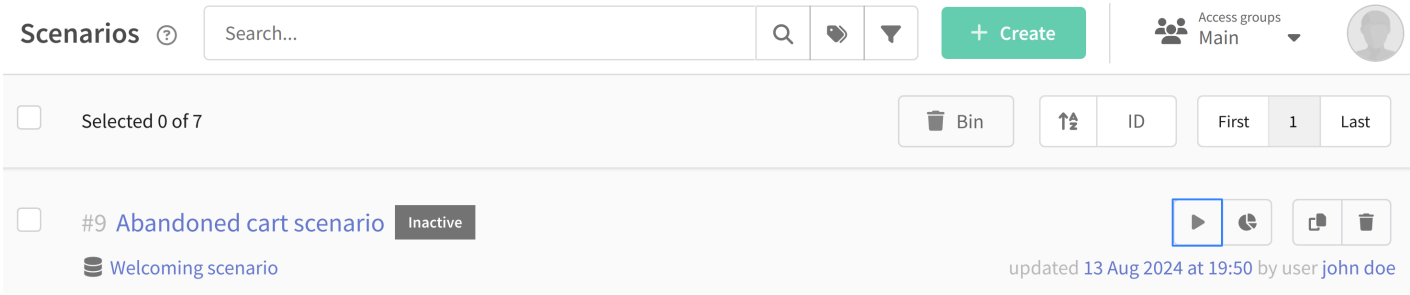
Step 7. Activate the scenario and evaluate its effectiveness

The finished scenario looks like this:



The scenario can be expanded. For example, if the subscriber has not completed the purchase, give them a promo code or send a compilation of related products.

After saving, activate the scenario in the quick actions menu:



You can evaluate the effectiveness of the scenario by clicking on the corresponding icon in the quick actions menu or in the channel report.

Market

Market allows you to transfer product and order data to the platform and use it in customer communications. Data can be transferred automatically via API or a product feed, or manually by uploading a file.

The imported data can be used for the following purposes:

- **inserting order details into message templates;**
- **triggering events based on order status changes;**
- **segmenting customers based on purchase data.**

Before you get started, make sure you have the required roles enabled for the user and token:

To work with the Market module, we recommend using API requests. If you are using the platform interface, please follow the instructions below:

Market settings

	This instruction contains information on such market entities as sales channels, custom fields, order item statuses, regions, categories and currencies. To get started, select the "Market" section, and then "Settings".

Sales channels

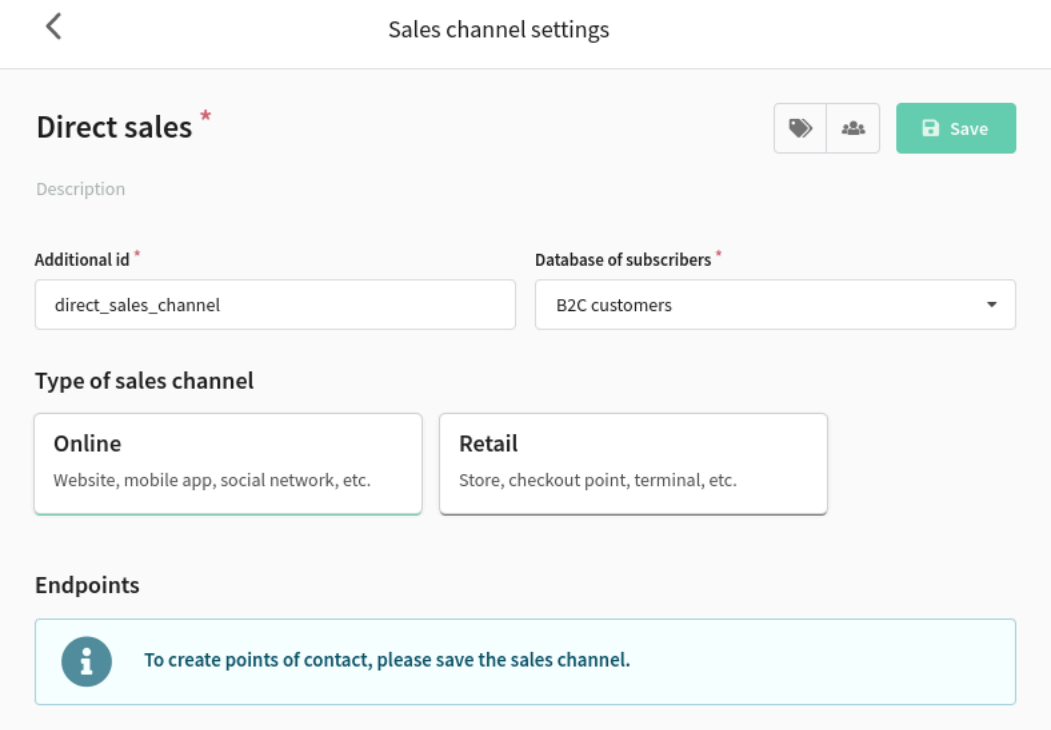
A sales channel is the way customers get to a company to order products they are interested in. It covers platforms and tools used for sale (stores, websites, terminals or cash registers where the order is placed). The key task of the channel is to determine the customer base and touchpoints.

Click Create to create a new sales channel. You will see a window, configure the channel in it:

- **type a name and description for the channel**
- **additional ID**
- **add tags**

  and access groups

- **specify the profile database (channel purchasers)**
- **select the type of channel (online or retail)**



CAUTION

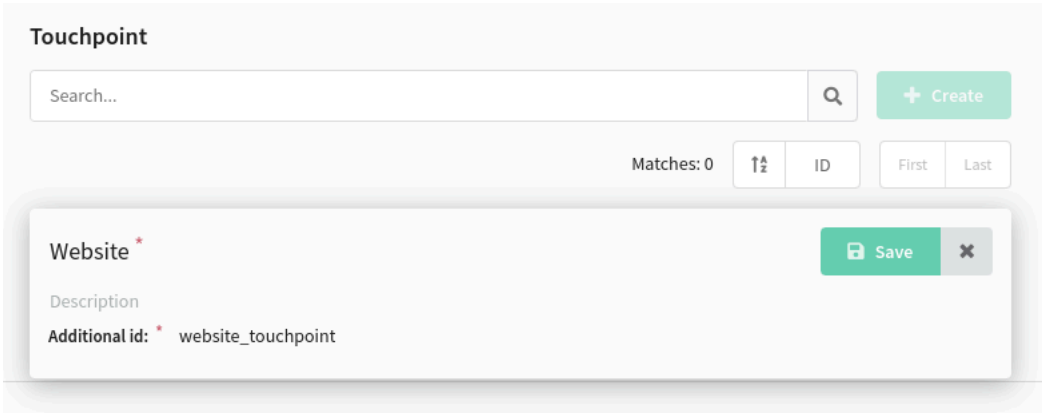
Once the channel has been created, the specified profile database cannot be changed.

Save the channel and then select it from the general list to create a touchpoint.

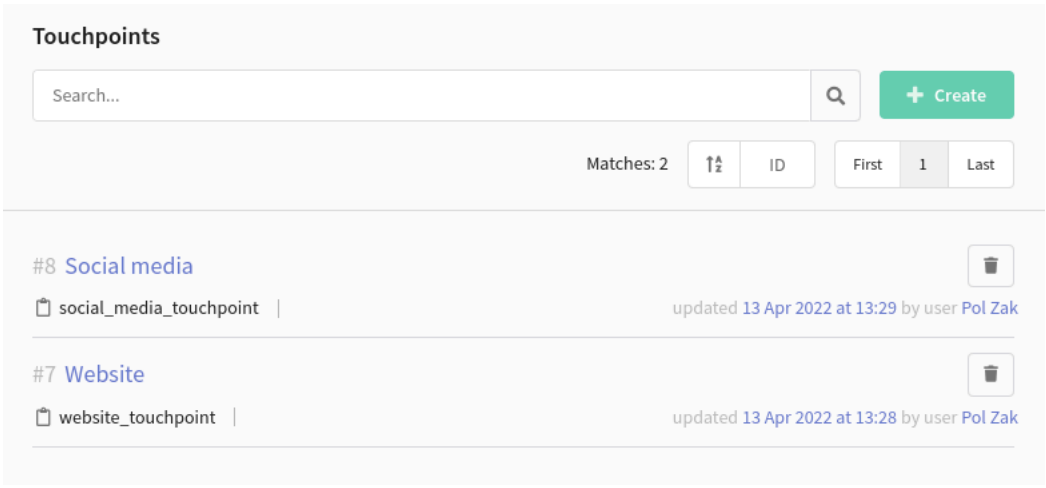
TIP

A touchpoint is a specific platform or tool by means of which you sell a product to your customer. Touchpoints connect the sales channel and the order. Some examples of touchpoints: a website, a store, a parcel terminal, a cash register where the order is placed, a social network, a call centre, etc.

Type a name, description and additional ID for the touchpoint. Then click "Save:



Each touchpoint can be linked to only one channel. At the same time, the channel may have several touchpoints:

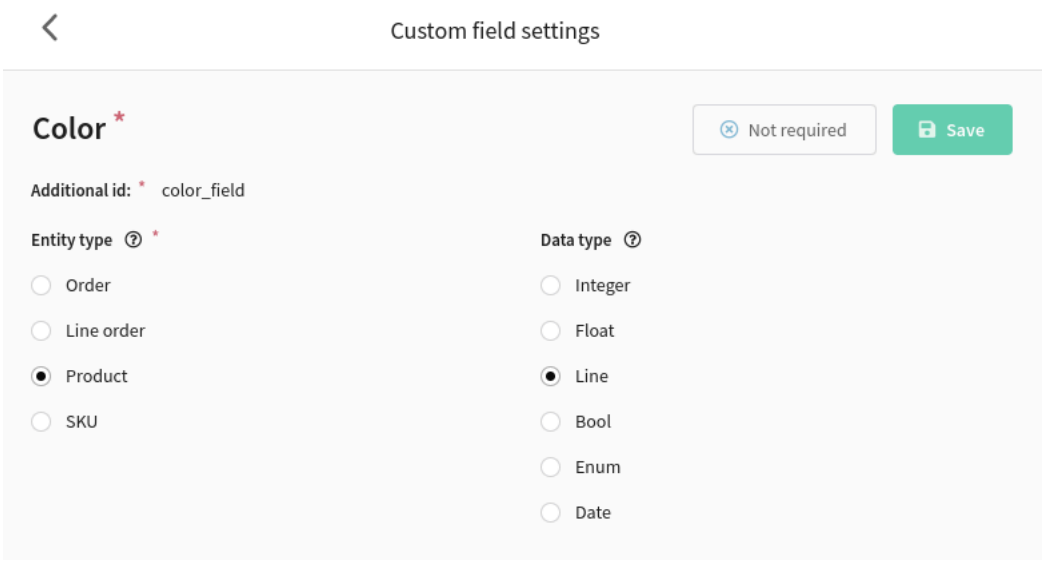


Custom fields

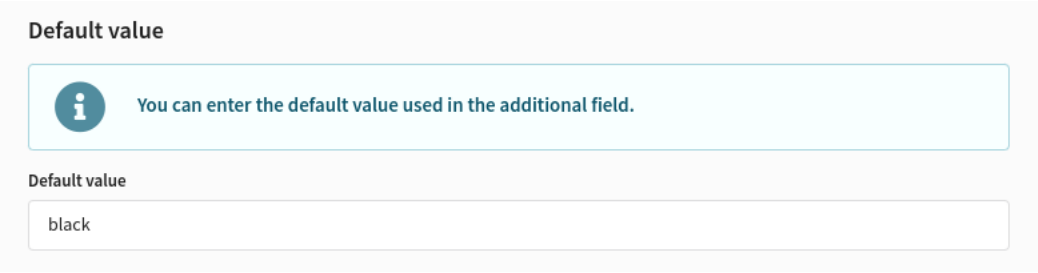
Custom fields are set for entities such as order, order item, product and SKU. There you can store information about important product features — colour, size, volume, material, model, weight, etc.

Select the type of entity and specify the type of data that will be stored in the new field:

- **whole number**
- **floating-point number**
- **line**
- **boolean**
- **list**
- **date**



It is possible to set a default data value. It means that the value will be automatically substituted in an custom field of the specified entity.

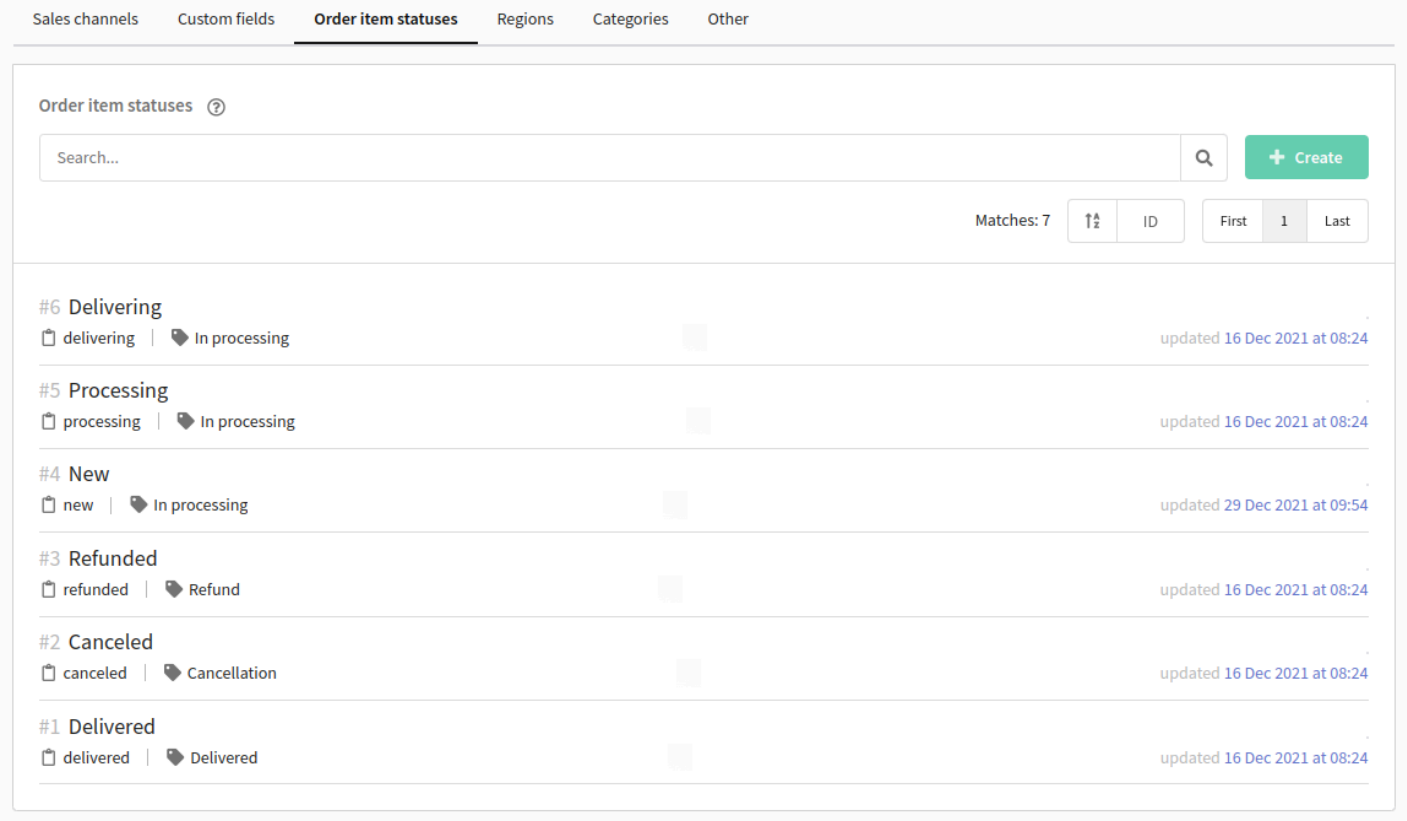


If you need to make the field required, change Optional to Required":

Order item statuses

Statuses are available by default:

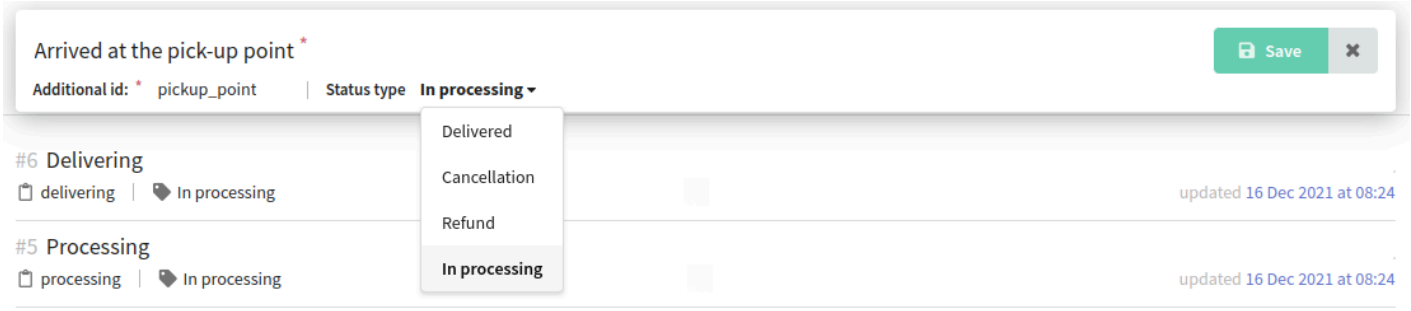
- **delivering**
- **processing**
- **new**
- **refunded**
- **canceled**
- **delivered**



You can also create your own statuses. For example, if you need to segment the delivery process into several stages:*"received by delivery service", "arrived at the pick-up point", "arrived at the local courier facility".*

Use the "Create button to add a new status. Enter a name, ID and type for the status. Four types of statuses are available:

- **delivered**
- **cancellation**
- **refund**
- **in processing**

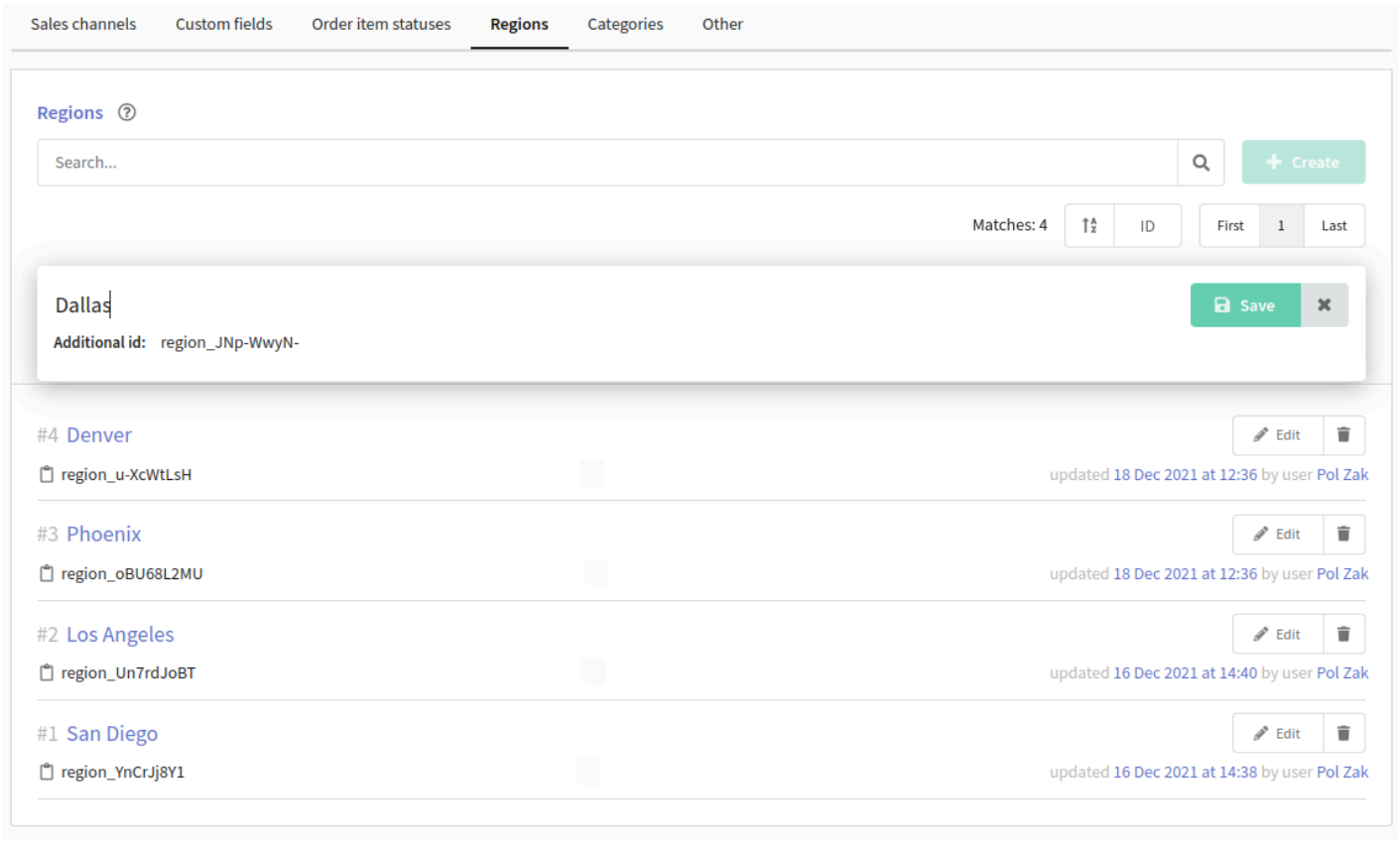


After filling in all the fields, click Save.

Regions

Regions are specified when setting up the order. You can set regional data in products and SKUs. For example, availability, price, delivery details, etc.

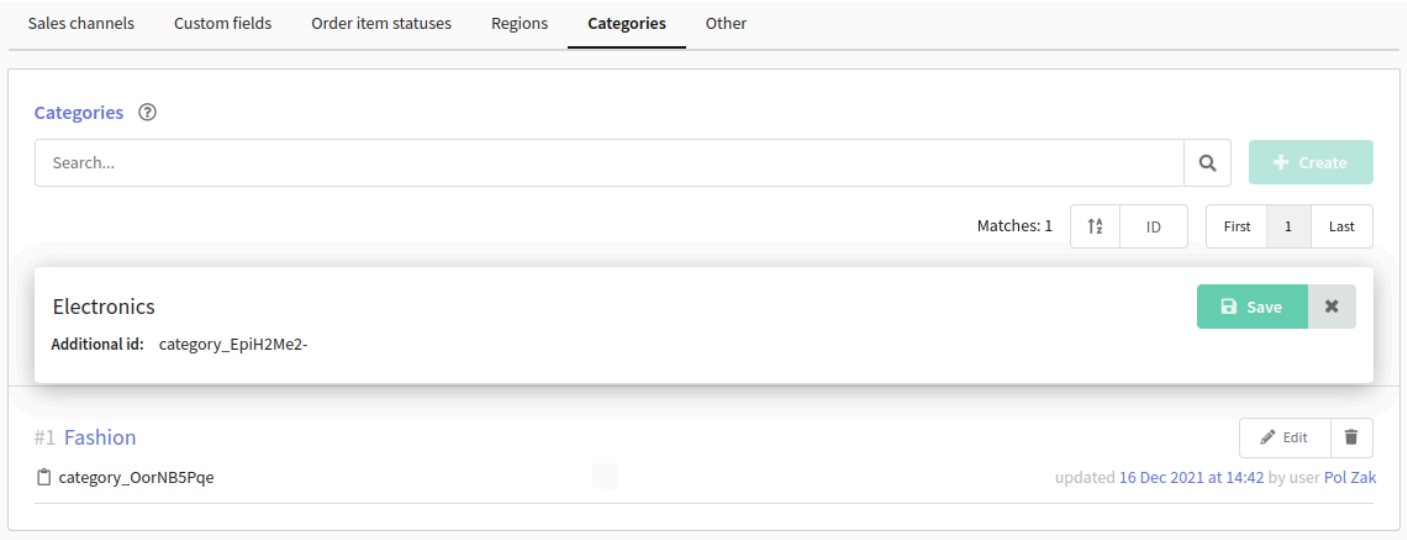
When creating a region, enter its name and edit the ID if necessary. Then click Save:



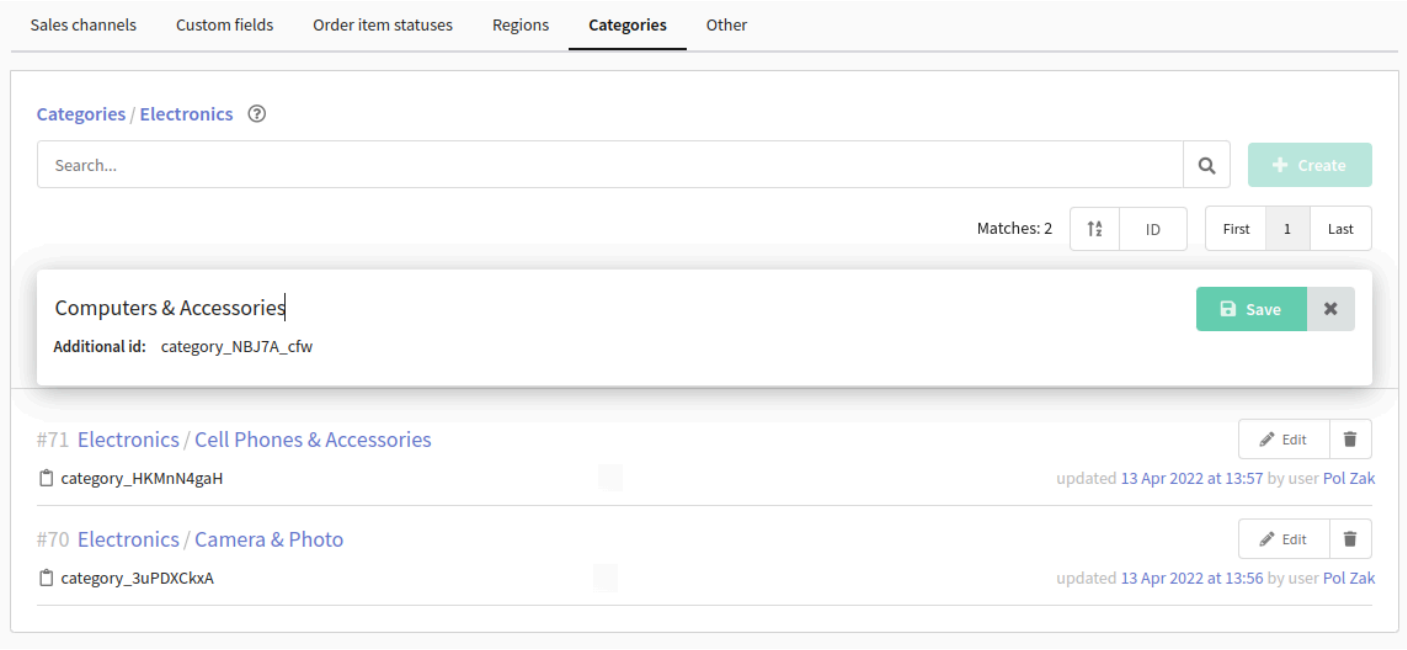
Categories

A category is a group of products that seem similar to the customer. You can combine products and SKUs in a category.

Click Create" to create a new category. Type a name for the category and change its ID if necessary. Then save the category:



Each category may be divided into several subcategories. Click on the name of the category if you want to create a subcategory. Create and save the object:

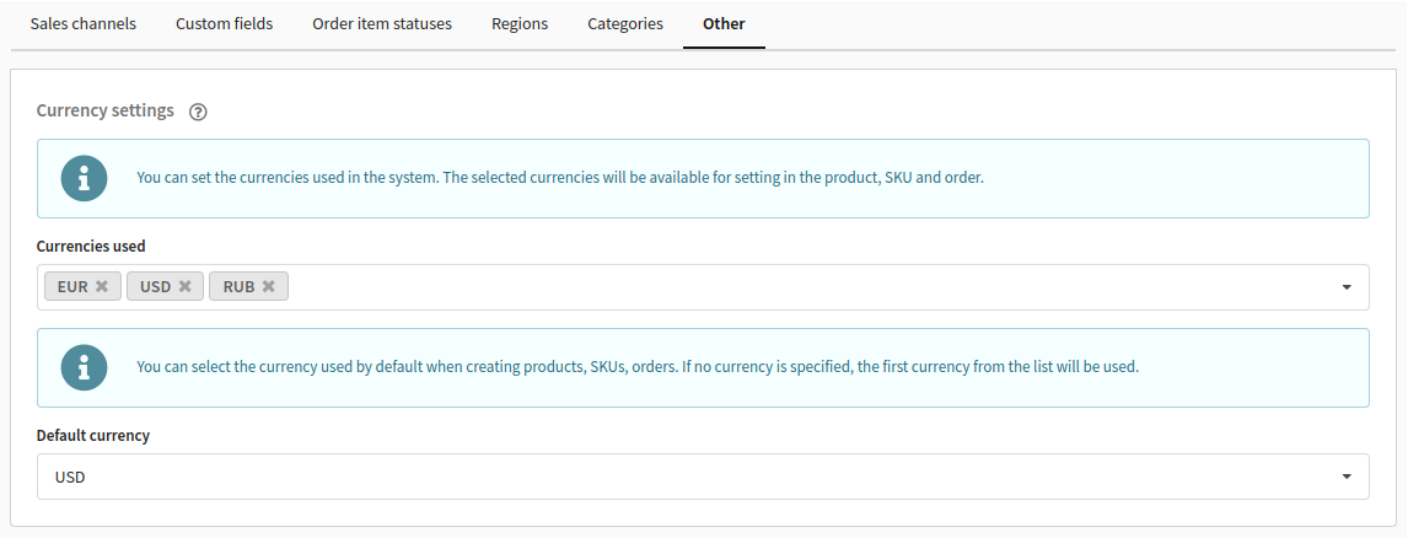


Other

Here you can set currencies used in the market. Select necessary currencies from the drop-down list.

CAUTION

Product or SKU will not be imported through API if its currency is not stated in the Currencies field.



You can select a default currency. It will be automatically set when creating products, SKUs and orders.

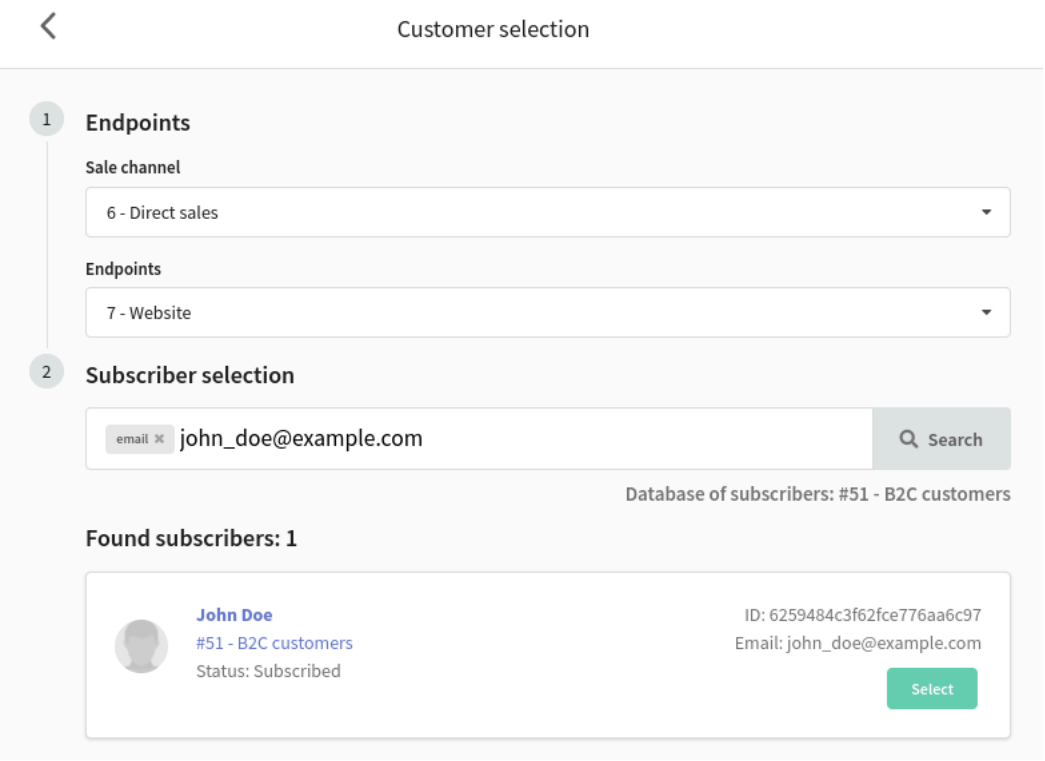
Orders

	You can create and process orders coming from various sales points in the "Orders" section. It stores information on orders that have already been placed, including their current statuses. To create a new order, select a customer and add products or SKUs to the order. You can [import orders using API](/en/developer-guide/market/orders/56249746/) or create it manually in the platform interface. To get started with orders, go to "Market" → "Orders".

Selecting a customer

To create an order manually, enter its number. Then specify the buyer by clicking " Select a customer ":

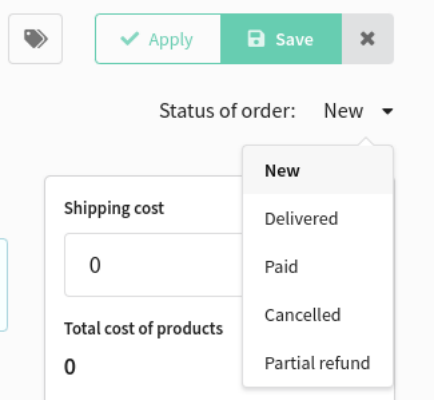
A new window will open. Select a sales channel and a touchpoint in it. Each channel has a profile database attached to it. Use the “Select customer” button to find the desired profile in the database by its email, phone number, custom field or any other available identifier. The window will automatically close when you click “Select”:



Current order status

Available order statuses:

- **new**
- **delivered**
- **paid**
- **cancelled**
- **partial refund**

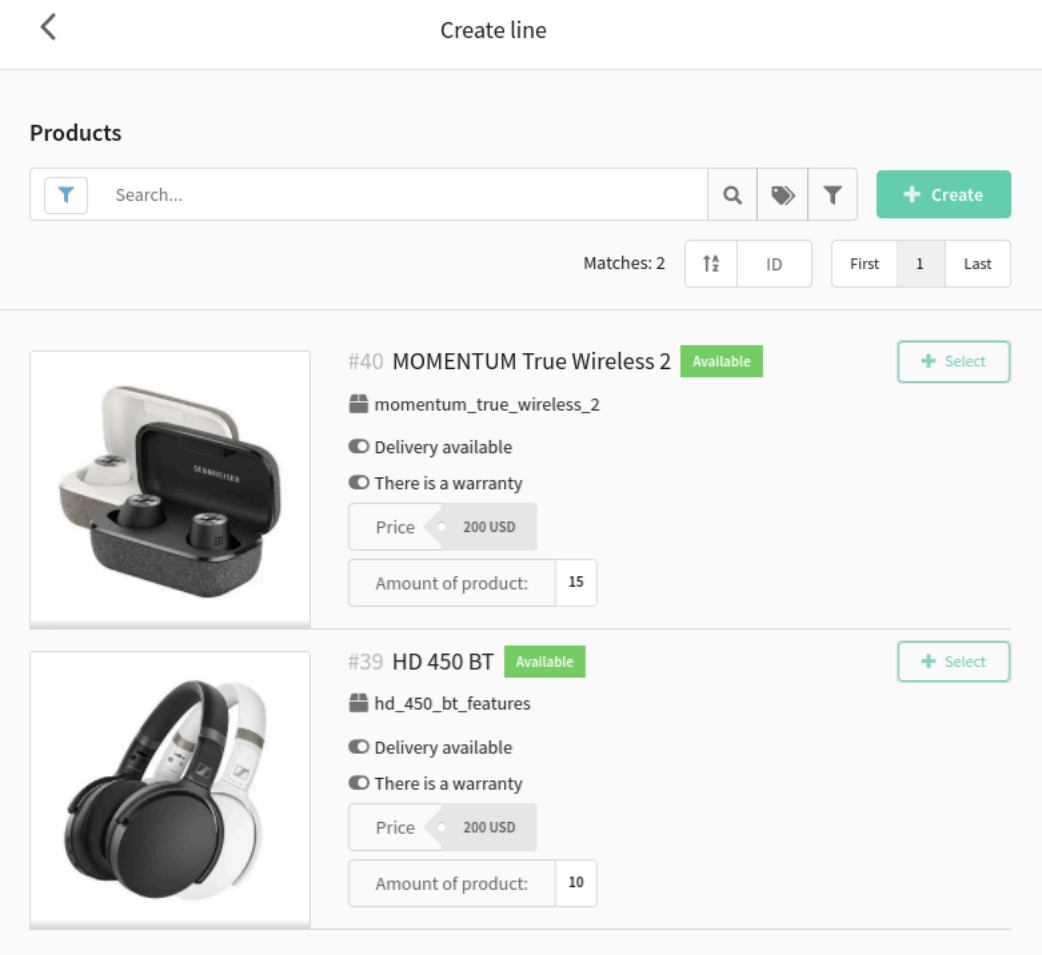


Adding an item to an order

An order comprises items. These can be products or SKUs. Several items of the same product in an order count as one item.

To add new items to an order, click “Add”:

In the new window, select the required items:

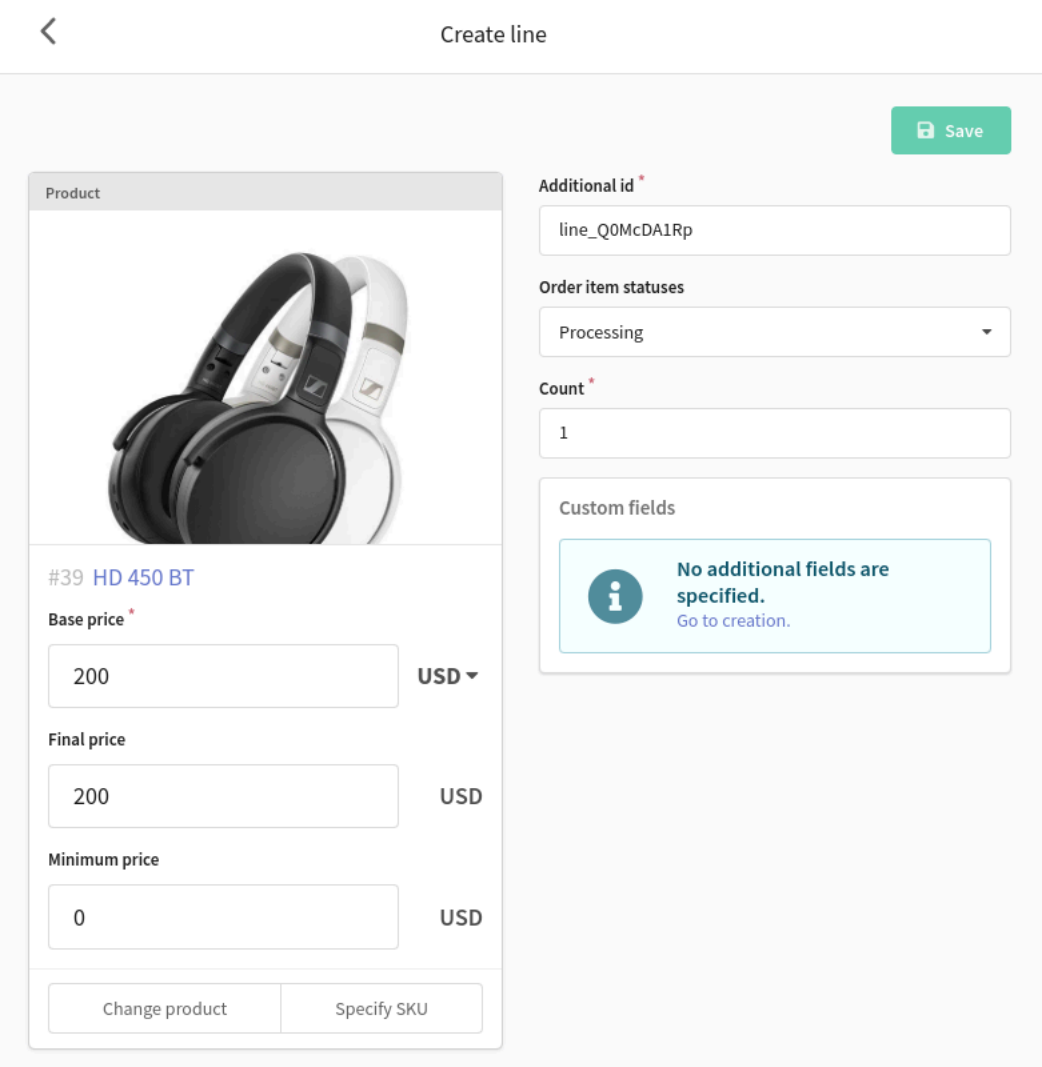


Configuring an item

You can edit each item if necessary. To do this, click on an item in the general list — a window for editing the following fields will appear on the right:

- **product price**
- **additional identifier (used in API requests);**
- **current status of the order item;**
- **number of products in the item;**
- **custom fields.**

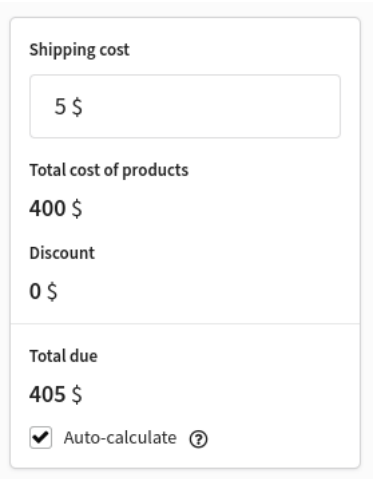
In this window, you can also change the product or specify the SKU.



Order price

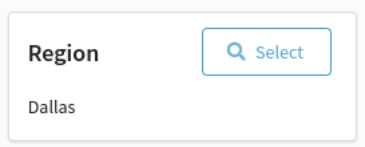
The total price of the order is shown in the right window. It comprises delivery price and the price of each item in the order.

If you need to recalculate the price manually, activate the “Auto Calculate” option.



Region

Select the region where the order was placed. Some characteristics of the order (price, values of custom fields, stock availability) may change depending on the region.

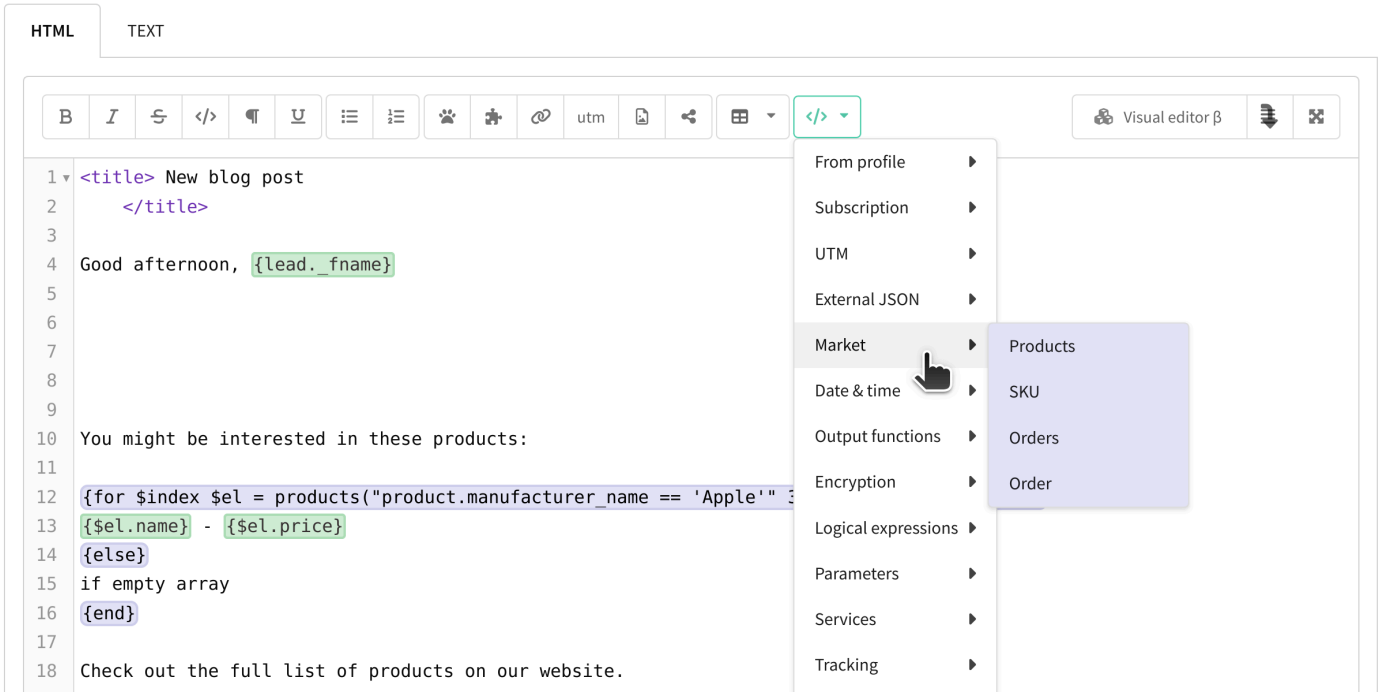


The order configuration is complete. Save it using the top bar buttons.

Market variables in message template

To use information about orders and products in your mailings, you need to add Market variables to your message template.

Open the message template and click the `</>` button in the editor. In the list, select Market and insert one of the variables:



TIP

For more information on using marketplace variables in messages, please refer to this article.

Guide: how to send an order confirmation email

Description

This guide will show you the algorithm for working in the Altcraft Platform market.

Let's imagine that we have an online store for smartphones. A buyer visits a website, adds the product to the cart and places the order. Our task is to send them an order confirmation email with the order number and price.

The guide comprises the following steps:

1. **Creating a sales channel and touchpoins**
2. **Importing product and SKU data using a yml file**
3. **Setting up an API request to add order data to the market**
4. **Creating a template and substituting order data using variables**
5. **Setting up a trigger mailing**

Step 1. Set up a sales channel

You need a sales channel to start working in the market. This channel defines a profile database and touchpoints.

A profile database contains the contact details of customers who place orders in a certain channel. You need to import their data into a new or existing database. To do this, use import from a file or via API.

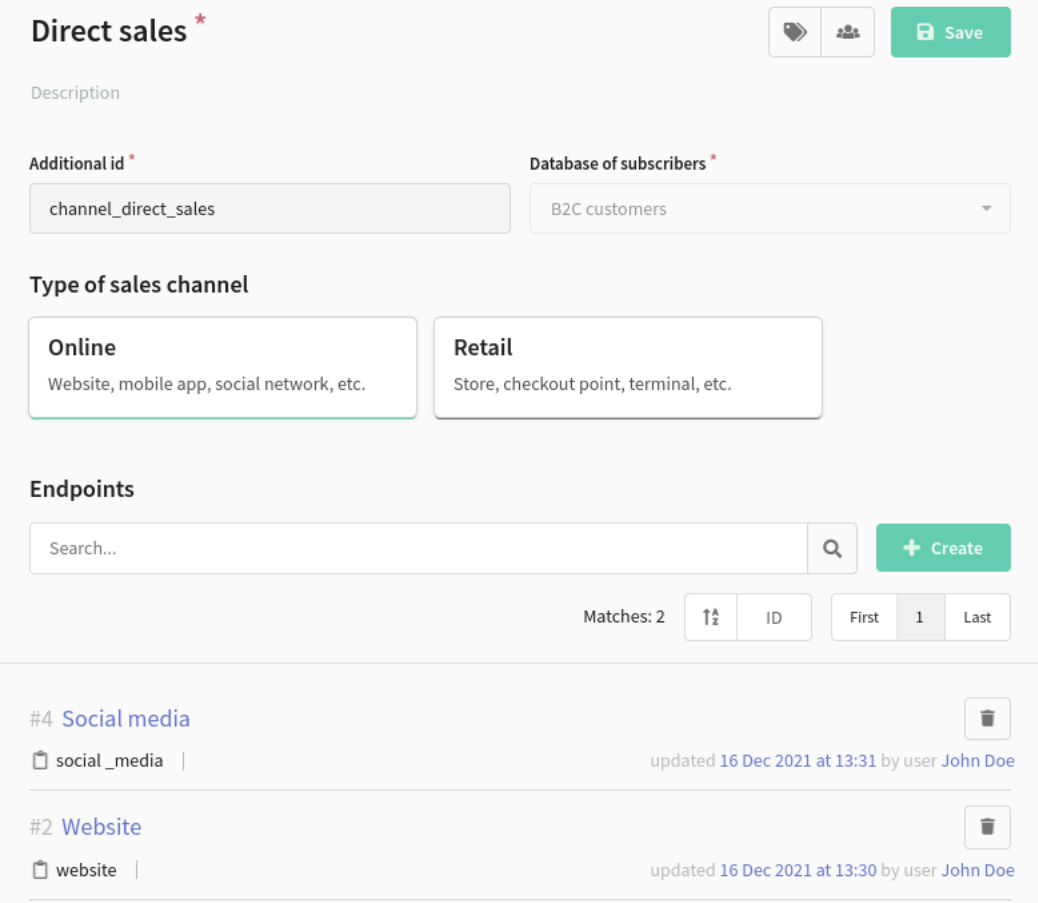
A touchpoint is a platform or a tool using which you sell products and services to your customers (a website, a store, a parcel terminal, a social network, a call center and so on). A touchpoint connects a sales channel a an order.

To create a channel, go to Market —> Settings in the main menu. Create a new channel on the Sales channels tab. Specify:

- **a channel name**
- **an additional identifier**
- **a profile database**
- **a channel type: online or retail**

Next, save the channel and go back to its settings. Create touchpoints.

Please note that each touchpoint can only be linked to one channel. The touchpoint ID will be used in the API request to import the order.



Step 2. Upload product data to the platform

Products in the market are represented by two entities: products and SKU.

INFO

A product is goods or services for sale. SKU is a product modifier, it is used for variations of one product (such characteristics as colour, size, weight, model, configuration and so on can vary).

SKU is connected to a certain product and cannot exist without it. For example, `Apple iPhone 12` **is a product, but** `Apple iPhone 12 128GB` **and** `Apple iPhone 12 256GB` **are its SKUs.**

There are three ways to add product data to the platform:

- **manually**
- **import it from a file**
- **import it via API**

Let's import product data from a file. To do this, you need a yml file.

TIP

If you have any questions about import, read this detailed guide.

Below you can see examples of yml files:

Product

A file with such content will add two products to the market: *Apple iPhone 12* and *Xiaomi 12X.*

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2017-02-05 17:22">
    <shop>
    <name>BestSeller</name>
    <company>Tne Best inc.</company>
    <url>http://best.seller.ru</url>
    <categories>
      <category id="10">Смартфоны</category>
      <category id="101" parentId="10">Apple</category>
      <category id="102" parentId="10">Xiaomi</category>
    </categories>
    <offers>
      <offer id="iphone12" available="true">
        <name>Apple iPhone 12</name>
        <vendor>Apple</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12348</url>
        <price>83990</price>
        <oldprice>89990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>101</categoryId>
        <picture>https://example.com/img/apple_phone.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0156789012</barcode>
          </offer>
      <offer id="xiaomi12" available="true">
                <name>Xiaomi 12X</name>
        <vendor>Xiaomi</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12345</url>
        <price>62990</price>
        <oldprice>69990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>102</categoryId>
        <picture>https://example.com/img/xiaomi_phone.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0123456789379</barcode>
          </offer>
    </offers>
  </shop>
</yml_catalog>
```

SKU

A file will add two SKUsfor *Apple iPhone 12*: *Apple iPhone 12 128GB* and* Apple iPhone 12 256GB*.

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2017-02-05 17:22">
    <shop>
    <name>BestSeller</name>
    <company>Tne Best inc.</company>
    <url>http://best.seller.ru</url>
    <offers>
      <offer id="apple12_128gb" group_id="iphone12" available="true">
        <name>Apple iPhone 12 128GB</name>
        <vendor>Apple</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12358</url>
        <price>71990</price>
        <oldprice>72990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>101</categoryId>
        <picture>https://example.com/img/apple_phone_128.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0156789012</barcode>
          </offer>
      <offer id="apple12_256gb" group_id="iphone12" available="true">
                <name>Apple iPhone 12 256GB</name>
        <vendor>Apple</vendor>
        <url>http://best.seller.ru/product_page.asp?pid=12359</url>
        <price>83990</price>
        <oldprice>89990</oldprice>
        <currencyId>RUB</currencyId>
        <categoryId>102</categoryId>
        <picture>https://example.com/img/apple_phone_256.png</picture>
        <delivery>true</delivery>
        <manufacturer_warranty>true</manufacturer_warranty>
        <barcode>0156789012</barcode>
          </offer>
    </offers>
  </shop>
</yml_catalog>
```

TIP

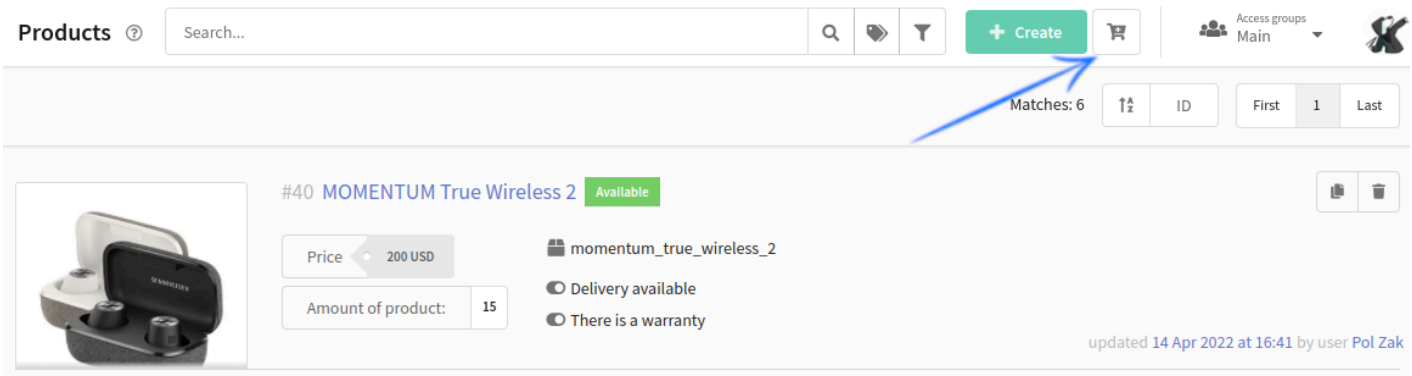
A list of all products and SKUs is stored in the `offers` element. One product or SKU is stored in a separate `offer` element.

The `id` attribute, which is unique for each product, is specified for the `offer` element.

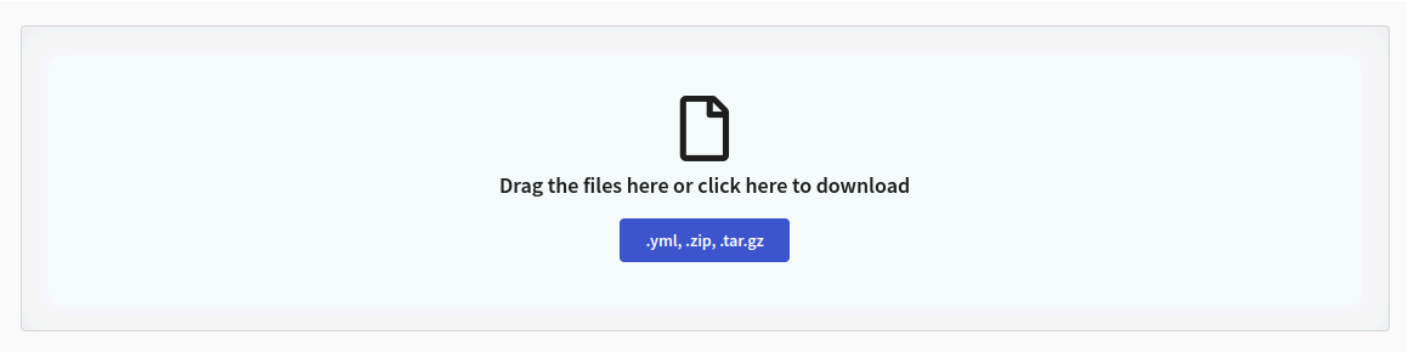
The structure of the yml file is almost the same for products and SKUs. The only difference is the following: when importing SKU for the `offer` element you need to specify not only `id`, but also `group_id` — a product identifier.

Please note that SKUs are different variations of one product, so the value of the `group_id` attribute must be the same for all SKUs of this product.

To import a file with a list of products, go to Market —> Products. Click on the icon on the top toolbar, and then select what exactly you want to import — a product.



Click on the download area and select a file on your device:



Start the import process by clicking "Import" on the top toolbar.

Then import SKUs. To do this, follow all the steps described in this paragraph.

Step 3. Set up sending an API request when adding an order

When a visitor of an online store places an order, an API request with the order data must be sent to the platform.

URL

Method: POST

`https://example.com/api/v1.1/market/orders/import`

Request parameters

Parameter	Type	Example	Required	Description
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Да**	**API token. Make sure that roles that allow you to work with the market are available for the token.**
data	**array of objects**	[{ … }]	**Да**	**Order data. Learn more.**
Data object				
external_id	**string**	**“1234001”**	**Yes**	**Order ID**
matching	**string**	**“email”**		**Subscriber search mode. By default — email. For each type of matching, certain fields must be passed in the request body. You can learn more about these fieldshere.**
endpoint	**string**	**“website”**	**Yes**	**Touchpoint ID**
status	**string**	**“new” “delivered” “paid” “canceled” “part_refunded”**	**Yes**	**Order status ID**
total_price	**string**	**“71990”**	**Yes**	**Total cost of the order. It consists of the price of each line in the order and the cost of delivery.**
currency	**string**	**“RUB”**	**Yes**	**Currency of the order price.**
create_time	**string**	**“2022-04-29T12:48:23Z”**	**No**	**Order creation time**
update_time	**string**	**“2022-04-29T12:48:23Z”**	**No**	**Order update time**
lines	**array**	[{ … }]	**Yes**	**Order lines data. At least one item.**
Lines object				
external_id	**string**	**“line_01”**	**Yes**	**Line ID**
product	**string**	**“iphone12”**	**Must not be present in the request if `sku` is specified**	**Product ID**
sku	**string**	**“apple12_128gb”**	**Must not be present in the request if `product` is specified**	**SKU ID**
base_price_per_item	**string**	**“71990”**	**Yes**	**Base price**
min_price_per_item	**string**	**“71990”**	**Yes**	**Minimum price**
final_price_per_item	**string**	**“71990”**	**Yes**	**Final price**
count	**int**	**1**	**Yes**	**Quantity of goods per item**

Request example

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "data": [
    {
      "external_id": "1234001",
      "matching": "email",
      "email": "johndoe@example.com",
      "endpoint": "website",
      "status": "new",
      "total_price": "71990",
      "create_time": "2022-04-29T12:48:23Z",
      "update_time": "2022-04-29T12:48:23Z",
      "currency": "RUB",
      "lines": [
        {
          "external_id": "line_01",
          "product": "iphone12",
          "base_price_per_item": "71990",
          "min_price_per_item": "71990",
          "final_price_per_item": "71990",
          "count": 1
        }
      ]
    }
  ]
}
```

Response example

```
{
  "error": 0,
  "error_text": "",
  "result": [
    {
      "error": 0,
      "external_id": "1234000"
    }
  ]
}
```

Response parameters

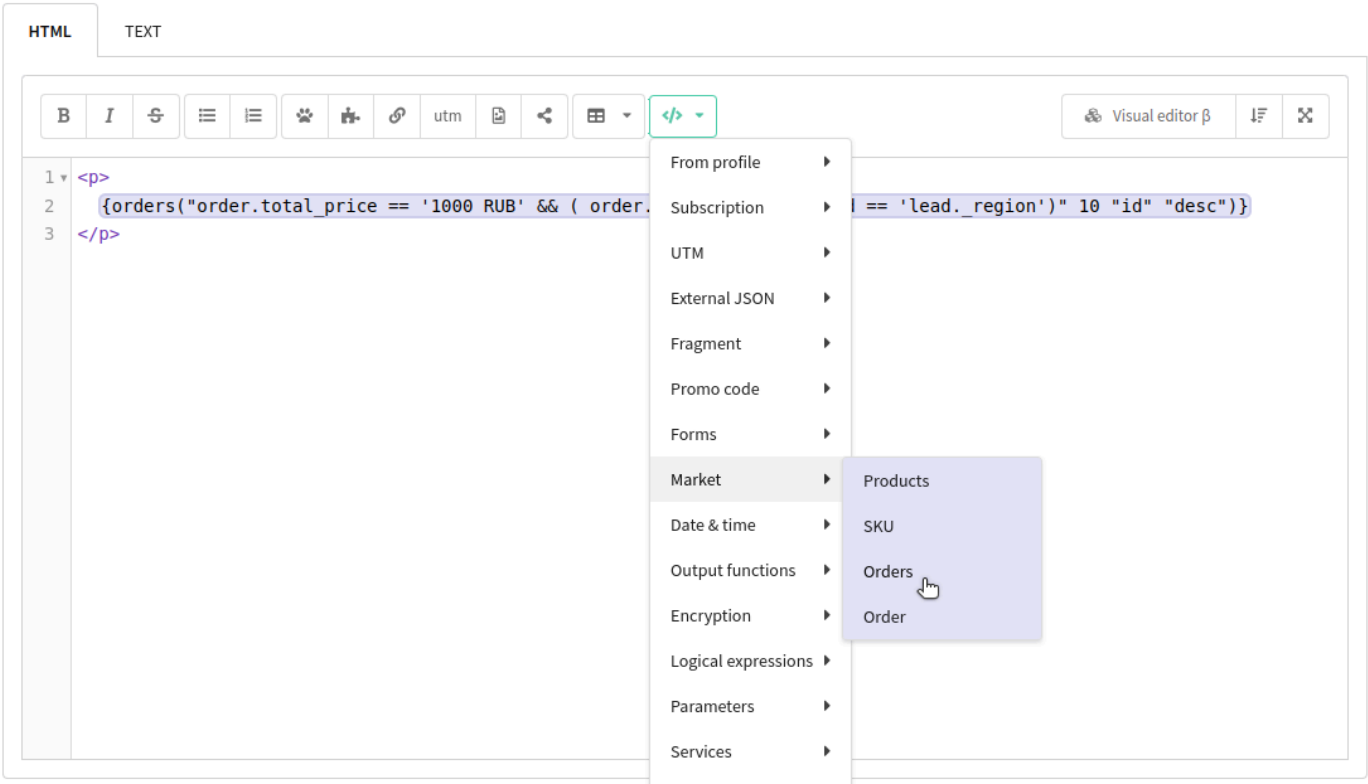
Parameter	Type	Description
error	int	Error code
error_text	string	Error text
result	array	IDs of orders that were successfully imported

Step 4. Create an email template

Go to the Templates section. Add a new template for the email channel.

Add the market variables to the email template to use the order information in your mailing. To do this, click `</>` in the editor. In the list, select “Market and insert the Orders” variable.

Theordersfunction pattern will be added to the editor. This function selects orders from the list of profile orders.



We only need to add information about the last order with the status “New” to our email. To do this, we will need to change the arguments of a function:

```
{orders("order.status == '1'" 1 "create_time" "desc")}
```

The function consists of four arguments:

Argument	Example	Description
Filter	**“order.status == ‘1’”**	**The function will select only those orders that match the condition “order status is new”.**
Limit	**1**	**The function will select only the first element in the list of orders.**
Sorting field	**“create_time”**	**Selected elements will be sorted by the “create_time” field (order creation time).**
Sorting direction	**“desc”**	**The elements will be sorted in descending order of the value in the “create_time” field.**

An array of orders is the result of the orders function.In this case, it will be an array with one item. Use loop to iterate an array.

In the email template, substitute two variables:

- `{$el.external_id}` — **additional order ID.**
- `{$el.total_price}` — **total order price.**

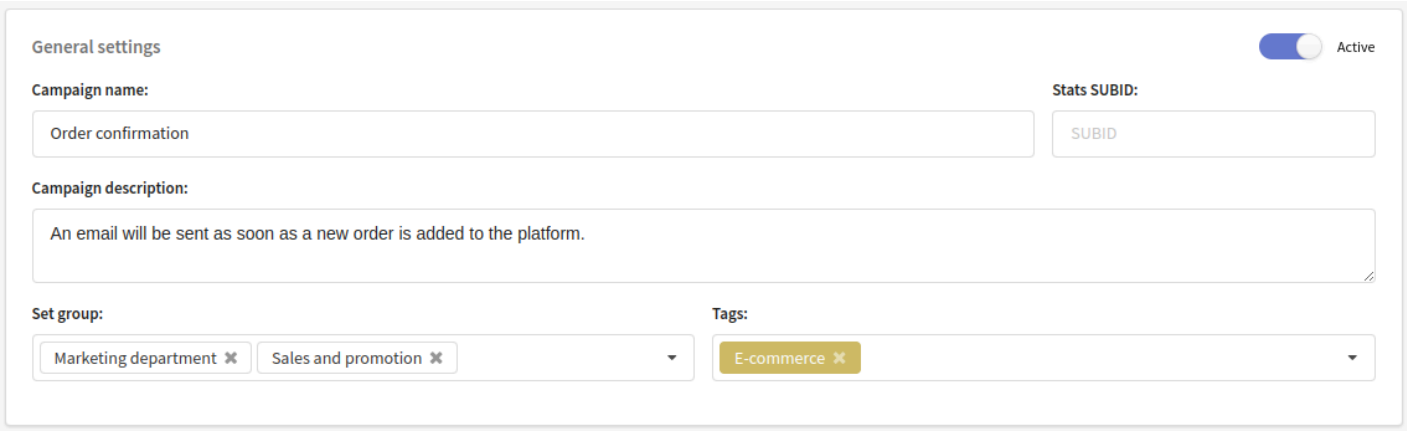
```
{for $index $el = orders("order.status == '1'" 1 "create_time" "desc")}
Thanks! Your order no.{$el.external_id} is confirmed.
Order total: {$el.total_price}
{else}
if empty array
{end}
```

TIP

Read more about market variables in this article. There you will find information about fields that can be used in the filter or inserted into the template.

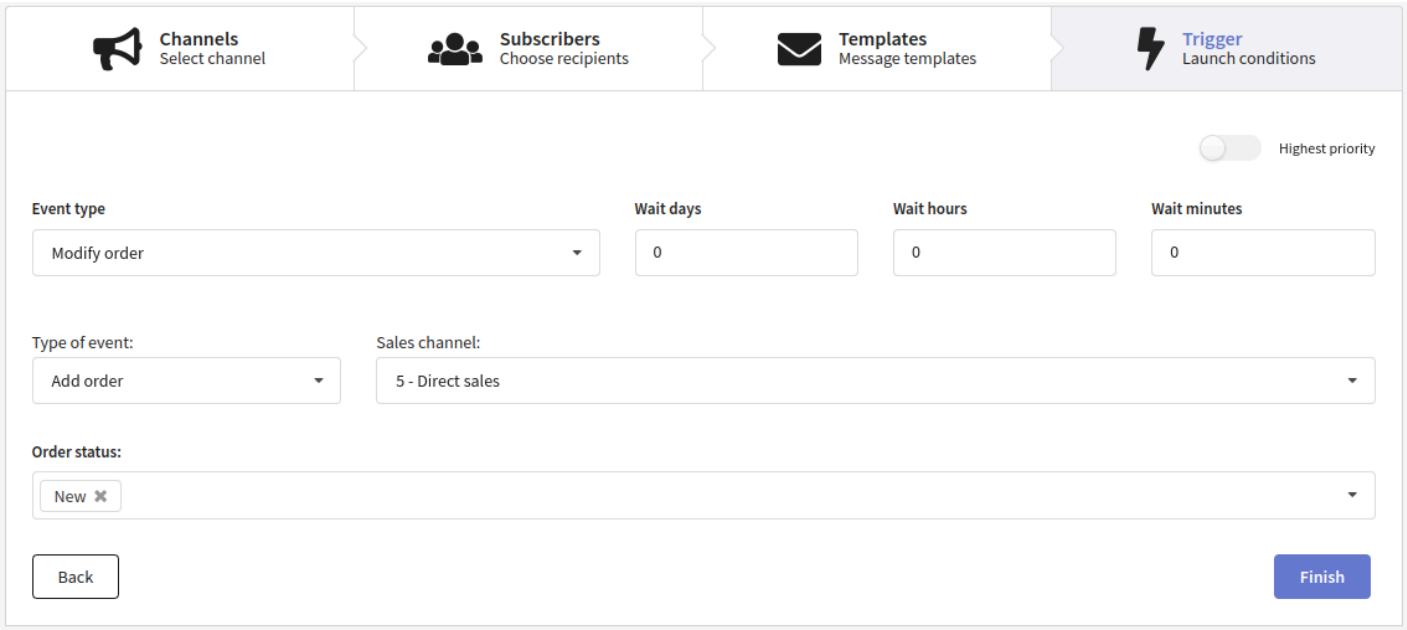
Step 5. Set up a trigger mailing

Open the Mailings section → Triggers. Create a new mailing. Don’t forget to activate it.



Select the email channel and add the desired email template. On the Trigger tab specify:

- **Event type: when changing an order —> adding an order. The trigger will be activated as soon as a new order is added to the market.**
- **Sales channel: here specify the profile database with customer data of your online store.**
- **Order status: new. The trigger will be activated if an order with the status "new" will be added to the market.**



Click Finish and save the mailing.

TIP

To test how the trigger works, you can add an order manually.

Products

	Products are goods or services for sale. Each product has its own card with a name, description, image, and other information. You can add multiple SKUs (product modifiers) to one product. To start working with products, open the " Market" section → "Products". </td> </tr>

There are several ways to create a product or SKU:

1. **Manually**
2. **Import a product from a file**
3. **Import via feed on schedule**
4. **Import via API**

How to create a product manually

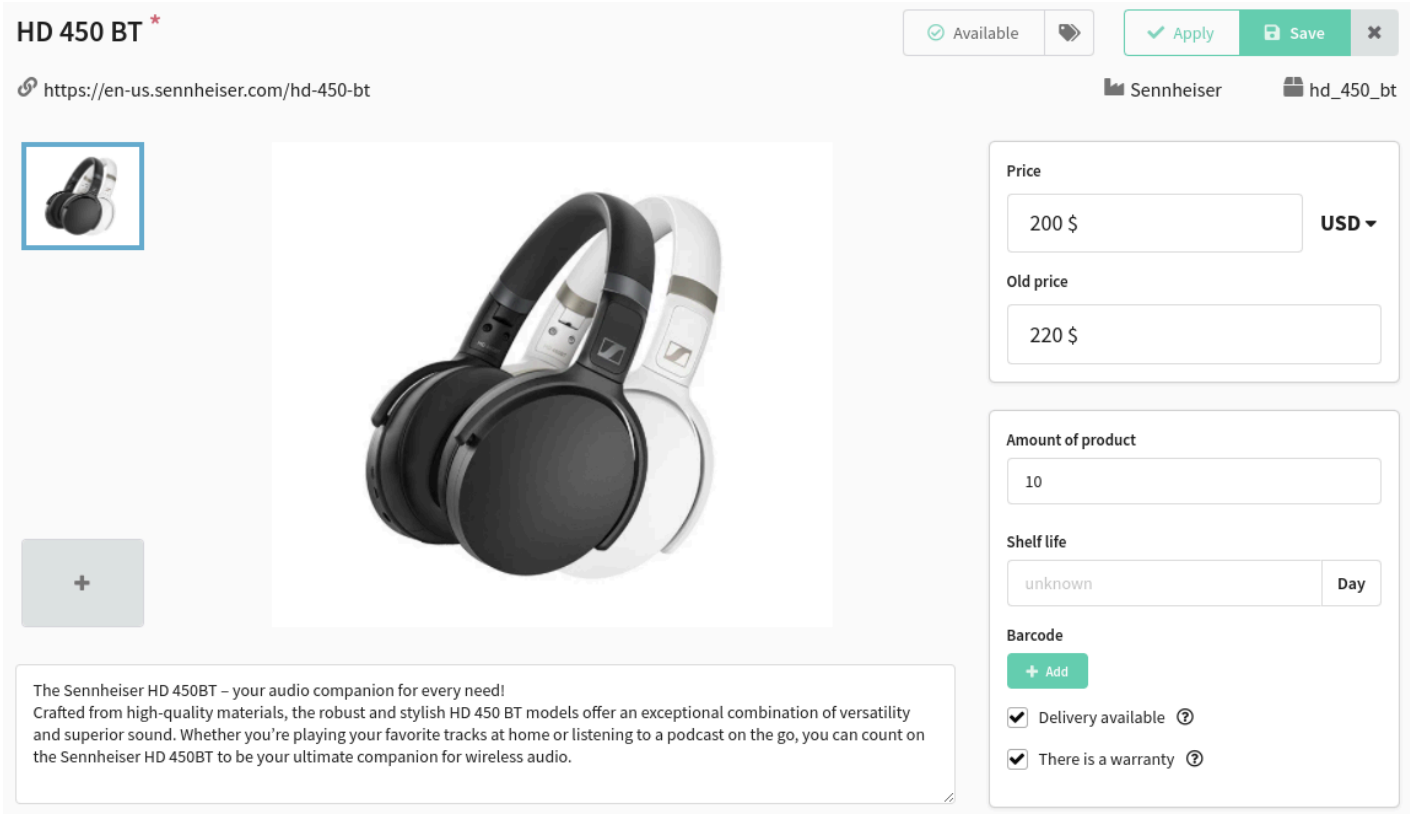
	In this guide, you will learn how to create a product or SKU manually in the Altcraft market. To get started, go to the "Products" section and click "Create" in the upper panel. </td> </tr>

Basic product information

When creating a new product you must specify its name and additional id. The latter can consist only of numbers and Latin letters. It can contain up to 20 characters.

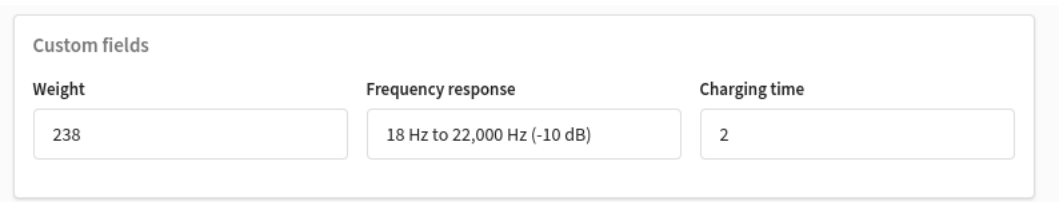
Other fields are optional:

- **product link — the URL of a product page on the store's website**
- **product images**
- **description**
- **available/unavailable — in stock or out of stock**
- **tags for quick search**
- **manufacture 's name**
- **price (select currency) — the current price of a product**
- **old price**
- **amount of product available for sale**
- **shelf life or service life (in days)**
- **barcode from the manufacturer in one of the formats (EAN-13, EAN-8, UPC-A, UPC-E), it is possible to add multiple barcodes.**
- **delivery — the possibility of courier delivery (for all regions to which the store delivers).**
- **warranty — whether or not the product has an official manufacturer's warranty**



Custom fields

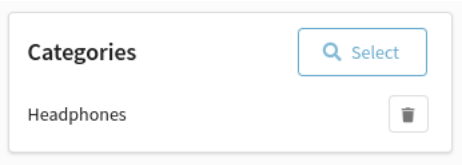
Fill in the custom product fields. For example, you can specify important characteristics of a product such as colour, size, amount, material, weight, model, etc.



You can create custom fields in "Settings".

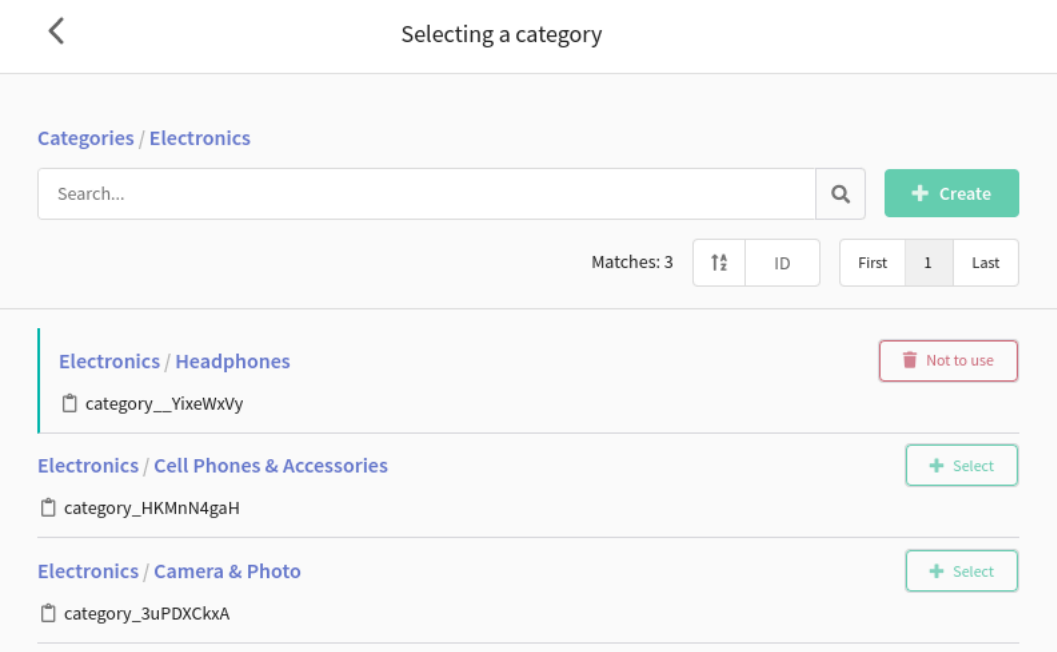
Categories

Specify categories for a product by clicking " Select":



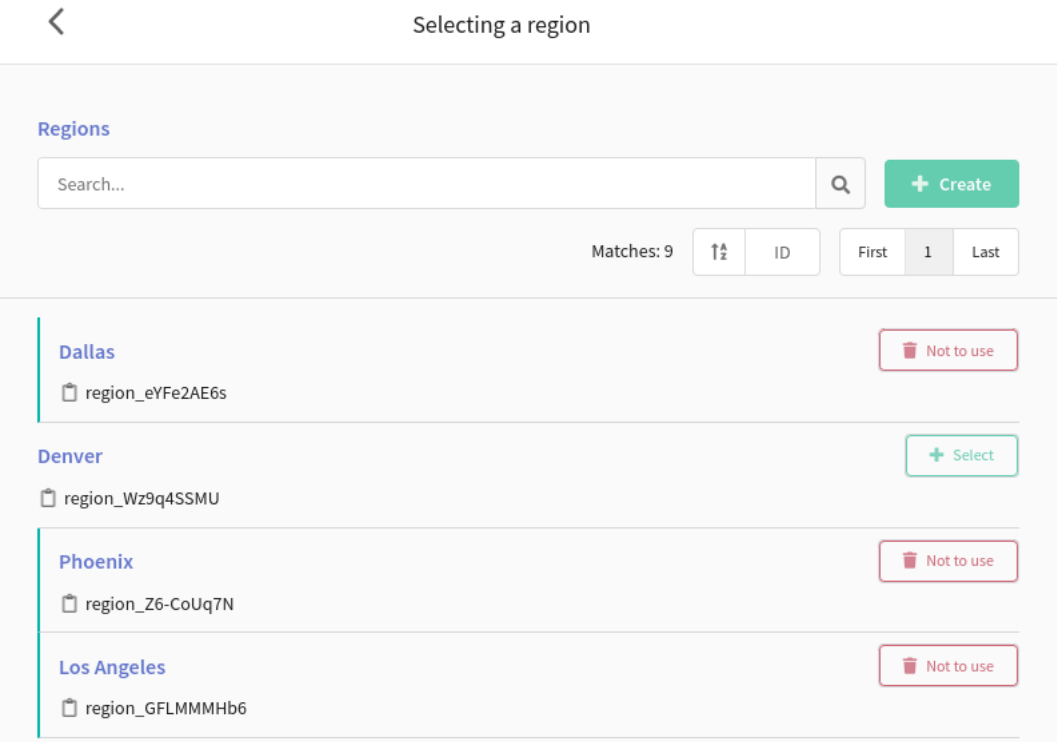
You can create categories in the market settings. Click "Create" to quickly jump to this section.

If you have already created the necessary categories, click "Select" to add a product there:



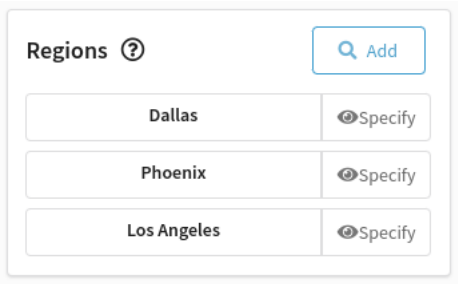
Regions

Add regions where your product is available:



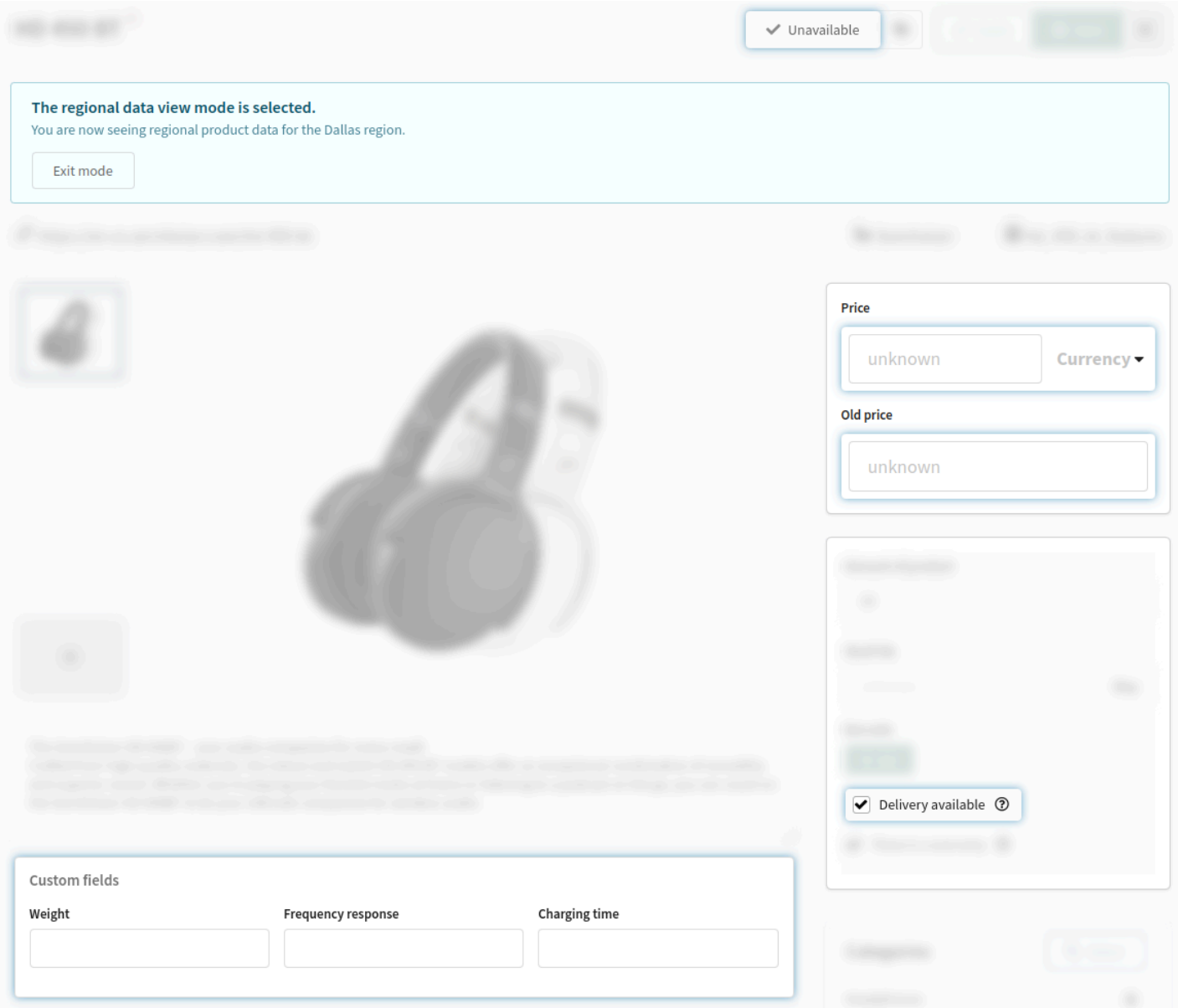
You can change product parameters depending on the region. If you do not deliver your product to some region, change the "Delivery is available" parameter for it.

Click "Specify" to edit product information for a certain region — you will switch to the regional data view mode:

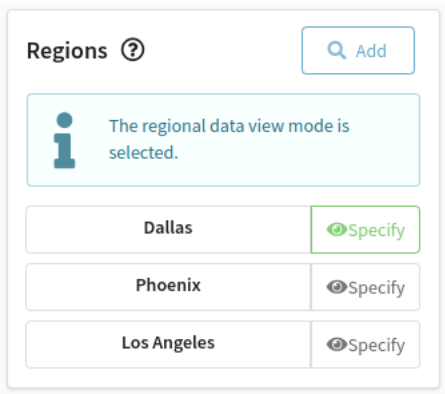


In this mode, you can change the following product information:

- **availability in the selected region;**
- **price;**
- **old price;**
- **delivery to the selected region;**
- **additional fields.**



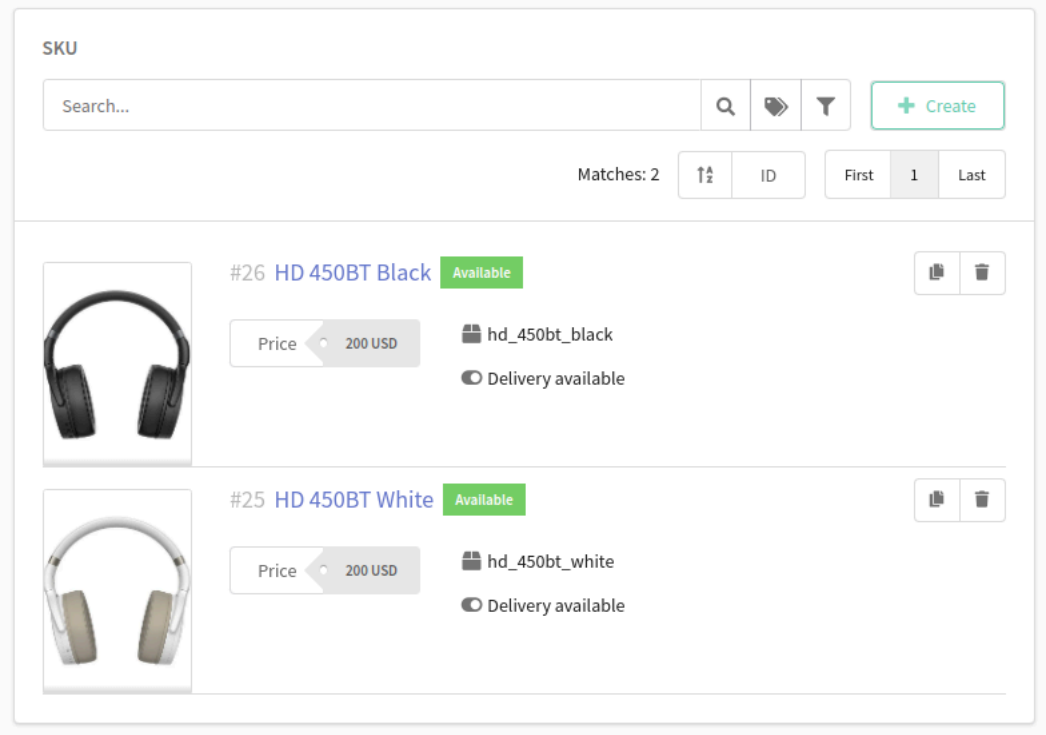
After filling in the regional data, click “Specify” next to a region to save changes:



SKU

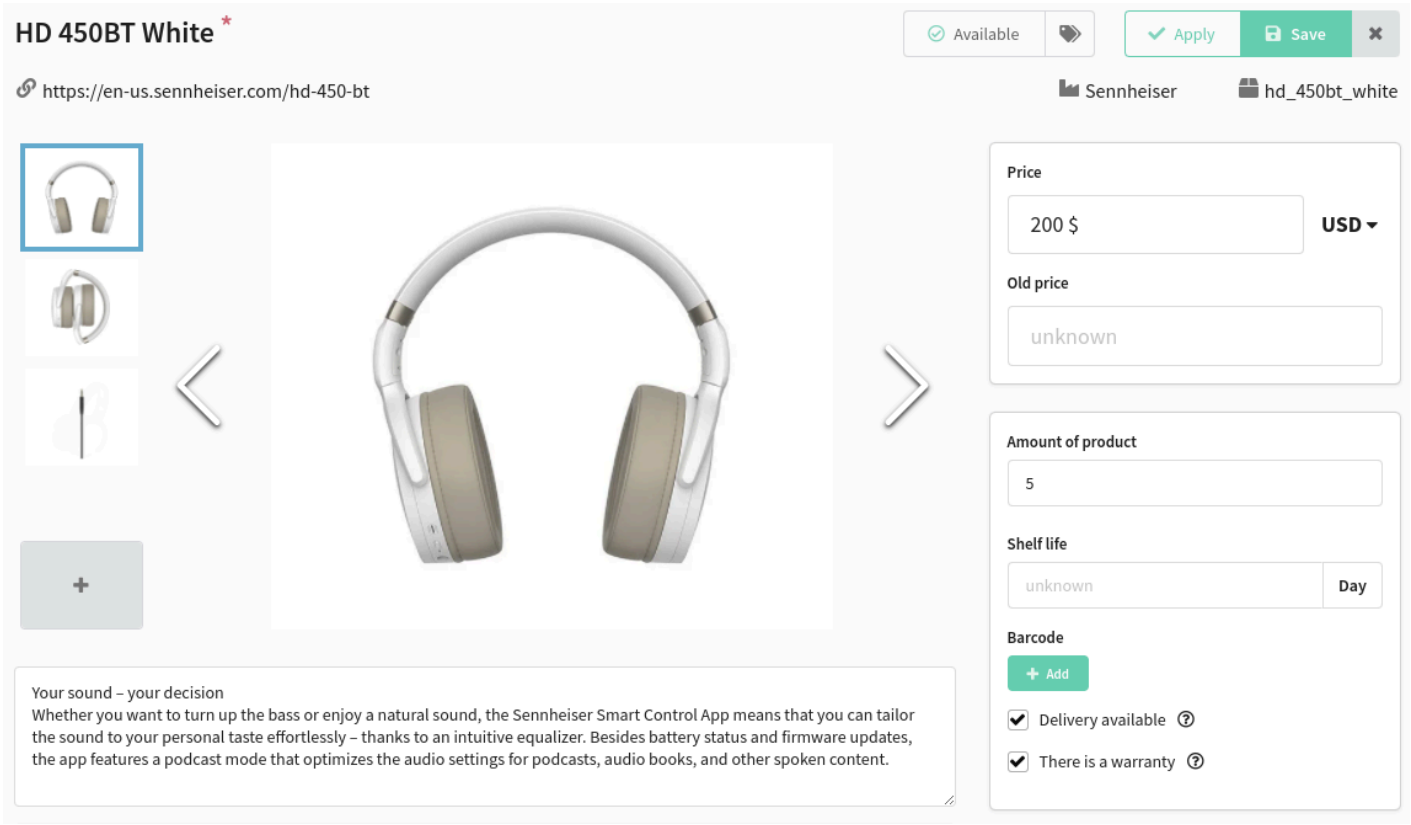
TIP

A stock keeping unit (SKU) is a product modifier that is attached to a group of products that differs from the rest by some characteristics (colour, size, weight, model, configuration, etc.). Each product has a unique SKU.



The process of creating a SKU is similar to creating a product:

- **enter a name and an additional identifier;**
- **fill in a product card (link, image, manufacturer, price, delivery, warranty, and so on);**
- **add a SKU to a specific product category;**
- **add regions and specify regional data for each of them;**
- **fill in the custom fields.**



Then save a product. Now you can use this product or its SKU when creating an order.

How to import a product from a file

This article describes how to import product data from a file into the Altcraft platform market. Products in the Market are represented by two entities: products and SKU.

Product is a good or service for sale. SKU is a product modifier, i.e. a group of goods that stands out in one product according to certain parameters (color, size, weight, model, equipment, etc.).

A SKU is tied to a specific product and cannot exist without it. For example, Apple iPhone 12 is a product, and Apple iPhone 12 64GB and Apple iPhone 12 128GB are its SKU.

To upload product data to the platform, use a file in yml format or a .zip/.tar.gz archive containing yml files.

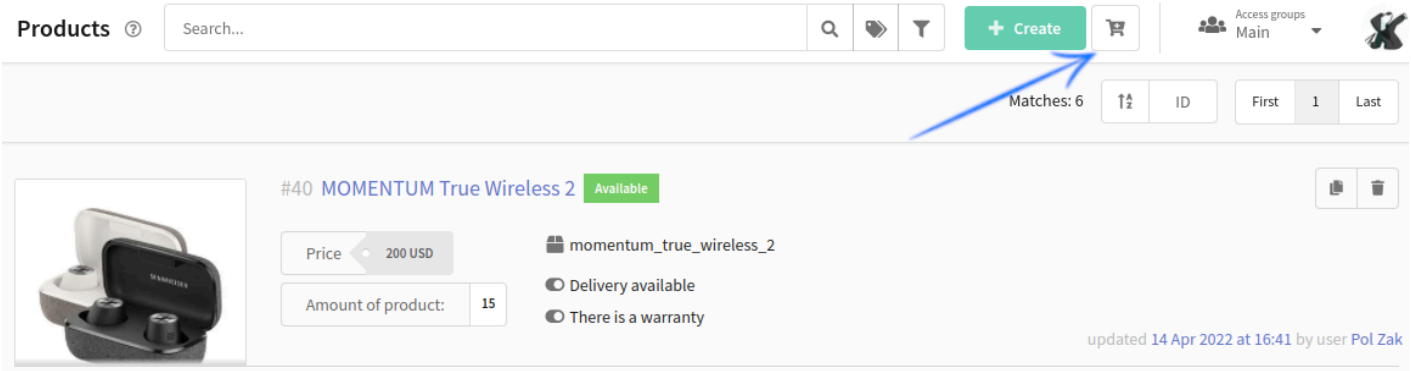
INFO

The YML (Yandex Market Language) format is a data format for product platforms developed by Yandex. Detailed information about the structure of the yml file, its parameters and properties, as well as examples of ready-made yml feeds with products, can be found there.

File uploading

Click

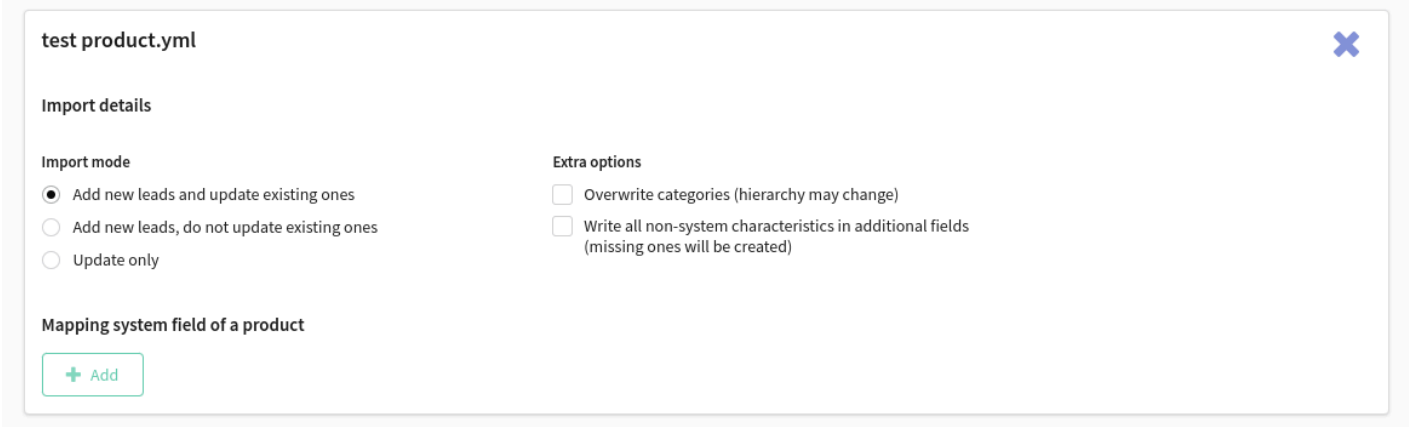
on the upper panel to import a product or SKU from a file. Select what you want to import: product or SKU.



Import details

Specify the import mode. It depends on whether you are adding data about new products or updating information on the products that are already in the market:

- **Add new products and update existing ones**
- **Add only new products and do not update existing ones**
- **Update only existing ones, do not add new ones**



Extra options are activated in the “Import details” section:

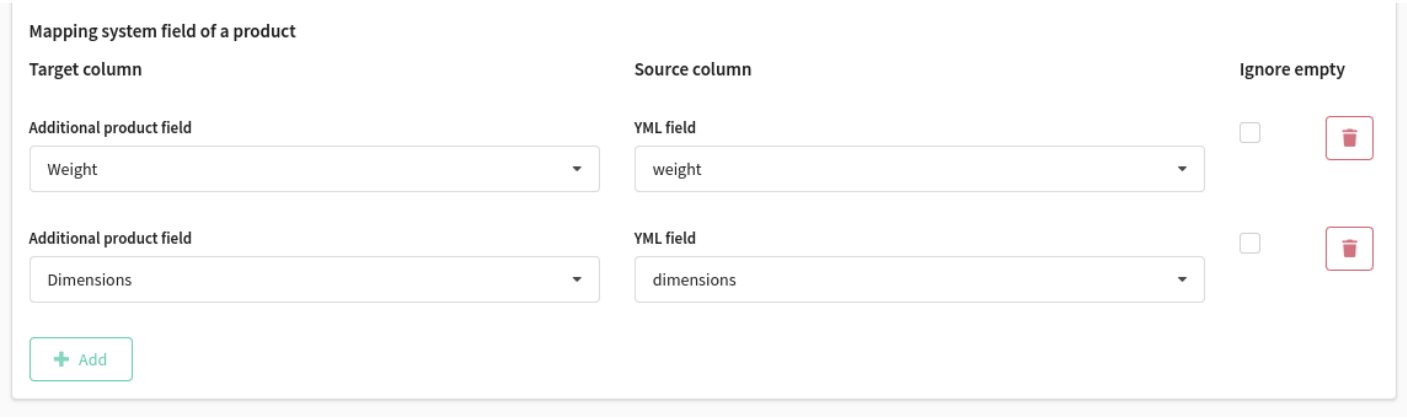
- **Overwrite categories (hierarchy may change)**
- **Write all non-system characteristics in additional fields (missing ones will be created)— the platform will generate custom fields for the param element in a yml file.**

Mapping system fields

In this block, you can configure the import of additional product information into the platform.

The target column is a custom field in the market, which is created in the “Settings” section.

The source column is a standard field in the yml file, the data from which you want to write to a custom field.



Now the import setup is complete. Start the process by clicking “Import” in the upper panel.

Scheduled product import

This article describes how to synchronize products and SKUs in Market with your product feed data. This feature allows you to avoid having to download a new yml file every time your feed is updated.

After setting up the import, new information about products will be automatically uploaded to the Market according to the specified schedule. Information will be loaded from the yml feed with products, to which you will need to provide a link.

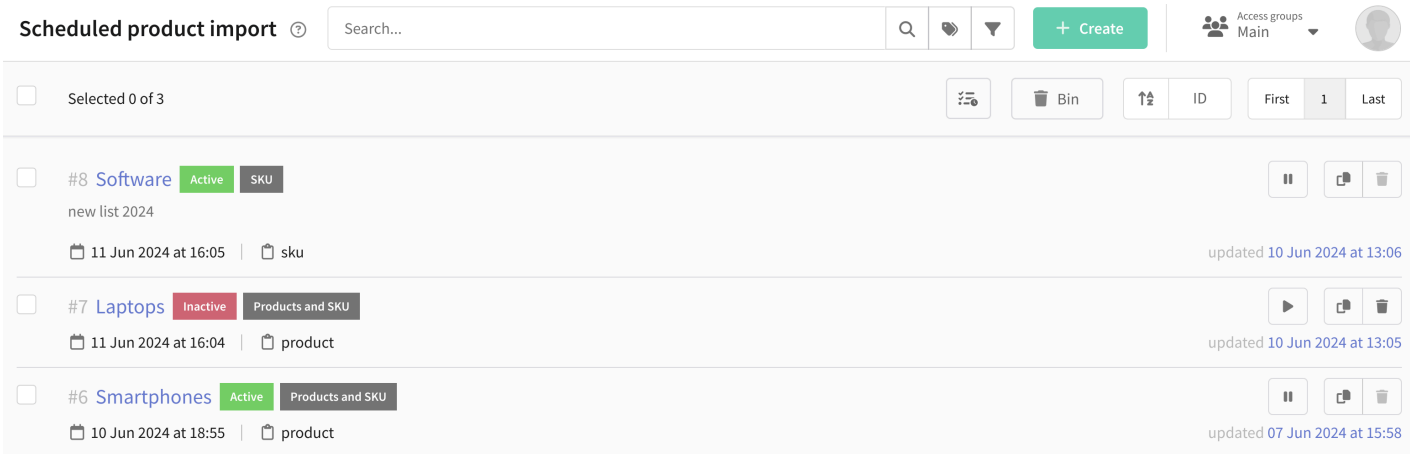
INFO

The YML (Yandex Market Language) format is a data format for product platforms developed by Yandex. Detailed information about the structure of the yml file, its parameters and properties, as well as examples of ready-made yml feeds with products, can be found here.

Step 1. Create a new import

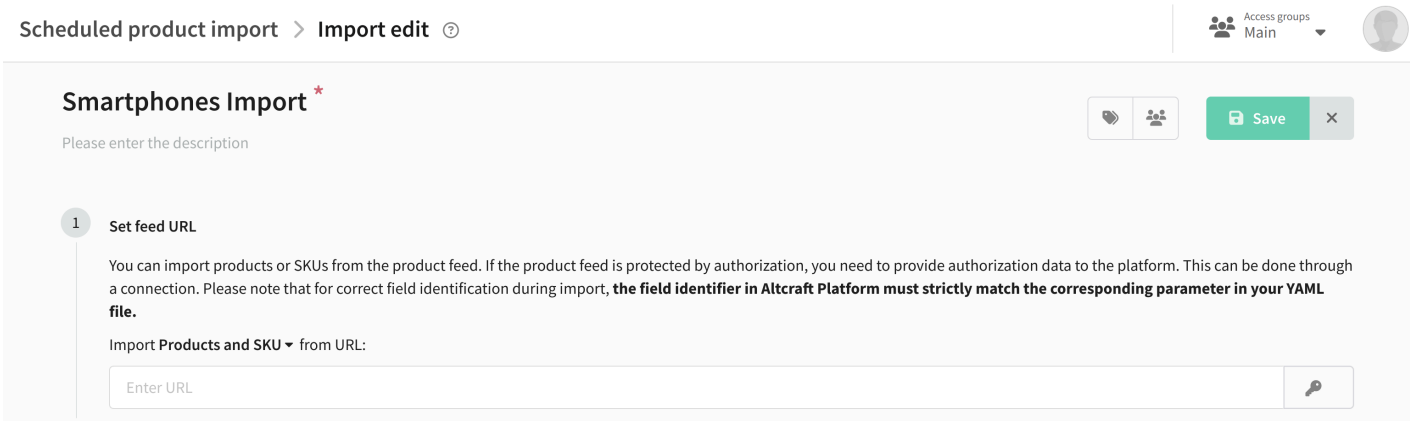
Go to the Market section, select Scheduled product import. A list of existing imports will appear in front of you with brief information about them. Here you can also start, pause, clone or delete an import.

Click the Create button in the upper right corner:



Step 2. Specify the feed URL

Insert the URL of your feed into the corresponding field. In the item Import products and SKU from URL you can choose what type of objects will be loaded from the feed - products and their SKUs or only SKUs. In the first case the platform will first import all products and then add all relevant SKUs, in the second case - new products will be completely ignored and SKUs will be loaded for existing products:

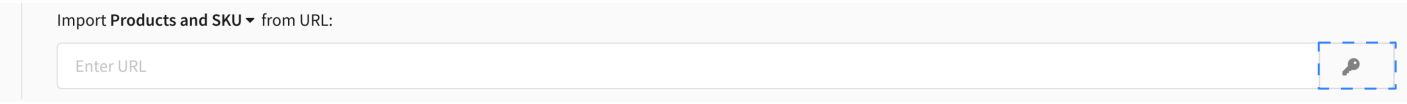


CAUTION

If your product feed has fields that are not provided by default in the Market platform, make sure you have created the appropriate additional fields in Market Settings, otherwise the import will fail.

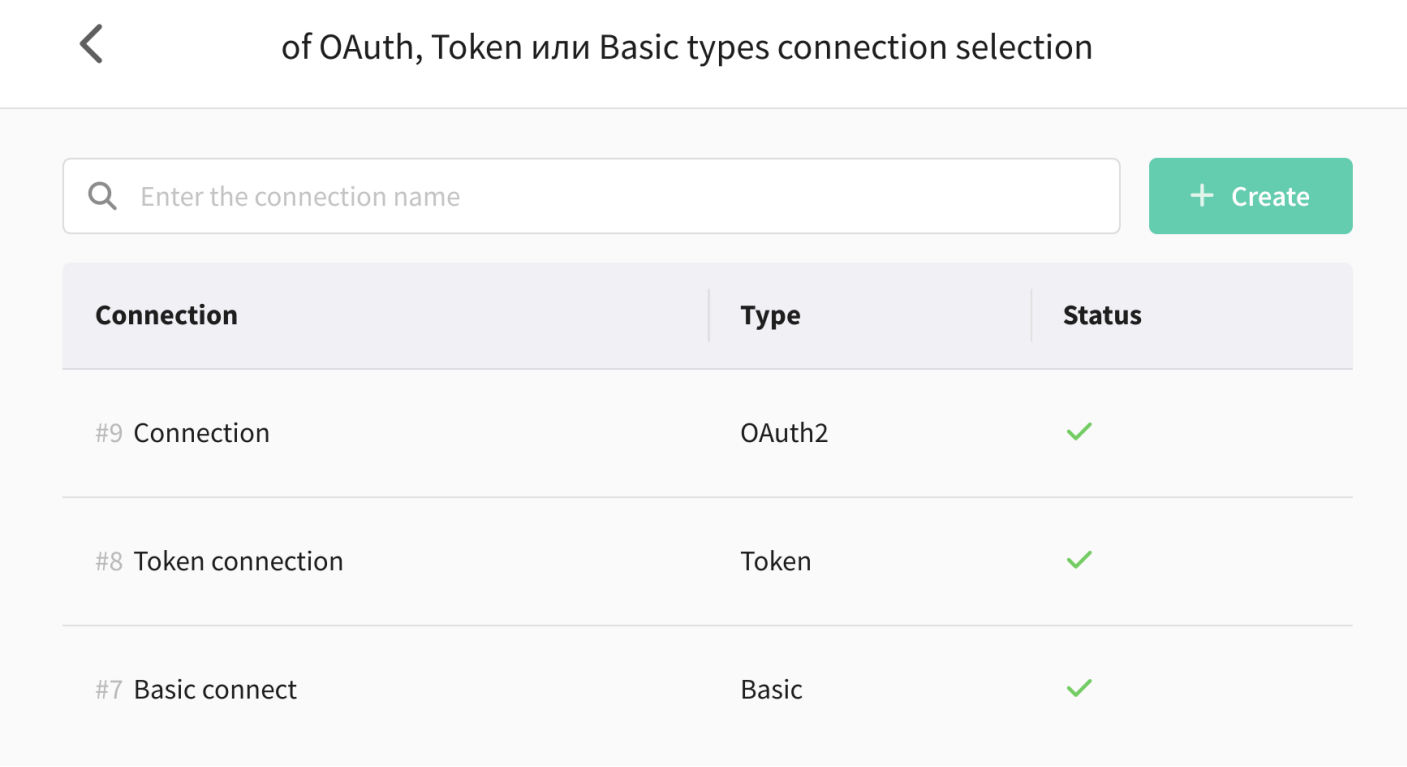
Connection selection

If your product feed is protected by authorization, you need to add connection to the import. You can do this by clicking on the button with the key icon:



A connection selection window will appear in front of you. There are three types of authorization connections on the Altcraft platform:

- **OAuth2 is an authorization protocol that allows you to grant one service (application) the rights to access user resources on another service. The user does not need to trust the application with a login and password.**
- **Basic Authentication - HTTP authentication protocol that uses login and password to gain access to protected information.**
- **Token Authentication - a protocol that uses a special token that is provided to the user after successful authentication to access protected resources.**



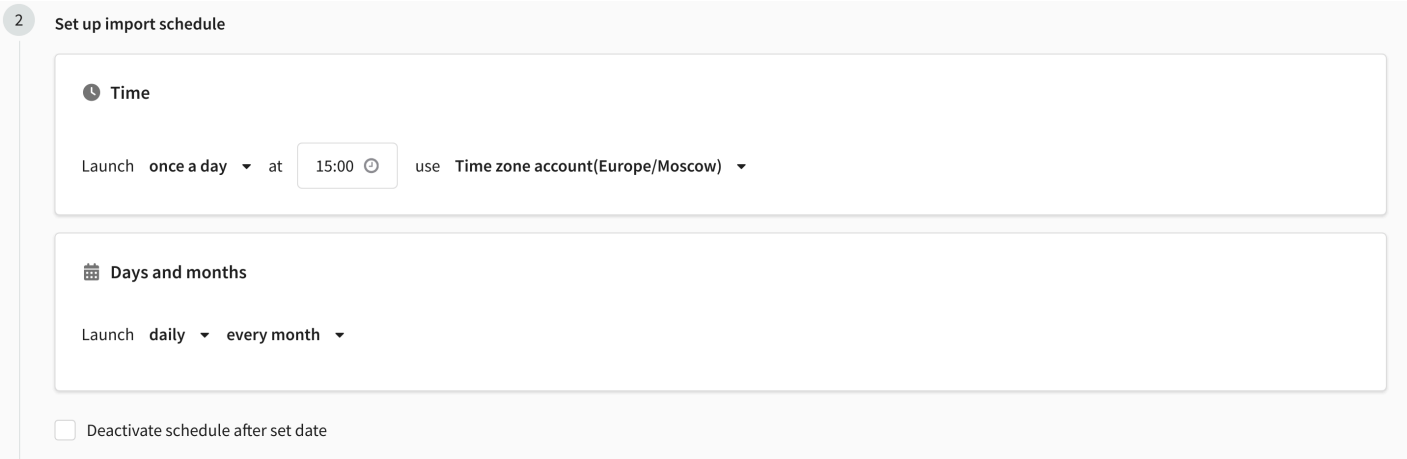
If there are no connections, you can create a new one directly in this window. To do this, click + Create and select the type of connection: OAuth2, Basic Authentication or Token Authentication. Fill in the required fields and click **Save **.

TIP

More detailed information about creating and configuring a connection is available in the "Connections" section.

Step 3. Set up the import schedule

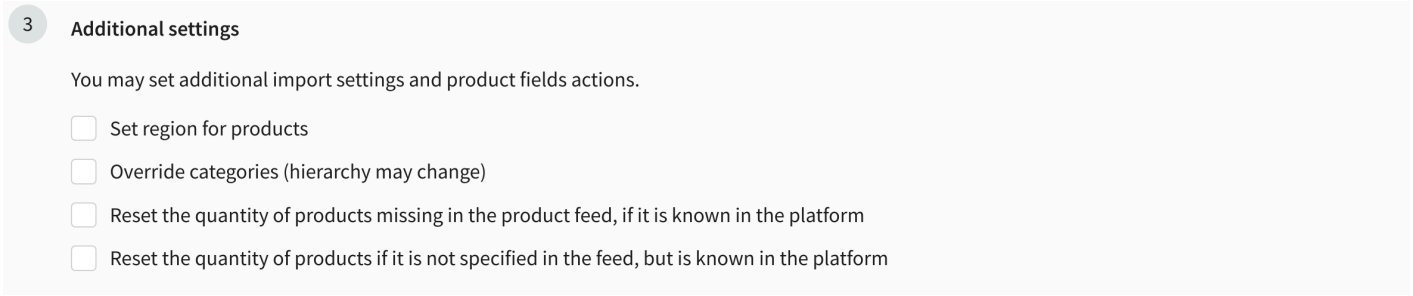
From the drop-down menu, select a schedule plan and customize depending on how often you want the product list to be updated. For each schedule plan, you can set specific time zone. You can also specify the import deactivation date:



Step 4. Specify additional settings

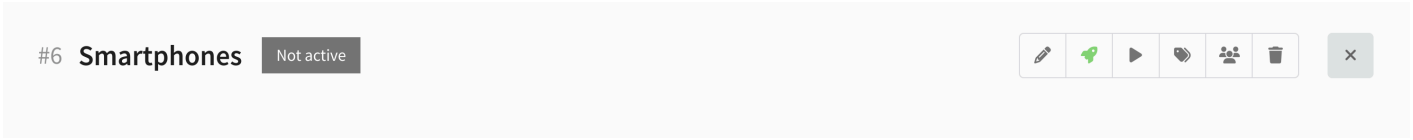
After setting the basic parameters, you can specify the necessary additional options:

- **Specify region for products - sets the region for imported products. The region must be created on the platform in advance.**
- **Override categories (hierarchy may change) - product categories will be overwritten in accordance with the content of the feed.**
- **Reset the quantity of products missing in the product feed, if it is known in the platform - if any product is not specified in the feed, then its quantity on the platform will become 0.**
- **Reset the quantity of products if it is not specified in the feed, but is known in the platform - if the quantity of a product is not indicated in the feed, then its quantity on the platform will become 0.**



Step. 5. Saving and starting import

To save the created import, click the Save button. You will see an overview of your import.



The following functions are available in the context menu:

— edit import

— launch and activate import right now

— activate/deactivate import

— assign tag

— assign access groups fot his import

— delete import

After the first launch, the imports history and information about added products will appear in the table below. There you can also see the status and number of errors that occurred during the import. To switch the type of data displayed (product data or SKU data), activate the Products/SKU switch:

#6 Smartphones

Active

Latest launch

completed

Total products
100

Updated **99**
Added **1**
Skipped **0**

Launch start
07-06-2024 18:58

Launch end
07-06-2024 18:58

Next launch
08-06-2024 18:55

Import settings

General info
ururugu.yml
Not selected

Schedule
At 18:55
Daily every month

Imports history

Products/SKU

Start	End	Products				Categories		Status	Errors
		Total	Added	Updated	Skipped	Added	Updated		
07-06-2024 18:58	07-06-2024 18:58	100	1	99	-	-	3	completed	0

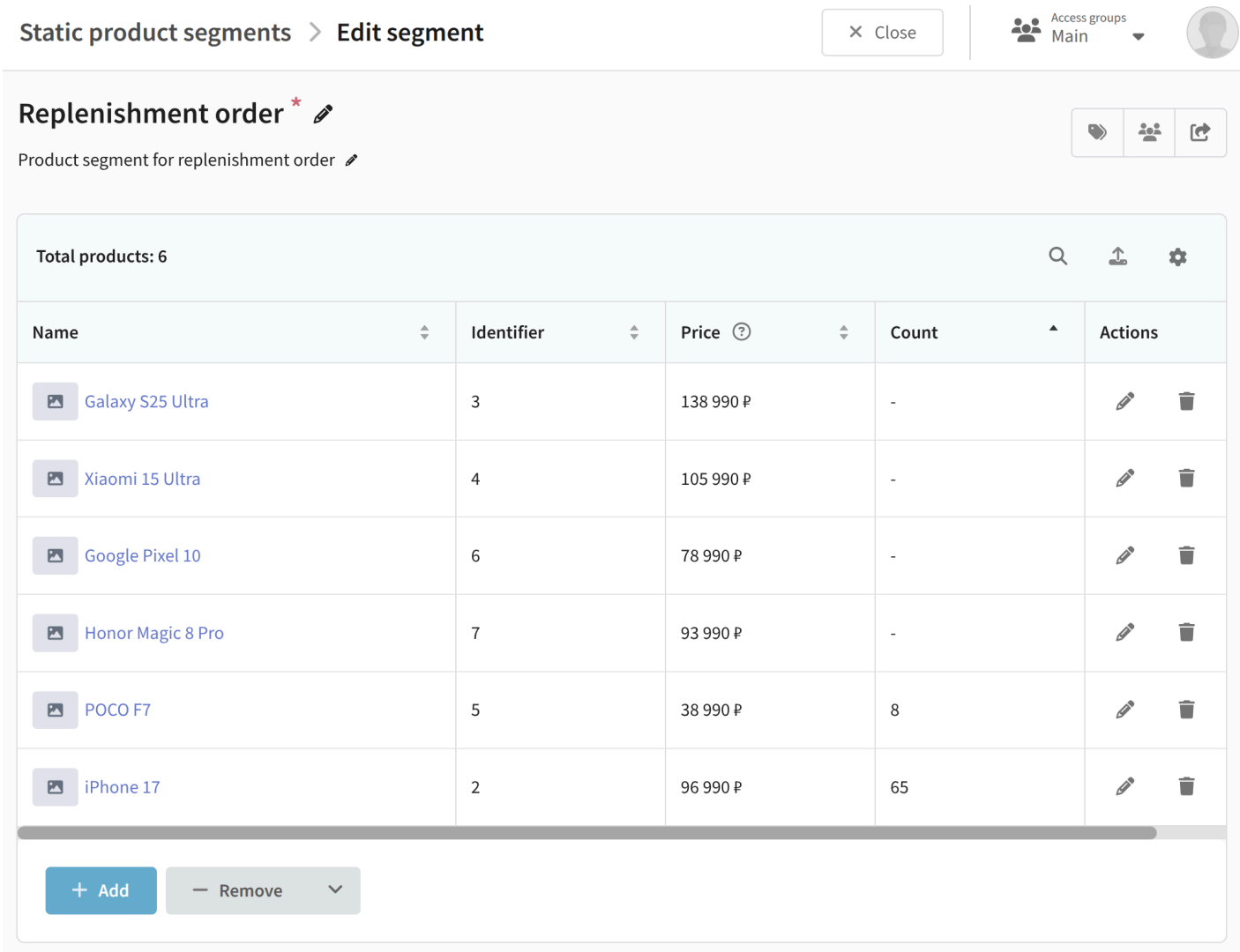
Product and SKU Segments

TIP: Segmentation in Altcraft

Product and SKU segmentation works the same way as profile segmentation in the platform. More details about the condition builder and segmentation logic are available in this section.

Description

Product and SKU segments are sets of products and their modifiers (SKUs) grouped by a specific criterion. A finished segment can be exported and used for sales data analysis. There are two types of segments: static and updatable.

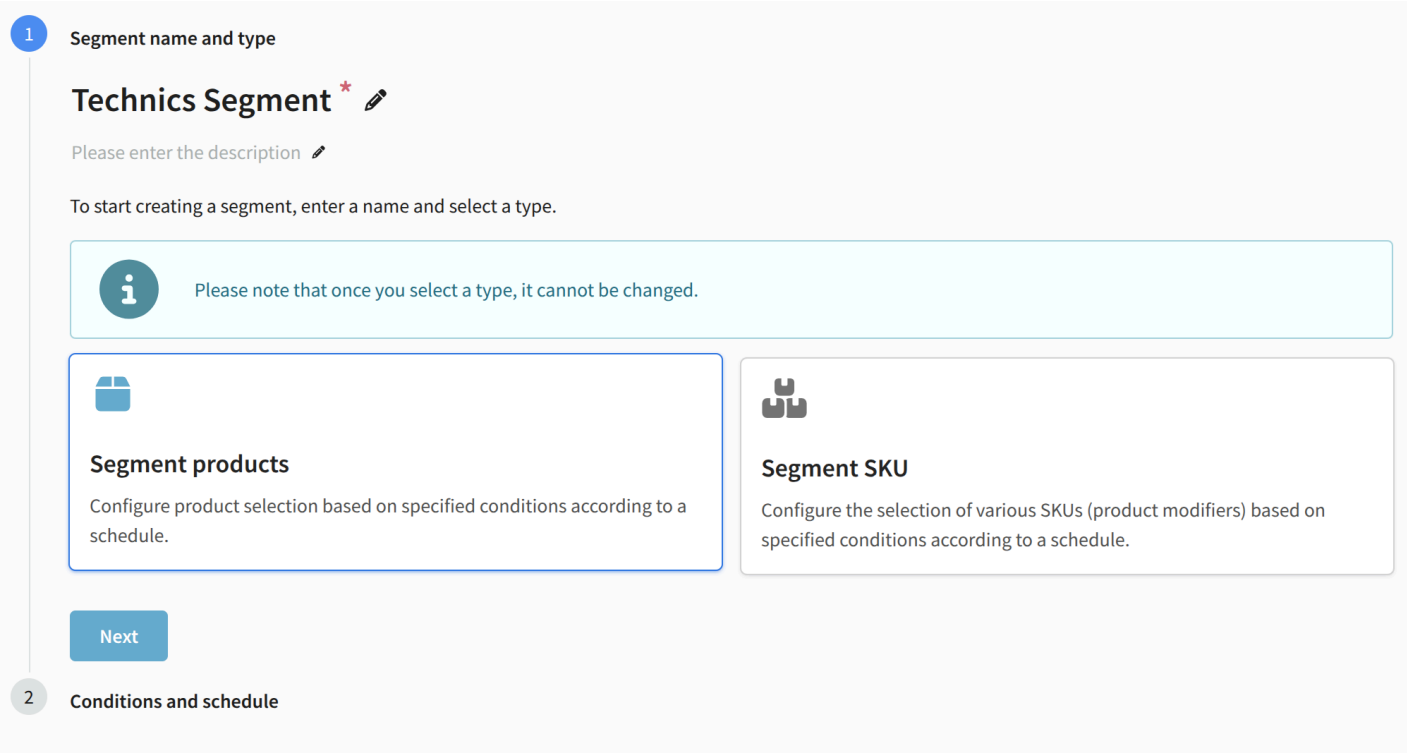


Updatable Segments

Updatable segments are product and SKU sets that are recalculated on a schedule based on defined conditions. Such a segment can also be recalculated manually outside the schedule.

To create an updatable segment, go to Market — Updatable segments:

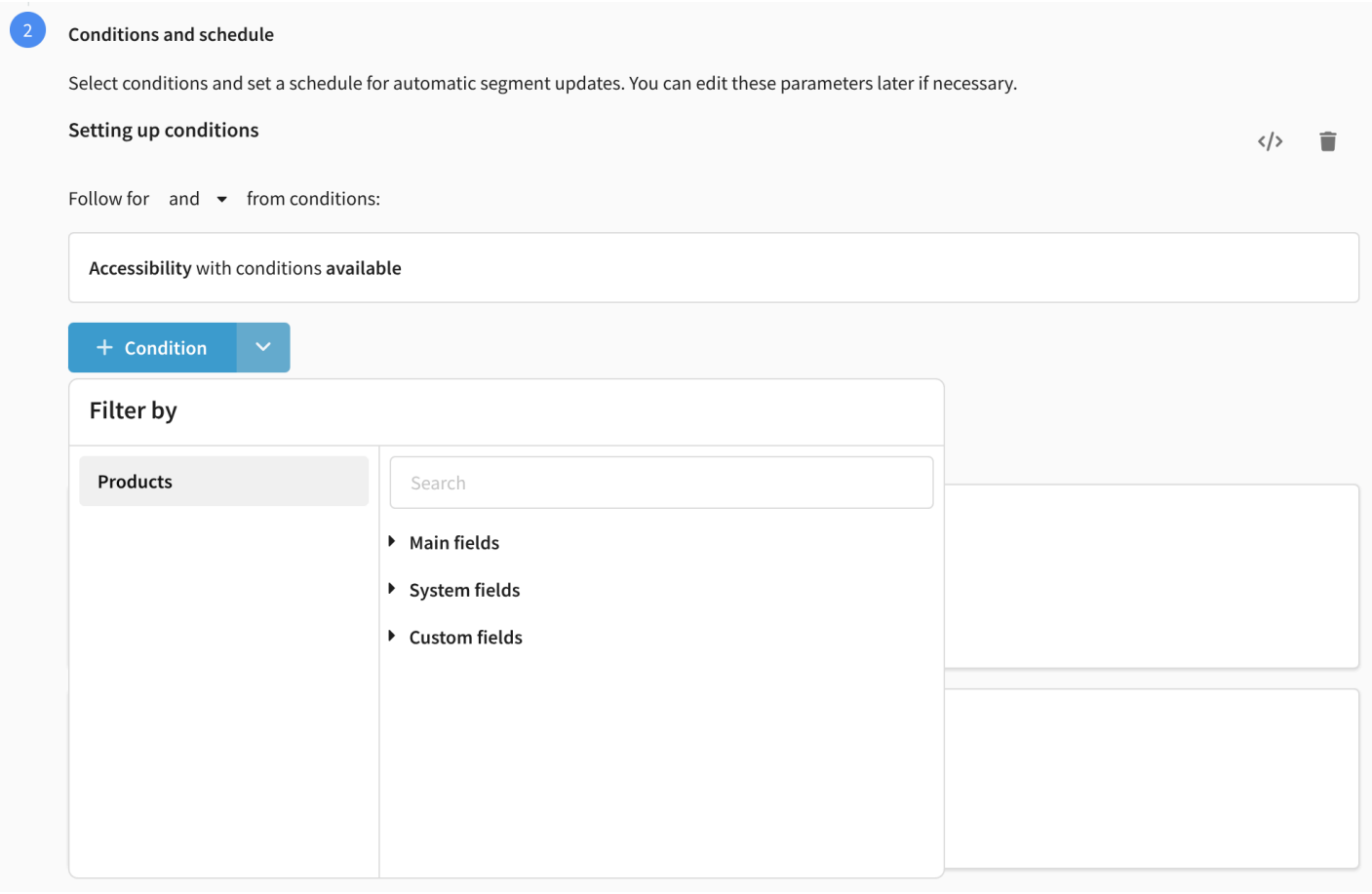
Click “Create”. Specify a name, description, and select entities for segmentation — products or SKU. After selection, the segment type cannot be changed:



Then the Conditions and schedule block opens. Select conditions or condition groups and set a schedule for automatic updates. Conditions inside a group can be combined using match all conditions (AND) or match any condition (OR). These settings remain editable later.

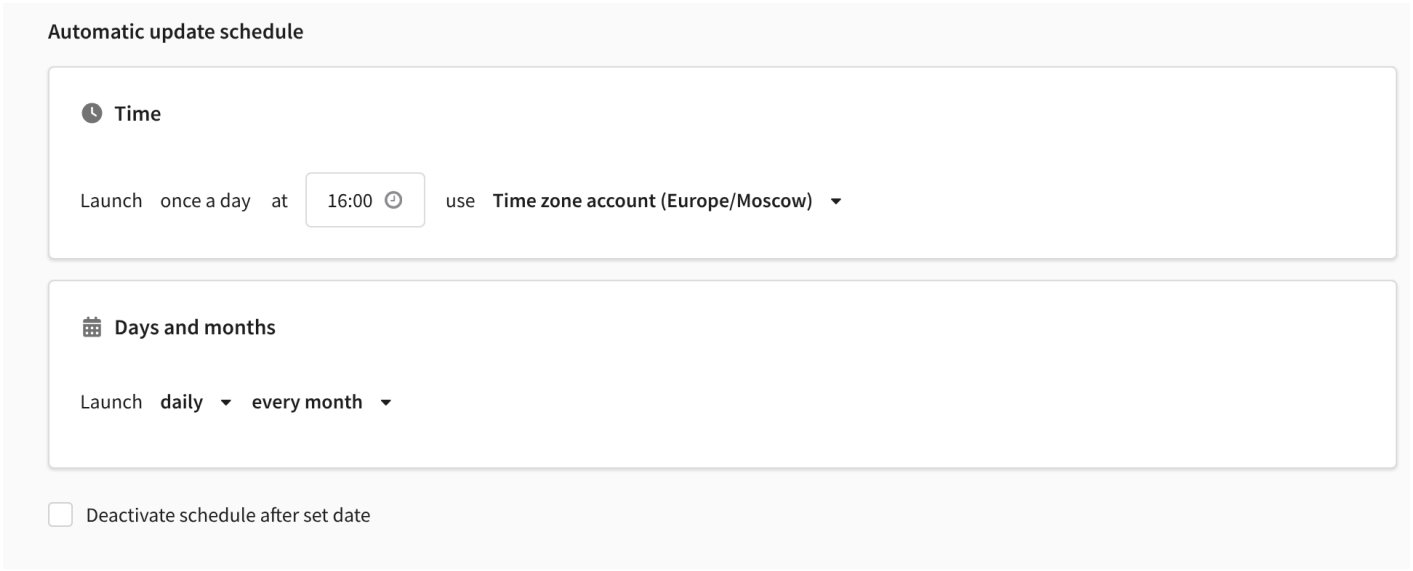
Conditions are divided into three groups:

- **Core fields — filtering by name, region, category, product quantity, expiration date, price, delivery, warranty, availability, manufacturer, and tags.**
- **System fields — filtering by identifier, additional (string) identifier, creation date, and update date.**
- **Custom fields — filtering by user-defined fields.**



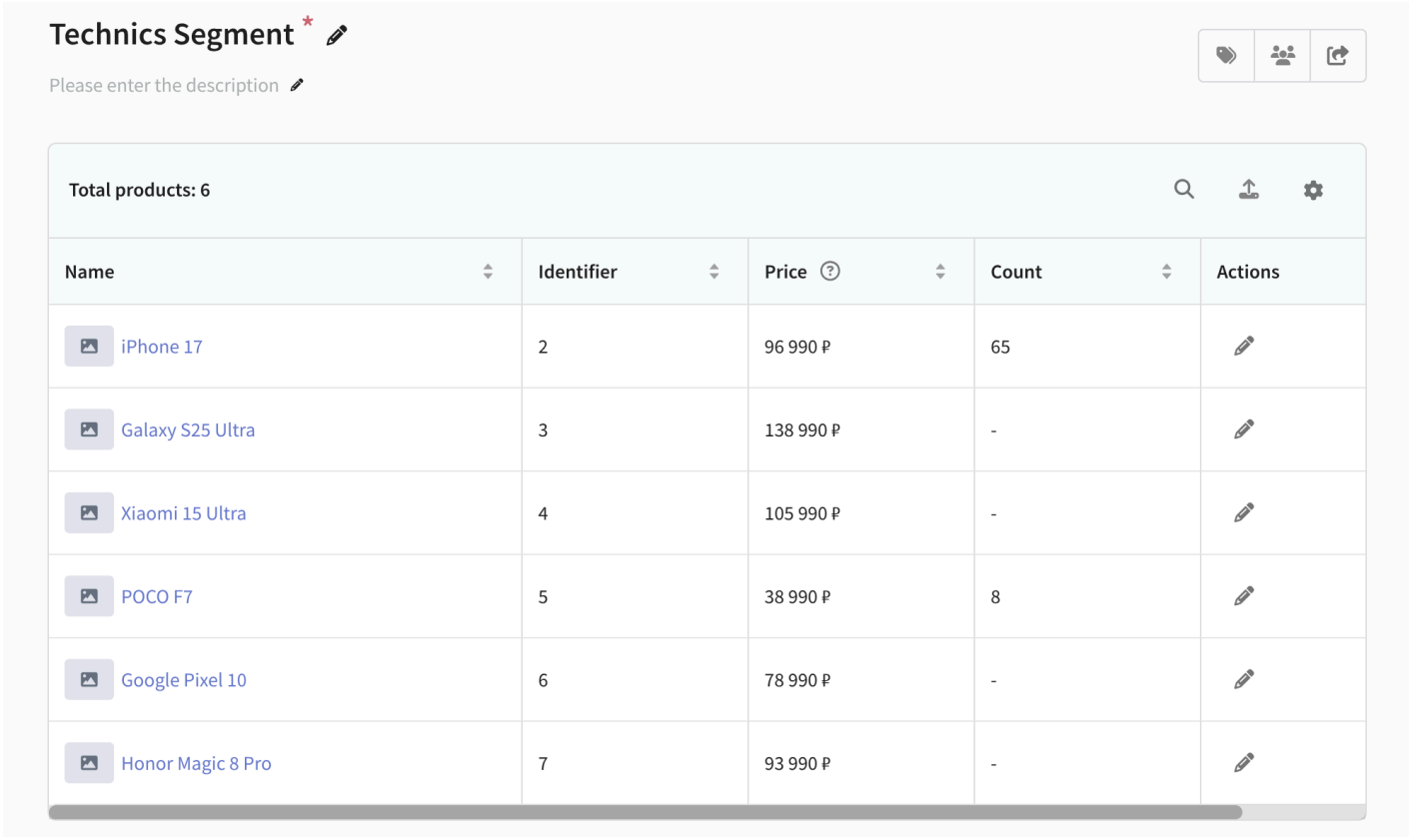
The same block contains the Script button, which opens an editor for manual condition writing. After setting conditions, you can define a product count limit. This limit applies to the selection result after condition filtering.

A segment can be updated daily or on specific days. Update time and time zone are set by the user:



You can also configure automatic segment deactivation after a specified date.

After calculation, the segment view displays a table with all products included in the segment. It shows the fields “Name”, “Identifier”, “Price”, “Quantity”, and a button to open product editing:



Available actions for this table:

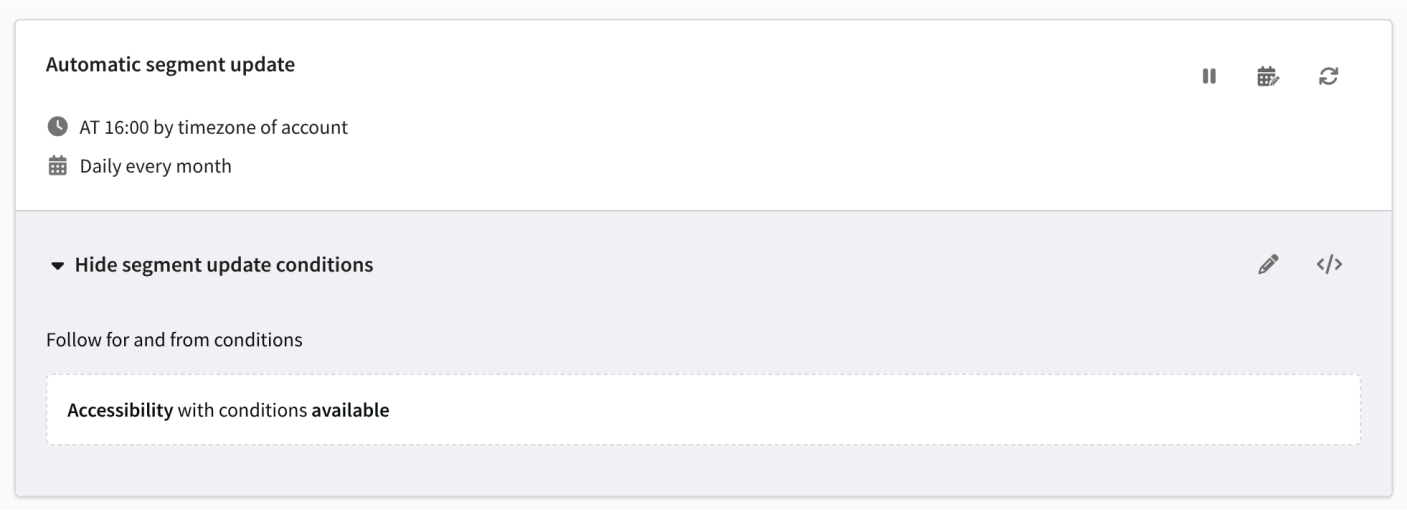
- **Search**

  : find a specific product.
- **Export products**

  : export the displayed list to an external file.
- **Configure displayed fields**

  : add or remove table columns by dragging a Field from the Add field sidebar to Displayed fields or back.

Below the product table are the segment update schedule and conditions. Here you can edit them or recalculate the segment outside the schedule:



Below is the Update history block. It visually shows changes in the total number of products in the segment for the selected period. You can specify a date range and group data by days or weeks.

Below the update history is the Log. It contains a detailed record of all operations affecting the segment composition. For each operation, the following is shown:

- **Start date: time when the segmentation process started (adding/removing by conditions).**
- **End date: time when the process finished.**
- **Initiator: name of the user who initiated the operation.**
- **Added: number of products added as a result of the operation.**
- **Removed: number of products removed as a result of the operation.**
- **Status: current operation state (for example, “Completed”, “In progress”, “Error”).**

Log

Start date	Finish date	Type	Initiator	Added	Deleted	Status
10-01-2026 15:36	10-01-2026 15:36	Manual	Marketing User	65	0	Completed
13-01-2026 15:57	13-01-2026 15:57	Manual	Marketing User	13	0	Completed
15-01-2026 15:58	15-01-2026 15:58	Manual	Marketing User	19	8	Completed
17-01-2026 16:00	17-01-2026 16:00	Manual	Marketing User	29	0	Completed

TIP: Segment actions

At the top of the screen, a quick actions menu is available, where you can assign tags, configure access groups, and copy products to another segment.

Static Segments

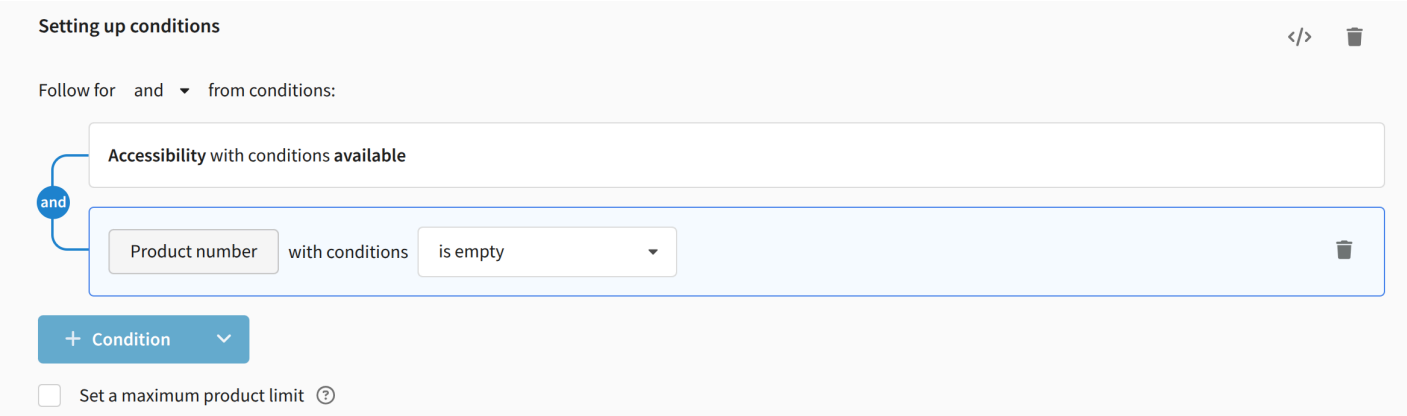
Static product segments are sets of products and their modifiers (SKU) grouped via API or manually using selection conditions.

To create a static segment, go to Market — Static segments:

Click “Create”. Specify a name, description, and select entities for segmentation — products or SKU. After selection, the segment type cannot be changed:

The segment is created. To add products, click “Add” — the Conditions block opens.

To add a rule, click + Condition. The dropdown menu contains the Add condition group option. Conditions inside a group can be combined using match all conditions (AND) or match any condition (OR):



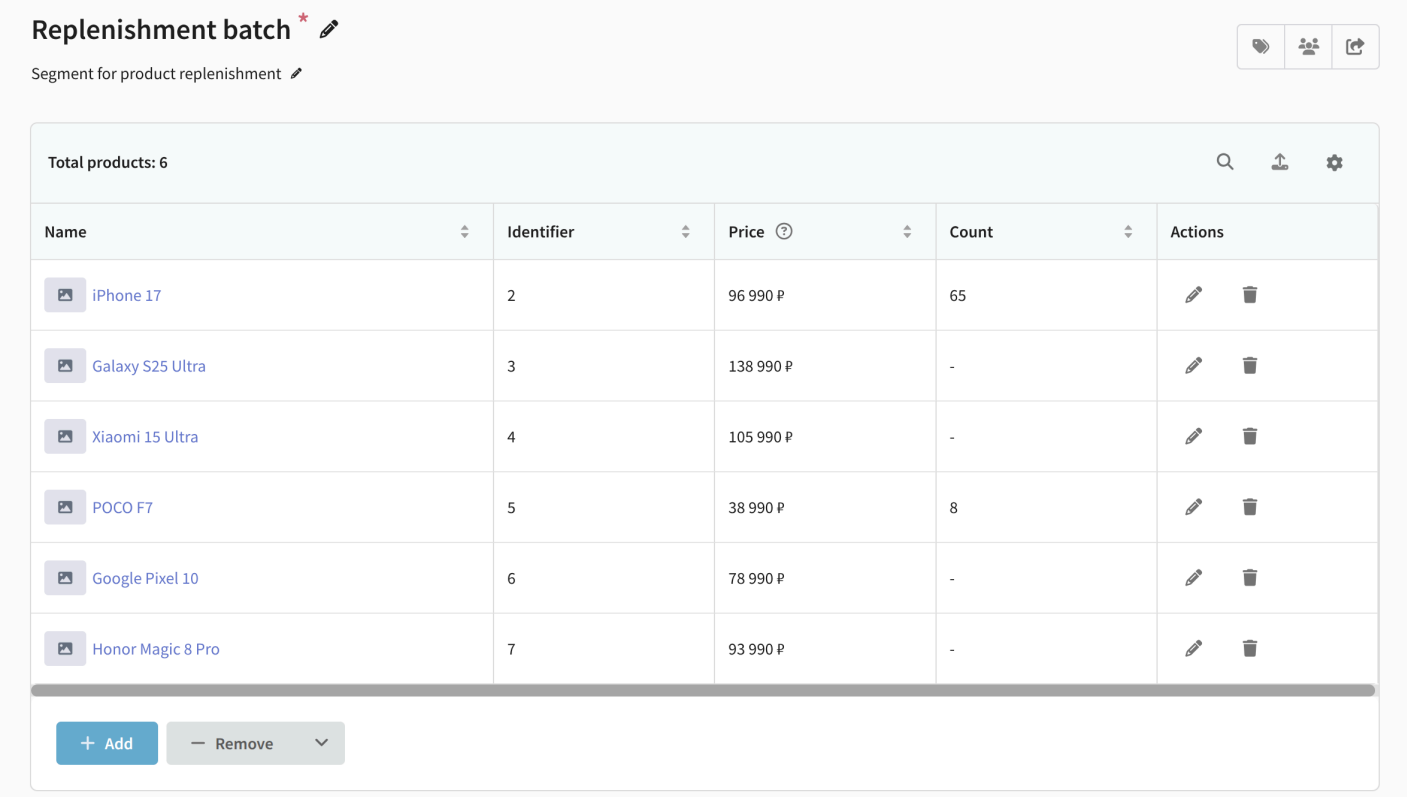
Conditions are divided into three groups:

- **Core fields — filtering by name, region, category, product quantity, expiration date, price, delivery, warranty, availability, manufacturer, and tags;**
- **System fields — filtering by identifier, additional (string) identifier, creation date, and update date;**
- **Custom fields — filtering by user-defined fields.**

The same block contains the Clear button, which fully removes all defined rules, and the Script button, which opens an editor for manual condition writing. After setting conditions, you can define a product count limit. This limit applies to the selection result after condition filtering.

After clicking “Add”, products matching the selection rules appear in the segment. To remove products from the segment, click Remove. The removal function works the same way as adding products.

After calculation, the static segment view displays a table with all products included in the segment. It shows the fields “Name”, “Identifier”, “Price”, “Quantity”, as well as a quick actions menu for removal or opening product editing:



Available actions for this table:

- **Search**

  : find a specific product.
- **Export products**

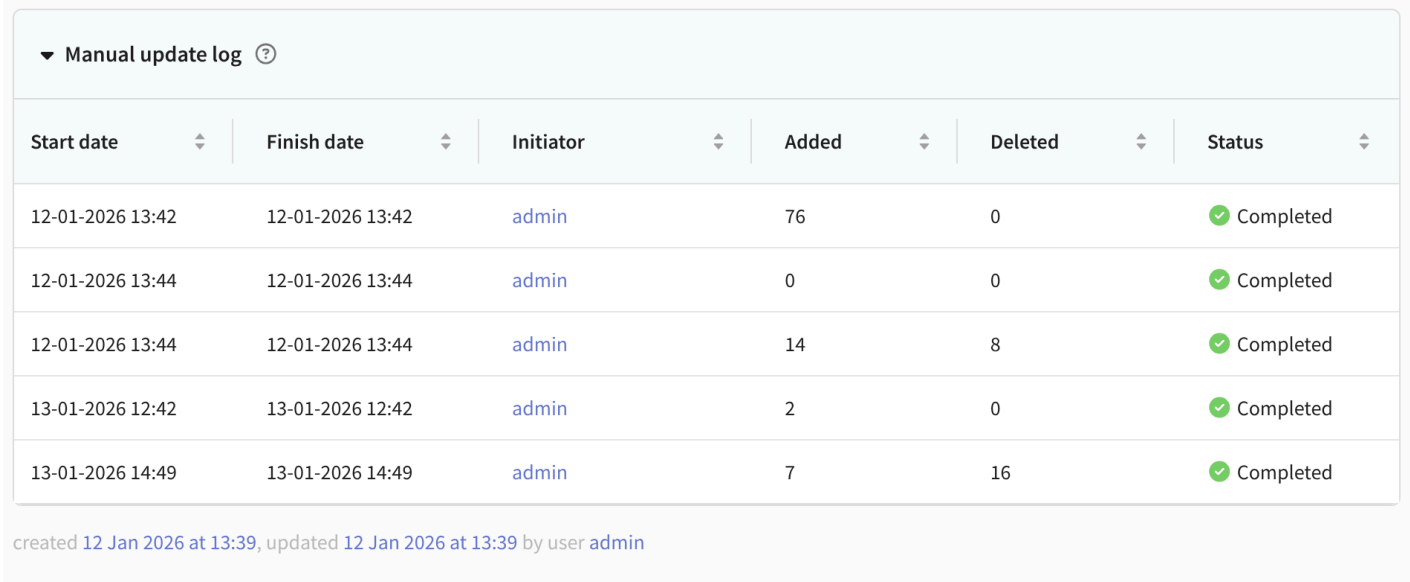
  : export the displayed list to an external file.
- **Configure displayed fields**

  : add or remove table columns by dragging a Field from the Add field sidebar to Displayed fields or back.

Below is the Update history block. It shows changes in the total number of products in the segment for the selected period. You can specify a date range and group data by days or weeks.

Below the update history is the Manual changes log. It contains a detailed record of all operations that affected the segment composition. For each operation, the following is shown:

- **Start date: time when the segmentation process started (adding/removing by conditions).**
- **End date: time when the process finished.**
- **Initiator: name of the user who initiated the operation.**
- **Added: number of products added as a result of the operation.**
- **Removed: number of products removed as a result of the operation.**
- **Status: current operation state (for example, “Completed”, “In progress”, “Error”).**



TIP: Segment actions

At the top of the screen, a quick actions menu is available, where you can assign tags, configure access groups, and copy products to another segment.

Conditions

Conditions for selecting products and SKUs are divided into three groups: Main Fields, System Fields, and Additional Fields. The available set of operators depends on the type of selected field.

Main Fields

Main fields include standard product or SKU fields: name, category, price, quantity, and others.

Product Name

A string field containing the product name.

- **is equal to / is not equal to**
- **contains / does not contain**
- **begins with**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Region

A field for specifying the sales region or product availability.

- **exists / does not exist**
- **is equal to / is not equal to**
- **is in list / is not in list**

Category

A field containing one or more categories to which the product belongs.

- **exists / does not exist**
- **contains at least one value from the list**
- **does not contain any values from the list**
- **contains all values from the list**

- **does not contain all values from the list at the same time**

Product Number

An integer reflecting the remaining stock or available quantity.

- **is empty / is not empty**
- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **in the range**

Expiration Date

A *date* type field indicating the product's expiration date.

- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **exists / does not exist**
- **in the range**

Price

A numeric field with the current product price.

- **exists / does not exist**
- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **is in list / is not in list**
- **in the range**

Old Price

A numeric field with the price before discount (if applicable).

- **exists / does not exist**
- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **is in list / is not in list**
- **in the range**

Delivery

A boolean field (true/false) indicating whether delivery is required for the product or available.

- **true**
- **false**

Guarantee

A boolean field (true/false) indicating whether the product has a warranty.

- **true**
- **false**

Accessibility

A switch field reflecting the product's availability status.

- **available**
- **not available**

Manufacturer

A string field with the manufacturer's company name.

- **exists / does not exist**
- **is equal to / is not equal to**
- **contains / does not contain**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Tags

A list field used for flexible product tagging.

- **exists / does not exist**
- **contains at least one value from the list**

- **does not contain any values from the list**
- **contains all values from the list**
- **does not contain all values from the list at the same time**

System Fields

Fields automatically filled by the platform or used for internal identification.

Identifier

A unique numeric product ID in the system.

- **is equal to / is not equal to**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Additional Identifier

A string field for storing an external ID (e.g., SKU from 1C or product code).

- **is equal to / is not equal to**
- **is in list / is not in list**
- **is in data table / is not in data table**
- **is in file in column / is not in file in column**
- **is in a field of post url response / is not in a field of post url response**

Product/SKU Creation Date

A *date* type field recording the moment the record was created in the platform.

- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **month is / month is not**
- **month is the same as today**
- **month day is / is not / is greater than / is less than / is the same as current**
- **day and month are / are not / are greater than / are less than / are the same as current**
- **is in list / is not in list**
- **is in a date range**
- **date relative to current date — selection based on a moment in the past or future (e.g., “created in the last 7 days”)**

Product/SKU Update Date

A *date* type field recording the moment the record was last modified.

- **is equal to / is not equal to**
- **is greater than / is less than**
- **is greater than or equal to / is less than or equal to**
- **month is / month is not**
- **month is the same as today**
- **month day is / is not / is greater than / is less than / is the same as current**
- **day and month are / are not / are greater than / are less than / are the same as current**
- **is in list / is not in list**
- **is in a date range**
- **date relative to current date**

Custom Fields

You can create your own fields in the catalog structure to store specific product information. For segmentation, operators corresponding to the created field type (string, numeric, date, boolean, etc.) are available.

Preparing the YML file

Products in the Market are represented by two entities: products and SKU.

A product is a good or service for sale. SKU is a product modifier, i.e. a group of goods that stands out in one product according to certain parameters (color, size, weight, model, equipment, etc.).

To upload product data to the platform, a file in yml format or a .zip/.tar.gz archive containing yml files is used. If you are setting up import on a schedule, you must provide a link to the yml file.

The YML (Yandex Market Language) format is a data format for product platforms developed by Yandex. You can view its full structure at this link.

The structure of such files looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2017-02-05 17:22">
  <shop>
    ...
    <offers>
      ...
    </offers>
    <gifts>
      ...
    </gifts>
    <promos>
      ...
    </promos>
  </shop>
</yml_catalog>
```

If you want to upload product data to the platform, use yml files or .zip/.tar.gz archives containing yml files.

A list of all products and SKUs to import is passed in the `offers` element. More information you will find here.

Each product and SKU is described in a separate `offer` element. In this element you need to specify an `id` attribute that is unique for each item:

```
<offer id="9012">
```

Importing a SKU is similar to importing a product. The only difference is the structure of a yml file.

When importing a SKU, you'll have to specify not only `id` , but also `group_id` (the product identifier) for the `offer` element. This is because the SKU is always tied to a specific product.

```
<offer id="1234" group_id="9012">
```

TIP

Please note that SKUs are different variations of the same product, so the value of the `group_id` attribute must be the same for all SKUs of the same product.

Examples

Product

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2020-11-22T14:37:38+03:00">
    <shop>
        <name>BestSeller</name>
        <company>Tne Best inc.</company>
        <url>http://best.seller.ru</url>
        <currencies>
        </currencies>
        <categories>
            <category id="1">Appliances</category>
            <category id="10" parentId="1">Small kitchen appliances</category>
        </categories>
        <delivery-options>
        </delivery-options>
        <offers>
            <offer id="9012">
                <name>Cream freezer Brand 3811</name>
                <url>http://best.seller.ru/product_page.asp?pid=12345</url>
                <price>200</price>
                <currencyId>USD</currencyId>
                <categoryId>10</categoryId>
                <delivery>true</delivery>
                <delivery-options>
                </delivery-options>
                <param name="Color">white</param>
                <weight>3.6</weight>
                <dimensions>20.1/20.551/22.5</dimensions>
            </offer>
        </offers>
        <gifts>
            <!-- gifts not from the price list -->
        </gifts>
        <promos>
            <!-- promotions -->
        </promos>
    </shop>
</yml_catalog>
```

SKU

```
<?xml version="1.0" encoding="UTF-8"?>
<yml_catalog date="2020-11-22T14:37:38+03:00">
    <shop>
        <name>BestSeller</name>
        <company>Tne Best inc.</company>
        <url>http://best.seller.ru</url>
        <currencies>
        </currencies>
        <categories>
            <category id="1">Appliances</category>
            <category id="10" parentId="1">Small kitchen appliances</category>
        </categories>
        <delivery-options>
        </delivery-options>
        <offers>
            <offer id="1234" group_id="9012">
                <name>Cream freezer Brand 3811 v.2</name>
                <url>http://best.seller.ru/product_page.asp?pid=12345</url>
                <price>200</price>
                <currencyId>USD</currencyId>
                <categoryId>10</categoryId>
                <delivery>true</delivery>
                <delivery-options>
                </delivery-options>
                <param name="Color">white</param>
                <weight>3.6</weight>
                <dimensions>20.1/20.551/22.5</dimensions>
            </offer>
        </offers>
        <gifts>
            <!-- gifts not from the price list -->
        </gifts>
        <promos>
            <!-- promotions -->
        </promos>
    </shop>
</yml_catalog>
```

Loyalty programs

Description

Loyalty programs can be used in marketing to increase customer involvement in buying process and to keep the customer until next purchase.

In this section

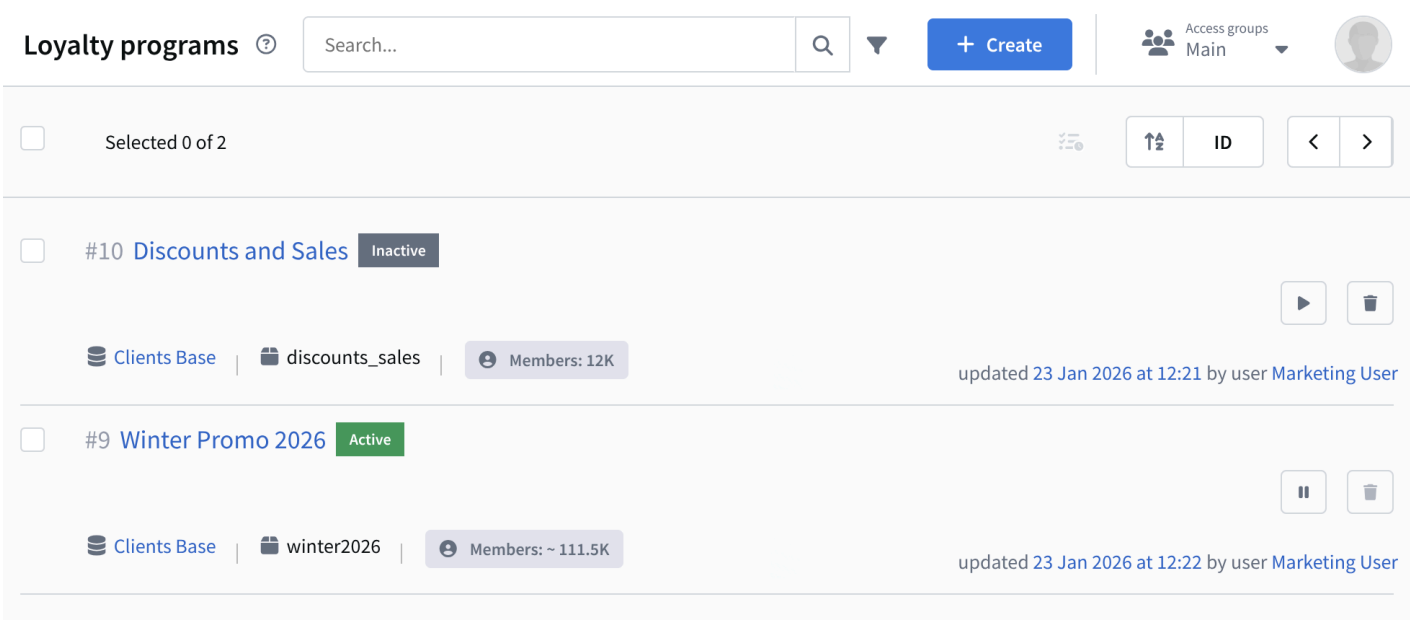
Loyalty programs

A loyalty program is a customer reward system for target actions such as purchases, registration, content viewing, and more. Customers earn points that can be spent on discounts, movies, plans, and services. The program includes tiers (bronze, silver, etc.), reporting, rules for points accrual and expiration, as well as integrations with other services and communication channels.

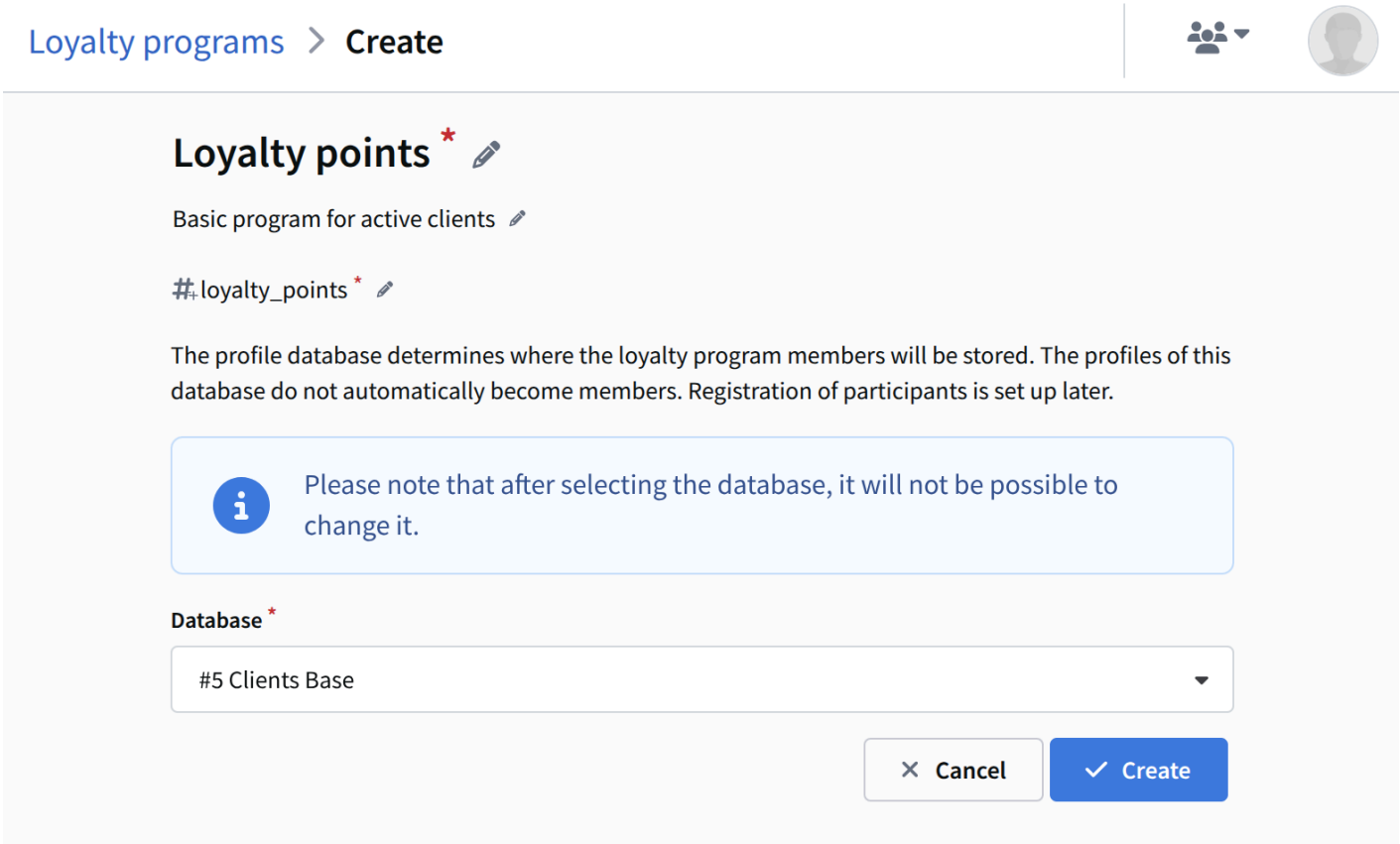
Creating a loyalty program

	To manage loyalty programs, go to **Loyalty — Loyalty programs in the main menu. A list of existing loyalty programs will open. To edit a program, click its title.**

To start a new loyalty program, click + Create at the top of the page:



The loyalty program settings window will open. Specify the name of your program and its external identifier. Optionally, add a description:



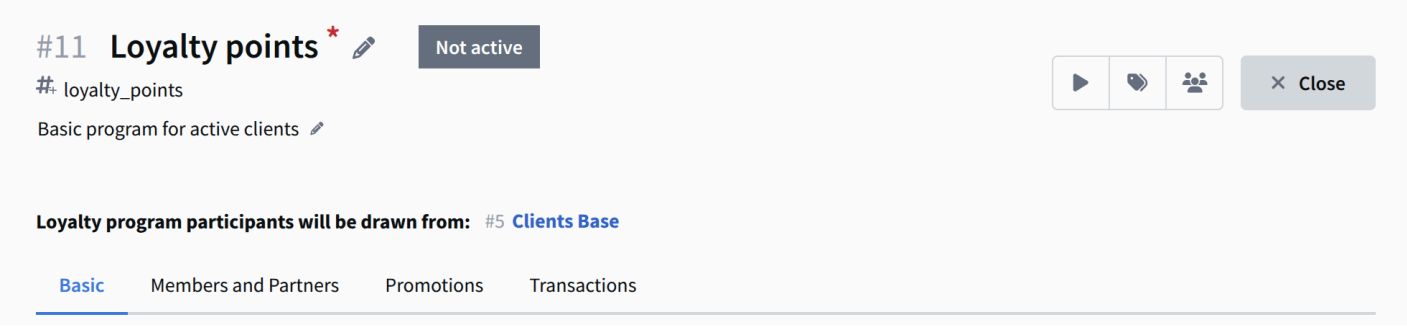
Select the database from which participants will be taken.

CAUTION

Please note: after selection, the database cannot be changed; only one loyalty program can be active in a single database at a time.

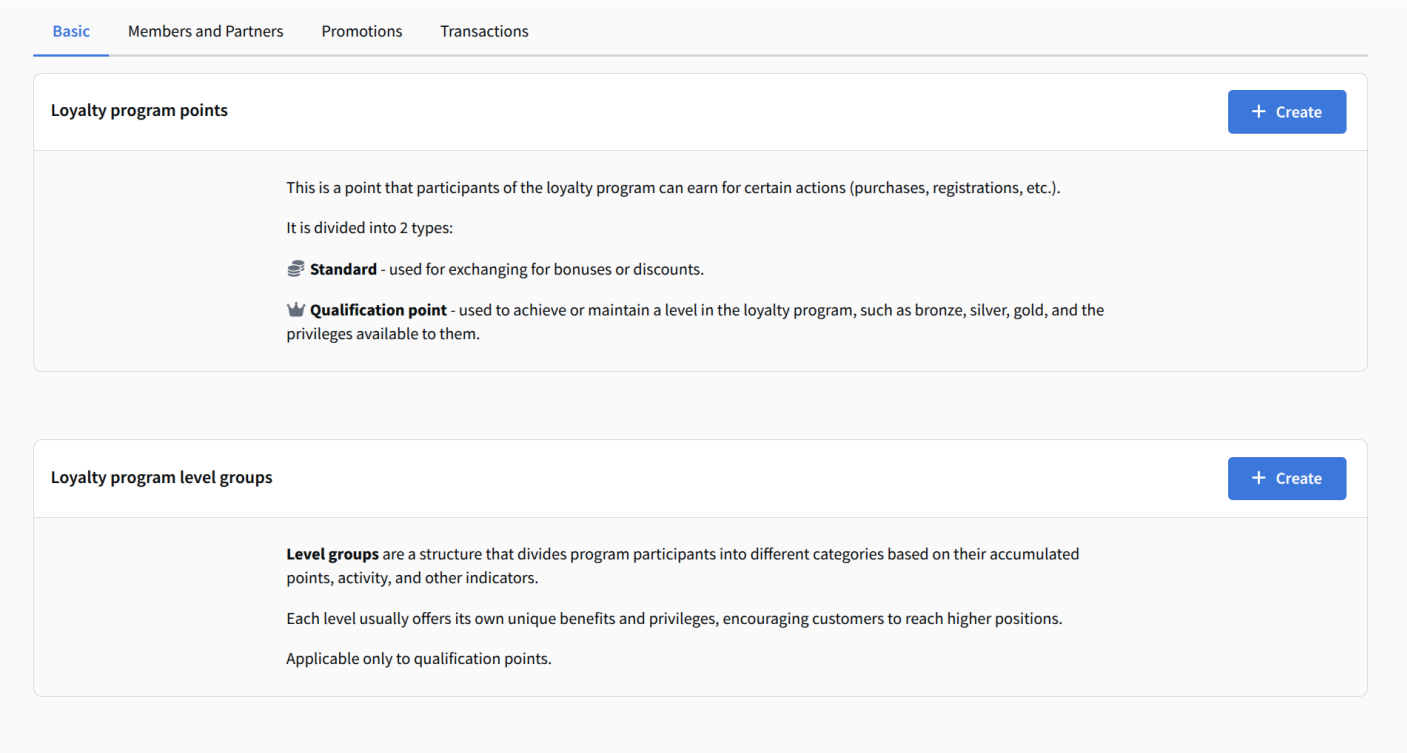
Editing a loyalty program

After creation, you will be taken to the loyalty program editor, which consists of several tabs: General, Members and partners, Promotions, and Transactions:



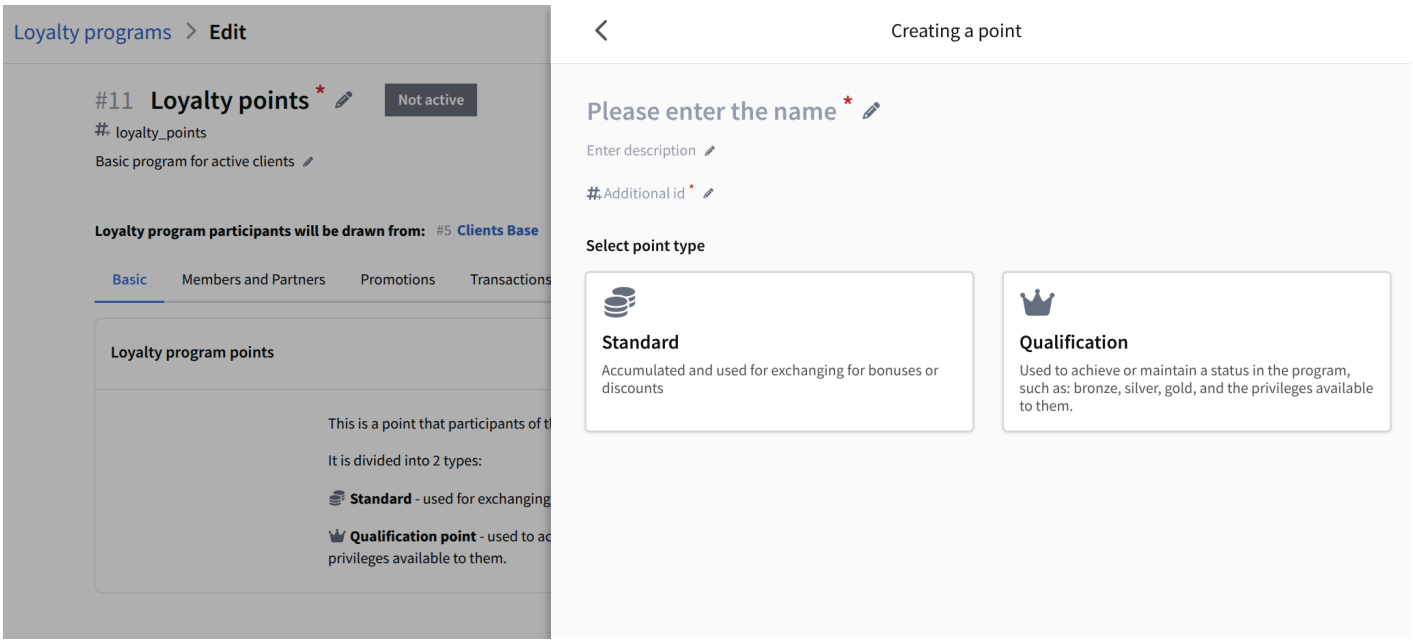
General

The General section is used to configure the basic structure of the loyalty program. It defines the key parameters that determine how points are accrued, used, and tracked, as well as how customers progress through loyalty tiers.



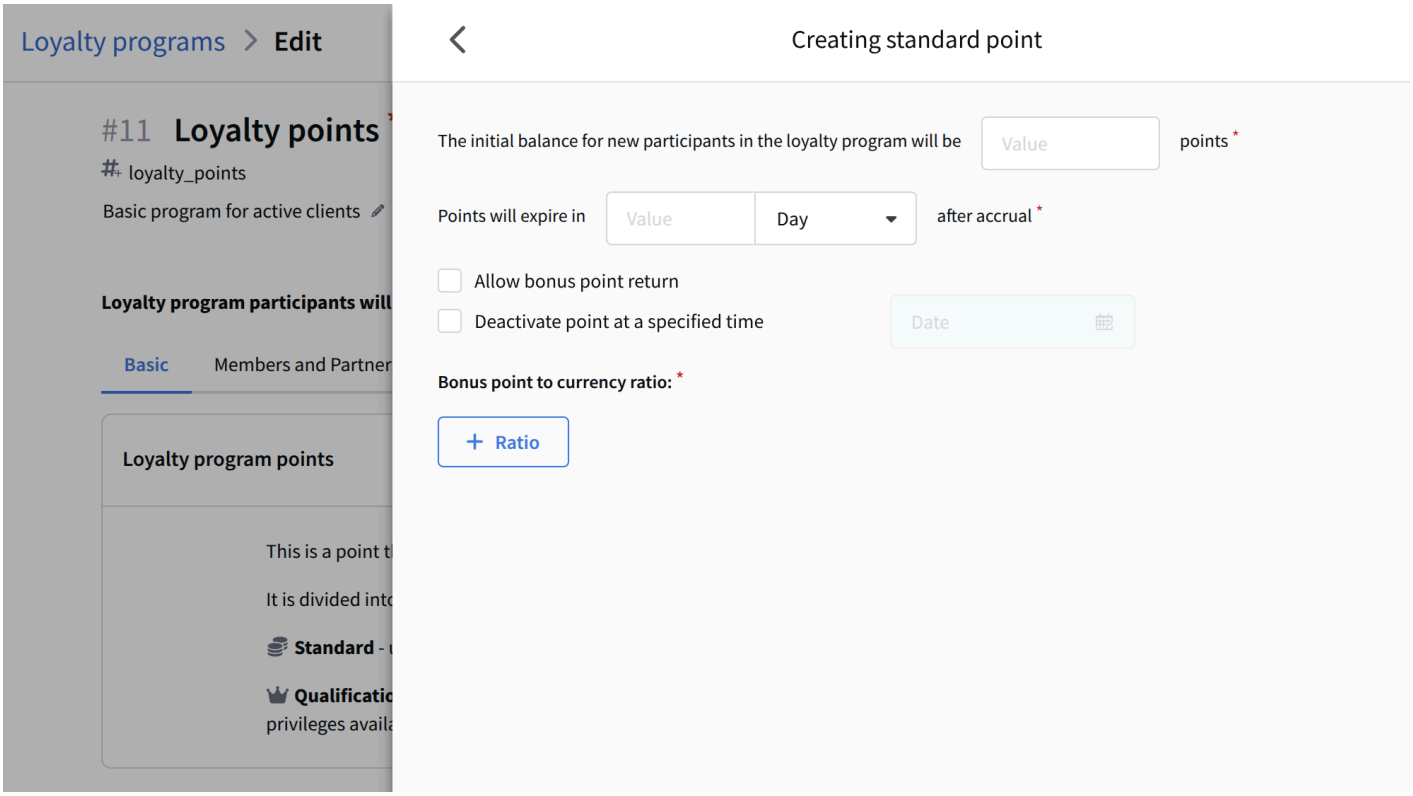
Loyalty program points

Loyalty program points — this block is used to create and manage points types within the loyalty program. Specify the points name and external identifier, and optionally add a description.



There are two types of points:

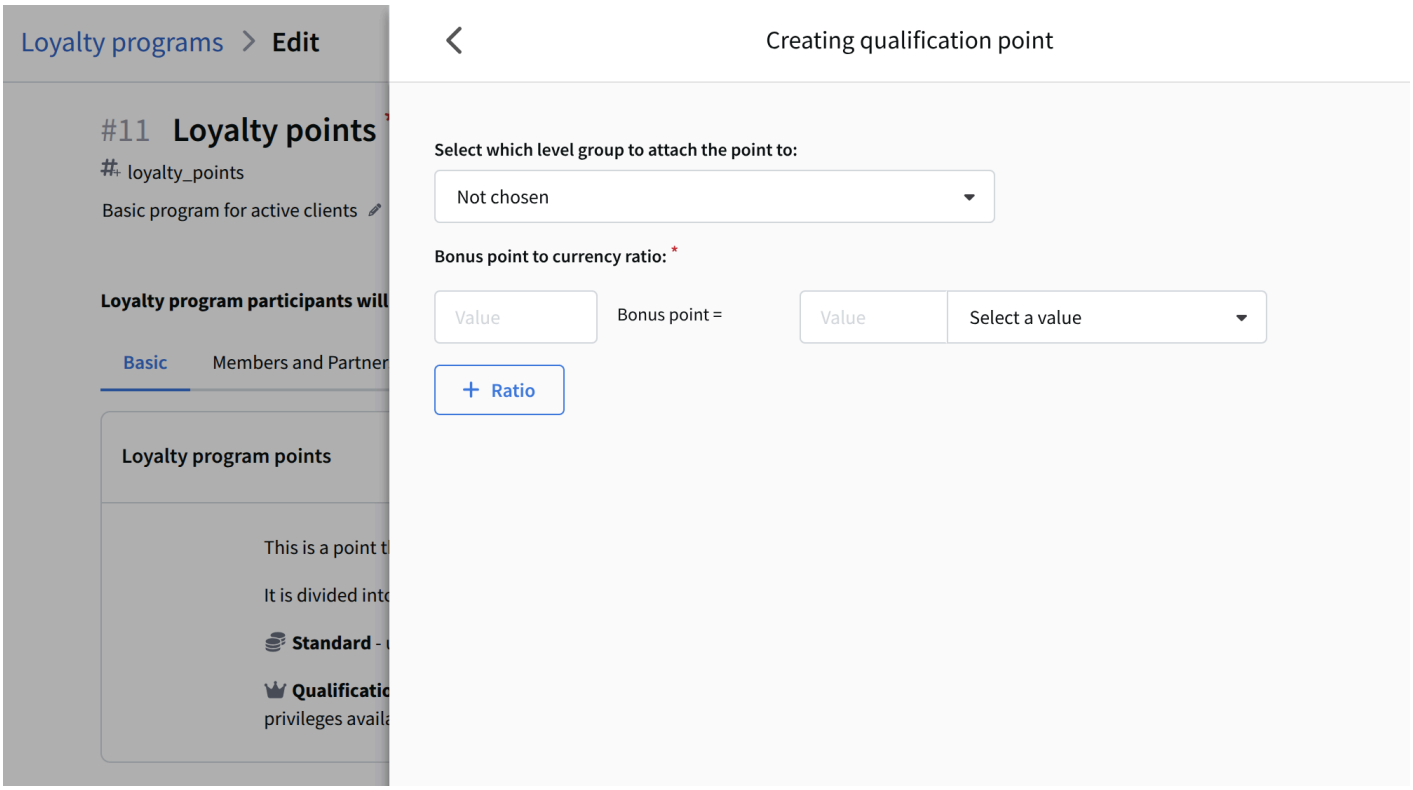
Standard points — points credited to the bonus account that the customer can later use to fully or partially pay for purchases.



When creating standard points, you can configure the following parameters:

- **Initial balance for a new loyalty program member — the number of bonus points automatically granted to a new member upon joining the program. The value is specified in points.**
- **Points validity period — defines how long accrued points remain valid. The period can be set in days, weeks, months, or years. After expiration, points are canceled.**
- **Allow bonus points refund — enables returning bonus points to a member, for example, in case of a product return or service cancellation.**
- **Deactivate points at a specific time — sets a date after which the points become inactive (not accrued or used). Useful for temporary or seasonal programs.**
- **Bonus-to-currency ratio — a required parameter defining how many bonus points correspond to a monetary amount, for example: 1 point = 1 USD or 10 points = 1 USD. Used for discount and redemption calculations.**

Qualification points — used to evaluate customer activity and determine their tier in the loyalty program (bronze, silver, gold, etc.).



When creating qualification points, you can configure:

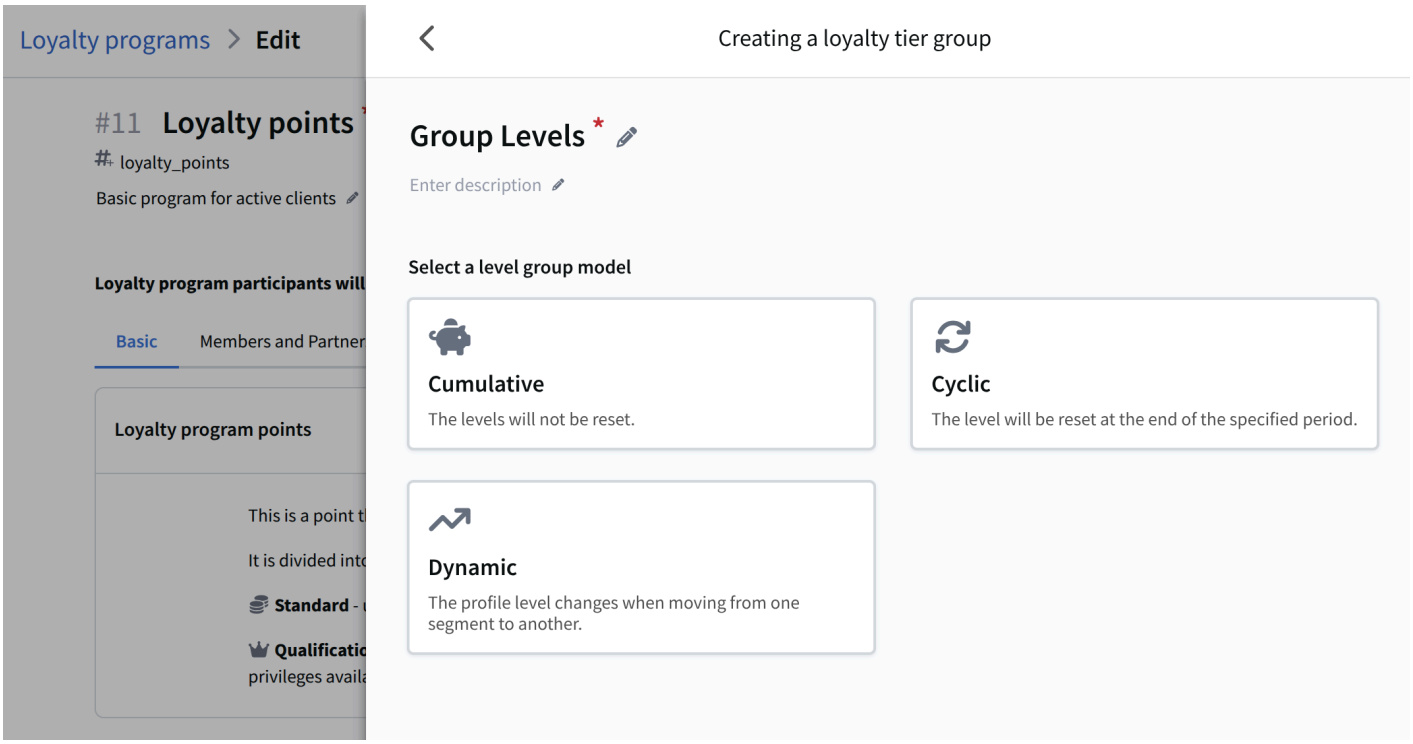
- **Allow bonus points refund — enables returning bonus points to a member.**
- **Tier group binding — links the points to a specific loyalty tier group (for example, Bronze, Silver, Gold). This can be used to configure different accrual rules or rates depending on the customer's tier.**
- **Bonus-to-currency ratio — defines the conversion rate between points and monetary value.**

Do not forget to activate the created points. If needed, points can be cloned to create another type based on them.

Loyalty tier groups

Loyalty tier groups allow you to segment loyalty program members into different levels or statuses based on activity, accumulated points, or other criteria.

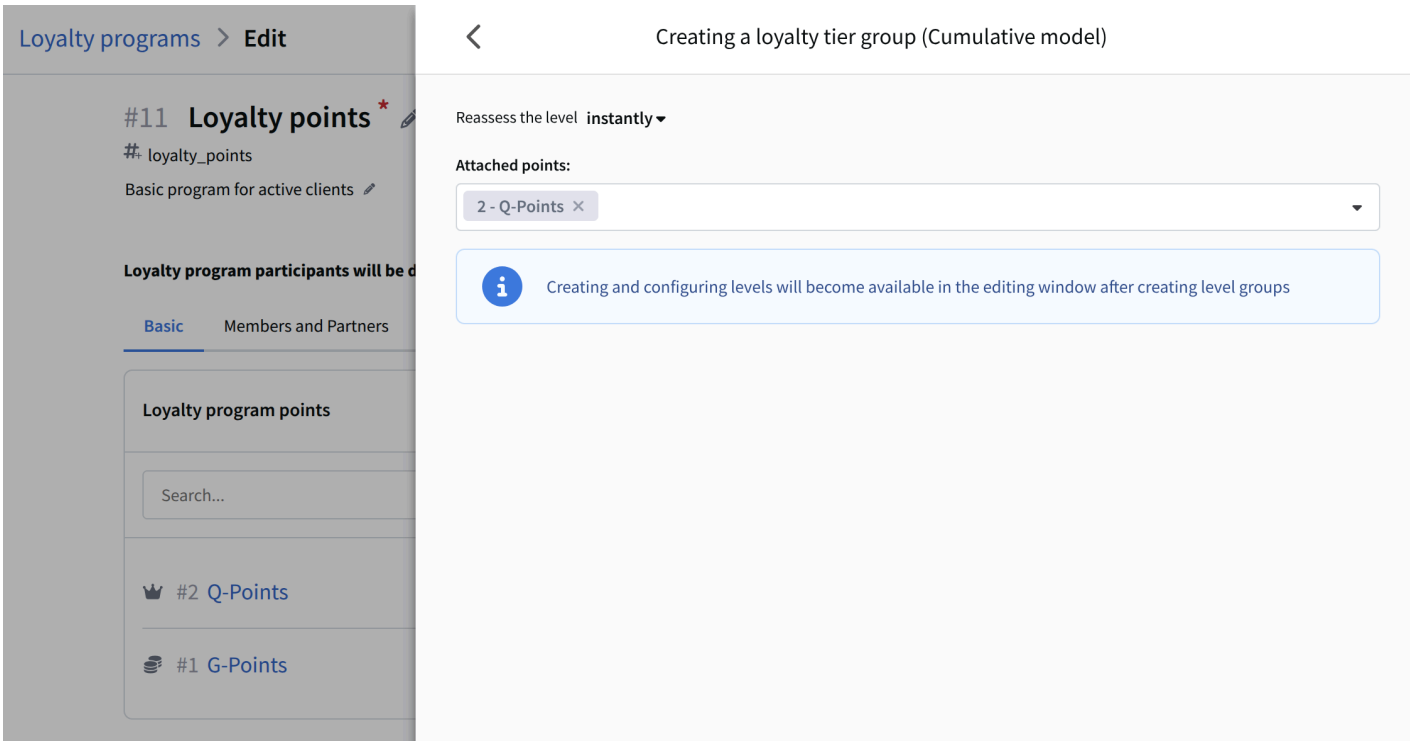
Specify the name and external identifier of the tier group, and optionally add a description:



Several tier group models are available:

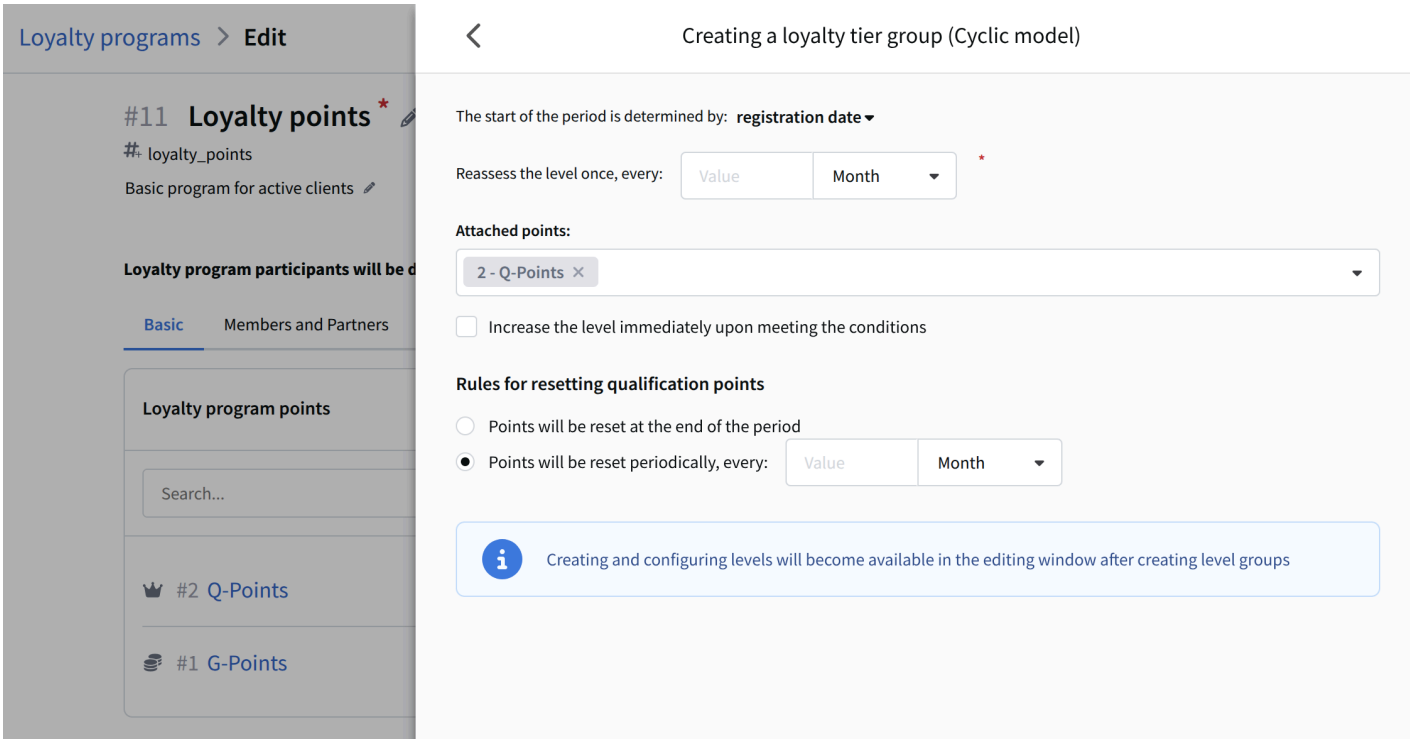
- **Cumulative model — once a member reaches a tier, it is retained permanently. After a user accumulates enough points or meets the necessary conditions, they receive a status that stays with them and does not reset over time. This is convenient for programs aimed at**

long-term motivation and rewarding loyalty.



Specify the points type bound to this model.

- **Cyclic model — the achieved tier is valid only for a defined period. After the period ends, the system reevaluates the member's status: if they have not confirmed their activity or met the required conditions, the tier may be lowered.**



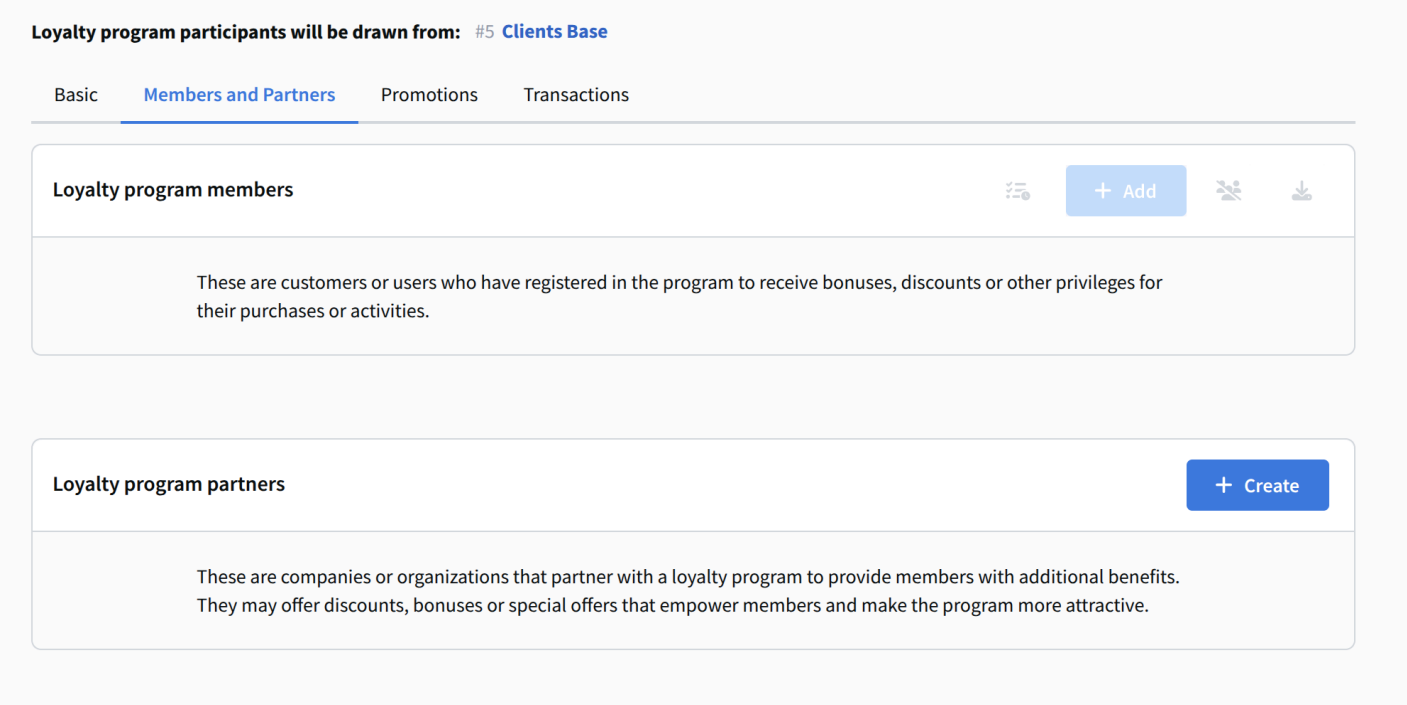
Configure the model:

- **Set the interval for tier reevaluation;**
- **Specify the points type bound to the model;**
- **Enable Upgrade tier immediately when conditions are met to apply tier changes instantly;**
- **Configure qualification points reset rules, you can choose reset at the end of each period or at fixed intervals (for example, every 6 months);**
- **Optionally enable Extend tier until the end of the month — the tier is retained until the end of the current calendar month, even if the member did not meet the conditions to maintain it.**
- **Dynamic model — the member's tier directly depends on their current characteristics or activity and changes automatically when conditions change.**

After completing the basic settings, go to Members and partners.

Members and partners

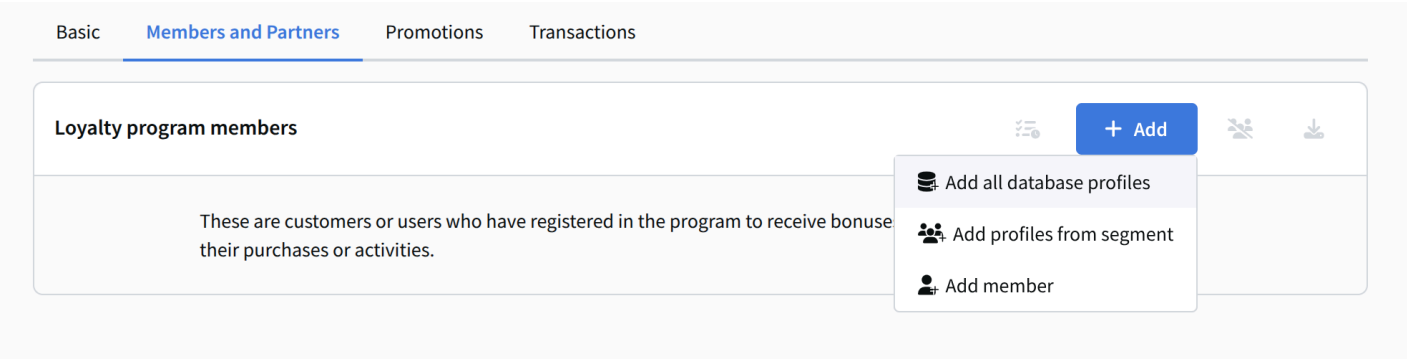
The Members and partners section is used to manage loyalty program members (end users) and partners (companies providing additional benefits).



This section includes the following elements:

Loyalty program members — customers registered in the loyalty system. They earn points for specific actions (for example, payments, content viewing, data confirmation) and can redeem points for discounts, content, or additional options.

To add members, click + Add. Three options are available:



1. Add all database profiles — mass registration of all customers in the database. 2. Add profiles from a segment — registration of a selected group of users, for example, active customers, those who recently paid for a service, etc. 3. Add a member — manual registration of a single customer.

INFO

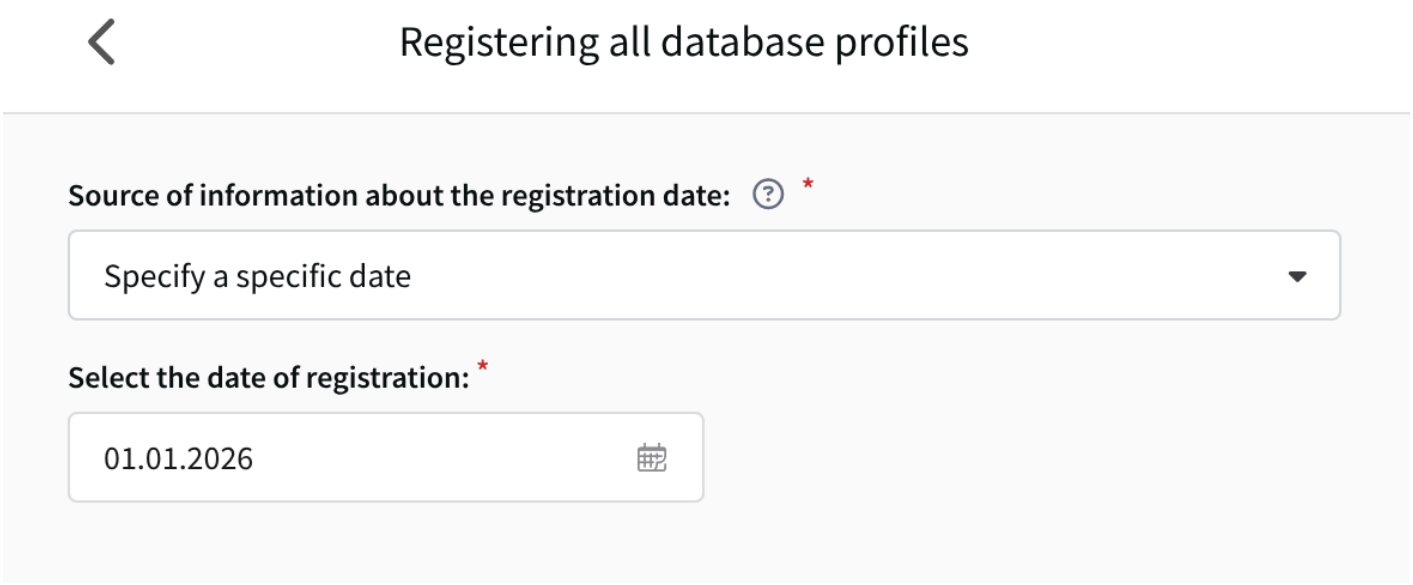
Additionally, the following methods for adding members are available:

- Automation via the "Loyalty Participation" node — automatically adding customers within scenarios when specified conditions are met, for example, after submitting a form.

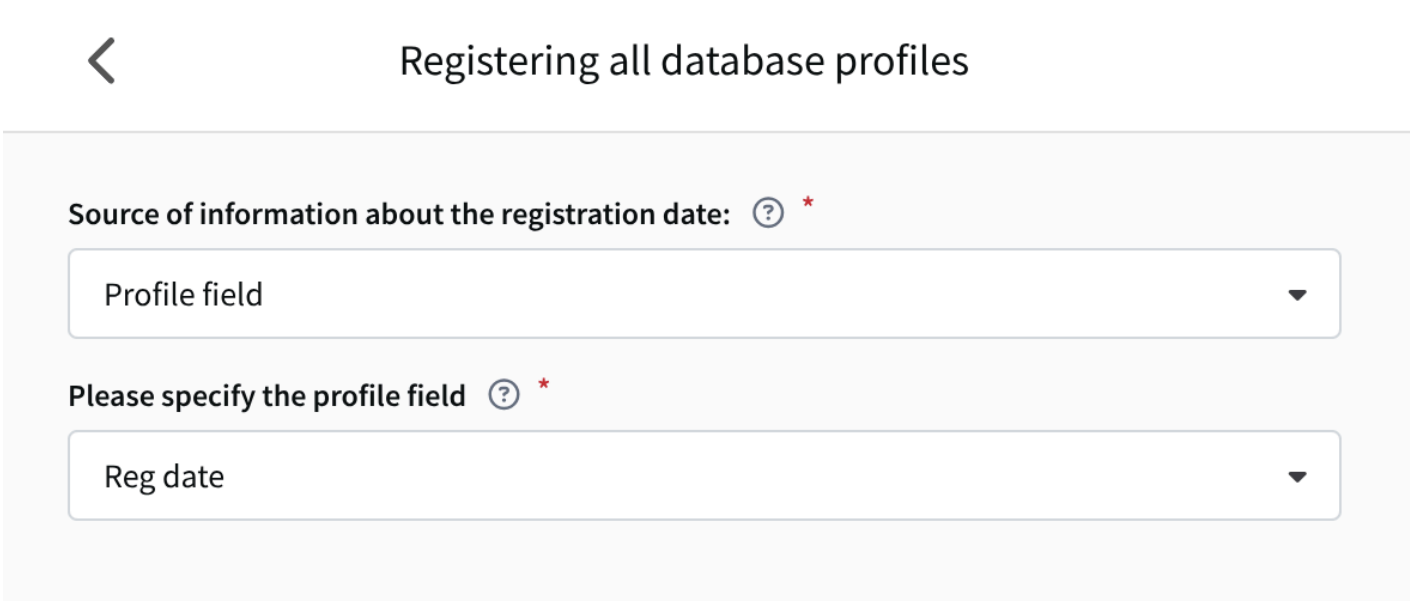
- Integration via API — adding members programmatically using an API call. Suitable for connecting external systems and services.

Specify the registration date source.

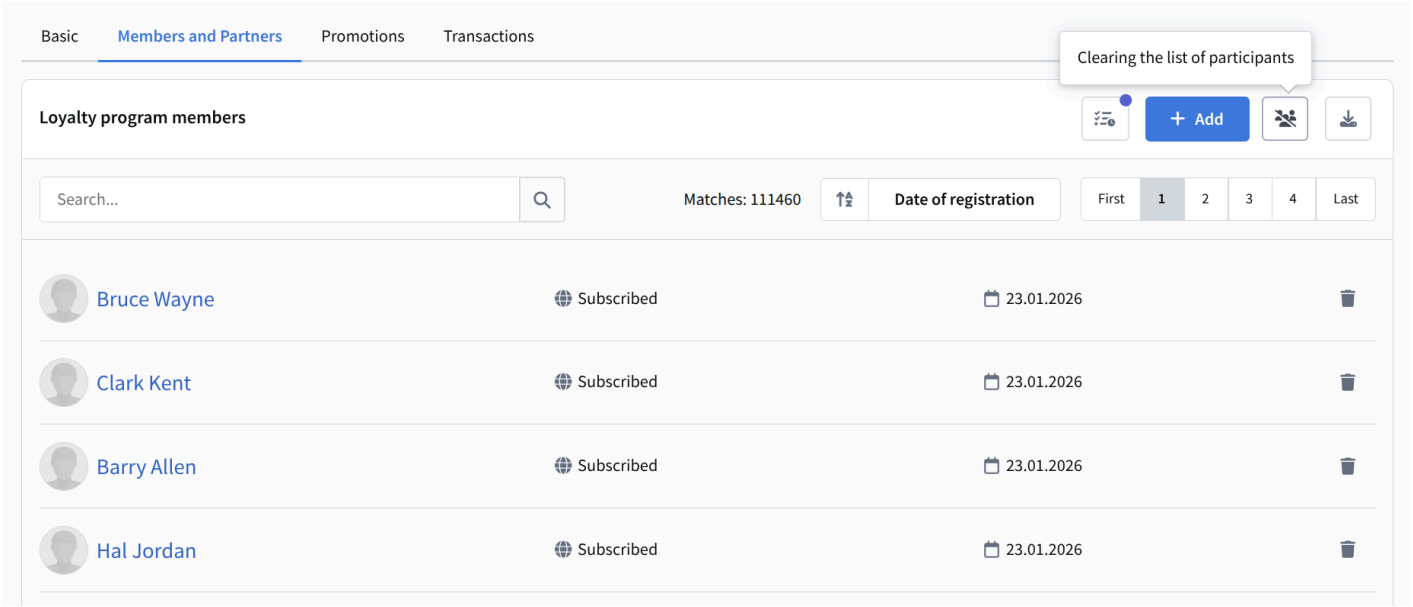
It can be a specific date:



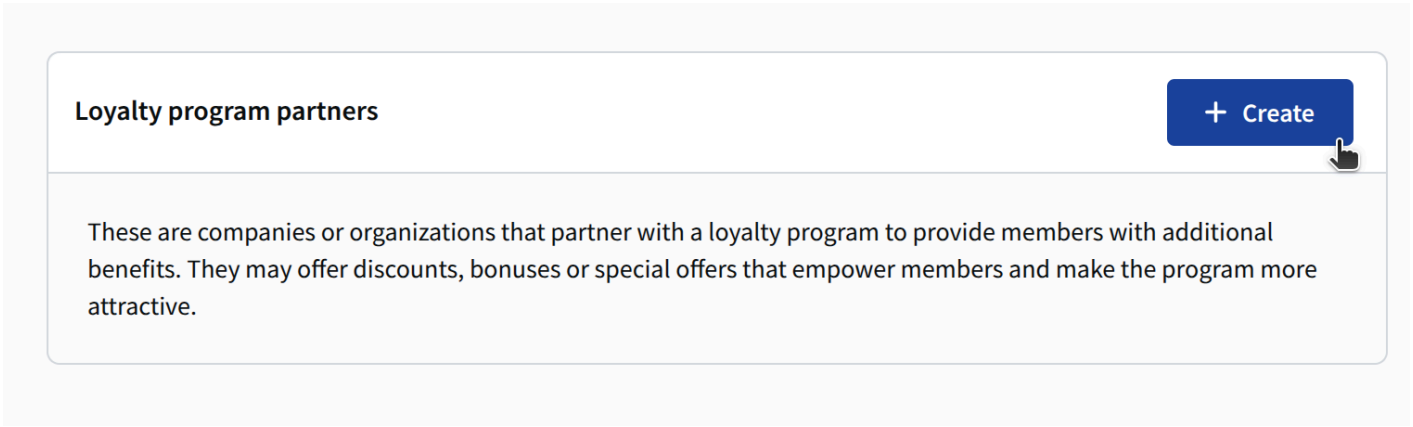
Or a profile field:



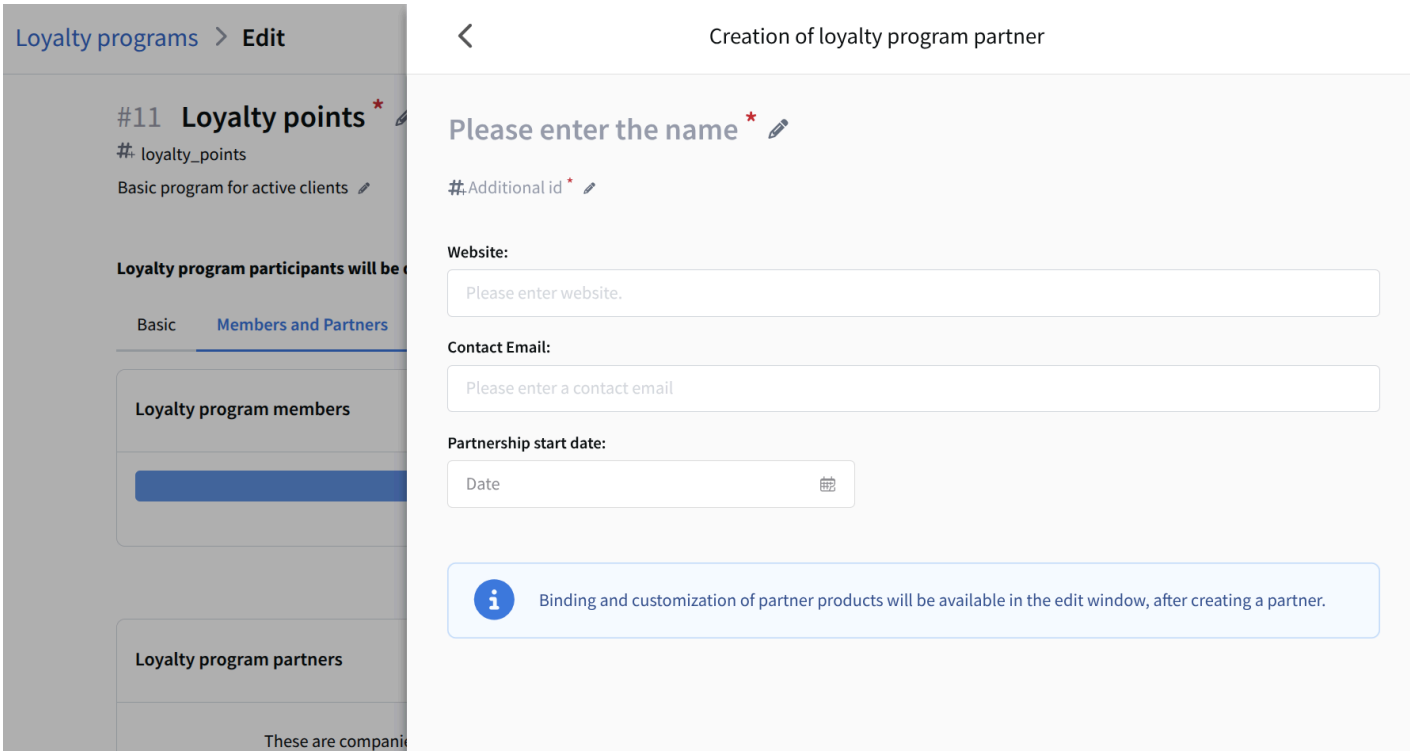
After members are added, you can remove individual members or all members using Clear members list, and export the list using the corresponding button.



Loyalty program partners — this section manages organizations cooperating with the loyalty program to expand its capabilities. Partners provide participants with additional privileges — from discounts to exclusive offers.



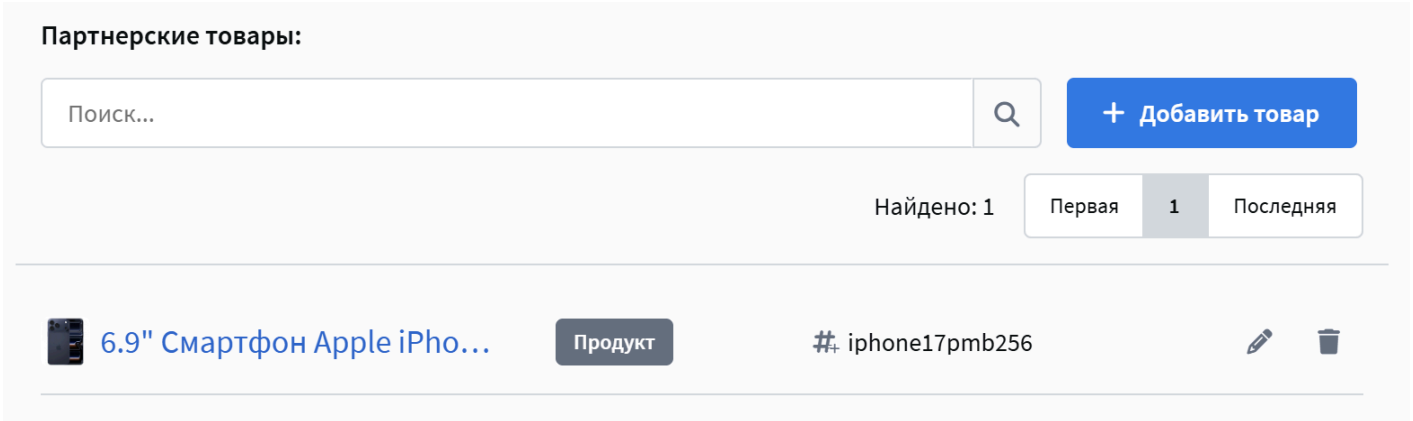
Click + Create to add a new partner. The partner settings window will open:



Specify the partner name and external identifier. Optionally, add a link to the partner's official website for reference, provide a contact email, and set the partnership start date.

After creating and saving the partner, you can link Partner products from the Market to which partner terms will apply.

For each product, a discount percentage is configured, which will be provided to loyalty program participants when purchasing partner products. These settings are used in promotions with the "Discount from a partner" benefit and are taken into account when calculating an order.



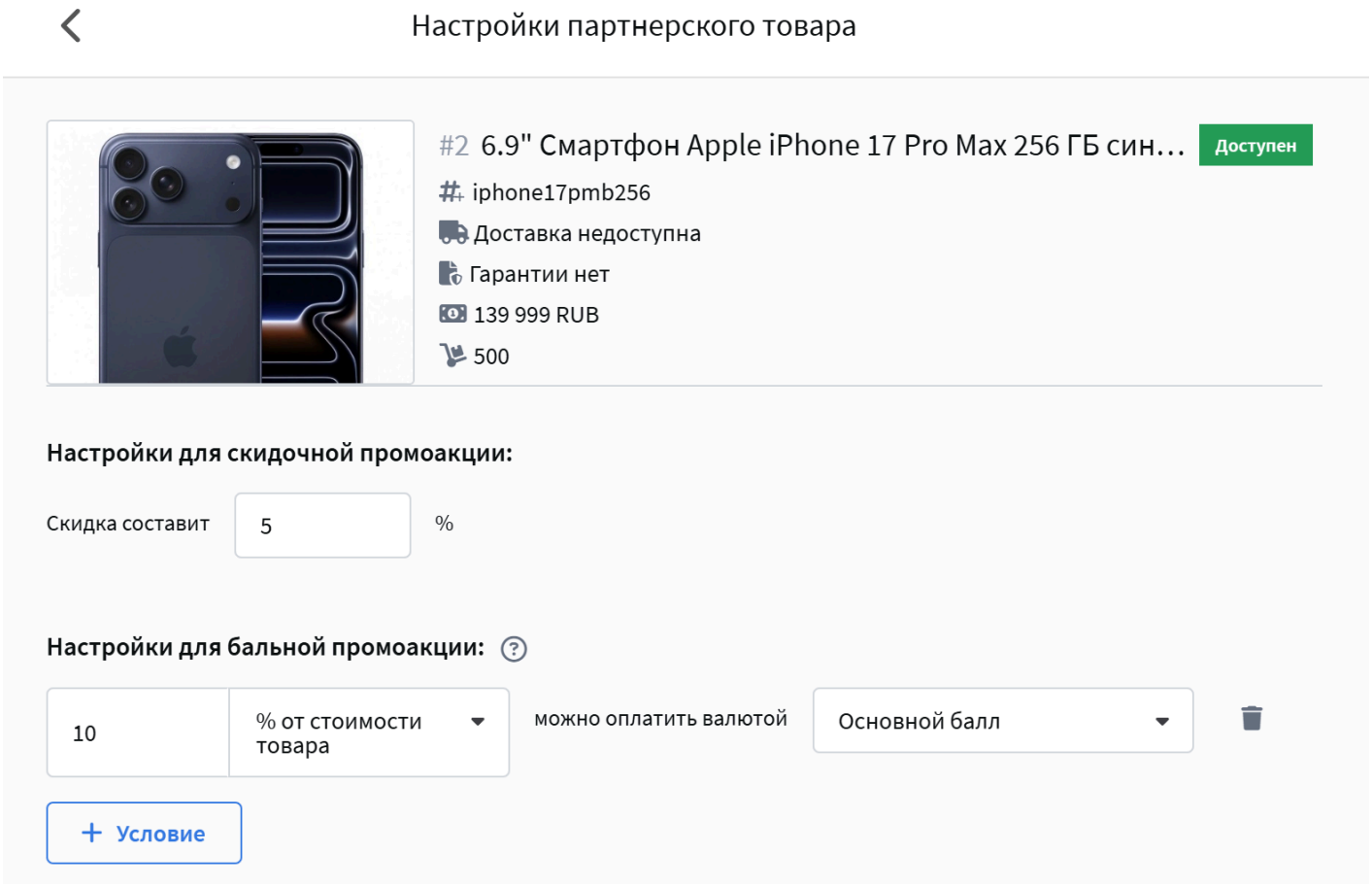
When adding partner products, you can also set application conditions depending on the type of promotion.

Settings for discount promotions:

- **Discount will be % — discount type;**

Settings for points promotions:

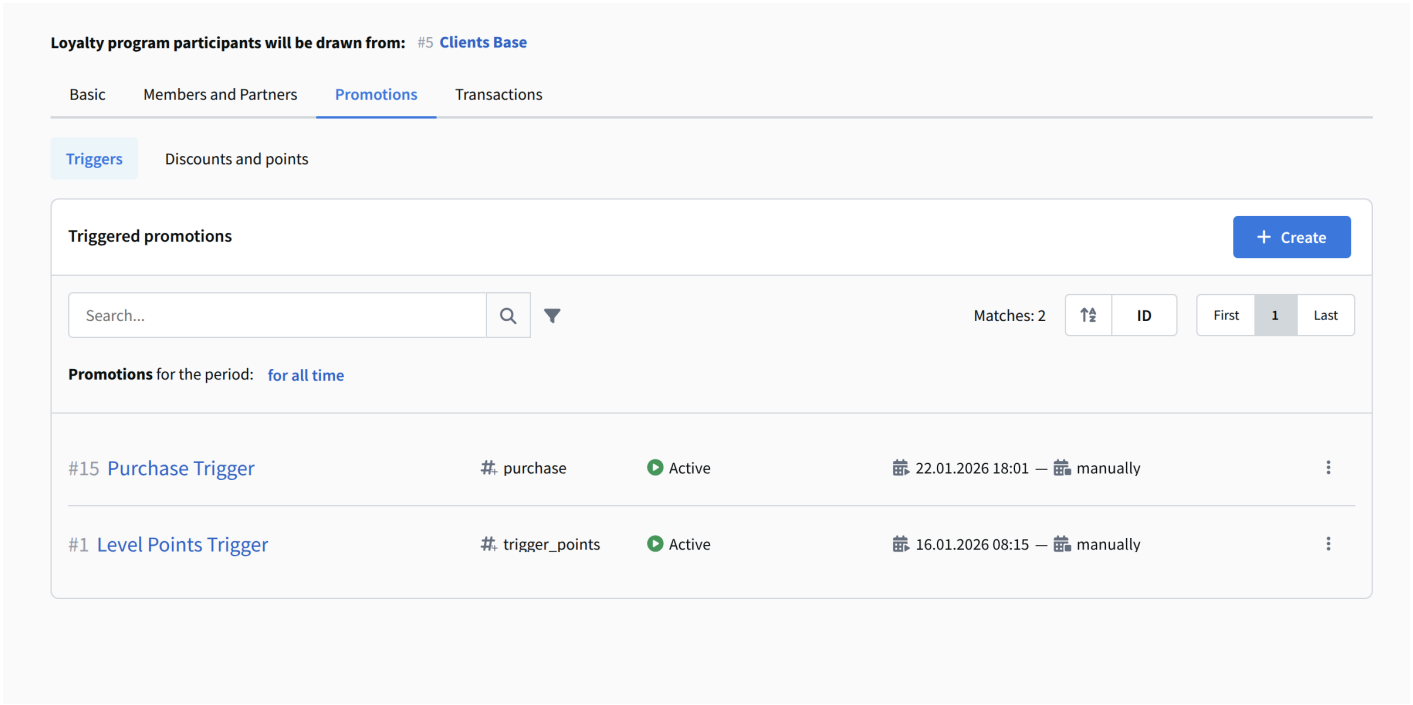
- **Value — the number of points or the percentage of the product price to be used for calculation;**
- **Points / % of product price — choice of calculation mechanics: points accrual/deduction or applying a discount in %;**
- **Can be paid with currency — determines how many points can be used as a monetary equivalent:**



Partner products are used to form special offers within the loyalty program. Such offers are available only to participants for whom the partnership terms apply and are configured through corresponding promotions with the “Discount from a partner” benefit.

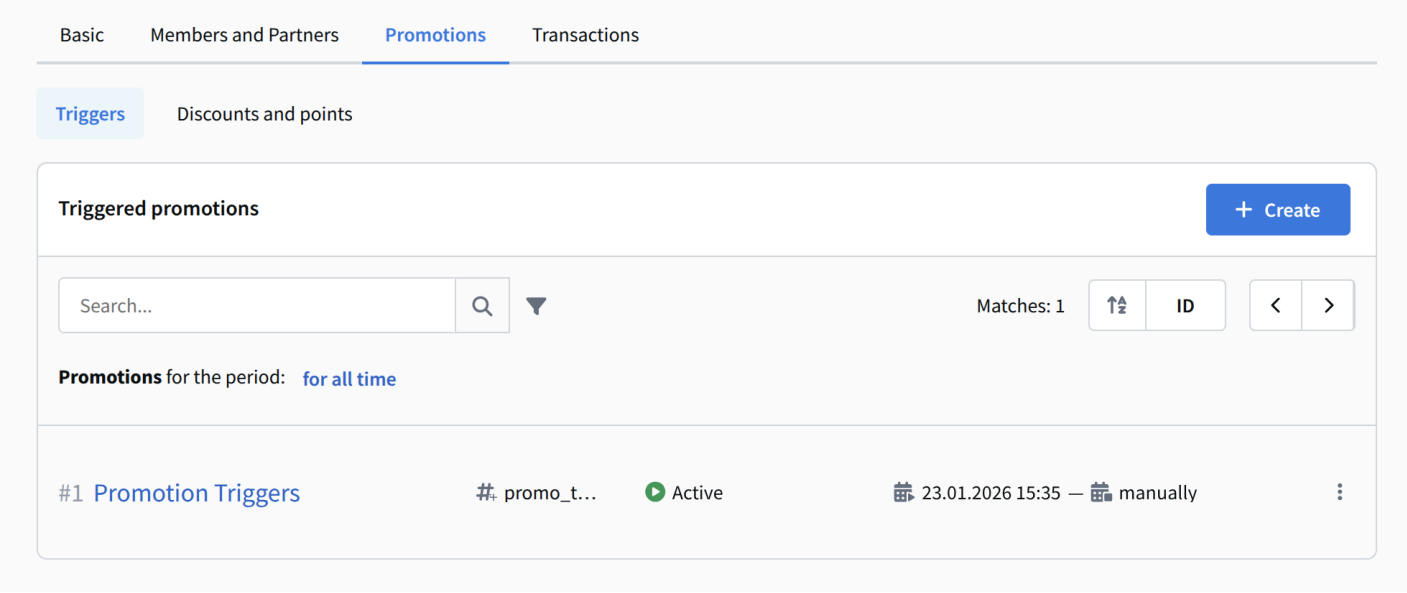
Promotions

The Promotions section is used to create and manage loyalty program promotions that automatically accrue and deduct points, apply discounts, and grant other benefits to orders and products in the Market.



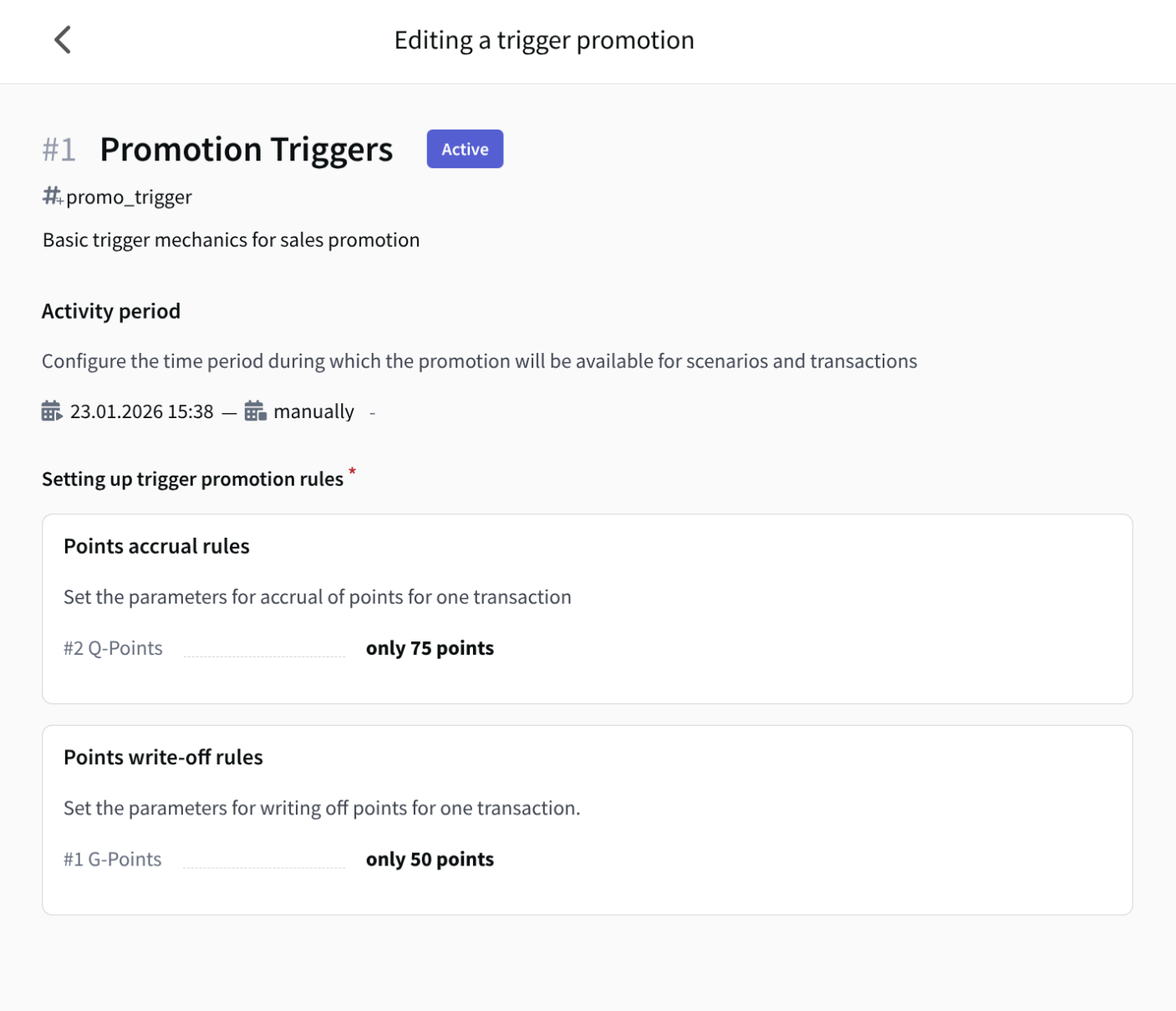
Promotions are divided into two groups: trigger and discount and points promotions.

Trigger promotions



Trigger promotions are marketing activities in which, when rules are met (triggers are activated), a participant becomes eligible to receive bonus points. In other words, this is a set of rules according to which a customer can earn (and later lose) loyalty points. Such promotions are linked to triggers when creating campaigns and scenarios.

To create a new trigger promotion, click + Create. A promotion setup window opens:

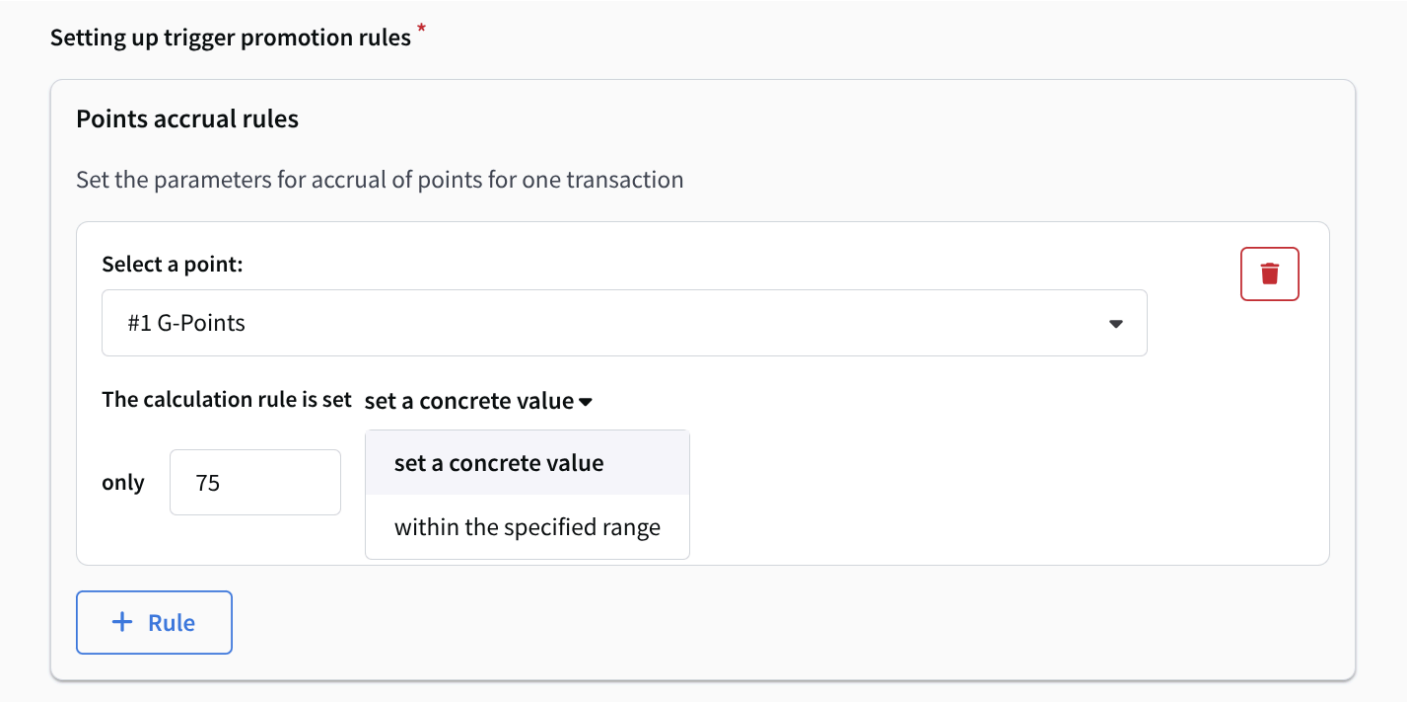


Specify the name of the promotion and its additional identifier. Add a description if needed.

Then set the start and end dates of the promotion:

- **Promotion start — starts immediately after activation or at a specific date and time;**
- **Promotion end — ends manually, at a specific date and time, or after a specified period.**

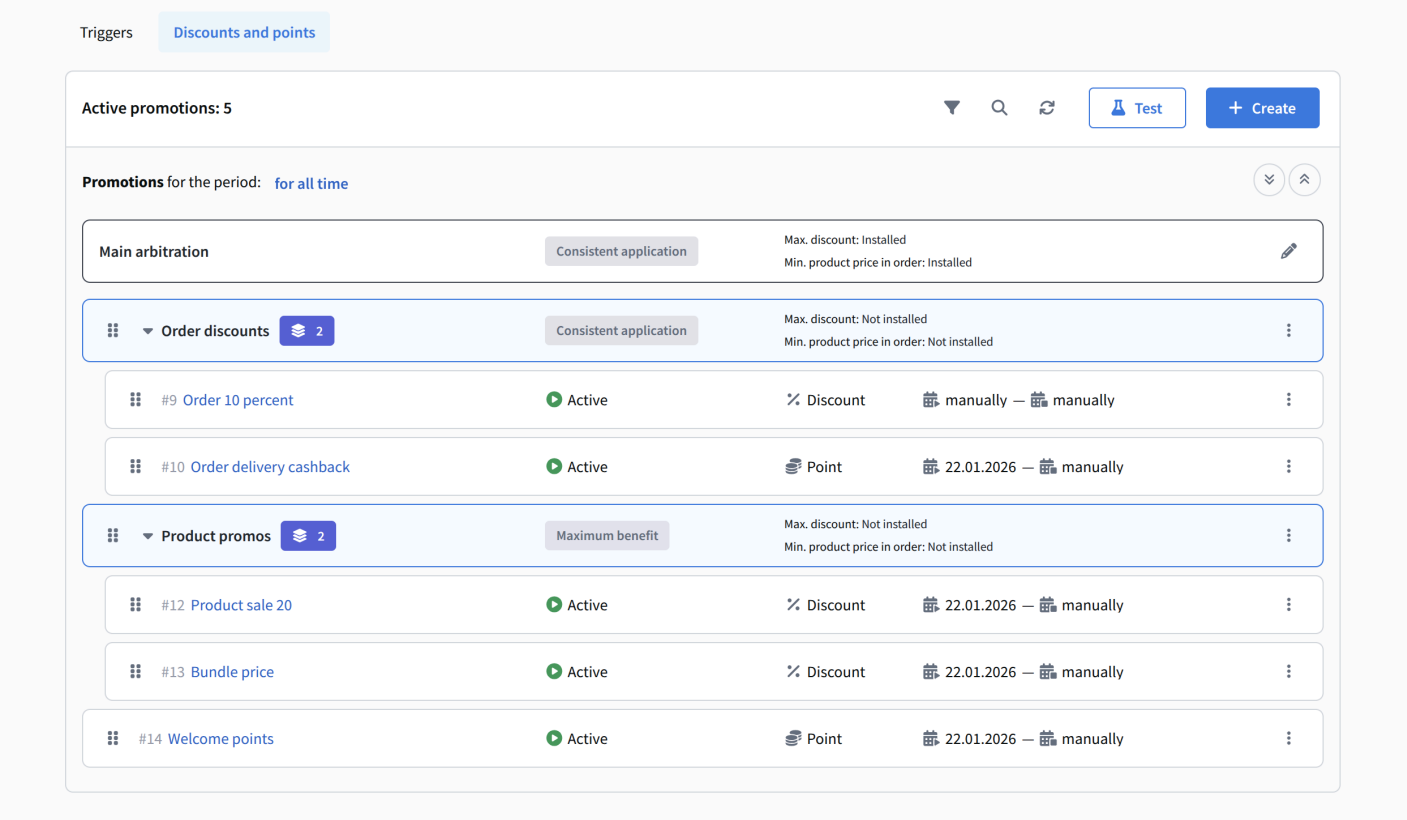
Next, define the promotion rules. They control how points can be accrued and deducted for participants. Accrual and deduction rules are configured as separate conditions. Specify the point type to which the rules apply and the exact value or range of points to be changed:



After saving and activating the promotion, it can be used in trigger-based campaigns and scenarios to automate point accrual. For example, you can set up a promotion that awards bonus points for placing a first order, purchasing products from a selected category, or reaching a specific order amount.

Discount and points promotions

Points and discount promotions are loyalty program promotions that are automatically applied at checkout or during order calculation via the Market. They manage point accrual and deduction, discount application, and other benefits based on configured conditions, audiences, and application rules.



Points promotions manage a participant's loyalty balance. They accrue or deduct points, both standard and qualification, when specified conditions are met, such as purchasing products, reaching an order total, participating in a promotion, or using points for payment. These

promotions do not change product prices directly; they affect the participant's future benefits and loyalty tier. Within points promotions, you can accrue both standard and qualification points, depending on the selected point type and configured benefits. This allows you to simultaneously manage the participant's bonus benefit and their loyalty program tier.

Discount promotions are applied to orders or individual products and change their price. They are used to grant purchase benefits such as discounts, special prices, and other incentives.

Arbitration

Promotions can run simultaneously and apply to the same order or product, so the system requires a mechanism to manage how they work together. For this purpose, the loyalty program includes arbitration.

Arbitration is a promotion application rule that defines how the system processes multiple eligible promotions at the same time. It determines:

- **which promotions are applied;**
- **the order in which they are applied;**
- **whether promotions can be combined;**
- **which promotions apply together and which do not.**

Arbitration is used whenever a single order or product meets the conditions of multiple promotions. It is configured at several levels and works hierarchically.

Main arbitration is the core arbitration applied to all points and discount promotions within the loyalty program. It defines the overall benefit calculation logic and always affects the final result, regardless of promotion structure.

Even if promotions are grouped or have their own rules, the final calculation always goes through the main arbitration.

Group arbitration allows promotions to be combined into groups, each with its own arbitration settings. It defines how promotions within the group interact. Groups can also contain subgroups.

Group arbitration works locally, only within its group. Its result is then passed upward and processed by the loyalty program's main arbitration.

In-promotion arbitration applies when a single promotion contains multiple benefits. It defines whether benefits are applied in parallel or sequentially.

There are several types of arbitration:

- **Consistent application — promotions are applied one after another in the order defined within the group. Each subsequent promotion is calculated based on the result of the previous one. For example, if one discount is applied first and another follows, the second discount is calculated from the reduced price. Suitable when order and step-by-step price or balance changes matter.**
- **Incompatibility — only one of the eligible promotions is applied. At the product or order level, the first promotion that meets its conditions is applied, and the rest are ignored. Used when promotions logically exclude each other, for example, "either a discount or a gift".**
- **Summation — all eligible promotions are applied simultaneously. Discounts are added together, and accrued points are summed. The result is calculated as the combined effect of all promotions in the group. Suitable when multiple promotions should reinforce each other.**
- **Maximum benefit — the system evaluates possible promotion combinations and selects the one that provides the greatest benefit to the participant. Used in complex promotional mechanics where the best option should be selected automatically without manual priority setup.**

CAUTION: Arbitration type limitations

Not all benefits are supported by some arbitration types:

For "Maximum benefit" arbitration, the following are unavailable:

- **Display templated text**
- **Issue a promo code**

For "Summation" arbitration, the following are unavailable:

- **Round price / Round order price**
- **Use price from field**
- **Award points from a custom field**

Additional limits can be configured for arbitration to control the final result and prevent excessive discounts or overly low prices:

- **Maximum discount — limits the maximum discount that can be applied regardless of the number or combination of triggered promotions. If the total discount exceeds the limit, the system caps it automatically.**
- **Minimum product price — sets the lowest allowed product price after all promotions are applied. If promotion calculations result in a price below this minimum, the system adjusts the result and stops further reduction.**

Limits configured in the main arbitration apply to all promotions in the loyalty program.

Arbitration keeps promotion outcomes predictable and controlled, even with complex discount and bonus structures.

TIP: Practical example

Assume a customer wants to buy an all-in-one PC for ₽100,000. They have access to a 20% birthday discount and can redeem 15% of the price with bonus points. Here is how it works with different arbitration types:

Consistent application 100,000 → −20% = 80,000 → −15% = ₽68,000 Promotions are applied one after another from the result of the previous one.

Incompatibility 100,000 → either −20% = ₽80,000, or −15% = ₽85,000 Only one promotion is applied.

Summation 100,000 → −(20% + 15%) = ₽65,000 All promotions are added together.

Maximum benefit Comparison: 80,000 / 85,000 / 68,000 / ₽65,000 The lowest price is selected.

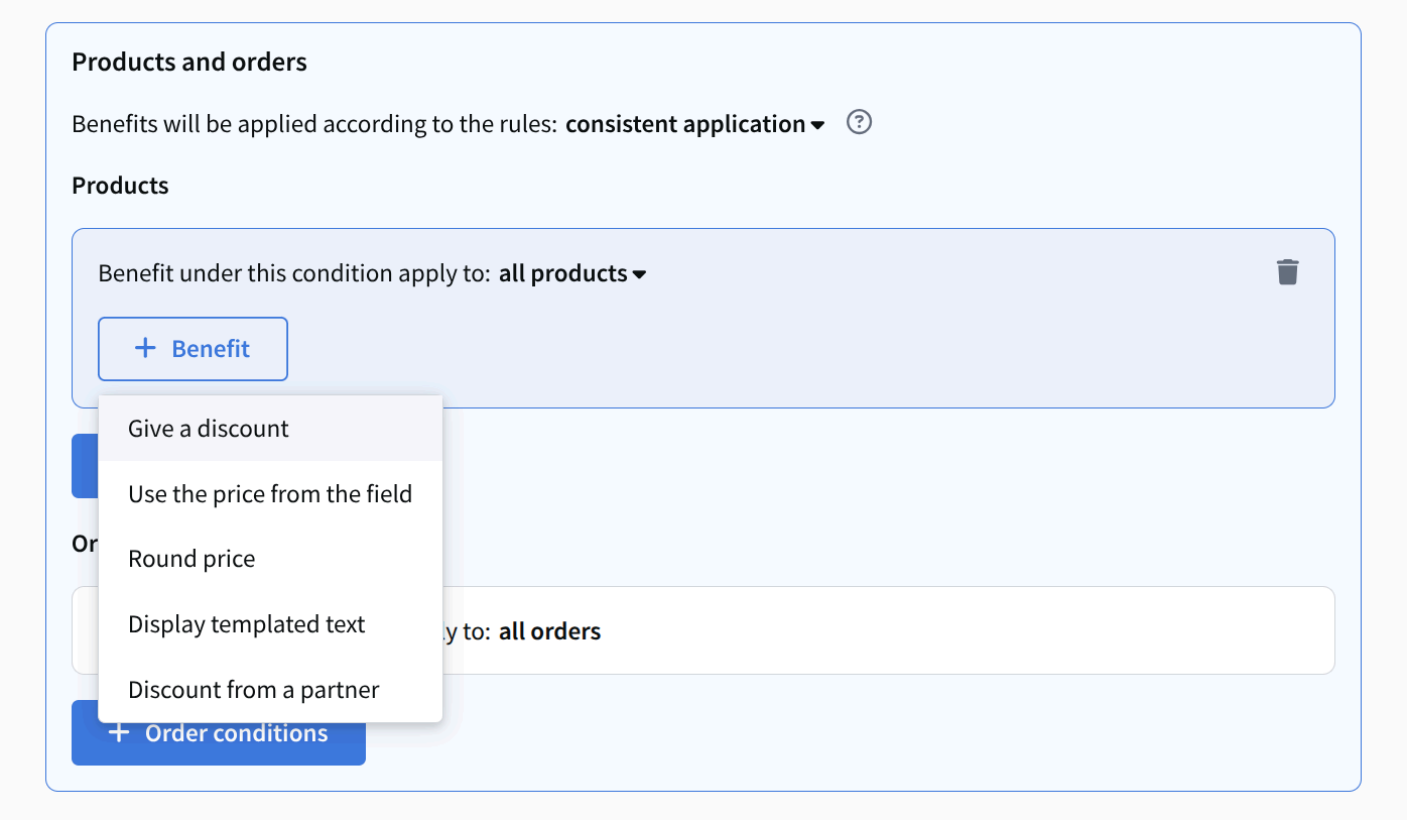
Benefits for discount promotions

For products:

- **Give a discount — a discount in a specified amount or percentage of the product price;**
- **Use the price from the field — uses a price from a custom product field. This price can be multiplied by a specified factor, for example 0.75 for a 25% discount;**
- **Round price — rounds the product price to an integer, half, nearest integer down, tens, or nearest ten;**
- **Display templated text — passes text with the order, supporting variables and emoji. The text is sent with order data and can be passed via API, for example to an email or POS terminal;**
- **Discount from a partner — grants a discount on products from selected partners according to partner product settings.**

For orders:

- **Give a discount — a discount in a specified amount or percentage of the order value. You can limit the number of discounted items per order;**
- **Round order price — rounds the final order total;**
- **Display templated text — passes text with the order;**
- **Issue a promo code — selects a promo code from a specified pool and links it to the participant. You can also configure a message template that the participant receives with the promo code.**



Benefits for points promotions

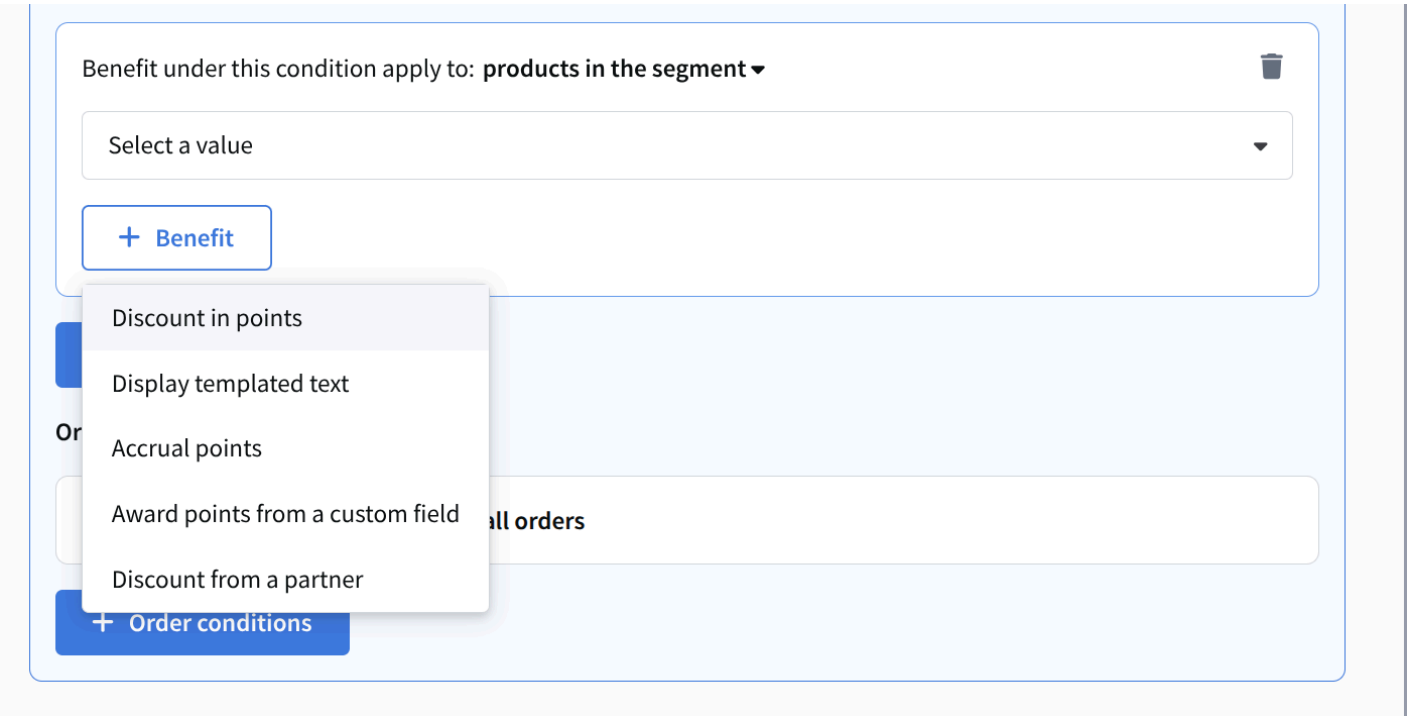
For products:

- **Discount in points — a discount issued in points in a specified amount or percentage of the product price, or a fixed number of points. You can limit the number of discounted items per order;**
- **Display templated text — passes text with the order;**
- **Accrue points — credits specified loyalty points to the profile. You can configure when points are credited, for example at order creation or after delivery;**
- **Accrue points from a custom field — credits the number of points taken from a custom product field. You can also accrue a certain number of points for each currency unit spent, for example, 200 points for every ₽1,500 spent;**

- **Discount from a partner — grants a points-based discount on products from selected partners according to partner product settings.**

For orders:

- **Discount in points — a discount issued in points as a percentage of the order value or a fixed number of points;**
- **Display templated text — passes text with the order;**
- **Accrue points — credits points for the entire order.**

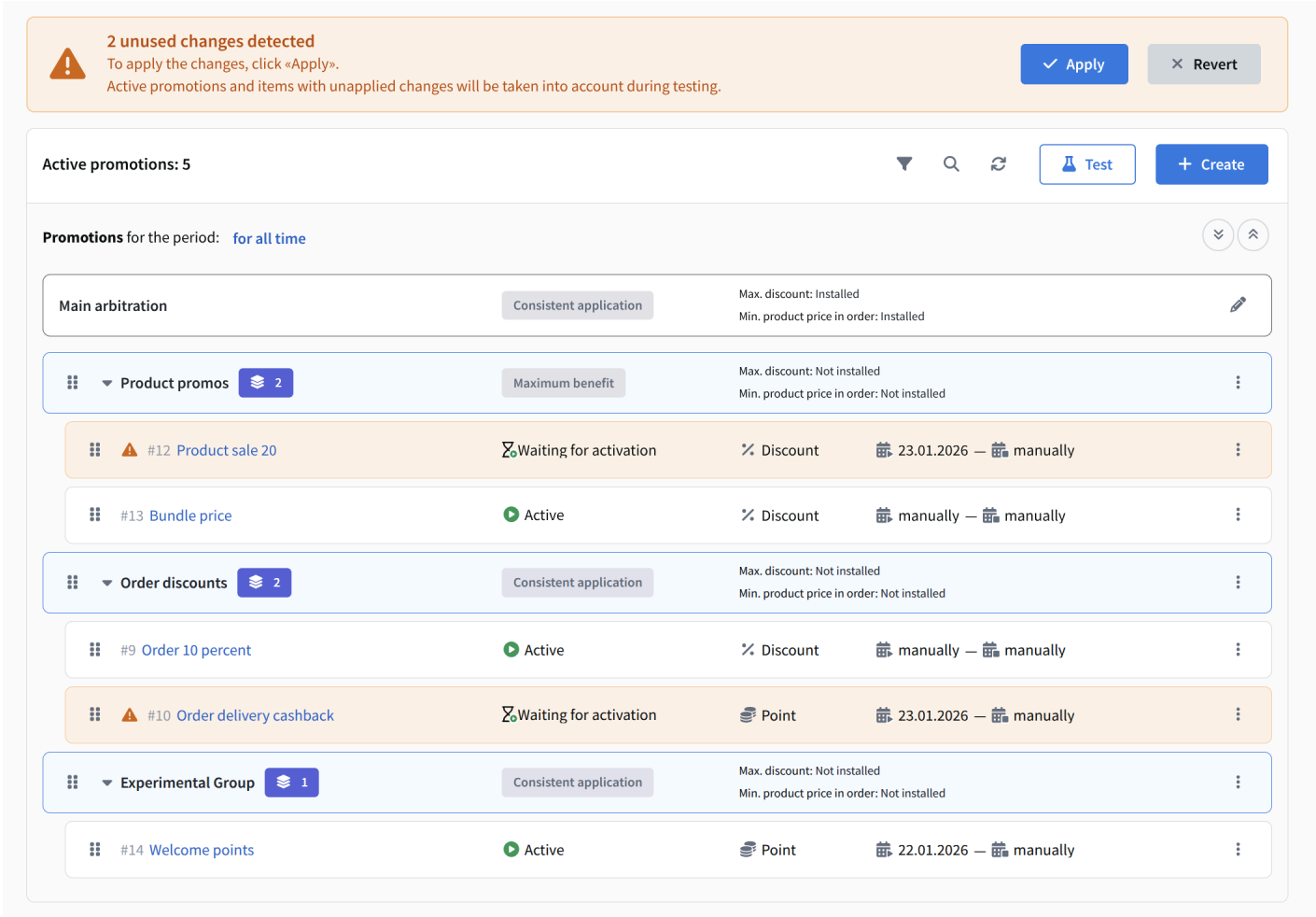


After completing all required sections, the promotion can be saved or activated. An activated discount promotion participates in order calculations according to configured arbitration and conditions. If saved, the promotion appears as a draft and must be activated later.

When working with promotions, the top of the page shows the change status and quick actions.

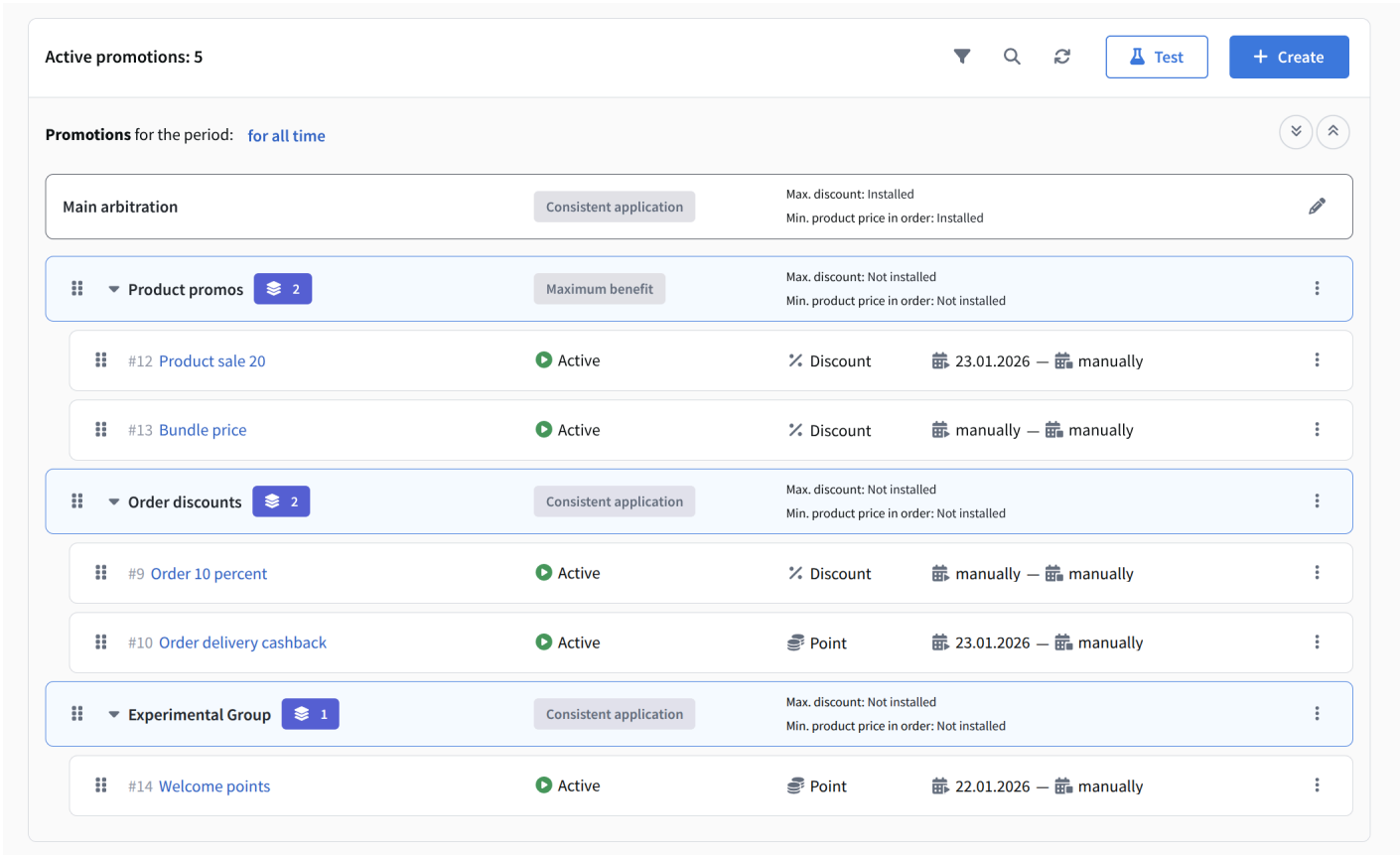
If there are unapplied changes in the tree, the system displays a notification: “Unapplied changes detected”, with two options:

- **Apply — makes changes effective for real calculations.**
- **Revert — discards unapplied changes and restores the last applied version.**



Unapplied changes are considered in test mode only. Real calculations use changes only after clicking Apply.

Below is the list of promotions and groups:



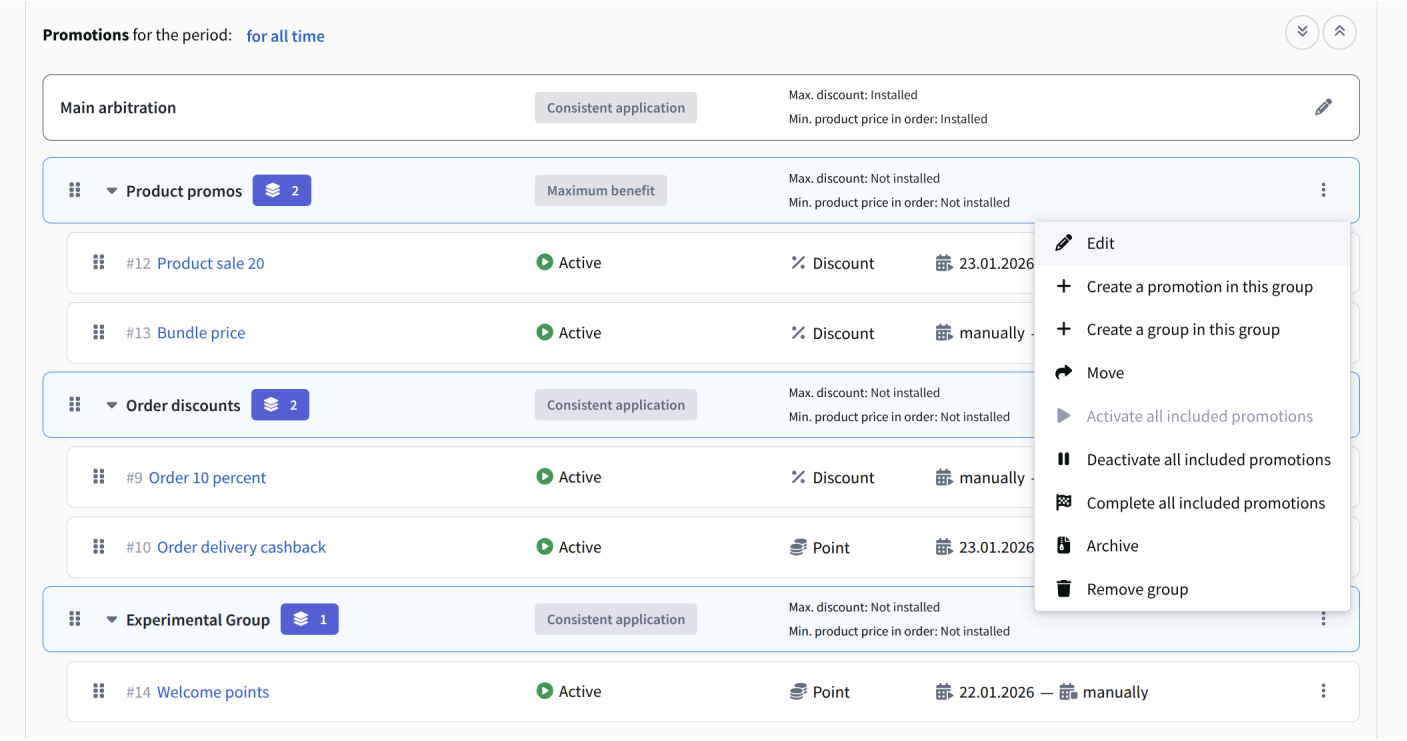
- **the header shows the number of active promotions;**
- **quick actions on the right: filters/search, Test, Create;**
- **the tree shows the Main arbitration and its limits, such as maximum discount and minimum product price;**

- **groups and promotions have action menus, and items can be dragged within the tree and moved between groups.**

For created promotions, a quick actions menu is available, which can be used to:

- **edit the promotion;**
- **deactivate it (if active);**
- **move it to another group;**
- **clone it;**
- **end the promotion (all related processes stop and it cannot be reactivated).**

Group management



Promotion groups are used to logically organize promotions and manage their interaction through separate arbitration.

A group is a node in the promotion tree where its own arbitration type is defined, limits for maximum discount and minimum price are configured, and the application order of included promotions is set.

Groups can be nested (no more than 9 levels) and placed inside other groups or within the loyalty program's main arbitration.

When creating a group, specify:

- **group name;**
- **parent group or main arbitration;**
- **arbitration type and limits.**

After creating a group, promotions can be:

- **moved into it during creation or editing;**
- **dragged between groups directly in the list;**
- **reordered within the group to affect the result (except for Summation);**
- **deactivated in bulk via the quick actions menu, and the group can also be archived.**

As with promotions, changes to group structure and promotion order must be applied. Until applied, changes are considered only in test mode. The final result of applying promotions is always additionally processed by the loyalty program's main arbitration. All actions applied to a group (activation, deactivation, archiving) cascade to all nested promotions and subgroups.

INFO

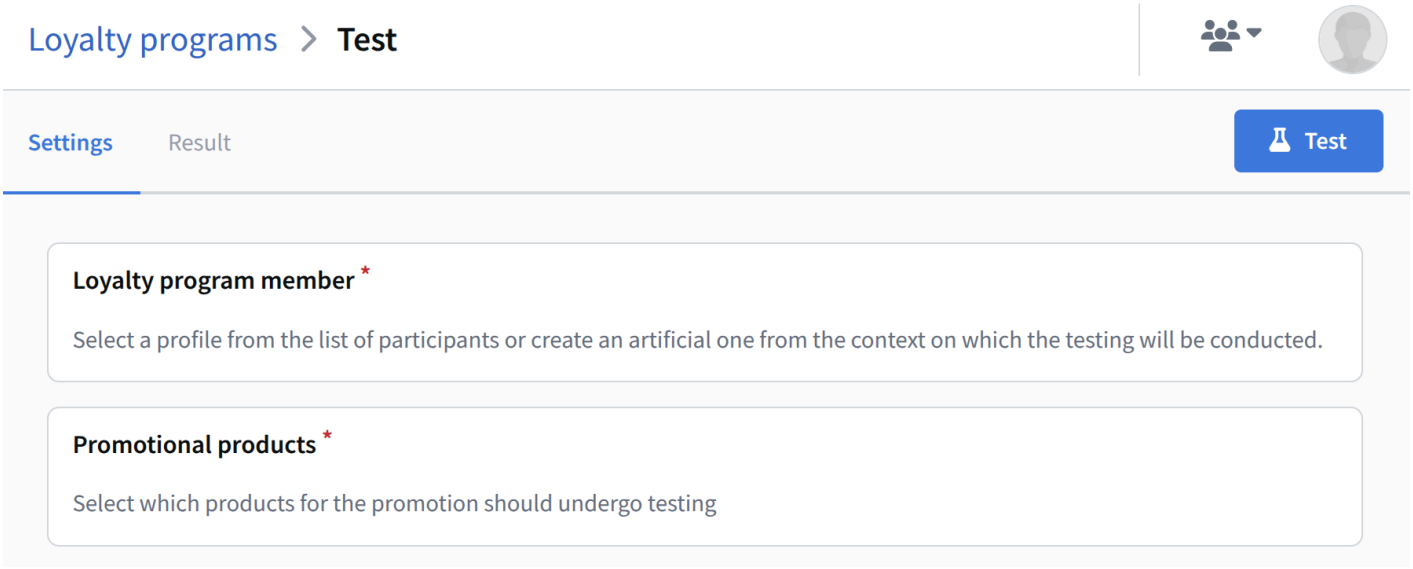
A group is deleted only if one of the following conditions is met:

- **there are no promotions inside the group;**
- **all promotions inside the group have the "Draft" status.**

Promotion testing

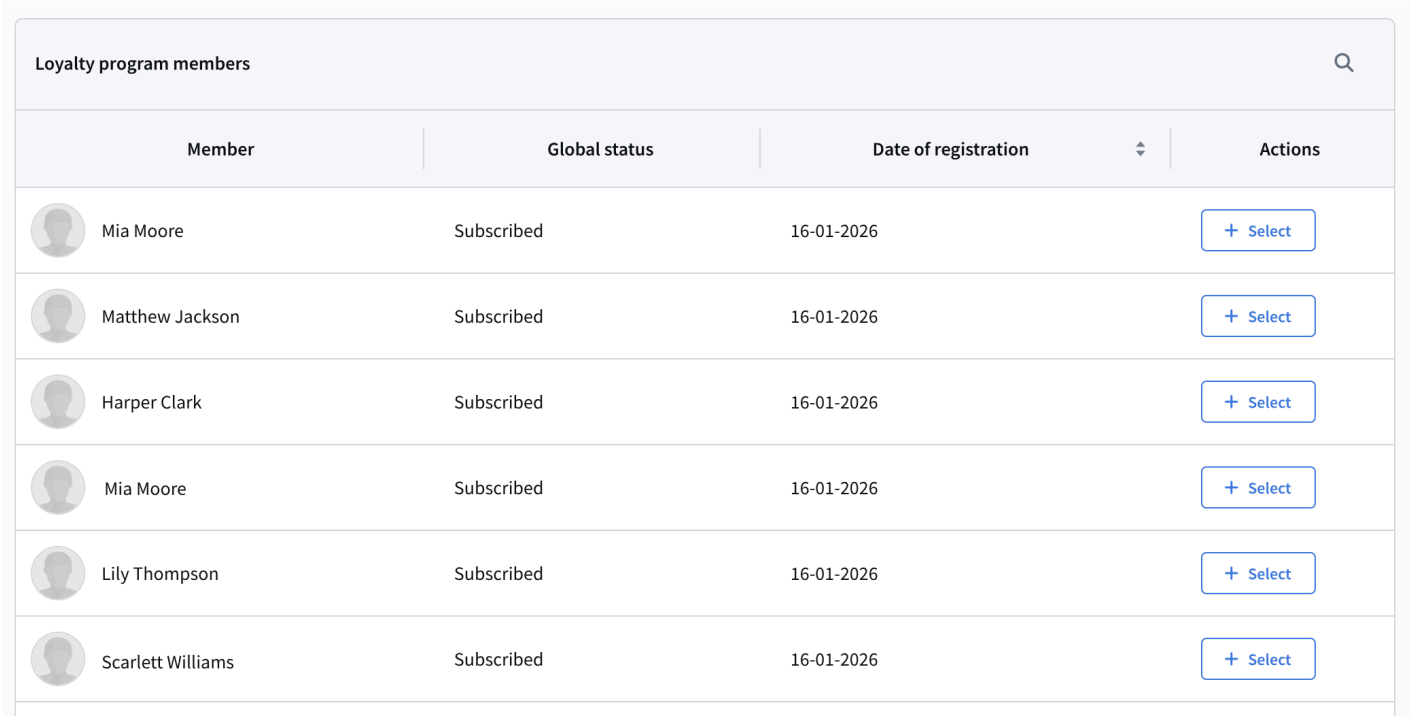
To check how promotions apply to an order, click Test to open test mode, where you can set order and participant parameters and view the result, including unapplied changes.

Select a member and products for testing:



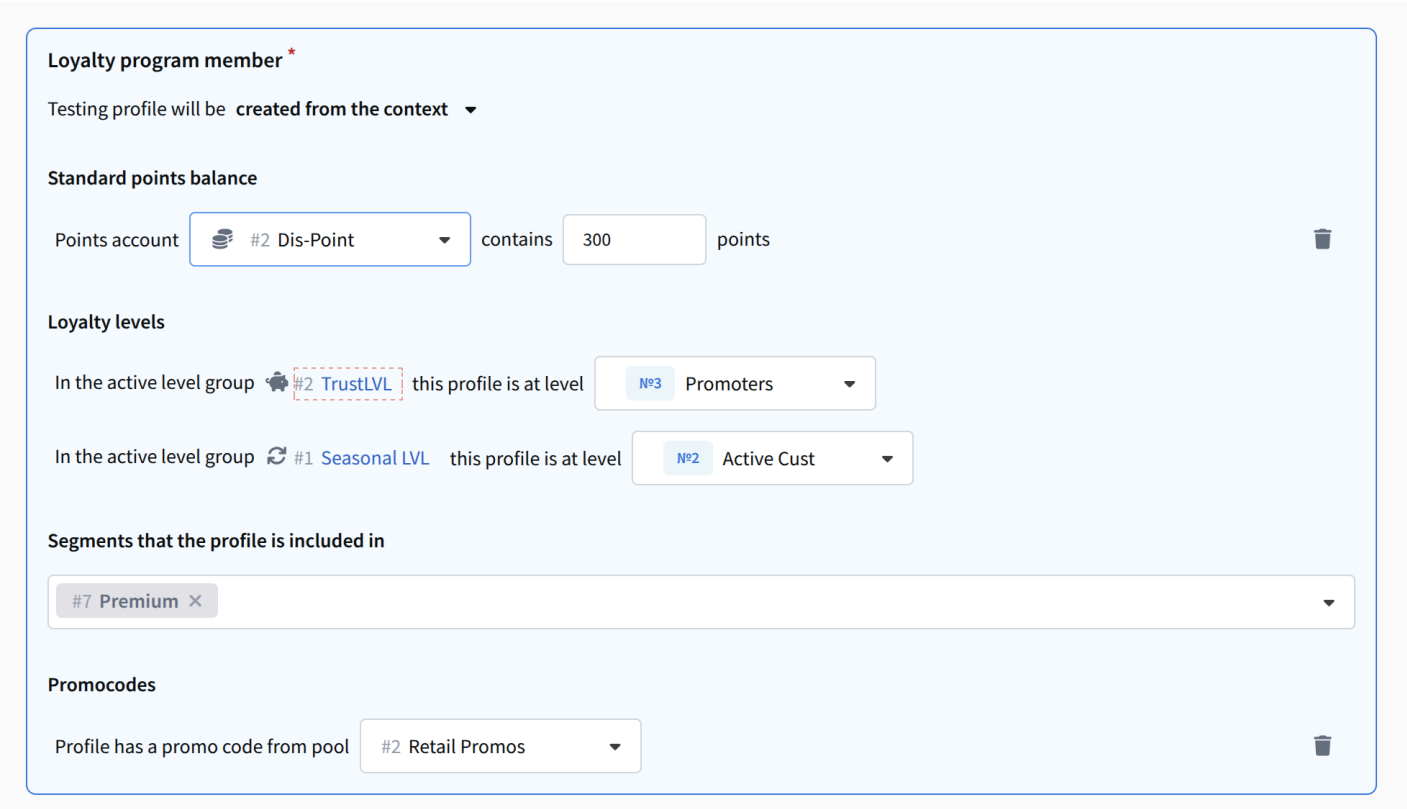
You can select an existing participant profile or create a profile from context.

When selecting an existing profile, click Select, choose a participant, and it will be fixed for testing:

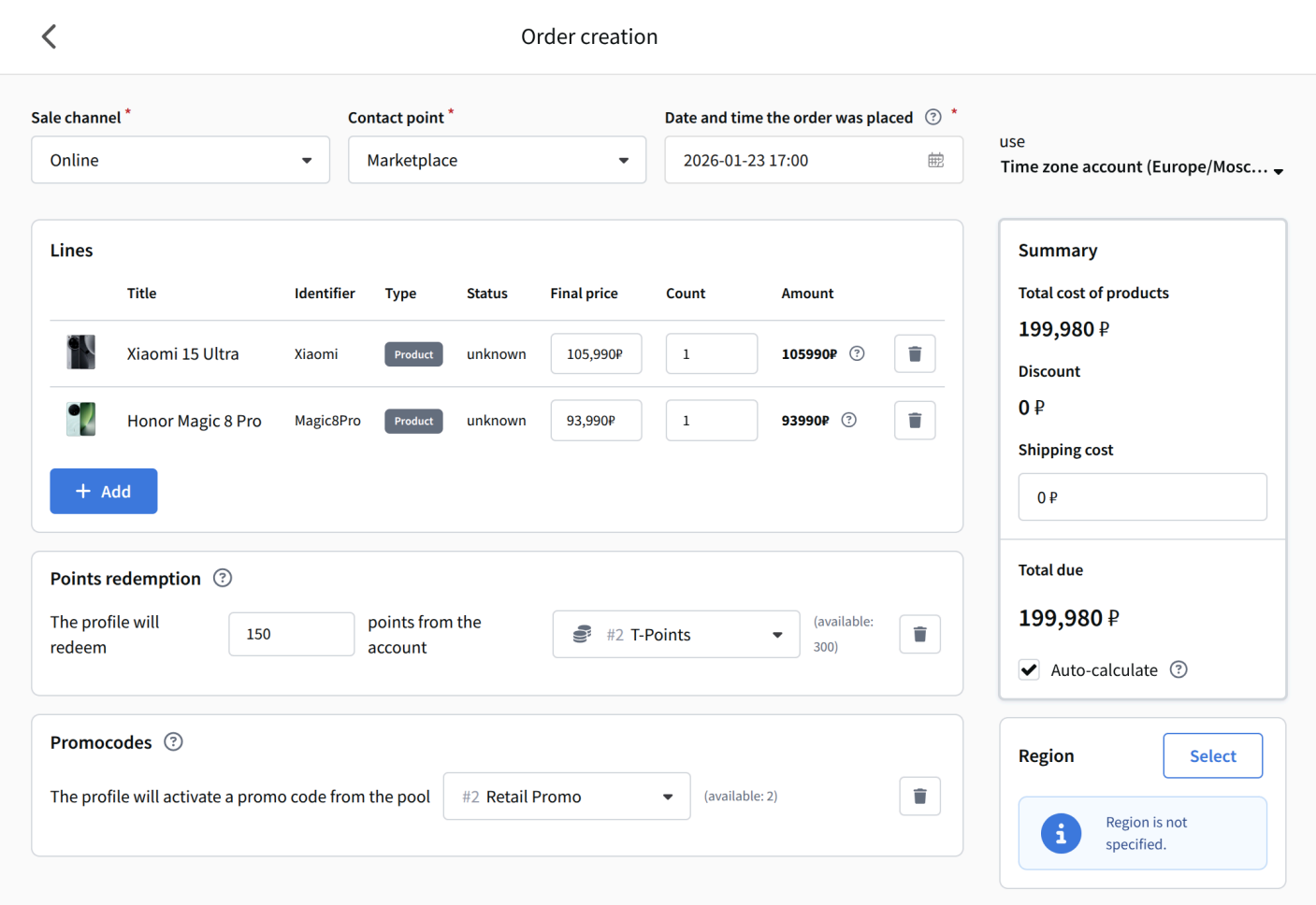


When building a profile from context, a test participant is created with configurable parameters:

- **point balance;**
- **loyalty levels;**
- **segments;**
- **linked promo codes.**



Next, define the products for the promotion within an order:



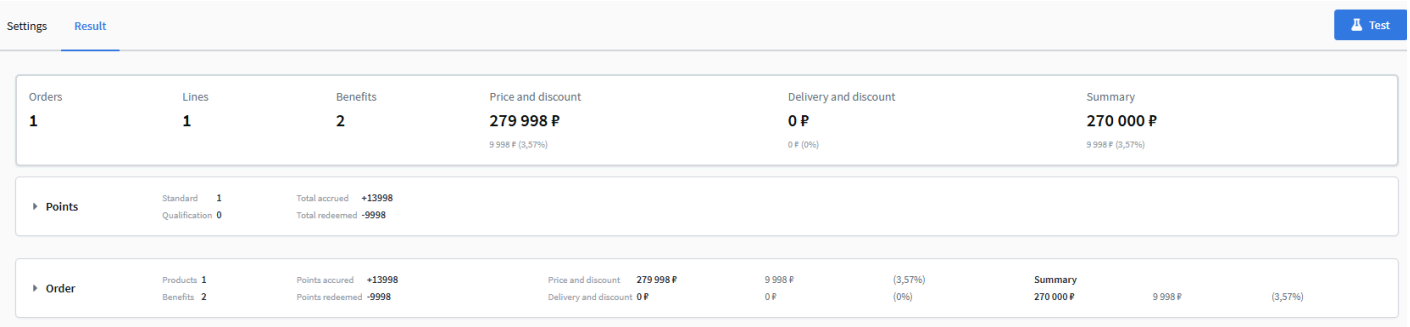
For a test order, specify:

- **Sales channel and contact point — the channel through which the customer placed the order;**
- **Order date and time, including time zone;**
- **Lines — products, quantity, and final price. After selecting a product, you can edit it. Changes apply only to the test order and do not alter Market data. This is useful for testing promotions on new prices;**
- **Point redemption — amount and type of points to deduct within the order;**

- **Promocodes — promo codes activated by the profile;**
- **Shipping cost;**
- **Region.**

If Auto-calculate is enabled, the final price is calculated as the sum of products and delivery.

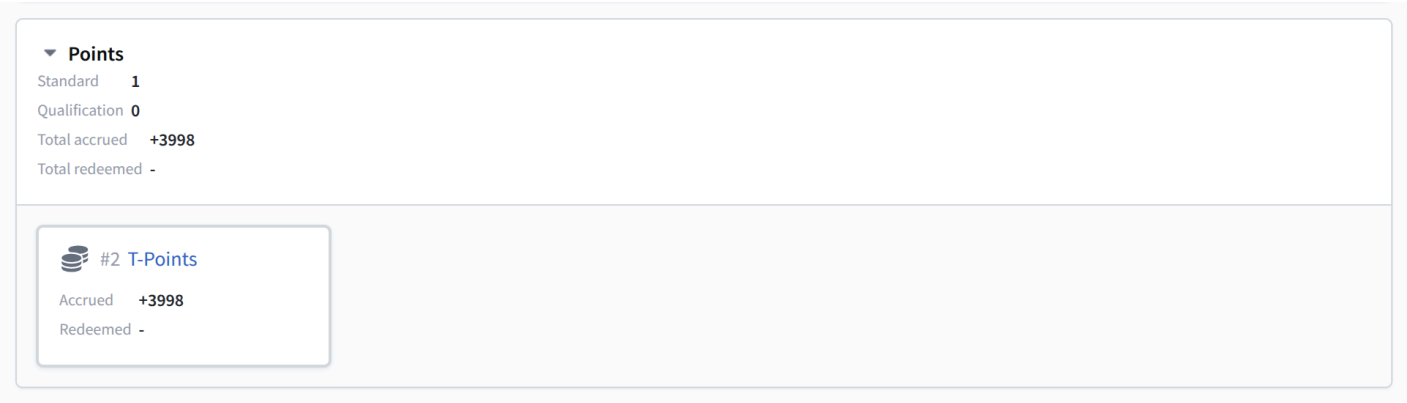
After configuring the participant and products, click Test. The Result tab shows the final benefit application:



The panel shows:

- **number of orders;**
- **number of items;**
- **number of applied benefits;**
- **total order value (before discount) and discount amount (under the value);**
- **delivery cost and discount;**
- **final order total with applied discount (discount amount shown under the total).**

The Points tab shows loyalty balance changes after the order:



The Order tab contains a table of benefit-related operations. Each row includes:

- **order item number;**
- **line (product);**
- **benefit;**
- **benefit result;**
- **applied promotion;**
- **promotion type.**

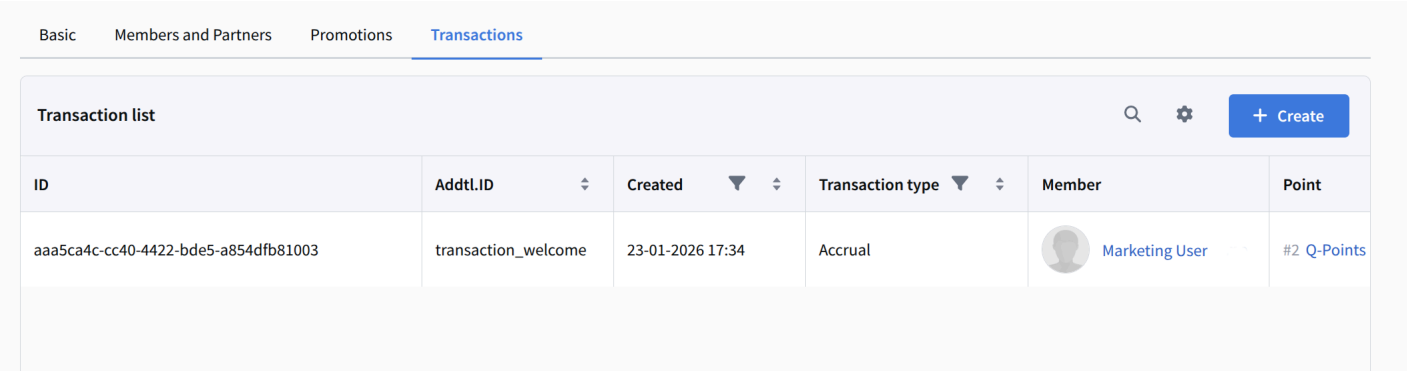
Positions details

Position number	Position	Benefit	Result	Promotion	Promotion type
1	Xiaomi 15 Ultra	Issue a discount	-15 898,5 ₽ (-15%)	#9 Order 10 percent	Discount
1	Xiaomi 15 Ultra	Accrue points	+1999 to #2 T-Points	#14 Welcome points	Points
2	Honor Magic 8 Pro	Issue a discount	-14 098,5 ₽ (-15%)	#9 Order 10 percent	Discount
2	Honor Magic 8 Pro	Accrue points	+1999 to #2 T-Points	#14 Welcome points	Points

The table supports search and configurable visible columns via the actions menu.

Transactions

The Transactions section contains a list of all transactions within the loyalty program. Here you can also create a new transaction to accrue or redeem points.



The transactions table provides detailed information about each balance change. It includes the following columns:

- **ID — transaction ID**
- **External ID — external transaction identifier**
- **Created at — transaction creation date**
- **Transaction type — transaction type**
- **Member — loyalty program member profile**
- **Points — points type used**
- **Balance before — balance before the transaction**
- **Value — number of accrued or redeemed points**
- **Balance after — balance after the transaction**
- **Source type — entity type that initiated the transaction**
- **Source — link to the source entity**

To create a new transaction, click + Create. The transaction setup window will open:

Create a points transaction

#welcoming

Promotion member *

The member whose account needs to be manipulated

John Wayne

Trigger promotion *

Select the TPA within which to conduct the transaction

1 - Promotion Triggers

Points *

Select the points to whose account the transaction will be made

2 - Q-Points

Operation value *

Configure the type and value of manual transaction for selected member

- (•) **Accrue points to account**

  Specify the number of points to be credited to the account, and configure activation and expiration of credited points

  Accrue **only 75 points** according to the rule accrual

- ( ) **Redeem points from account**

  Specify the number of points to be debited from the account

Comment *

Welcoming gift

- **Optionally specify an external identifier;**
- **Select the member — any profile registered in this loyalty program;**
- **Select the trigger promotion under which the transaction will be conducted;**
- **Select the points type;**
- **Specify the operation type — accrual or redemption;**
- **Add a transaction comment;**
- **Optionally specify additional JSON fields.**

Integrating loyalty programs with an external order system

A loyalty program in Altcraft Platform can be used in two ways, depending on whether product and order data is transferred to the platform via the Market module.

Market allows you to transfer product and order data to the platform and use it in customer communications. Data can be transferred automatically — via API or product feed, or manually — by uploading a file.

The availability of data in Market determines which loyalty mechanisms are available to you:

- **Without Market, you cannot automatically calculate discounts, accrue or redeem points during checkout. Point accrual and redemption operations are only available through trigger promotions and the Loyalty API.**
- **With Market, in addition to basic operations via trigger promotions, discount and points promotions become available. These allow you to automatically accrue and redeem points during checkout, apply discounts, and calculate all applied benefits according to the configured arbitration. The partner products functionality also becomes available — for individual products, you can configure special terms, such as increased cashback or an exclusive discount for program participants. Partner products are taken into account when calculating an order via `preorder` / `preorder_commit` just like regular promotions.**

Loyalty with Market integration

This option is suitable for automatically accruing and redeeming points during checkout, as well as applying discounts through points and discount promotions. To make this work, you need to:

- **Upload product data to Market. Data can be uploaded via API, product feed, or from a file.**
- **Create discount and points promotions — these will be automatically applied when calculating an order.**

Order processing with calculation of applied benefits occurs through a two-stage API:

- **At the first stage, the `preorder` method is called — it performs a preliminary calculation when the customer is building a cart. The request includes the order composition and participant data (email, phone, profile_id). The platform returns the final price with discounts, the amount of points accrued and redeemed, and a list of triggered promotions. The order is not created at this stage — it's just a calculation.**
- **At the second stage, after the order is placed or confirmed, the `preorder_commit` method is called. The platform creates the order, credits points to the participant's account, deducts used points, and records all applied promotions.**

INFO: Please note

Automatic point accrual and redemption during checkout works only when using the two-stage API (`preorder` — `preorder_commit`).

If an order is created in any other way (for example, via import, direct addition in the interface, or custom logic), the loyalty mechanisms do not trigger. In such cases, point management must be implemented separately — through scenarios with trigger promotions or direct calls to the accrual and redemption API.

Loyalty without Market integration

In this scenario, points are accrued and redeemed for user actions: filling out forms, installing apps, achievements. Discount and points promotions and related functions (testing, benefit application calculation within arbitration) are unavailable.

Point management is based on using trigger promotions — they set the rules for accrual and redemption but are not tied to orders. The moment a promotion is executed is determined either in automation scenarios (via the "Loyalty points management" node) or through direct calls to the Loyalty API (accrual, redemption, member registration). The `preorder` and `preorder_commit` methods are not used in this scenario — they are only available when working with Market.

Examples of implementing this approach — accruing points for app installation, form submission, and notifying about point expiration — can be found in the article Basic loyalty program use cases.

Basic loyalty program use cases

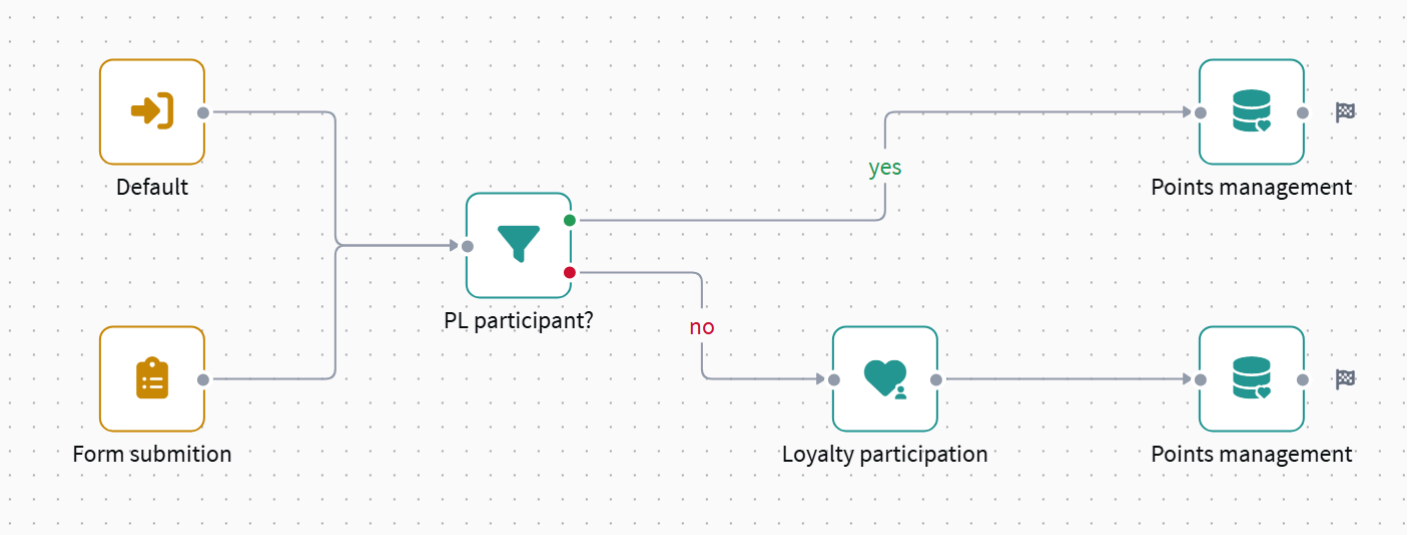
A loyalty program in Altcraft can be used for automatically rewarding users and for personalized communications at different stages of their interaction with the product. Below are basic practical cases that show how to configure point accrual, adding profiles to the loyalty program, and sending notifications based on user actions and their balance status. All examples are designed for typical scenarios and can be adapted to specific business tasks.

CAUTION

Altcraft Platform provides technical tools for storing and processing data, including custom fields that can be used to record consent to participate in the loyalty program. The platform is not responsible for the legal correctness of the method for obtaining, storing, and using such consent. Issues of compliance with legal requirements (including those related to personal data and marketing communications) are the user's responsibility, taking into account applicable regulations and the advice of legal professionals.

Rewarding a profile for filling out a form

This case allows you to automatically add a user to the loyalty program immediately after they fill out a form on a website or in an application. The logic works correctly for both new participants and those already registered in the loyalty program, also rewarding them for filling out the form:



INFO: Please note

If the initial balance in the point settings is zero, after the "Loyalty participation" node, you must also add a "Loyalty points management" node with a trigger promotion to award the bonus for filling out the form.

Loyalty

Before creating a scenario, you need to prepare the basic configuration of the loyalty program. In the Loyalty — Loyalty programs section, create the program itself.

Configure standard points. When creating standard points, you must fill in the required parameters:

- **point name and external identifier;**
- **initial balance for a new loyalty program member;**
- **points expiration period (after a specified number of days, months, or years after accrual);**
- **point-to-currency ratio.**

These parameters determine the balance with which a profile will be added to the program and how the accrued bonuses will be accounted for in the future:

A trigger promotion is used as a universal mechanism for accruing or redeeming points. Despite the name, such a promotion does not have its own trigger — the moment of its execution is determined by the scenario.

Configure the point accrual rules.

- **Select the created point type;**
- **Set a range or a fixed amount:**

INFO

If a range is specified, the exact number of points to be accrued is determined in the scenario node settings. This allows you to use the same trigger promotion for different cases, for example, awarding 500 points for downloading a mobile app and 300 points for filling out a form, without creating separate promotions for each case.

After saving and activating the loyalty program, the trigger promotion becomes available in automation scenarios. In the scenario, it is added as a Loyalty points management node, and it is there that the exact moment of execution and the precise number of points to be accrued or redeemed are determined.

Database and Form

In the Profile database, create an additional Boolean field to record the user's consent to participate in the loyalty program:

Then, create a Form and add this boolean field to it. It is recommended to make the field required so that a profile cannot submit the form without confirming consent. After publishing the form, data from it will be saved to the user's profile, including the value of the consent field:

Scenario

Once the loyalty program and trigger promotion are configured, you can proceed to create a scenario. In the scenario, disable re-entry for profiles to prevent the same profile from entering the scenario again. Select Form submission as the entry trigger:

In the scenario, add a condition check element to verify the profile's participation in the loyalty program:

If the profile is already a member of the program, the "Loyalty participation" node is not used, and the trigger promotion is called directly to accrue, for example, 500 bonus points for filling out the form:

If the profile is not yet a member of the loyalty program, the scenario adds the profile to the program. Since the consent field in the form is required, no additional consent check is needed:

When a profile is added to the program, points are accrued automatically — according to the set initial balance value for the standard points.

INFO: Please note

If the point settings specify a zero initial balance upon registration, you must add a Loyalty points management node (similar to the previous one) to manually accrue the required number of points.

Automatically rewarding a profile for installing a mobile app

This case allows you to automatically add a user to the loyalty program or accrue points immediately after they install a mobile application. This uses a pixel that triggers upon app installation.

INFO: Please note

If the initial balance in the point settings is zero, after the “Loyalty participation” node, you must also add a “Loyalty points management” node with a trigger promotion to award the bonus for downloading the app.

Loyalty

Before creating a scenario, you need to prepare the basic configuration of the loyalty program. In the Loyalty — Loyalty programs section, create the program itself.

Configure standard points. When creating standard points, you must fill in the required parameters:

- **point name and external identifier;**
- **initial balance for a new loyalty program member;**
- **points expiration period (after a specified number of days, months, or years after accrual);**
- **point-to-currency ratio.**

These parameters determine how many points a profile will receive after installing the app:

A trigger promotion is used as a universal mechanism for accruing or redeeming points. The moment of its execution is determined by the scenario.

Configure the point accrual rules:

- **Select the created point type;**
- **Set a range or a fixed amount.**

INFO

If a range is specified, the exact number of points to be accrued is determined in the scenario at the moment the promotion is called. This allows you to use the same trigger promotion for different cases: for example, 300 points for installing iOS and 500 points for installing Android.

After saving and activating the loyalty program, the trigger promotion becomes available in automation scenarios. In the scenario, it is added as a Loyalty points management node, where the moment of point accrual is determined.

Pixel

Create a pixel for the mobile application. The pixel is used for a single purpose — to record the fact of app installation.

- **Event type: App install goal.**
- **It is activated automatically upon installation.**
- **For correct operation, ensure the pixel is integrated into the mobile application via mSDK.**

Scenario

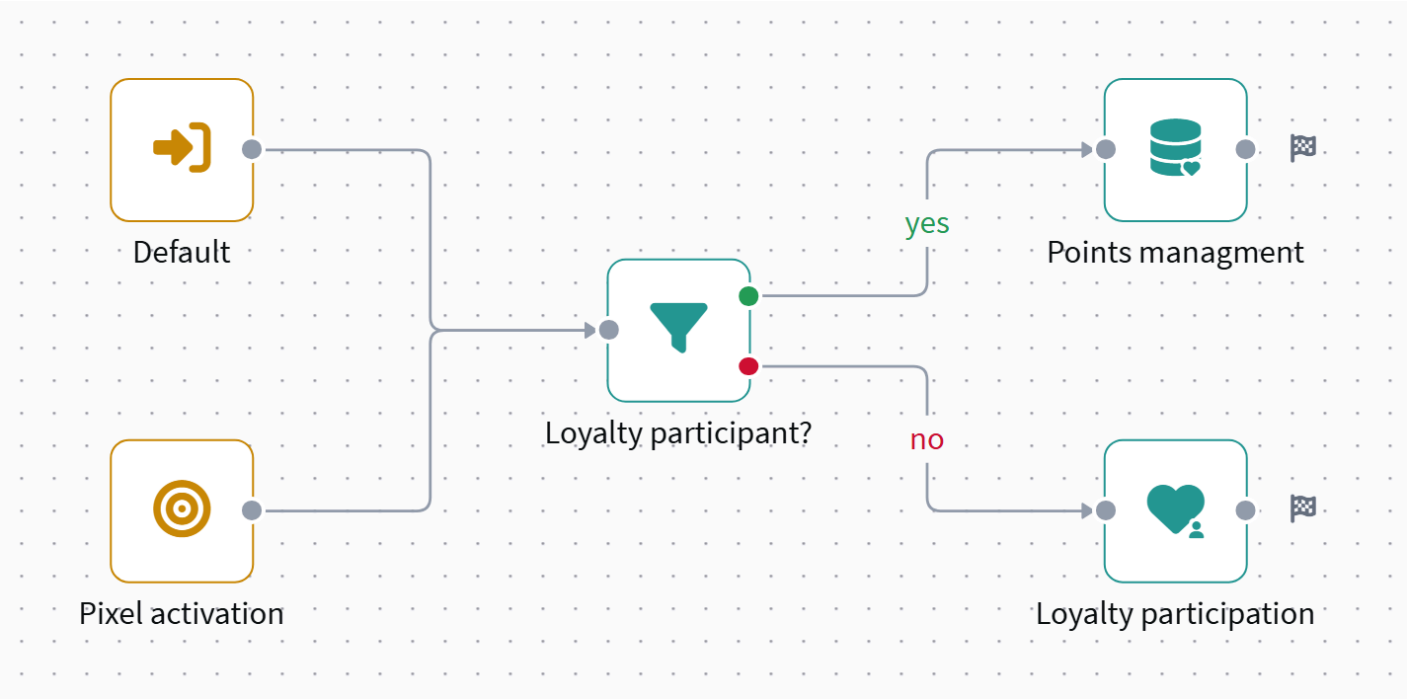
Once the loyalty program and trigger promotion are configured, create a scenario. In the scenario, disable re-entry for profiles to prevent the same profile from entering the scenario again.

Select Pixel event — app install as the entry trigger:

In the scenario, add a condition check element to verify if the profile is already a member of the loyalty program:

If the profile is already a member of the program, the “Loyalty participation” node is not used, and the trigger promotion is called directly to accrue points (for example, 500 points for installing the app):

If the profile is not yet a member of the loyalty program, the scenario adds the profile to the program. After this, starting points are automatically accrued due to the configured initial balance of the standard points.



Notification about imminent loyalty points expiration

In this case, users receive a Notification that their loyalty program points are about to expire. This uses a dynamic segment that selects profiles with the soonest point expiration, and a regular campaign executed on a schedule.

Segment

Create a dynamic segment for profiles whose points are about to expire soon.

In the segment conditions, use the filter Interaction with entities — Loyalty — Points expiration

- **Condition: in the next [X] days**
- **Value X:** `10`
- **Loyalty program: select the desired program**
- **Points: specify the required point type**

Such a segment automatically includes profiles that have points of the specified type expiring within the next 10 days.

SMS Template

Create an SMS message template that will display information about the imminent points expiration.

Use the variable `{loyalty_expiration_points(<Loyalty_Program_ID> "<Point_SID>" x y "desc")}` **to output the data.**

The variable returns information about the nearest expiration of points for the selected currency:

- `<Loyalty_Program_ID>` **— loyalty program identifier;**
- `<Point_SID>` **— point SID;**
- `x` **— points validity period;**
- `y` **— transaction count limit;**
- `"desc"` **— sort by transaction date (descending).**

Example SMS text:

```
{for $i $p = loyalty_expiration_points(2 "bonus1" 10 1 "desc")}{lead._fname}, {$p.Amount} bonus points will expire in 10 days. Spend them now!
{end}
```

Mailing

Create a regular maling, select the created dynamic segment as the audience. Choose the SMS template with the points expiration information. Configure the schedule (e.g., “Every day at selected time” “Send in the profile’s timezone if known”).

After activating the campaign, the system will send notifications daily only to those profiles whose points are genuinely nearing expiration.

INFO

Using a dynamic segment and a regular campaign allows you to notify users about points expiration without scenarios and manual control. Notifications are sent only to the current audience and automatically stop when the conditions change.

Order Segments

Order segments are a tool for fine-tuning the conditions under which benefits are applied in discount and bonus promotions within the Loyalty Programs module.

Order segments work similarly to dynamic segments: you define a set of conditions, and compliance with these conditions is checked in real time when benefits are applied to an order. This makes it possible to build flexible marketing mechanics such as discounts based on the number of items in the cart or special offers for specific regions or sales channels.

TIP: Use case example

For example, a marketer wants to launch a promotion that gives a 10% discount only for orders placed through the online store. To do this, they create an order segment where in the Sales channels section they select the is in list operator, specify the Website sales channel and the Website cart contact point.

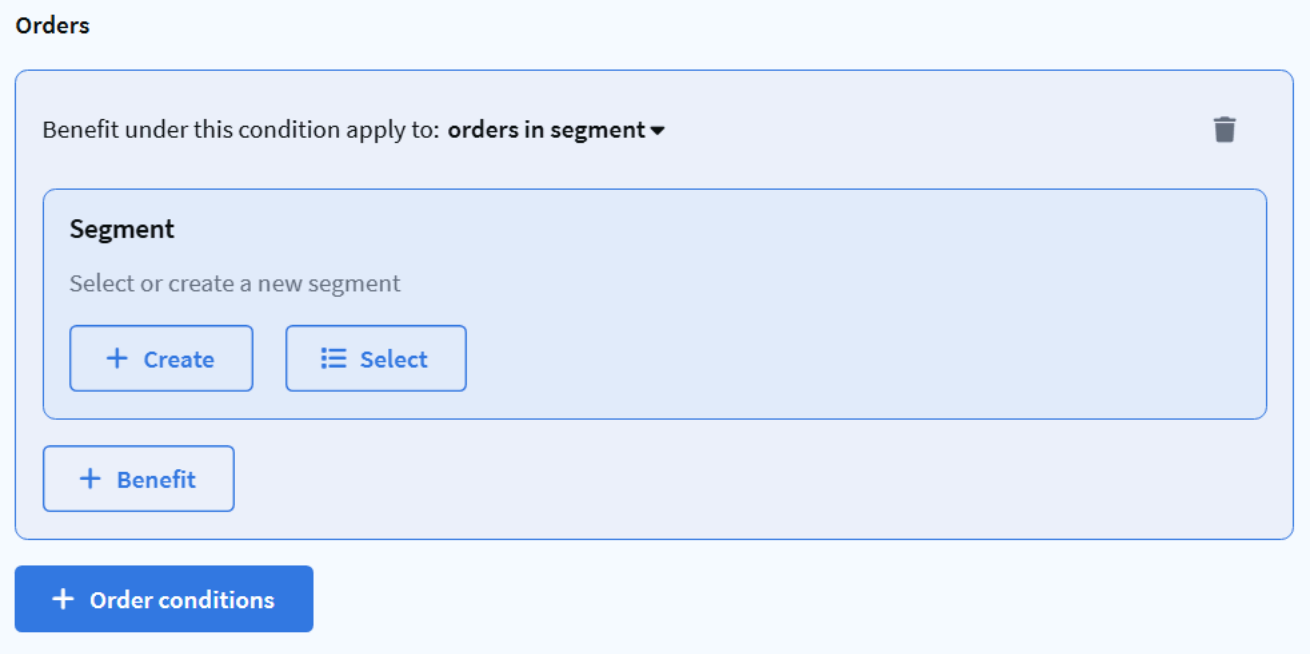
Then, in the discount promotion settings, in the Orders section, they select orders in segment from the dropdown and choose the created segment.

As a result, the 10% discount will be applied only to orders placed through the website cart. Orders from other channels (for example, retail stores or phone orders) will not receive this discount.

Configuring an order segment

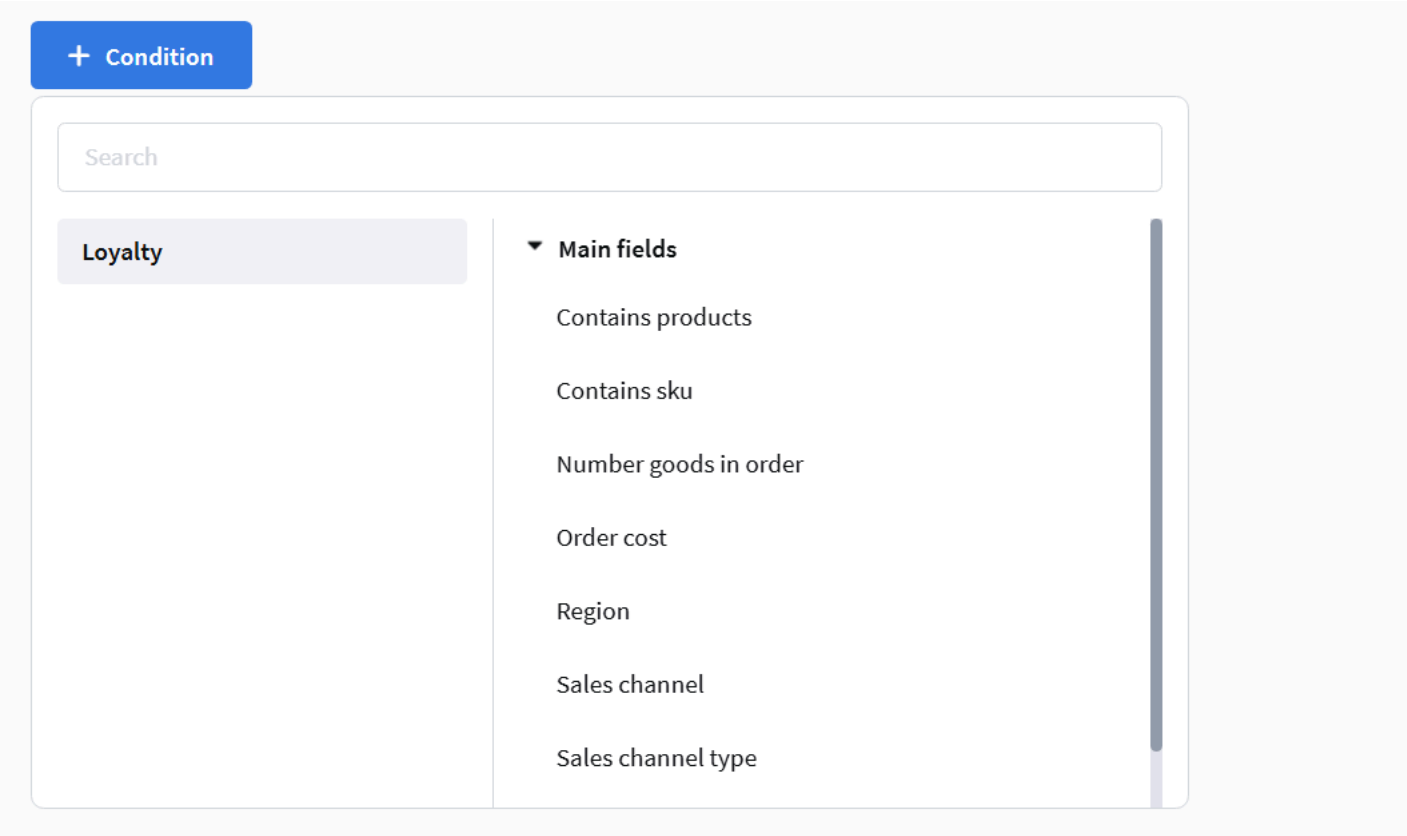
To use an order segment in a promotion, go to the Orders section during promotion setup and select orders in segment from the dropdown. Then click Select segment.

In the window that opens, you can select an existing segment or create a new one by clicking + Create.



Creating a new segment

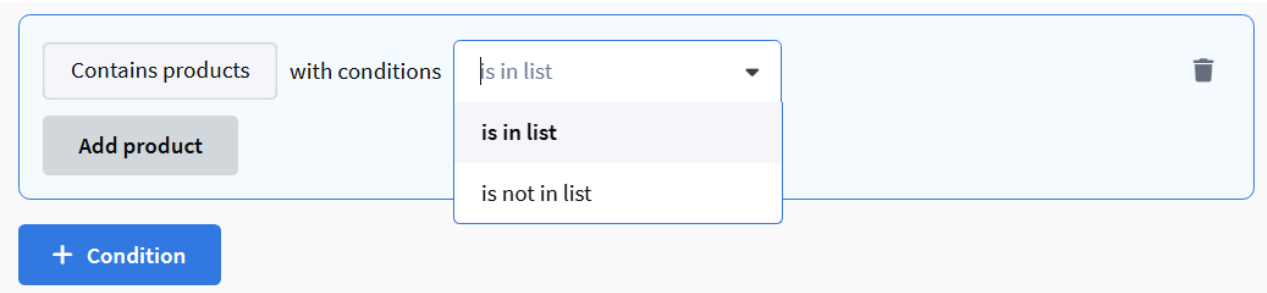
When creating a segment, specify its name and, if needed, a description. Then configure the conditions that the order must meet:



The following conditions are available:

Contains products

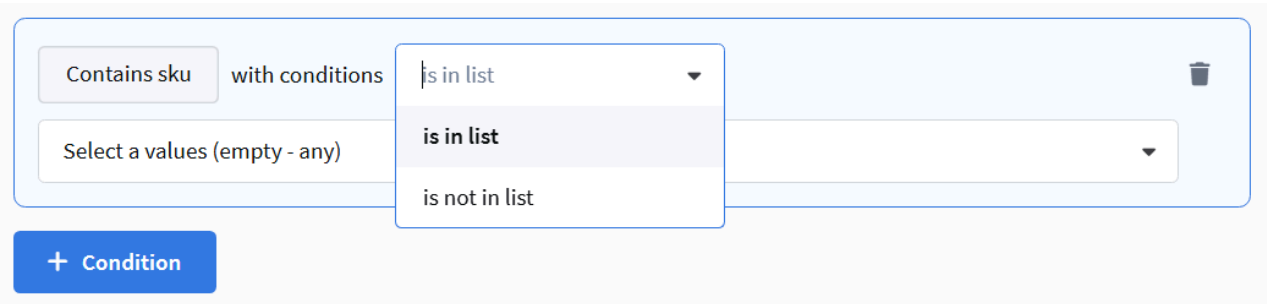
Checks whether the order contains specific products from a list.



- **is in list — benefits are applied if the order contains products that are included in the selected list;**
- **is not in list — benefits are applied if the products in the order are not included in the selected list.**

Contains SKU

Checks whether the order contains SKUs from a list.



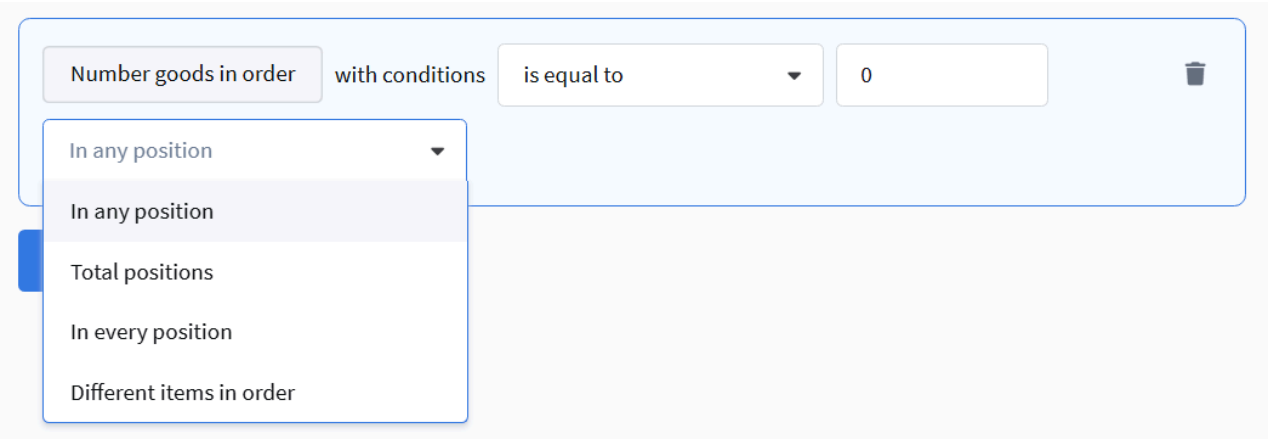
- **is in list — benefits are applied if the order contains SKUs included in the selected list;**
- **is not in list — benefits are applied if the order does not contain SKUs included in the selected list.**

Number of goods in order

Checks the number of goods in the order. Available calculation methods:

- **In any position — checks if there is at least one line item with the specified quantity.**
- **Total positions — checks the total quantity of goods across all line items.**

- **In every position — checks that each line item has the specified quantity.**
- **Different items in order — checks the number of unique products in the order.**

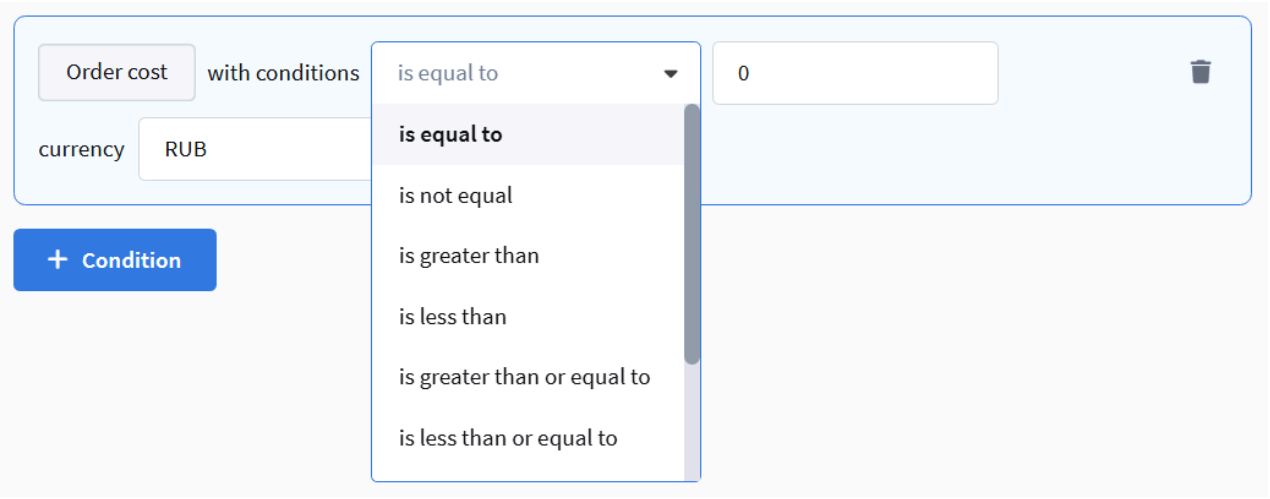


For the selected calculation method, specify an operator and a value:

- **is equal to — the quantity matches the specified number.**
- **is not equal — the quantity does not match the specified number.**
- **is greater than — the quantity exceeds the specified number.**
- **is less than — the quantity is less than the specified number.**
- **is greater than or equal to — the quantity exceeds or equals the specified number.**
- **is less than or equal to — the quantity is less than or equals the specified number.**
- **in the range — the quantity is within the specified range (inclusive).**

Order cost

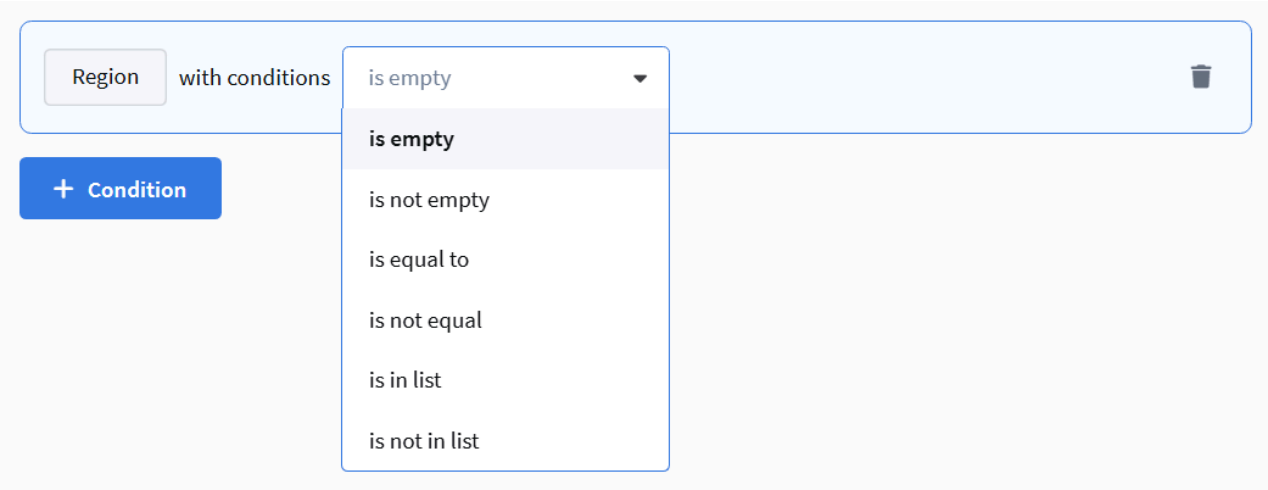
Checks the order cost. Specify an operator, amount, and currency:



- **is equal to — the order cost exactly matches the specified amount.**
- **is not equal — the order cost does not match the specified amount.**
- **is greater than — the order cost exceeds the specified amount.**
- **is less than — the order cost is less than the specified amount.**
- **is greater than or equal to — the order cost exceeds or equals the specified amount.**
- **is less than or equal to — the order cost is less than or equals the specified amount.**
- **is in list — the order cost matches one of the values in the list.**
- **is not in list — the order cost does not match any value in the list.**
- **in the range — the order cost is within the specified range (inclusive).**

Order region

Checks the order region. Available operators:

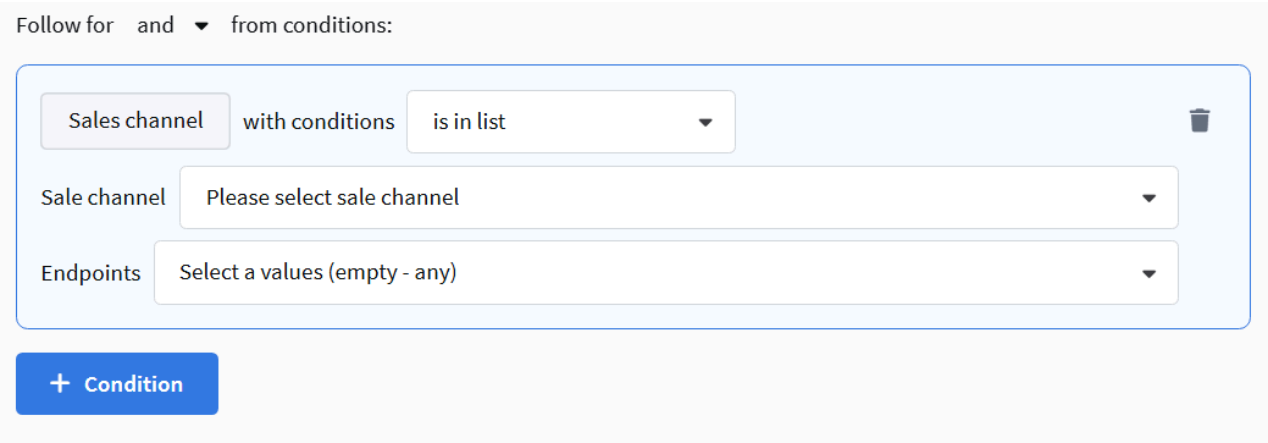


- **is empty — no region is specified for the order.**
- **is not empty — any region is specified.**
- **is equal to — the region matches the selected value.**
- **is not equal — the region does not match the selected value.**
- **is in list — the region is included in the selected list.**
- **is not in list — the region is not included in the selected list.**

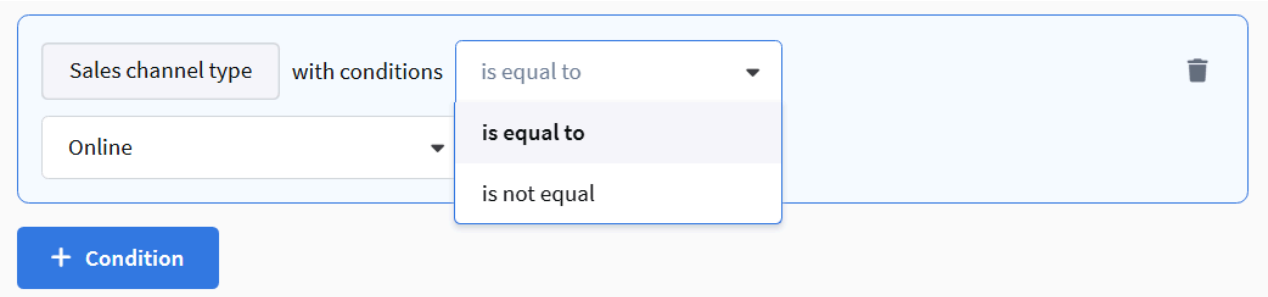
Sales channels

Checks through which sales channel and contact point the order was placed.

- **is in list — benefits are applied if the order was placed through one of the selected sales channels and contact points.**
- **is not in list — benefits are applied if the order was not placed through the selected sales channels and contact points.**



Sales channel type

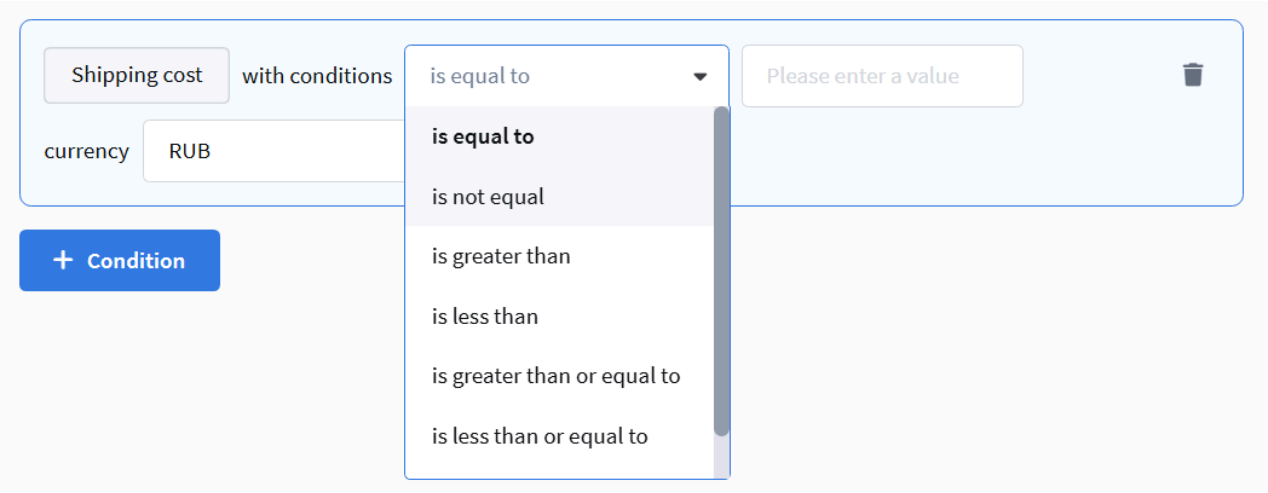


Checks the sales channel type (Online / Retail). Available operators:

- **is equal to — the sales channel type matches the selected value.**
- **is not equal — the sales channel type does not match the selected value.**

Shipping cost

Checks the order shipping cost. Specify an operator, amount, and currency:



- **is equal to — the shipping cost exactly matches the specified amount.**
- **is not equal — the shipping cost does not match the specified amount.**
- **is greater than — the shipping cost exceeds the specified amount.**
- **is less than — the shipping cost is less than the specified amount.**
- **is greater than or equal to — the shipping cost exceeds or equals the specified amount.**
- **is less than or equal to — the shipping cost is less than or equals the specified amount.**
- **in the range — the shipping cost is within the specified range (inclusive).**

Promotion codes

Description

You can create and manage customizable promotion codes to lead efficient loyalty programs using Altcraft Loyalty tool. These promo codes can be used in variety of cases where you want to give your customers personal offers or set limited access to your services.

- **Personalpromo code can be attached to certain customer profile for personal offers.**
- **Universal promo code can be attached to several profiles, It can be used to organize massive marketing mailings.**

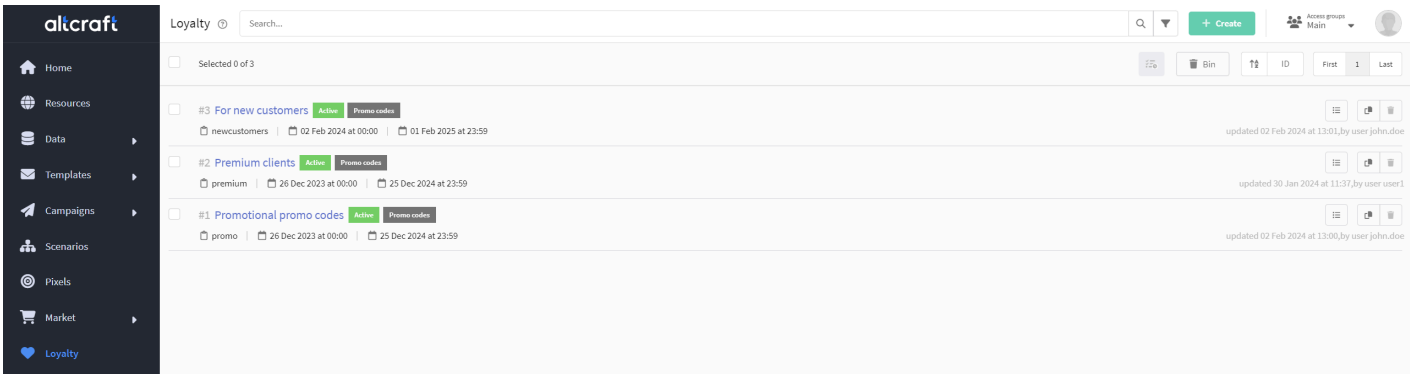
Promo codes can be attached to customers manually or automatically within mailings and workflows. Promo codes are generated in Loyalty program interface, or they can be imported from file.

> TIP
>
> **You can use API for advanced promo codes configuration. For example, you can track customers who used their offer. Loyalty program API documentation is available there.**

Managing promo codes

Go to the Loyalty section in the main menu to open the list of existing loyalty programs. Using context menu, you can clone, remove, activate or deactivate, and view the list of promo codes created in a loyalty program. To edit a program, click on its title.

To create a new program, click + Create on the top panel.



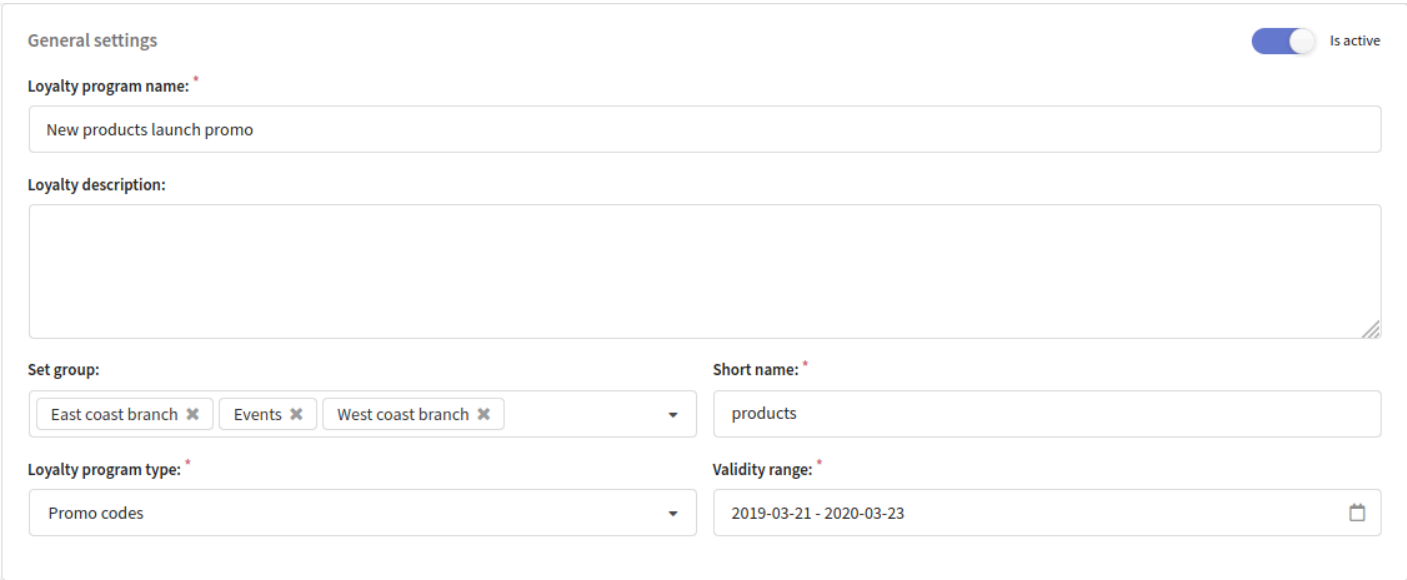
Promo code data

Each promo code contains the following information:

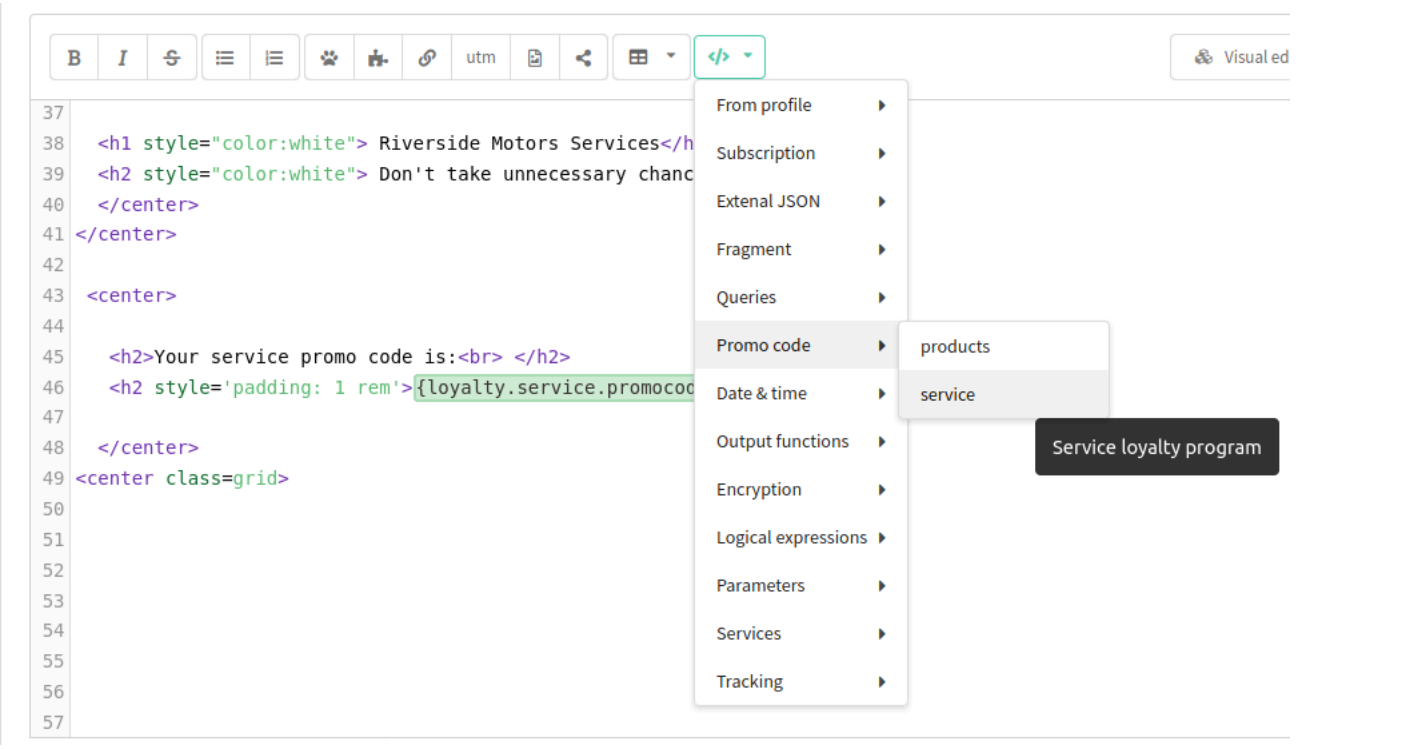
- **Promo code creation date, loyalty program ID**
- **Promo code attachment sign and date**
- **Promo code life time**
- **Promo code activation sign and date**
- **Additional data fields**

General settings

In general settings, enter loyalty program name, its short name, and set access group if necessary.



Loyalty program can be used in templates by its short name in variables menu or macro `{loyalty.name.promocode}`.



Loyalty program validity range defines time when promotion codes can be sent to your customers. If a code is outside validity range, messages containing it won't be sent to recipients. This parameter does not affect promo codes that have been already sent to the customers. Select dates range and press Apply button.

> **TIP**
>
> **At the moment, only one loyalty program type is available - promo codes.**

You can import your own custom promotion codes by pressing Import from file button or via a special API call. The other option is to generate promo codes right away in Altcraft platform.

Once imported or generated the are processed in a same manner. In case you generate promo codes - several formatting options are available.

Generating promo codes

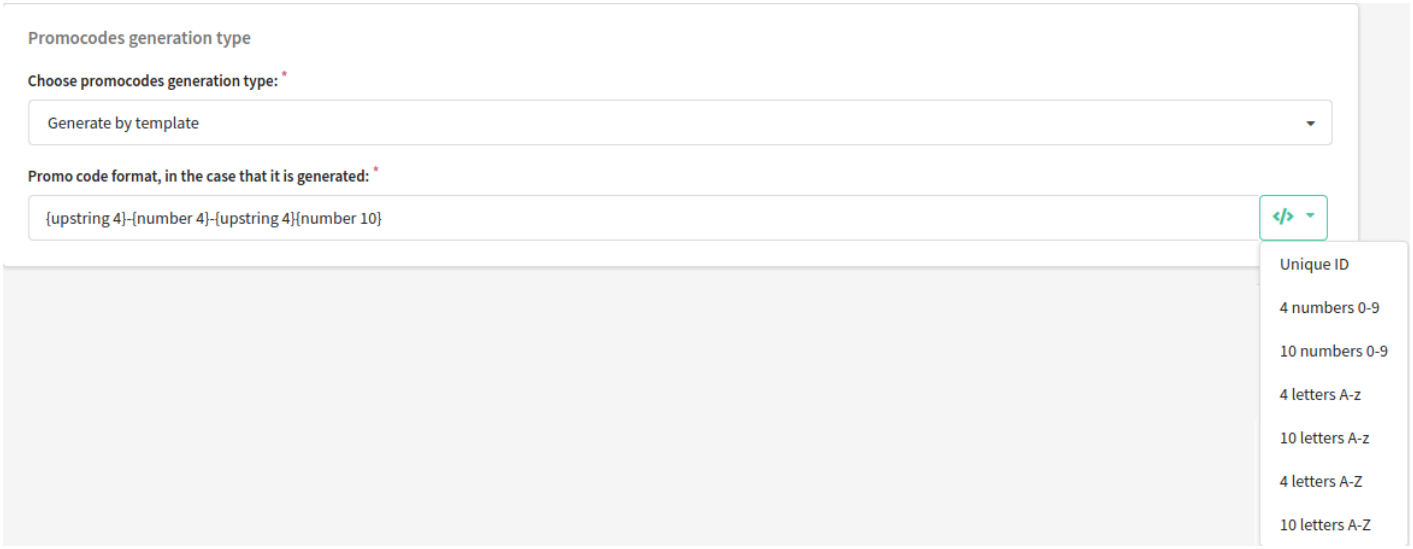
> **CAUTION**
>
> **If you plan to import promo codes from file, you can skip this paragraph.**

You can configure promo codes generation in Promocodes generation type section.

Generating by template

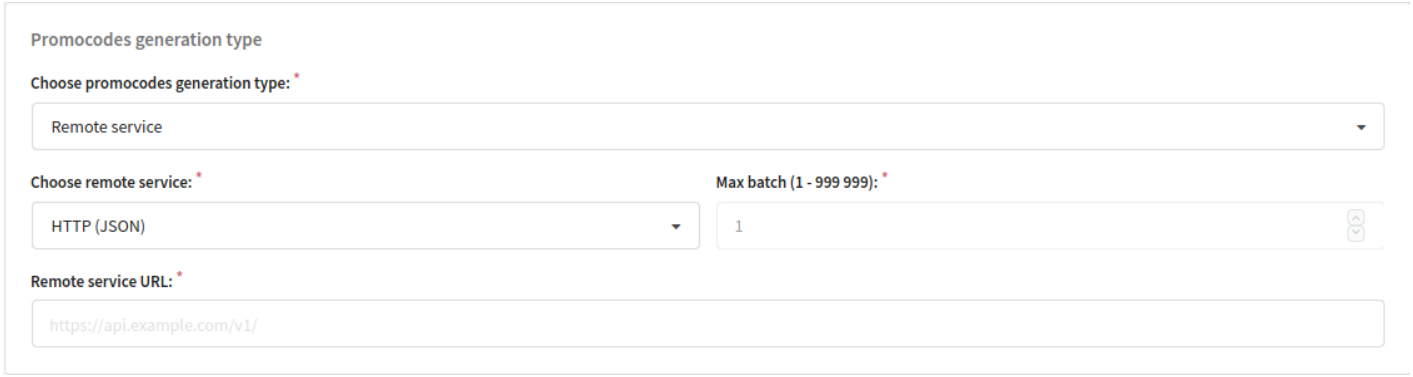
Altcraft Platform has its own tool to generate promo codes, which you can use by selecting Generate by template option. This tool can be useful if you do not have your own system for configuring promo codes, or promo codes list.

Select symbols combination you want to use from the `</code>` drop-down menu:



Generating by remote service

An alternate way to generate promo codes is a request to external service, which will create promo codes. To use it, select Remote service option in the drop-down menu:



Enter service URL and max promo codes batch in one query. POST query with JSON data of the following format is used:

```
{
    account_id: Number (Account identifier),
    loyalty_id: Number (Loyalty program identifier),
    loyalty_name: Line (Loyalty program name),
    loyalty_shortname: Line (Loyalty program short name),
    campaign_id: Number (Mailing identifier, if it requests promo codes),
    campaign_name: Line (Mailing name, if it requests promo codes),
    workflow_id: Number (Workflow identifier, if it requests promo codes),
    workflow_name: Line (Workflow name, if it requests promo codes),
    count: Number (Amount of promo codes to be generated),
    is_test: Boolean (Test condition),
    fields: {
        <Field name>: <Value>
    }
}
```

If promo codes are requested by mailing or workflow, their identifiers will be given in request. If `is_test` flag is true, promo code must be used only for preview. All loyalty program fields will be given with the request.

Promo codes list in JSON format will be returned:

```
{
    codes: Lines array (promo codes)
}
```

If it is necessary to return promo codes with custom fields values, promo codes must be returned as objects array:

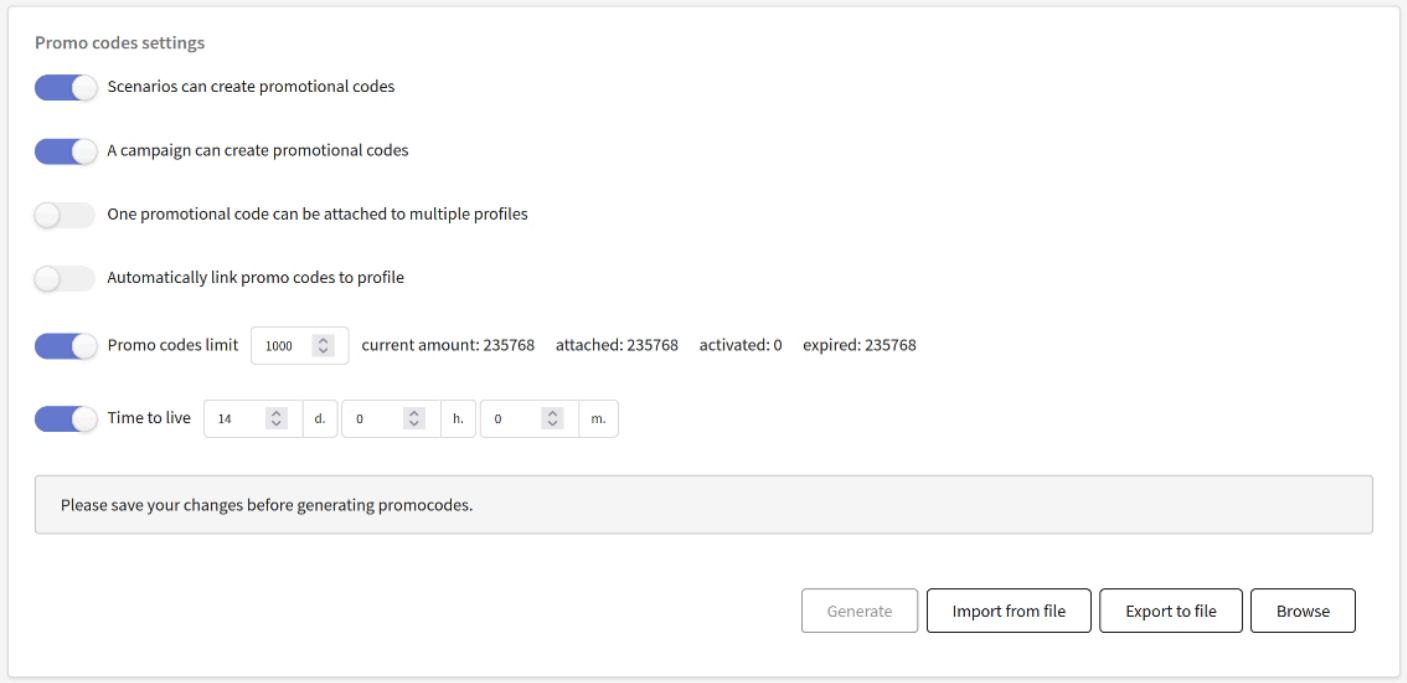
```
{
    codes: Objects array {
        code: Line (promo code),
        fields: {
            <Field name>: <Value>
        }
    }
}
```

TIP

You can use Generate button in Promo code settings block to check generating service. Here, you can also generate, import, export, and view promo codes.

Promo codes settings

You can configure the loyalty program to match your business logic in the Promo Code settings block. In addition to settings, this block provides commands for generating, importing, exporting, and viewing promo codes.



The following settings are available:

- **A workflow can create promotional codes — Workflow can create promo codes if there is not enough codes generated. Codes will be generated according to formatting scheme.**
- **A mailing can create promotional codes — Mailing can create promo codes if there is not enough codes generated. Codes will be generated according to formatting scheme.**

CAUTION

When testing a mailing, the loyalty program generates a new promo code instead of using an existing one.

- **One promotional code can be attached to multiple profiles — With this option, you can use one promo code for a group of customers. The mailing or scenario in this case will assign the first active promotional code of the loyalty program.**
- **Automatically link promo codes to profile — When generating or importing from a file, promotional codes will be assigned to certain profiles.**
- **Promo codes limit — Limit can be set for total promo code count**
- **Time to live — Time can be set for promo codes to be alive. When a specified time after generation runs away, promo codes will automatically get expired status.**

CAUTION

Promo code lifetime can be more than loyalty program validity range. If loyalty program is finished, promo codes can not be attached and sent to customers, but previously sent codes can be used by customers until their lifetime is not finished.

Automatically link promo codes to profile

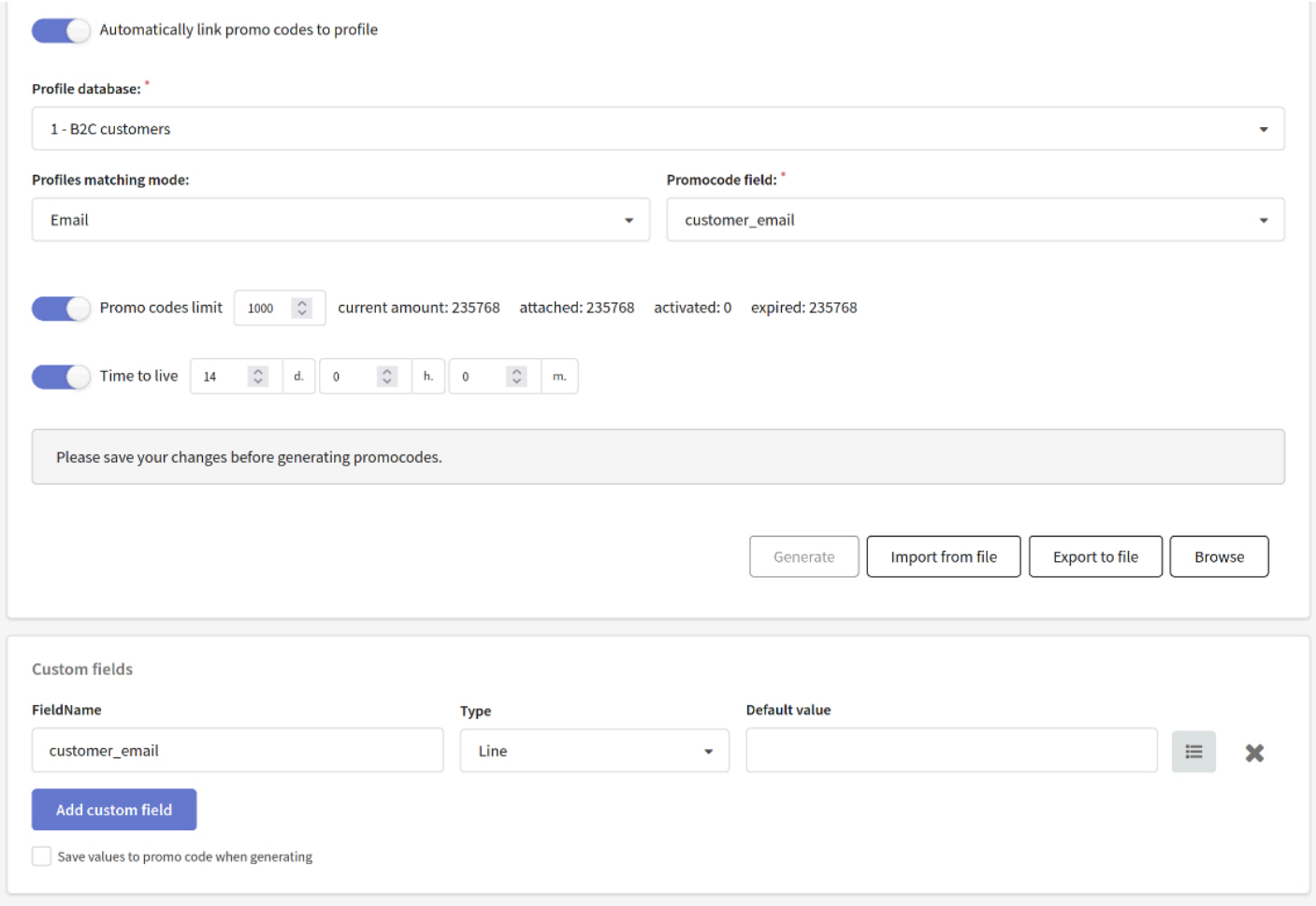
Select the database with customers to whom you want to assign promotional codes. Next, specify the profiles matching mode and the promocode field. The matching mode determines what data the platform will use to search for profiles. The following modes are available:

- **Email**
- **Phone**
- **By custom field**

If the value in the promo code field matches the value of the field specified as the matching mode, the platform will bind the promo code to the found profile.

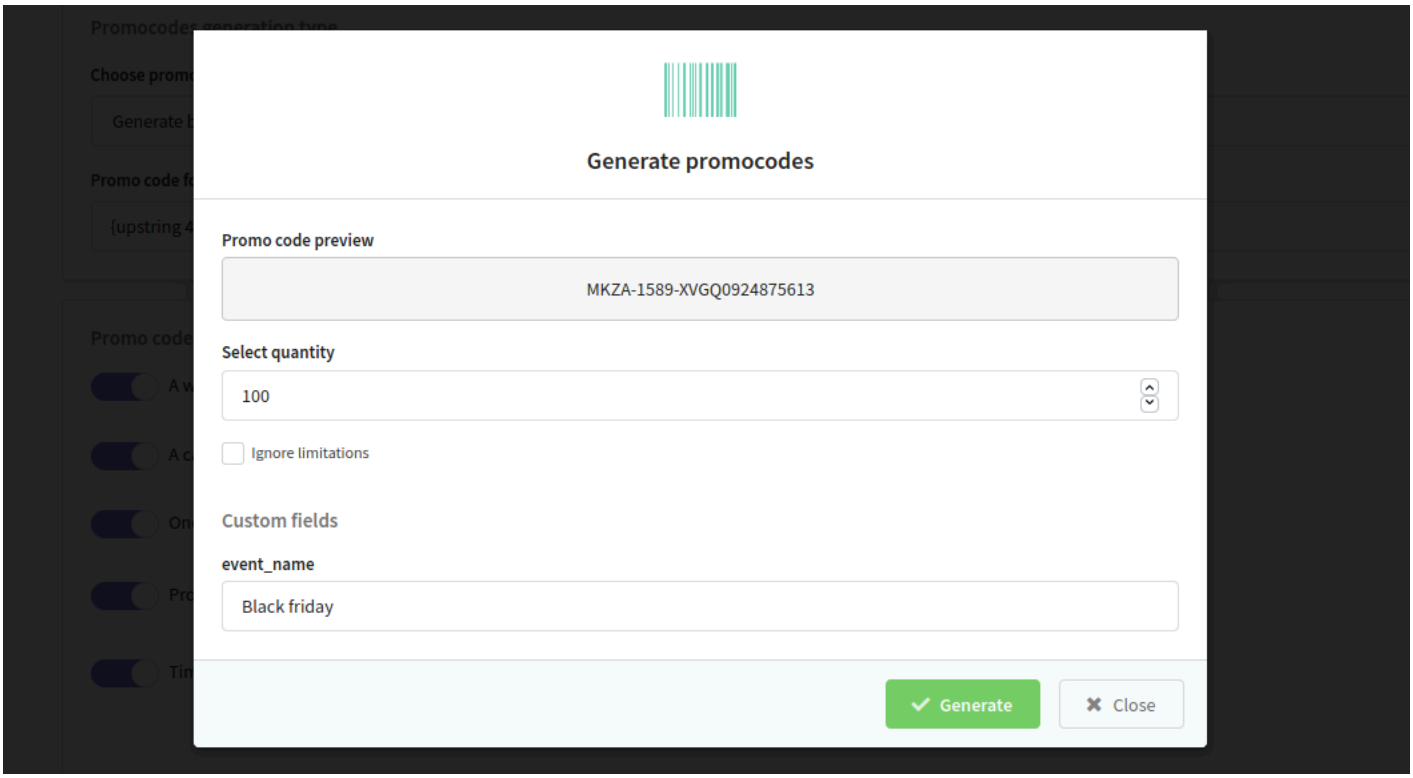
TIP

The promotional code field is created in the Custom fields section at the bottom of the page.



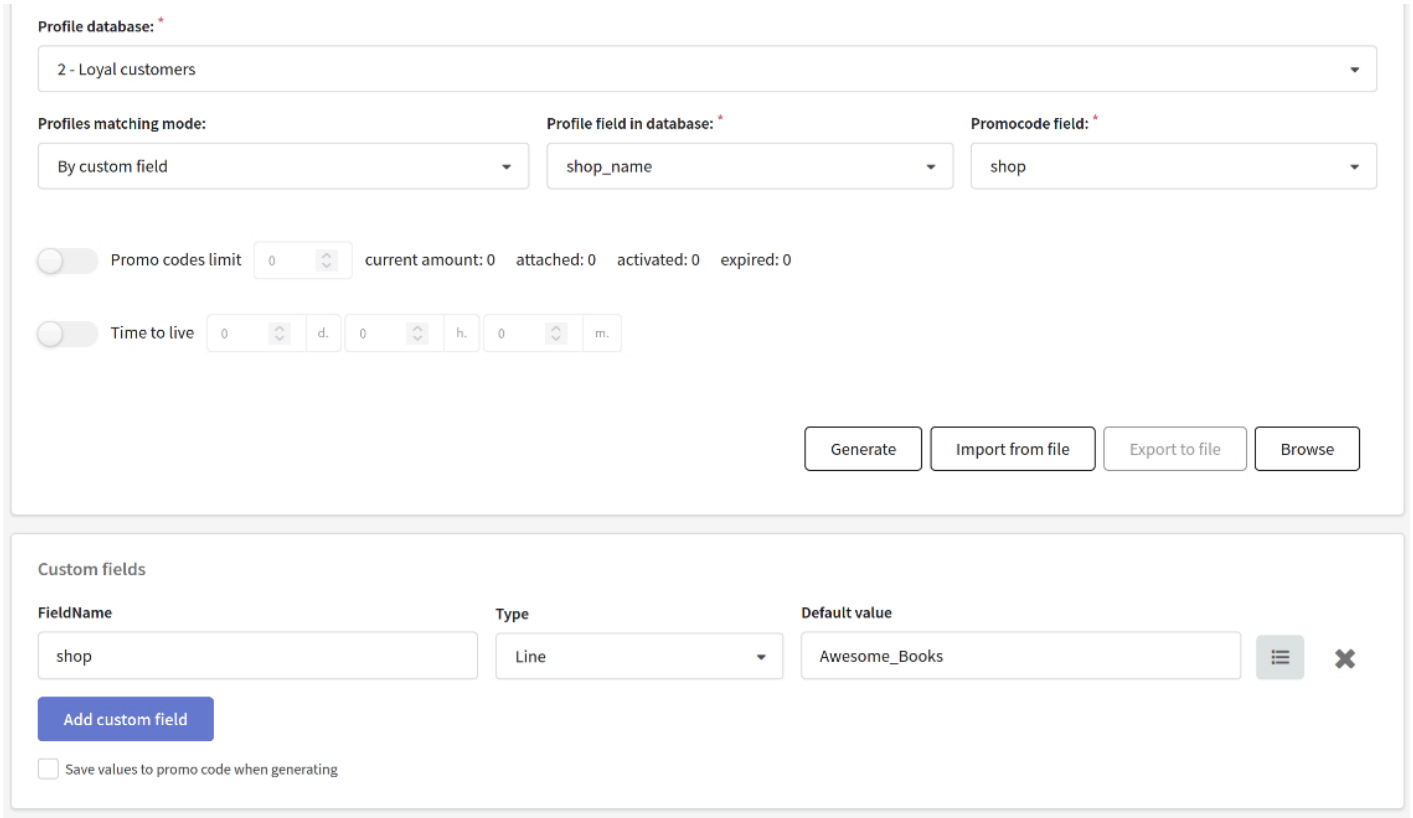
Generation from a template

After configuring generation options, press Generate button. Modal window with promo code preview will open. You can select promo codes quantity and select values for custom fields, if they are used in loyalty program. After confirming promo codes generation they will be updated in the interface at once, or a task for generating promo codes will be created.



To automatically link promo codes to profiles during generation, use the matching mode by a custom field. Create an additional promo code field and select it in the matching settings. Specify a default value. If the value of the custom field in the database matches the value of the custom field of the promo code, the promo code will be assigned to the found profile.

In the example below, promo codes will be assigned to profiles that have a custom field "`shop_name`" containing the value "`Awesome_Books`":

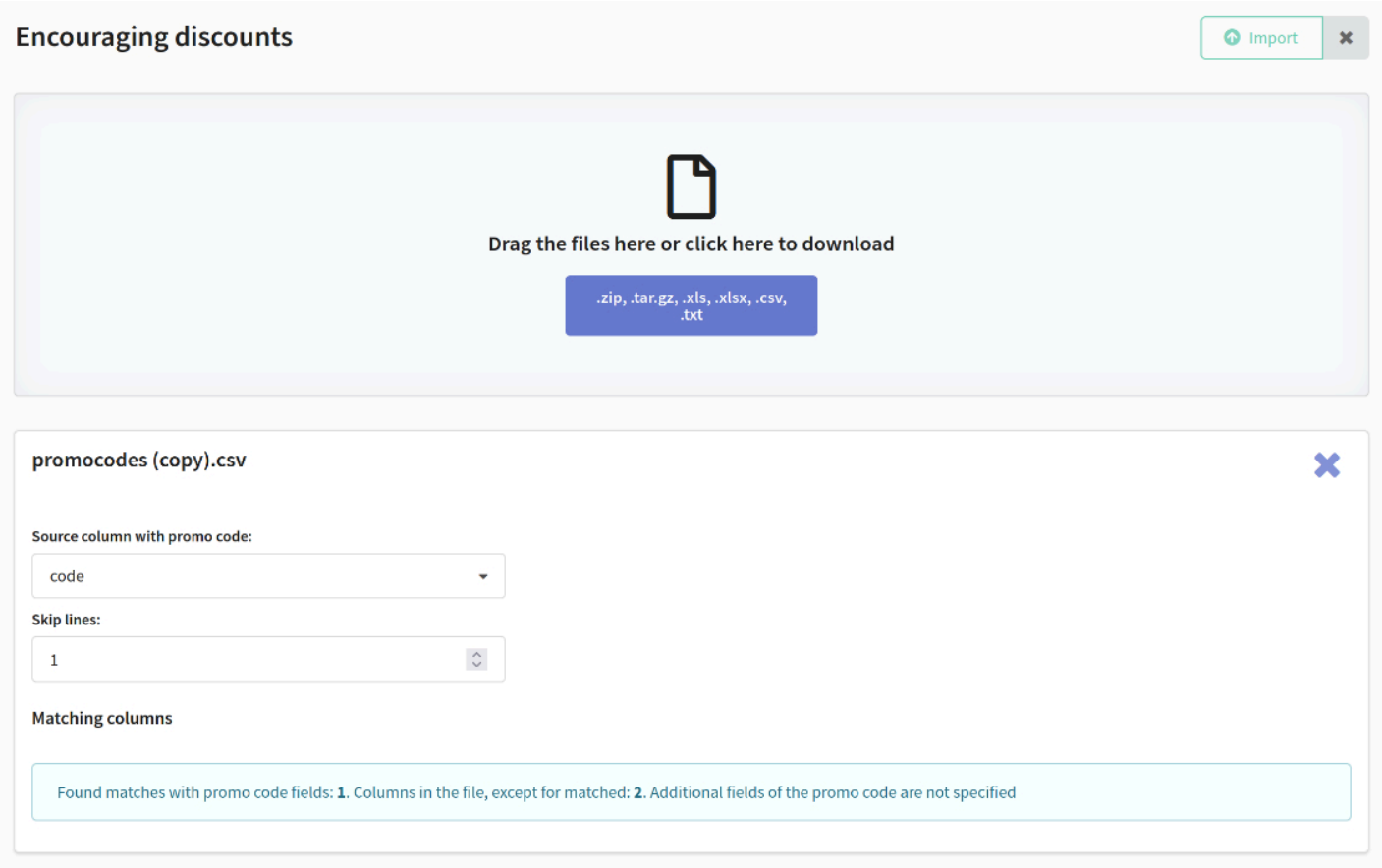


Import from file

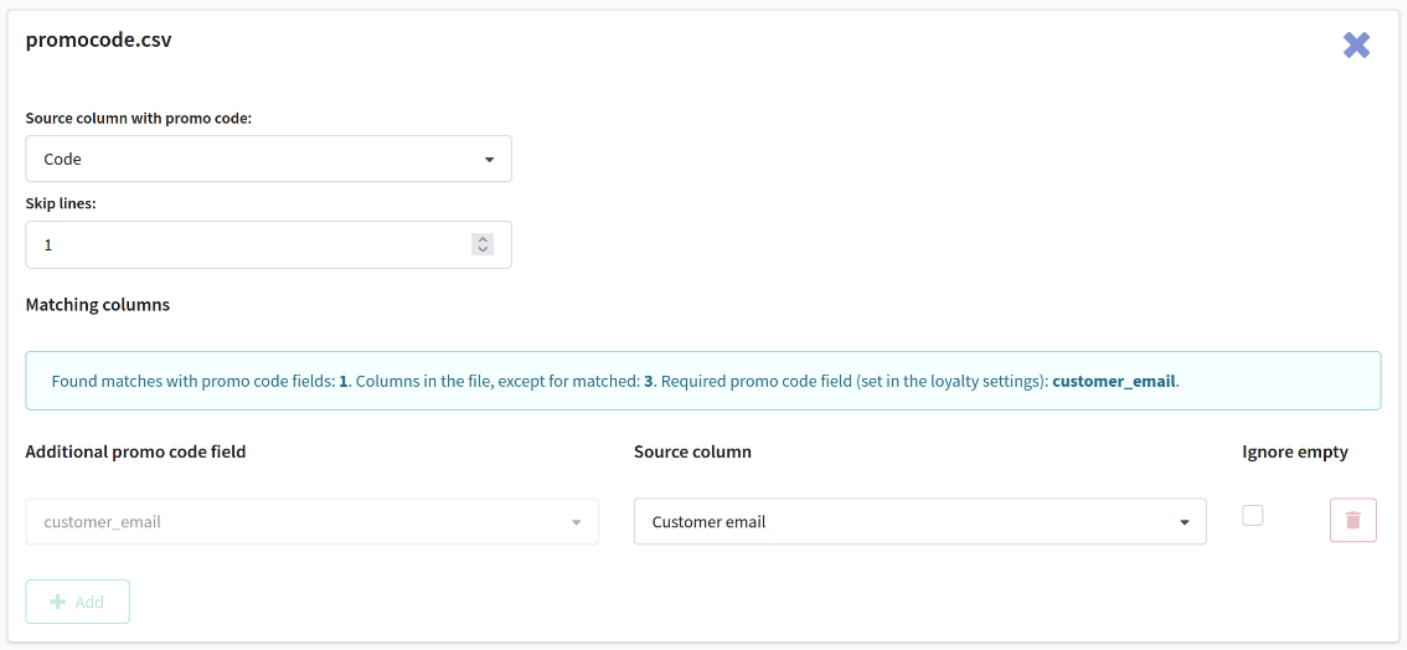
TIP

The import button will be available only after you save the loyalty program.

Press Import from file button to upload XLS or CSV file into the platform, select column with promo codes and press the button at the top part of the page to launch import:

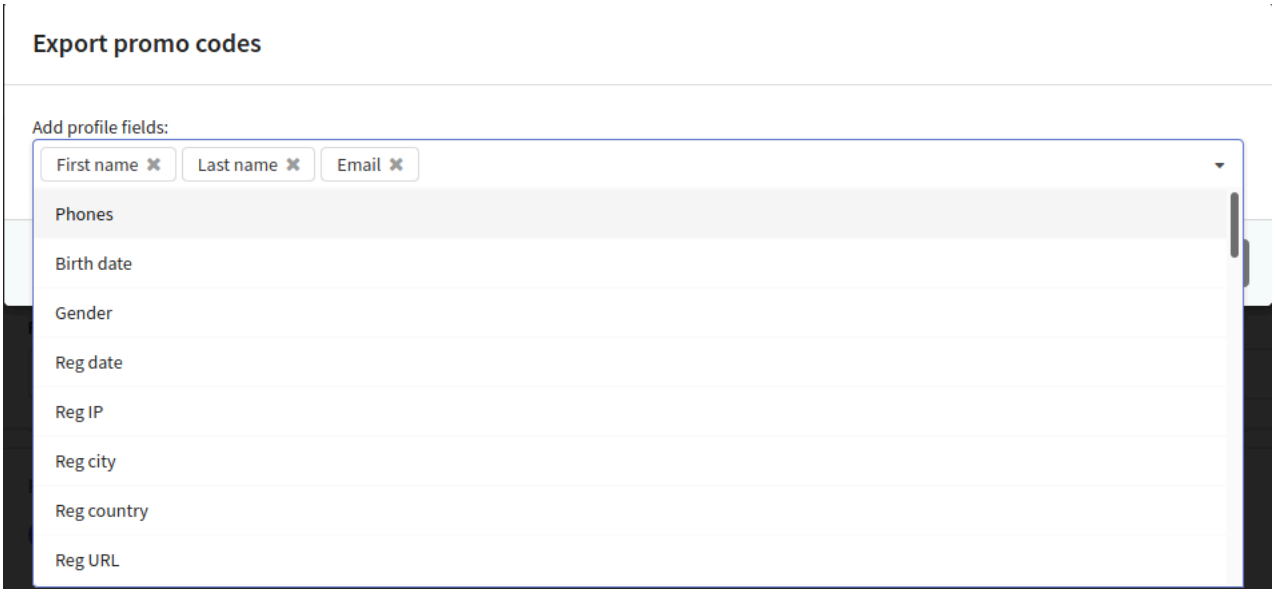


To automatically link promo codes to profiles during import from a file, set up columns matching. As the source column, select the column in the file that will be used to search for the required profile:



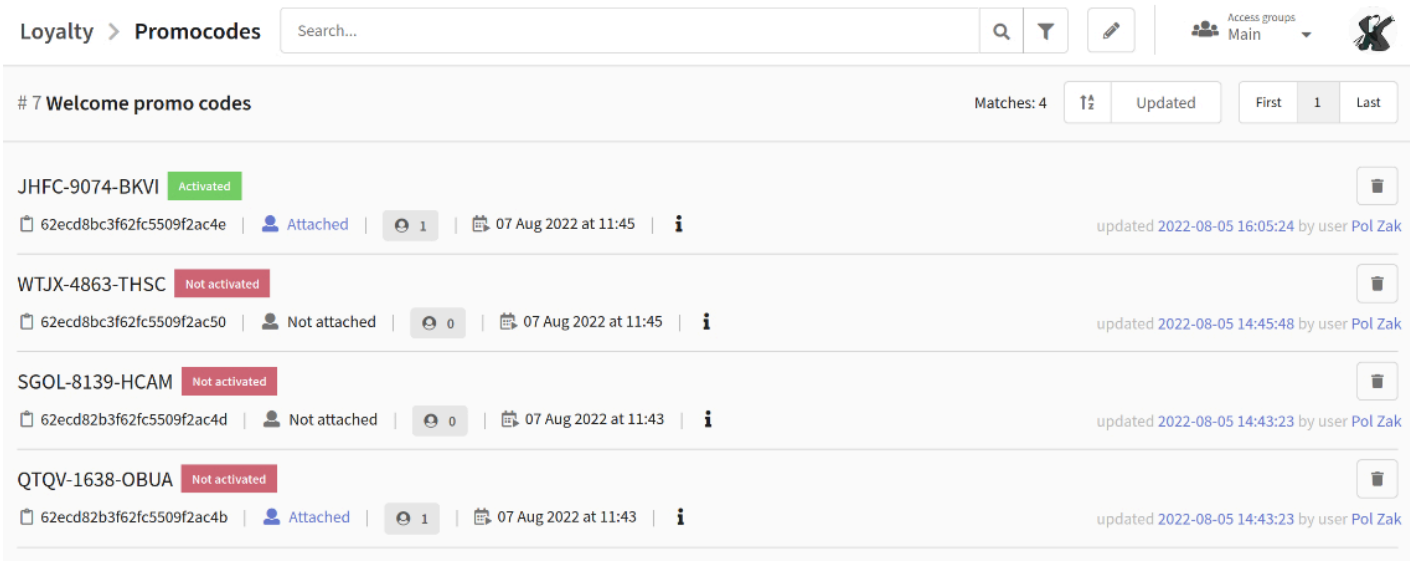
Export to file

Using Export to file, you can download promo codes and fields of profiles, which promo codes were attached to. Select profile fields in the drop-down menu and press OK button to export file in CSV format:



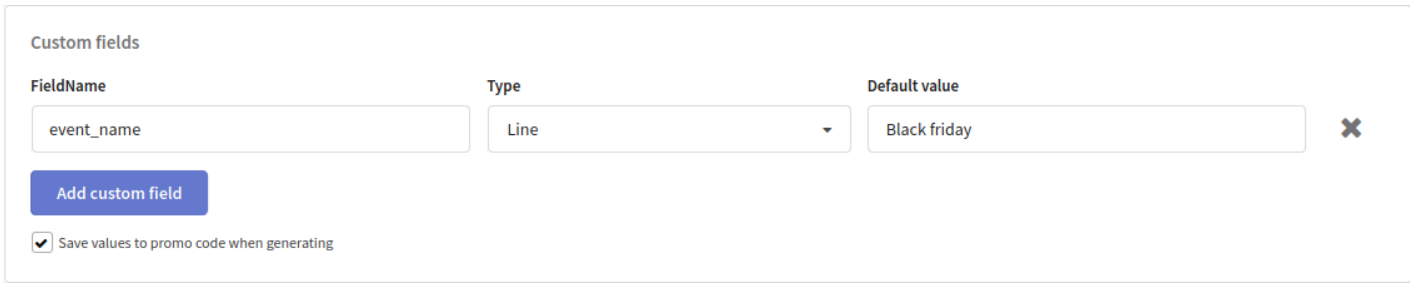
Promo codes browsing

Press Browse button to open list of promo codes created in loyalty program. Brief information will be displayed for each promo code. You can remove any promo code manually from the list:



Custom fields

Custom fields can be requested during promo codes generating and they will be sent to external service, if it is used. If necessary, these fields can be saved in the promo code. If external service returns fields with the same name, they will be also updated in promo code.



Reports and analytics

NOTE

All reports can be downloaded in CSV and XLSX formats by clicking the Export Report button in the action menu.

Channel report

Description

The channel report provides data on all events that are tracked in campaigns: sends, deliveries, opens, clicks, etc. You can get information either for one channel or for all of them at once. It is possible to group data according to various attributes, drill down into the report or use filters to get only the information you need. You can also compare metrics, for example, for different time periods or for different templates.

Get started by selecting Analytics —> Channels in the main menu.

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Report settings

Channel report settings include:

- **grouping data,**
- **adding filters**
- **saving the report.**

On the right you can see the following buttons:

****List of reports** — open the list of ready reports;**

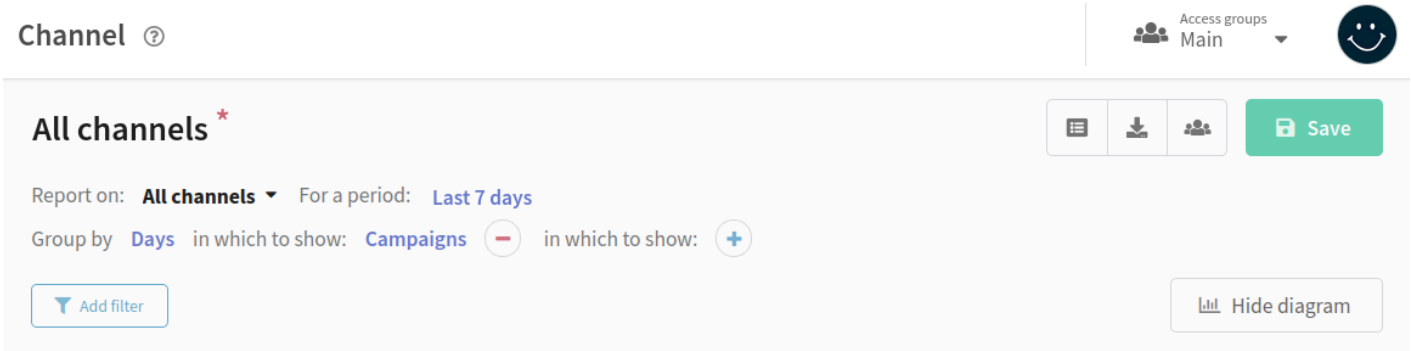
****Export report** — download the report in csv or xlsx formats;**

****Assigned groups** — assign access groups**

Grouping data

By default, the report is built for all channels — Email, SMS, Push, Telegram, WhatsApp*, Viber™, Notify, MAX, as well as custom ones. To get data for a specific channel, select it in the "Report by field.

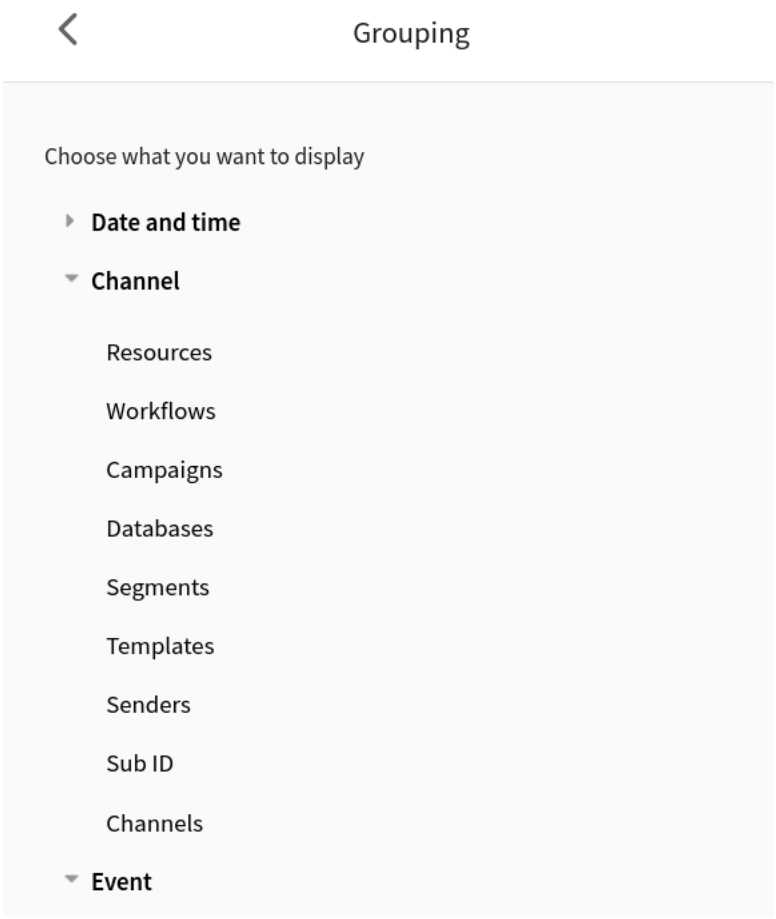
Next, set a period. You can set the date range in the calendar manually or choose one of the presets.



Select the attribute by which you want to group the data in the report. After that, you can detail the grouping by selecting the attributes one by one in the in which to show" field. Grouping is possible up to three levels.

You will see the attributes for grouping in the window on the right. They are divided into three categories:

- **Date and time. Group data by hours, days, weeks, months, quarters and years.**
- **Channel. Group by platform objects (resources, scenarios, campaigns, mailings, segments, etc.).**
- **Event. Group by link, domain, domain group, or provider.**



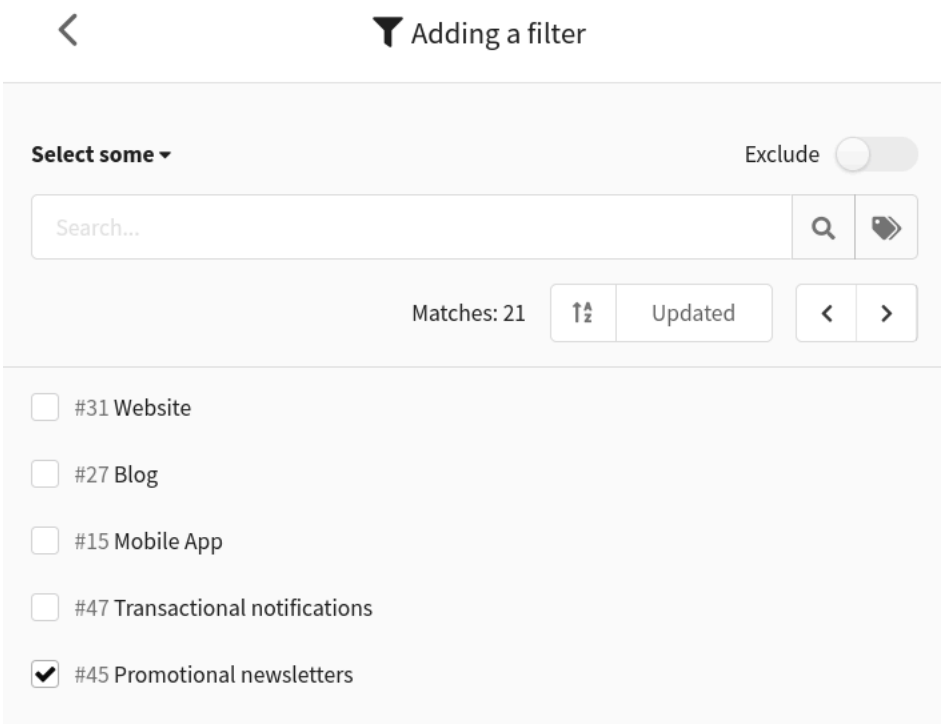
Adding filters

Categories	Available filters	Application
Date and time	• **Hours (24)** • **Days** • **Weekdays** • **Year and weeks** • **Months** • **Quarters** • **Years** • **Year and quarters**	**Using filters you can add to the report or exclude from it data for a certain period of time. Such filters are convenient to view statistics only for specific days or remove statistics for holidays and weekends from the report.** **You can also exclude data collected during periods when technical work was being done on your side and mailings were not being sent.**
Channel	• **Campaigns — statistics for all mailings and workflows included in selected campaigns** • **Channels — filter used to display data for a specific channel when the report was built for all channels** • **Resources — statistics for all mailings launched within the resource** • **Workflows — statistics for all mailings launched as part of an automation scenario** • **Mailings — statistics for specific mailings** • **Databases — statistics for mailings whose recipients are included in the selected database** • **Segments — statistics for mailings whose recipients are included in the selected segments** • **Templates — statistics for all mailings that use the selected templates** • **Senders — virtual senders**	**Mailings filter helps get data only for a specific type of mailings. For example, if you want to analyze only trigger mailing metrics, select the appropriate mailings from the list. To search for needed mailings, you can enter their name or ID in the search bar or specify tags.** To view metrics for mailings with a specific tag, switch Select some to Select by tag. A similar option is provided for any object in the "Channel" category. </td>
Event	• **Vendor — add or exclude mailing statistics depending on the Vendor field value in the profile database** • **Domain group — data for emails sent to a specific domain group (Gmail, Hotmail, Yandex)** • **Domain — data for emails sent to a specific domain (gmail.com, example.com, altcraft.com, etc.)** • **Link — mailing events associated with specific links in the template. Links are selected by the `name` attribute**	**In case of deliverability issues, you can view data for each ISP separately by setting up a domain group filter.** **If you don't want the report to include data of recipients with your corporate email, select the domain filter, enter the corporate domain and exclude it.**

To filter data, click the “Add filter” button. A window will appear on the right where you can choose which data to include in the report.

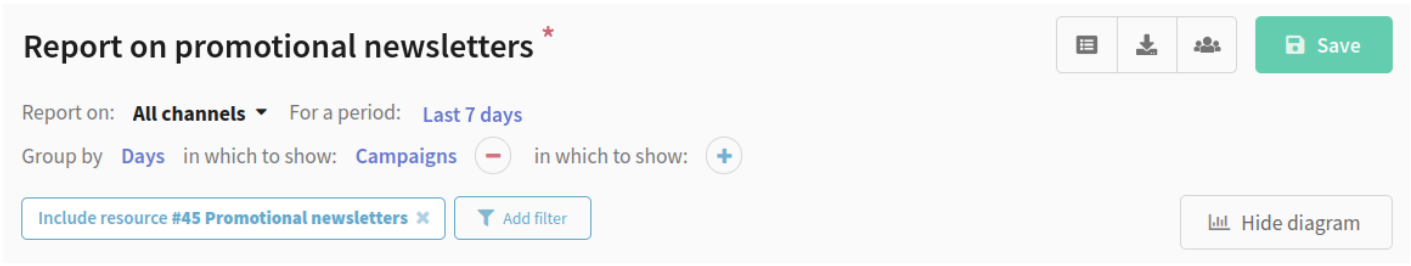
If you need to remove some data from the report, click on the Exclude switch in the upper right corner.

After setting the filter, click Apply at the bottom of the page.



The added filters will be displayed in the report. There is no limit to the number of filters you can use. Click

to remove the filter.



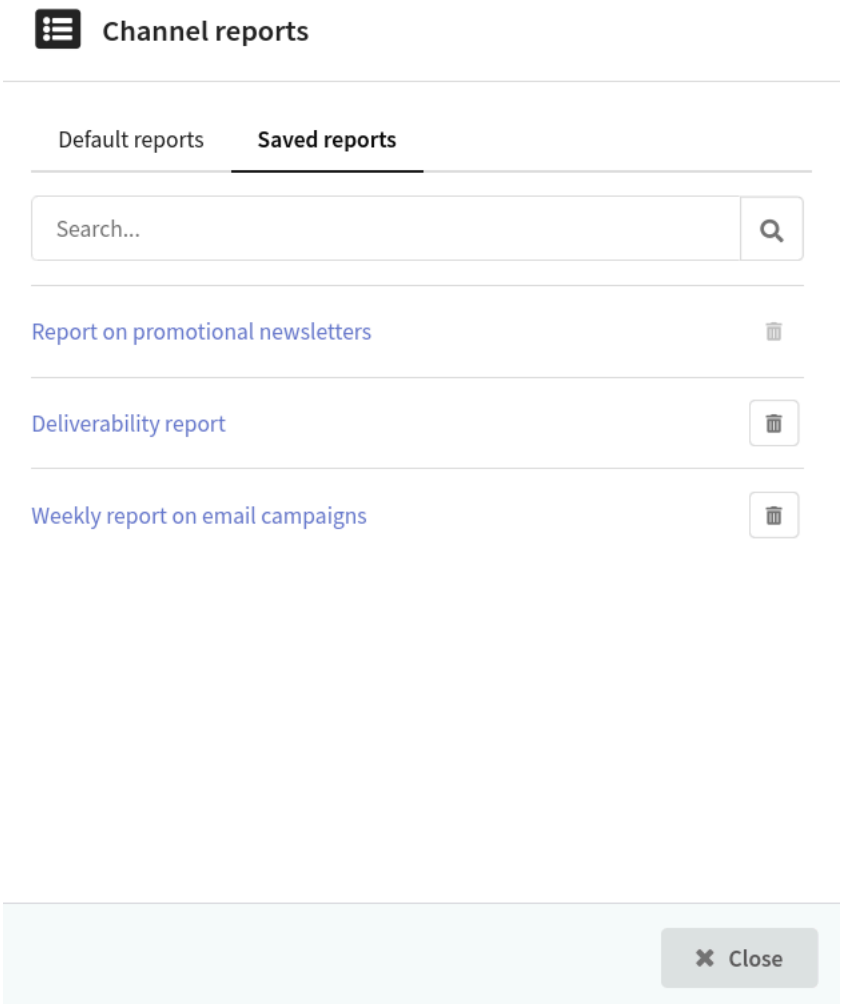
Saving a report

You can save the current report settings. To do this, edit the name of the report (the default is “All Channels”) and click the “Save” button. This is useful if you frequently query data using the same groupings and filters.

The saved report will be available to all users within group visibility.

Click

to open the list of reports. In a new window you will find default (system) and saved reports. Click on the name of the report to view it.



To delete a saved report, click

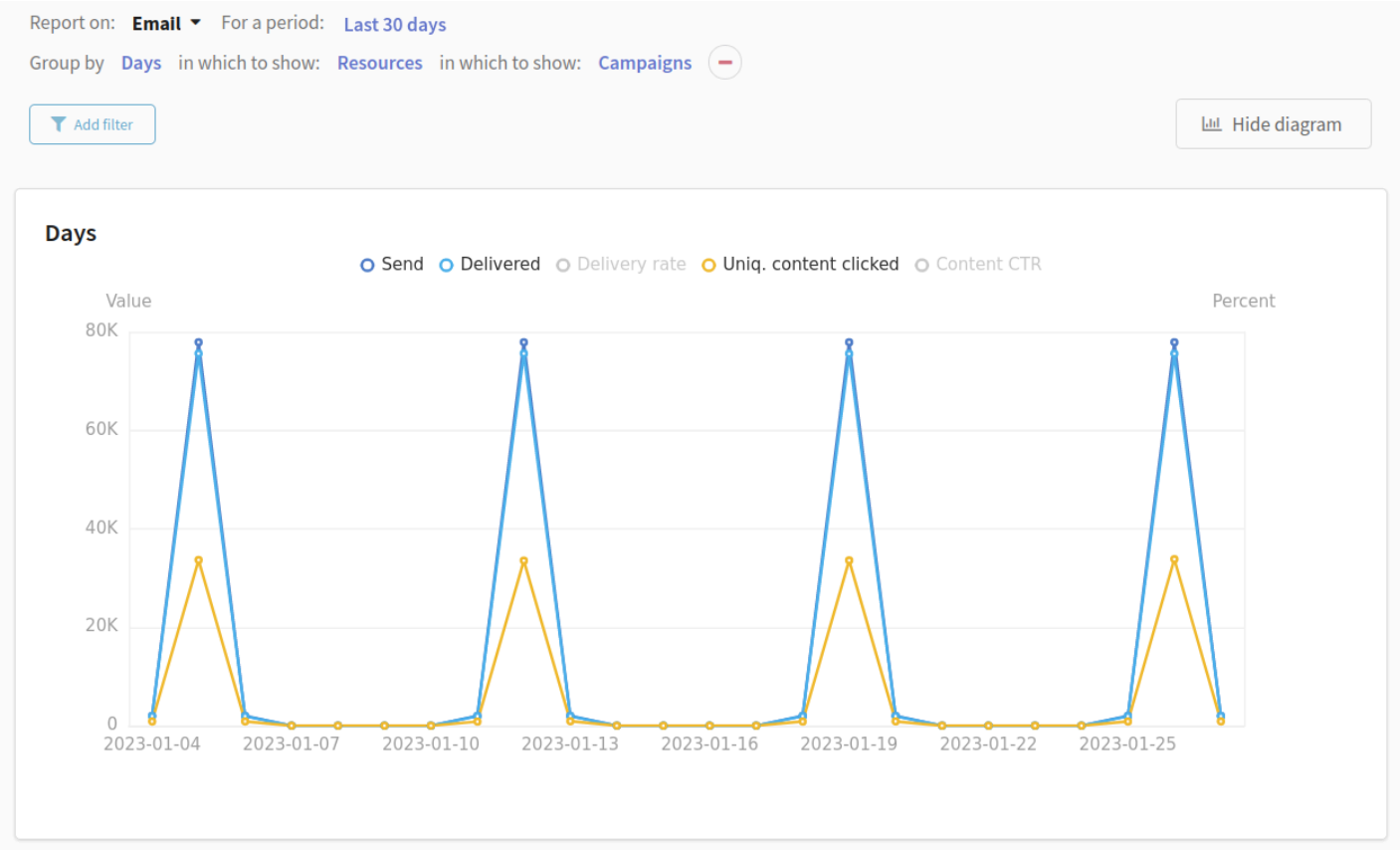
. However, you cannot delete the report that you are currently viewing.

Viewing a report

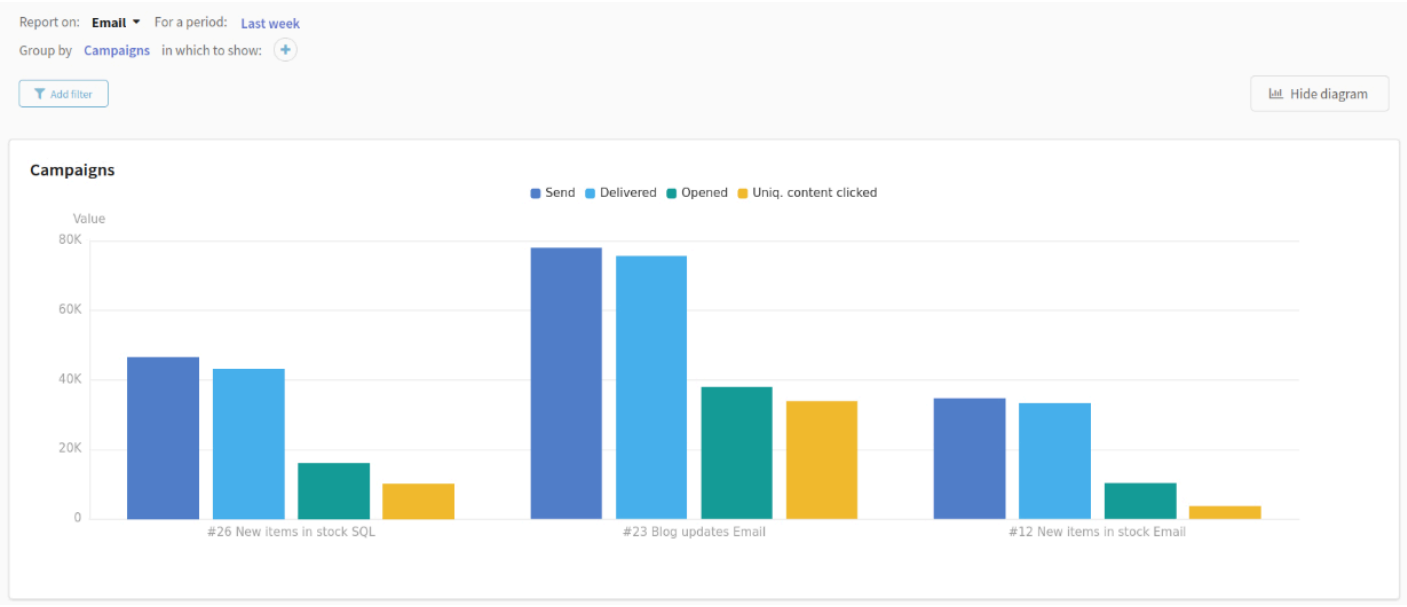
The report includes a table with data and a corresponding diagram. The chart can be collapsed by clicking on the Hide diagram button.

Diagram

The chart displays data based on grouping. When the report is grouped by days, weeks and quarters, the diagram shows the dynamics of changes in the metrics for the selected period:



When grouped by hour, day of the week, month, year, or channel and event, the chart shows the ratio of quantitative or percentage metrics in the selected period:



When you hover over the chart, you can see the exact figures.

The header contains the metrics displayed on the chart. Any column of the table can be displayed on the diagram, but the maximum number of indicators is 7. Click the metric to hide it.

Data table

The table contains quantitative and percentage metrics of campaigns (sends, deliveries, clicks, unsubscribes, complaints, etc.). Description of metrics can be found below.

Metrics are grouped based on the report settings. Depending on the details of the grouping, the table structure may have nested elements.

For example, with the setting “Group by Campaigns, in which to show Mailings, in which to show Channels” the table will display the hierarchical structure of marketing strategy effectiveness.

With the setting “Group by days, in which to show Resources, in which to show Mailings” the table will display the following data:

Name	Send	Delivered	Delivery rate	Uniq. content clicked	Content CTR
Total	8 303 211	7 287 553	87,77 %	2 150 570	29,51 %
2022-11-11	3 270	746	22,81 %	368	49,33 %
2022-11-10	74 032	71 739	96,9 %	31 980	44,58 %
#4 Service messaging (no opt-out)	863	838	97,1 %	442	52,74 %
#28 Welcome trigger	328	320	97,56 %	139	43,44 %
#6 Welcome trigger	349	337	96,56 %	144	42,73 %
#5 SMS opt-in reminder	186	181	97,31 %	159	87,85 %
#2 Blog subscription	73 169	70 901	96,9 %	31 538	44,48 %
2022-11-09	5 214	5 050	96,85 %	2 365	46,83 %
2022-11-08	1 984	1 931	97,33 %	1 275	66,03 %

To add a new metric to the table, hover over the column name and click

. Select a metric from the list. If you want to remove a column, click

.

To move a column to another part of the table, you need to hover over the column name, hold down the left mouse button and drag the column to the desired position.

Name	Send	Delivered	Delivery rate	Uniq. content clicked	Content CTR
Total	8 303 211	7 287 553	87,77 %	2 150 570	29,51 %
2022-11-11	3 270	746	22,81 %	368	49,33 %

Other options:

— switch quantitative metrics to percentages;

— display a column on a diagram.

Comparing data

The comparison mode allows you to see data on several objects at the same time. For example, you can compare the performance of two templates or evaluate the change in campaign metrics from period to period.

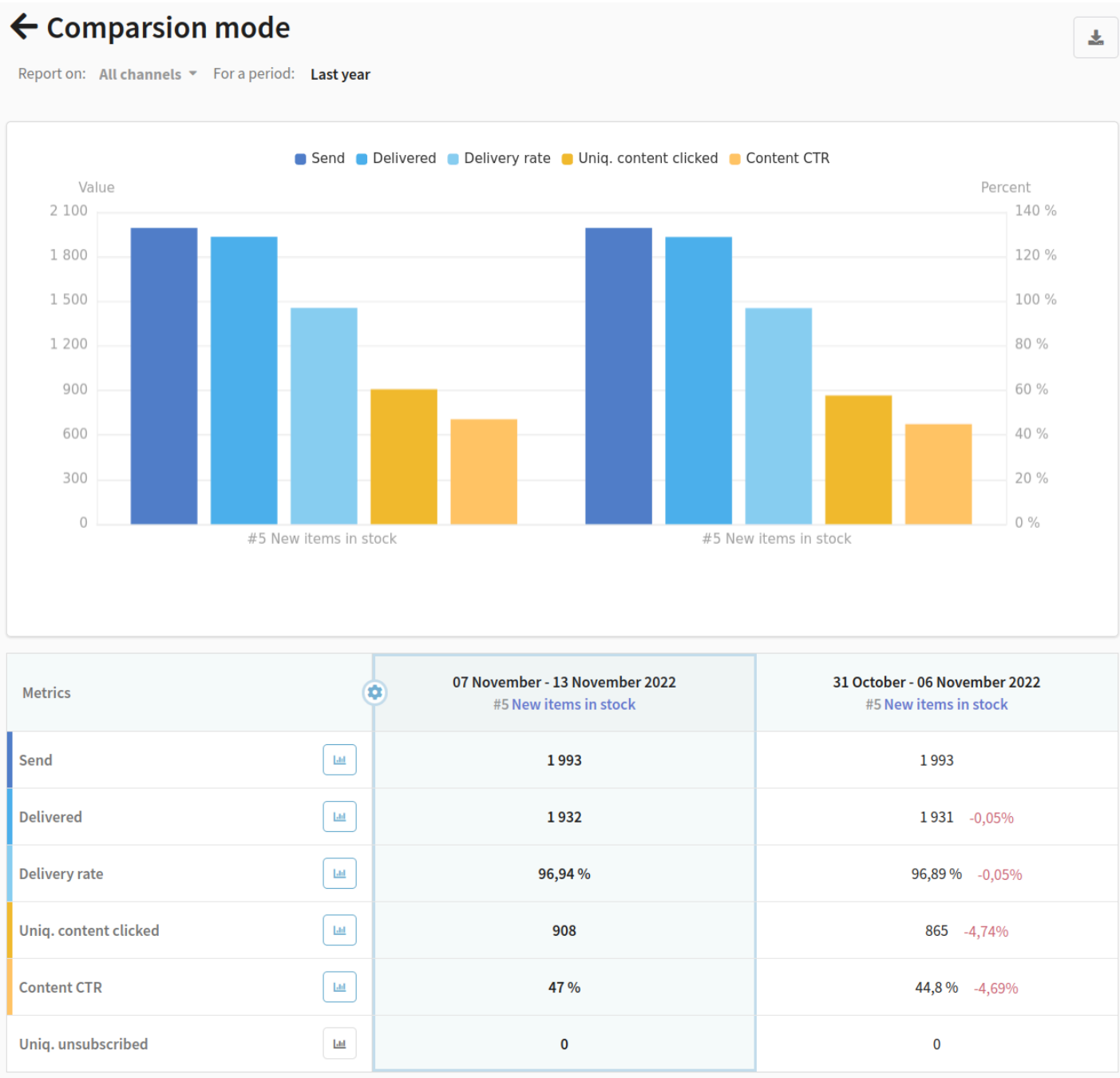
To switch to the comparison mode, select the objects in the table that you want to compare. Click “Compare”. It is possible to compare objects at different nesting levels.

Name	Compare	Send	Delivered	Delivery rate	Uniq. content clicked	Content CTR
Total		8 303 211	7 287 553	87,77 %	2 150 570	29,51 %
07 November - 13 November 2022		90 287	80 588	89,26 %	36 583	45,4 %
#9 Happy to meet you - join in!		2 352	1 517	64,5 %	687	45,29 %
#5 New items in stock		1 993	1 932	96,94 %	908	47 %
#3 Blog digest		74 833	70 578	94,31 %	31 391	44,48 %
#2 New items in stock		7 189	4 269	59,38 %	2 230	52,24 %
#1 Welcome message		3 920	2 292	58,47 %	1 367	59,64 %
31 October - 06 November 2022		87 776	81 327	92,65 %	36 780	45,22 %
#9 Happy to meet you - join in!		1 818	1 461	80,36 %	622	42,57 %
#5 New items in stock		1 993	1 931	96,89 %	865	44,8 %
#3 Blog digest		72 417	70 269	97,03 %	31 511	44,84 %
#2 New items in stock		8 671	5 341	61,6 %	2 356	44,11 %
#1 Welcome message		2 872	2 325	80,95 %	1 426	61,33 %

The comparison mode also contains a chart and a table with metrics. In the diagram, the metrics for each object are grouped together.

The first object in the table is the comparison column. Other objects are compared with this object. The percentage increase or decrease of the metric is calculated relative to this object. To change the comparison column, click

and select a different object.



Metrics	07 November - 13 November 2022 #5 New items in stock	31 October - 06 November 2022 #5 New items in stock
Send	1 993	1 993
Delivered	1 932	1 931 -0,05%
Delivery rate	96,94 %	96,89 % -0,05%
Uniq. content clicked	908	865 -4,74%
Content CTR	47 %	44,8 % -4,69%
Uniq. unsubscribed	0	0

To go back to the general mode, click ← Comparison mode.

Report metrics

INFO: Unique event

The uniqueness of any event is limited by the time specified in the platform configuration (`UNIQUE_ACTIONS_TTL_MIN` parameter). By default it is 1440 minutes. It means that a repeated event in a day will be counted as unique again. Check with your platform administrator for the uniqueness value set.

All indicators of unique events are marked in the table.

All metrics

Metric	Channel	Description
Send	**All**	**The total number of messages sent through all channels.**
Undelivered	**All**	**The total number of messages sent but not delivered through all channels. It consists of Hardbounced and Softbounced messages, delivery errors and messages which are not delivered due to undefined reason.**
Undelivery rate	**All**	**The percentage of undelivered messages through all channels.**
Delivered	**All**	**The total number of delivered messages through all channels.**
Delivery rate	**All**	**The percentage of messages delivered through all channels.**
Opened	**Email** **Push**	**The number of open messages.** **If the same profile opens a message multiple times, each open is counted.**

Uniq. opened	**Email** **Push**	**The total number of unique message opens through all channels.** **If the same profile opens a message multiple times, only one open is counted.**
Open rate	**Email** **Push**	**The percentage of profiles who opened the email or push.** **(Uniq. opened / Delivered) * 100%**
Replied	**Email**	**The number of replies to an email.**
Read	**Email** **WhatsApp*** **Viber** **Notify**	**The number of emails read.** **An event is logged if the recipient opens the message and views it for at least 8 seconds.**
Uniq. read	**Email**	**The number of unique reads of an email.**
Read rate	**Email**	**The number of unique reads of an email.** **(Uniq. read / Delivered) * 100%**
Glanced	**Email**	**An event is logged if the recipient opens the email and views it for less than 4 seconds.**
Skimmed	**Email**	**The event is logged if the recipient views the message for 4 to 8 seconds.**
Content clicked	**All**	**The number of clicks on links. When a recipient clicks multiple times on the same link, each click event is counted. For example, if there are 3 links in a message and the recipient clicks each link twice, 6 clicks are counted.**
Uniq. content clicked	**All, but Telegram Group and MAX Group**	**Number of profiles that clicked any link in the message at least once. Counts profile engagement with message links. If the same profile clicks on any part of the message multiple times, only one click on the content is counted. Used to calculate content CTOR and CTR.**
Uniq. link clicked	**Email** **Telegram Bot** **WhatsApp*** **Viber** **Notify** **MAX bot**	**The total number of unique clicks on links in messages. Tracks only the first engagement per link per profile. Used to calculate link CTOR and CTR.**
Content CTR	**All but Telegram Group and MAX Group**	**The percentage of users who click on at least one link within a piece of content out of the total number of users who received it.** **(Uniq. content clicked / Delivered) * 100%**
Content CTOR	**Email** **Push**	**The percentage of users who clicked on any link within a piece of content out of those who opened/viewed it (rather than just received it).** **(Uniq. content clicked / Uniq. opened) * 100%**
Link CTR	**Email** **Telegram bot** **MAX bot** **Whatsapp** **Viber** **Notify**	**The percentage of clicks on the link within a message out of the number of delivered messages.** **(Uniq. link clicked / Delivered) * 100%**
Link CTOR	**Email**	**The percentage of clicks on the link within a message out of the number of opens.** **(Uniq. link clicked / Uniq. opened) * 100%**
Camp. errors	**All**	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**All**	**Problems with protocol during sending a message.**
Suppressed	**All**	**The number of messages that were not sent because the recipient's email address or phone number has previously been included in the suppression list.**
Policy offenced	**All**	**The number of violations of the sending policy.**
Softbounced	**Email**	**The number of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailable.**
Softbounce rate	**Email**	**The percentage of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailable.**
Hardbounced	**Email** **SMS** **Push**	**The number of messages that were not delivered due to an invalid email address, phone number, or push subscription ID.**
Hardbounce rate	**Email** **SMS** **Push**	**The percentage of messages that were not delivered due to an invalid email address, phone number, or push subscription ID.**
Complained	**Email** **SMS** **Push**	**The number of messages marked as spam by recipients.**
Complains rate	**Email**	**The percentage of messages marked as spam by profiles.** **(Complained / Delivered)*100**
Subscribed	**All but Telegram Group and MAX Group**	**The number of subscriptions to resources in any channel.**
Unsubscribed	**Email** **SMS** **Push** **Viber** **Notify**	**The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message.**
Uniq. unsubscribed	**Email** **SMS** **Push**	**The percentage of unique unsubscribes from resources.**
Unsub to deliv rate	**Email** **SMS** **Push**	**(Unsubscribed / Delivered) * 100%**
Unsub to open rate	**Email** **Push**	**(Unsubscribed / Opened) * 100%**
Undefined	**-**	**Unable to determine reason for non-delivery.**

Email channel

Metric	Description
Send	**The number of sent email messages.**

Undelivered	**The number of undelivered email messages. It consists of Hardbounced and Softbounced messages and messages which are not delivered due to undefined reason.**
Undelivery rate	**The percentage of undelivered email messages.**
Delivered	**The number of delivered email messages.**
Delivery rate	**The percentage of delivered email messages.**
Opened	**The number of opened email messages. If the same profile opens an email message multiple times, each open is counted.**
Uniq. opened	**The number of unique email message openings. If the same profile opens a message multiple times, only one open is counted.**
Open rate	**The percentage of recipients who opened the email. (Uniq. opened / Delivered) * 100%**
Replied	**The number of replies to an email.**
Read	**The number of emails read. An event is logged if the recipient opens the message and views it for at least 8 seconds.**
Uniq. read	**The number of unique reads of an email.**
Read rate	**The number of unique reads of an email. (Uniq. read / Delivered) * 100%**
Glanced	**An event is logged if the recipient opens the email and views it for less than 4 seconds.**
Skimmed	**The event is logged if the recipient views the message for 4 to 8 seconds.**
Content clicked	**The number of clicks on links. When a recipient clicks multiple times on the same link, each click event is counted. For example, if there are 3 links in a message and the recipient clicks each link twice, 6 clicks are counted.**
Uniq. content clicked	**Number of profiles that clicked any link in the message at least once. Counts profile engagement with message links. If the same profile clicks on any link multiple times, only one click is counted. When grouped by links, it allows tracking which links users select first when engaging with the message.**
Uniq. link clicked	**The total number of unique clicks on links in email messages. Tracks only the first engagement per link per profile. If the recipient clicks on two different links in the email, 2 unique clicks on the link will be counted.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages. (Uniq. content clicked / Delivered) * 100%**
Content CTOR	**The percentage of users who clicked on any link within a piece of content out of those who opened/viewed it (rather than just received it). (Uniq. content clicked / Uniq. opened) * 100%**
Link CTR	**The percentage of clicks on the link within a message relative to the number of delivered messages. (Uniq. link clicked / Delivered) * 100%**
Link CTOR	**The percentage of clicks on the link within a message relative to the number of opens. (Uniq. link clicked / Uniq. opened) * 100%**
Camp. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Suppressed	**The number of messages that were not sent because the recipient's email address was included in the suppression list.**
Policy offenced	**The number of violations of the sending policy.**
Softbounced	**The number of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailable.**
Softbounce rate	**The percentage of emails not delivered due to the inbox being full or the recipient's mail server being temporarily unavailable.**
Hardbounced	**The number of messages that were not delivered due to an invalid email address.**
Hardbounce rate	**The percentage of messages that were not delivered due to an invalid email address.**
Complained	**The number of messages marked as spam by recipients.**
Complains rate	**The percentage of messages marked as spam by recipients. (Complained / Delivered) * 100**
Subscribed	**The number of subscriptions to the email channel in resources.**
Unsubscribed	**The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the email message.**
Uniq. unsubscribed	**The percentage of unique unsubscribes from resources.**
Unsub to deliv rate	**(Unsubscribed / Delivered) * 100%**
Unsub to open rate	**(Unsubscribed / Opened) * 100%**
Undefined	**Unable to determine reason for non-delivery.**

SMS channel

Metric	Description
Send	**The total number of sent SMS messages.**
Undelivered	**The number of undelivered SMS messages.**
Undelivery rate	**The percentage of undelivered SMS messages.**
Delivered	**The total number of delivered SMS messages.**
Delivery rate	**The percentage of delivered SMS messages.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**Number of profiles that clicked on any link in the message at least once. Counts profile engagement with message links. If the same profile clicks on any link multiple times, only one unique content click is counted.**
Content CTR	**The percentage of users who click on at least one link within the SMS message out of the total number of delivered messages. (Uniq. content clicked / Delivered) * 100%**
Camp. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Suppressed	**The number of messages that were not sent because the recipient's email address or phone number has previously been included in the suppression list.**
Policy offenced	**The number of violations of the sending policy.**
Hardbounced	**The number of messages that were not delivered due to an invalid phone number**
Hardbounce rate	**The percentage of messages that were not delivered due to an invalid phone number.**
Complained	**The number of messages marked as spam by recipients.**
Subscribed	**The number of subscriptions to the SMS channel in resources.**
Unsubscribed	**The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the SMS message.**
Uniq. unsubscribed	**The percentage of unique unsubscribes from resources.**
Unsub to deliv rate	**(Unsubscribed / Delivered) * 100%**

Undefined	**Unable to determine reason for non-delivery (unknown network error).**

Push channel

Metric	Description
Send	**The number of sent push notifications.**
Undelivered	**The number of undelivered push notifications.**
Undelivery rate	**The percentage of undelivered push notifications.**
Delivered	**The number of delivered push notifications.**
Delivery rate	**The percentage of delivered push notifications.**
Opened	**The number of opened push notifications.** ****Web push:**** **Number of opens equals number of deliveries.** ****Mobile push:**** **Open tracking depends on client-side implementation of event handling for links. Opens are recorded by the application itself and don't depend on whether the push notification contains links.**
Uniq. opened	**The number of delivered push notifications openings.** If the same profile opens an notifications multiple times, only one open is counted. </td> </tr> <tr> <td>Open rate</td> <td> The percentage of recipients who opened the push. (Uniq. opened / Delivered) * 100% </td> </tr> <tr> <td>Content clicked</td> <td>The number of clicks in the push notification (inluding links and buttons). For mobile pushes, clicks are counted only if the link includes tracking. Used to track redirects to websites.</td> </tr> <tr> <td>Uniq. content clicked</td> <td> If the same profile clicks the push notification multiple times, only one unique content click is counted. </td> </tr> <tr> <td>Content CTR</td> <td> The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages. (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Content CTOR</td> <td> The percentage of users who clicked on any link within a piece of content out of those who opened/viewed it (rather than just received it). (Uniq. content clicked / Uniq. opened) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Policy offenced</td> <td> The number of violations of the sending policy. </td> </tr> <tr> <td>Hardbounced</td> <td> The number of messages that were not delivered due to an invalid push subscription ID. </td> </tr> <tr> <td>Hardbounce rate</td> <td> The percentage of messages that were not delivered due to an invalid push subscription ID. </td> </tr> <tr> <td>Complained</td> <td>The number of messages marked as spam by recipients.</td> </tr> <tr> <td>Subscribed</td> <td>The number of subscriptions to the Push channel in resources.</td> </tr> <tr> <td>Unsubscribed</td> <td> The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message. </td> </tr> <tr> <td>Uniq. unsubscribed</td> <td>The percentage of unique unsubscribes from resources.</td> </tr> <tr> <td>Unsub to deliv rate</td> <td>(Unsubscribed / Delivered) * 100%</td> </tr> <tr> <td>Unsub to open rate</td> <td>(Unsubscribed / Opened) * 100%</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Telegram Bot channel

Metric	Description
Send	**The number of messages sent through in the Telegram bot.**
Undelivered	**The number of messages sent but not delivered in the Telegram bot.**
Undelivery rate	**The percentage of undelivered messages in the Telegram bot.**
Delivered	**The number of delivered messages in the Telegram bot.**
Delivery rate	**The percentage of messages delivered in the Telegram bot.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**If the same profile clicks on any link multiple times, only one unique content click is counted.**
Uniq. link clicked	**The number of unique clicks on links in messages.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages.** (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Link CTR</td> <td> The percentage of clicks on the link within a message relative to the number of delivered messages. (Uniq. link clicked / Delivered) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Policy offenced</td> <td>The number of violations of the sending policy.</td> </tr> <tr> <td>Subscribed</td> <td>The number of subscriptions to resources in the Telegram bot.</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Telegram Group channel

Metric	Description
Send	**The number of messages sent to the Telegram group.**
Undelivered	**The number of undelivered messages in the Telegram group.**
Undelivery rate	**Percentage of undelivered messages to the Telegram group.**
Delivered	**The number of messages delivered to the Telegram group.**
Delivery rate	**Percentage of messages delivered to the Telegram group.**
Content clicked	**The number of click on links.**
Camp. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Policy offenced	**The number of violations of the sending policy.**
Undefined	**Unable to determine reason for non-delivery (unknown network error).**

MAX Bot channel

Metric	Description
Undelivered	**Number of messages not delivered to the MAX group**
% Undelivered	**Percentage of messages not delivered to the MAX group**
Delivered	**Number of messages delivered to the MAX group**
% Delivered	**Percentage of messages delivered to the MAX group**
Content clicks	**If a recipient clicks the same link multiple times, each click event is recorded. For example, if a message contains 3 links and the recipient clicks each of them twice, 6 clicks will be recorded.**
Generation errors	**An error occurred while generating the template. It happens when the profile does not have a subscription to the resource selected in the Campaign.**
Transport errors	**An error occurred in the message delivery protocols.**
Policy limit exceeded	**Number of messages not sent because the limit set in the sending policy was exceeded.**
Unsubscribes	**The profile unsubscribed from the MAX Group channel.**
Reads	**Number of messages read by the subscriber.**
Undefined	**The reason for undelivery could not be determined (for example, an unknown network error occurred)**

MAX Group channel

Metric	Description
Undelivered	**Number of messages not delivered to the chat with the MAX bot**
% Undelivered	**Percentage of messages not delivered to the chat with the MAX bot**
Delivered	**Number of messages delivered to the chat with the MAX bot**
% Delivered	**Percentage of messages delivered to the chat with the MAX bot**
Content clicks	**If a recipient clicks the same link multiple times, each click event is recorded. For example, if a message contains 3 links and the recipient clicks each of them twice, 6 clicks will be recorded.**
Generation errors	**An error occurred while generating the template. It happens when the profile does not have a subscription to the resource selected in the Campaign.**
Transport errors	**An error occurred in the message delivery protocols.**

Policy limit exceeded	**Number of messages not sent because the limit set in the sending policy was exceeded.**
Unsubscribes	**The profile unsubscribed from the MAX Bot channel.**
Reads	**Number of messages read by the subscriber.**
Undefined	**The reason for undelivery could not be determined (for example, an unknown network error occurred)**

WhatsApp channel

Metric	Description
Send	**Number of messages sent on WhatsApp*.**
Undelivered	**Number of undelivered WhatsApp* messages.**
Undelivery rate	**Percentage of undelivered messages on WhatsApp*.**
Delivered	**Number of messages delivered on WhatsApp*.**
Delivery rate	**Percentage of messages delivered on WhatsApp*.**
Read	**The number of messages read by the user.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**If the same profile clicks on any part of the message multiple times, only one click on the content is counted.**
Uniq. link clicked	**The total number of unique clicks on links in messages.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages.** (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Link CTR</td> <td> The percentage of clicks on the link within a message out of the number of delivered messages. (Uniq. link clicked / Delivered) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Suppressed</td> <td> The number of messages that were not sent because the recipient's phone number has previously been included in the suppression list. </td> </tr> <tr> <td>Policy offenced</td> <td> The number of violations of the sending policy. </td> </tr> <tr> <td>Subscribed</td> <td>The profile subscribed to the WhatsApp* channel.</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Viber channel

Metric	Description
Send	**The total number of messages sent through on Viber.**
Undelivered	**The total number of messages sent but not delivered on Viber.**
Undelivery rate	**The percentage of undelivered on Viber messages.**
Delivered	**The total number of delivered messages on Viber.**
Delivery rate	**The percentage of messages delivered on Viber.**
Read	**The number of messages read by the user.**
Uniq. read	**The number of unique reads in Viber chat.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**If the same profile clicks on any link multiple times, only one click on the content is counted.**
Uniq. link clicked	**The number of unique clicks on links in messages.**

Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages.** (Uniq. content clicked / Delivered) * 100% </td> </tr> <tr> <td>Link CTR</td> <td> The percentage of clicks on the link within a message out of the number of delivered messages. (Uniq. link clicked / Delivered) * 100% </td> </tr> <tr> <td>Camp. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Deliv. errors</td> <td>Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.</td> </tr> <tr> <td>Suppressed</td> <td>The number of messages that were not sent because the recipient's email address or phone number has previously been included in the suppression list. </td> </tr> <tr> <td>Policy offenced</td> <td>The number of violations of the sending policy.</td> </tr> <tr> <td>Subscribed</td> <td>The number of subscriptions to resources in the channel.</td> </tr> <tr> <td>Unsubscribed</td> <td>The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message.</td> </tr> <tr> <td>Undefined</td> <td>Unable to determine reason for non-delivery (unknown network error).</td> </tr>

Notify channel

Metric	Description
Send	**The total number of messages sent through Notify.**
Undelivered	**The total number of messages sent but not delivered on Notify.**
Undelivery rate	**The percentage of undelivered on Notify messages.**
Delivered	**The total number of delivered messages on Notify.**
Delivery rate	**The percentage of messages delivered on Notify.**
Read	**The number of messages read by the user.**
Content clicked	**The number of clicks on links.**
Uniq. content clicked	**If the same profile clicks on any part of the message multiple times, only one click on the content is counted.**
Uniq. link clicked	**The number of unique clicks on links in messages.**
Content CTR	**The percentage of users who click on at least one link within a piece of content out of the total number of delivered messages. (Uniq. content clicked / Delivered) * 100%**
Link CTR	**The percentage of clicks on the link within a message out of the number of delivered messages. (Uniq. link clicked / Delivered) * 100%**
Camp. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Deliv. errors	**Problems with message template generation. It occurs when a profile does not have a subscription to the resource selected in the campaign.**
Suppressed	**The number of messages that were not sent because the recipient's email address or phone number has previously been included in the [suppression list](/en/user-guide/profiles-and-databases/66534).**
Policy offenced	**The number of violations of the sending policy.**
Subscribed	**The number of subscriptions to resources in the channel.**
Unsubscribed	**The number of unsubscribes from resources, i.e. clicks on the unsubscribe link in the message.**
Undefined	**Unable to determine reason for non-delivery (unknown network error).**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

*** Viber™ is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.**

Traffic report

Description

The traffic report provides data on profile traffic from sources such as channels, web forms, pixels, and pop-ups. You can group the data according to certain attributes, drill down into the report, or use filters to exclude redundant information. It is also possible to compare the data to see how switching to a different strategy have affected your performance metrics.

The report is based on hit. It is a unit of traffic measurement. A hit is logged when a user action on a website or mobile application sends a request to the server. The action can be a goal registration or a click in a message.

Report settings

Traffic report settings include:

- **grouping data,**
- **adding filters**
- **saving the report.**

On the right you can see the following buttons:

****List of reports** — open the list of ready reports;**

****Export report** — download the report in csv or xlsx formats;**

****Assigned groups** — assign access groups**

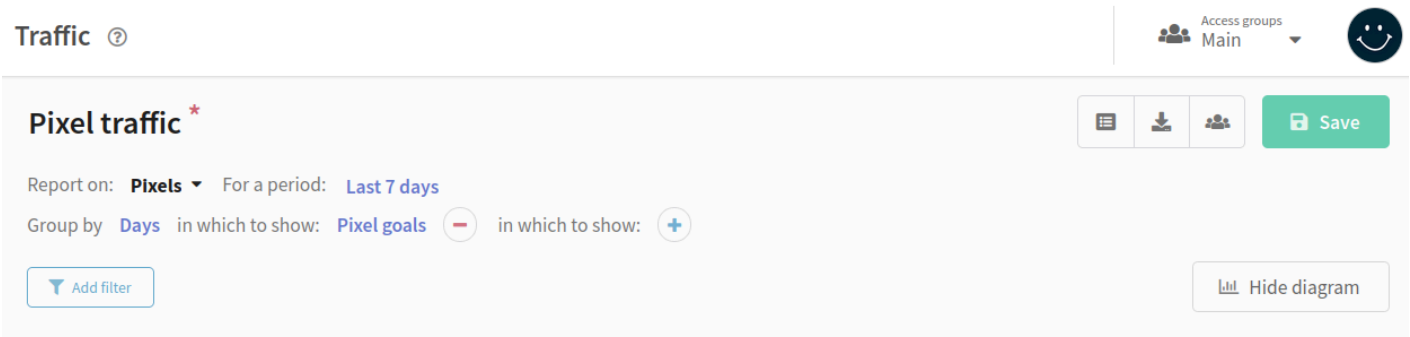
Grouping data

The report provides data on the following traffic sources:

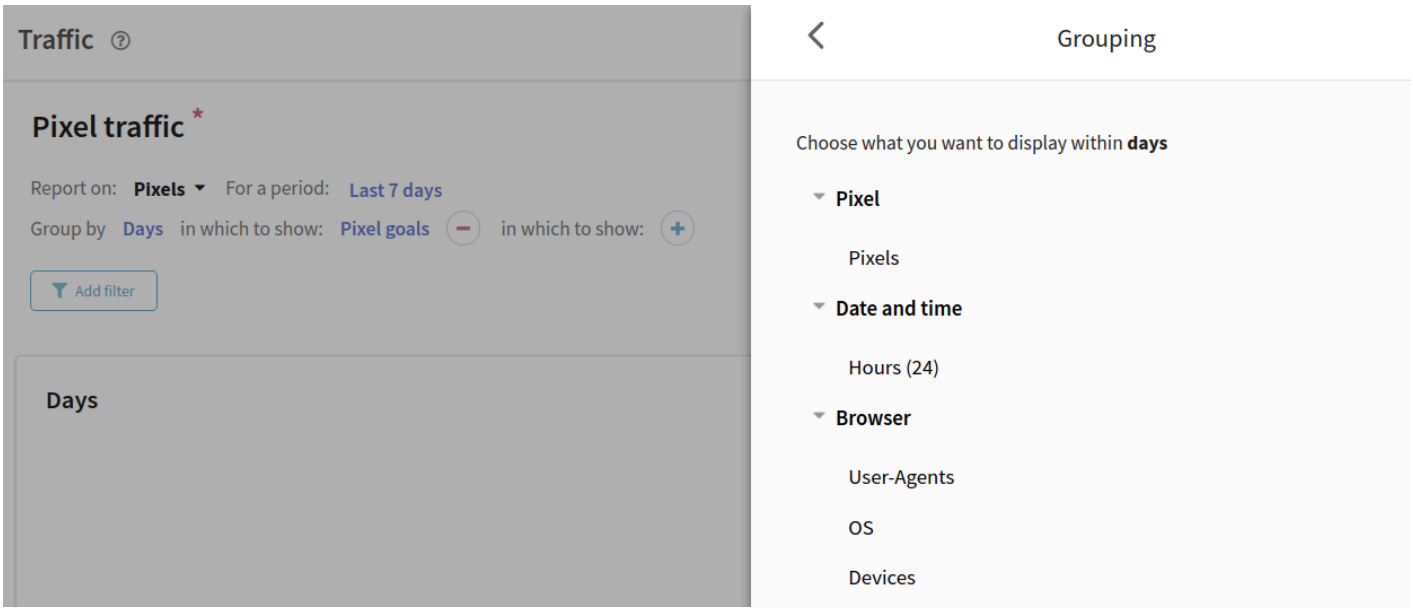
- **Channels. Statistics on subscriptions and unsubscribes, opens and clicks in messages.**
- **Pixels. Statistics on goals that are registered by pixels.**
- **Forms. Statistics onthe number of loads and submissionsof web forms.**
- **Pop-ups. Statistics on pop-ups shows and fills.**

To get data for a specific traffic source, select it in the Report by field.

Next, set a period. You can specify the date range in the calendar manually or choose one of the presets.



Select the attribute by which you want to group the data in the report. After that, you can detail the grouping by selecting the attributes one by one in the “in which to show” field. Grouping is possible up to three levels. You will see the grouping options in the window on the right:



TIP

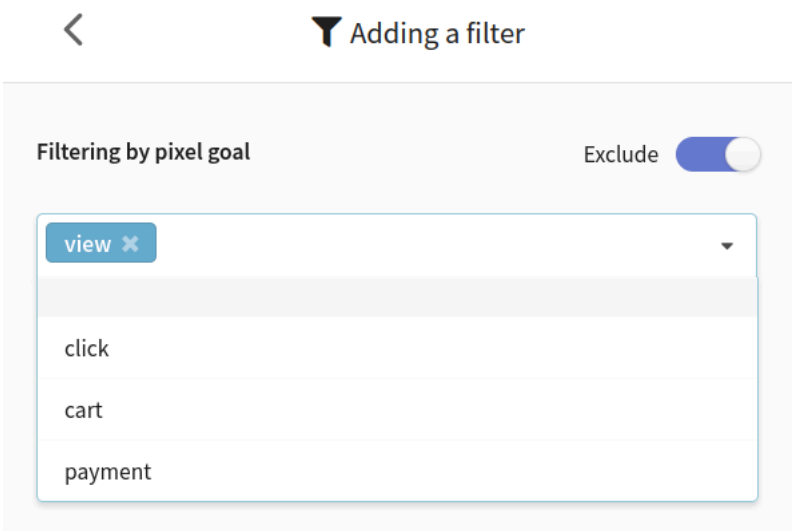
Not all browsers send responses specifying the device, browser, and operating system in the header. Groupings by “Device”, “OS” and “User Agent” contain “Other” for such cases. Also if a header was transmitted, but the system could not read it, then information about the event will be recorded in “Other”.

Adding filters

With the help of filters, you can leave only the necessary information in the report. Use filters if you want to view data by specific forms, goals, resources, etc. Filter categories are the same as grouping categories.

Click Add filter to filter the data. In the window on the right, you can choose which data to include in the report.

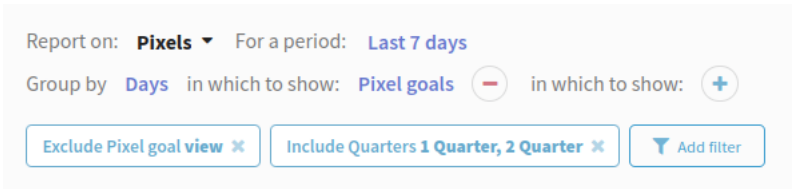
If you need to remove some data from the report, click on the Exclude switch in the upper right corner.



After setting the filter, click Apply at the bottom of the page.

The added filters will be displayed in the report. There is no limit to the number of filters you can use. Click

to remove the filter.



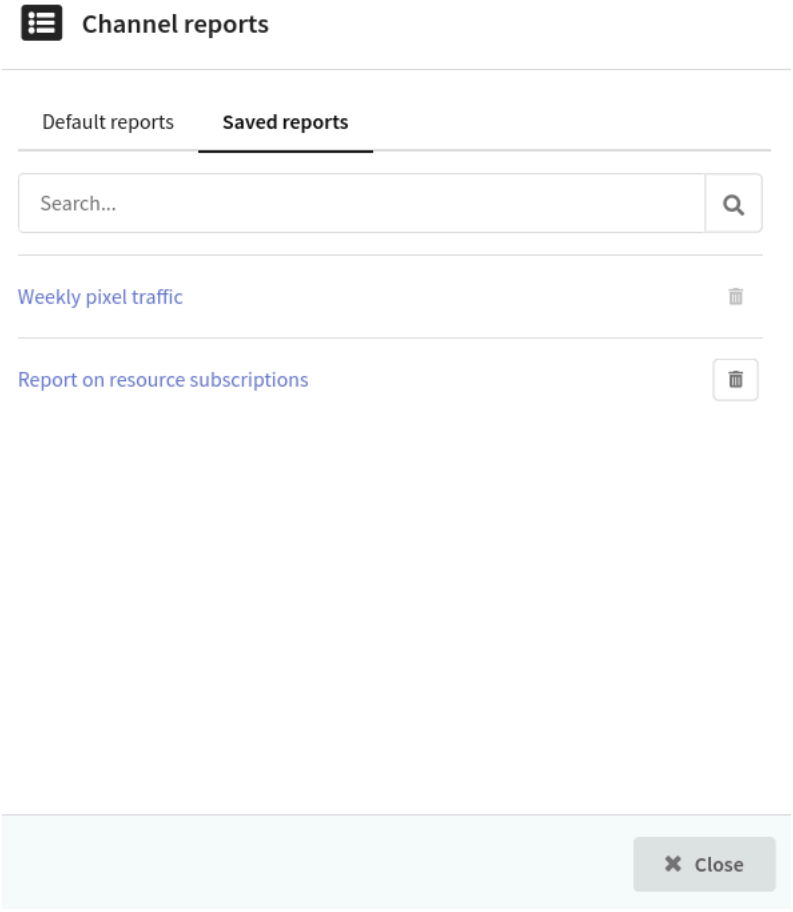
Saving a report

You can save the current report settings. To do this, edit the name of the report (the default is “All Channels”) and click the “Save” button. This is useful if you frequently query data using the same groupings and filters.

The saved report will be available to all users within group visibility.

Click

to open the list of reports. In a new window you will find default (system) and saved reports. Click on the name of the report to view it.



To delete a saved report, click

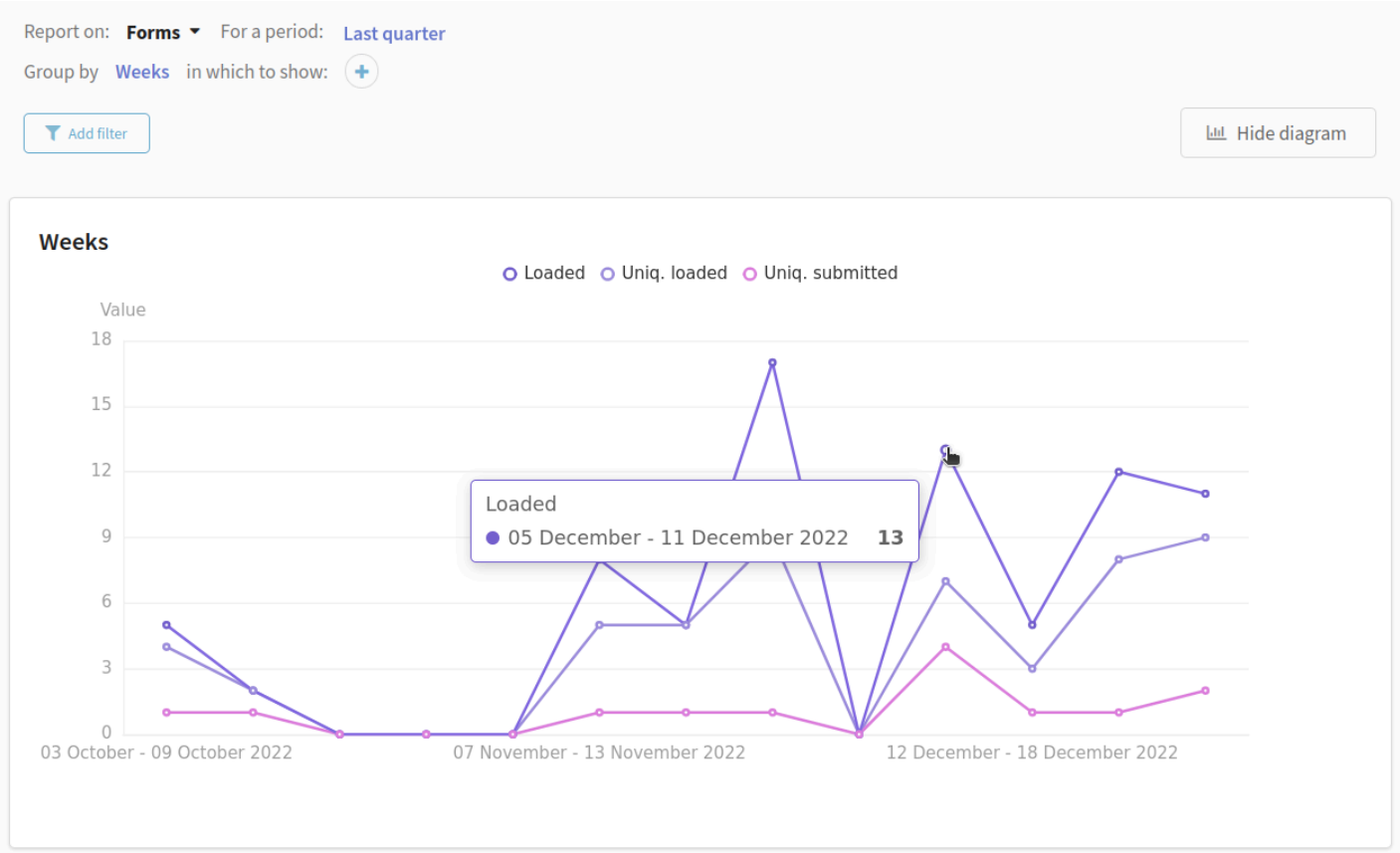
. However, you cannot delete the report that you are currently viewing.

Viewing a report

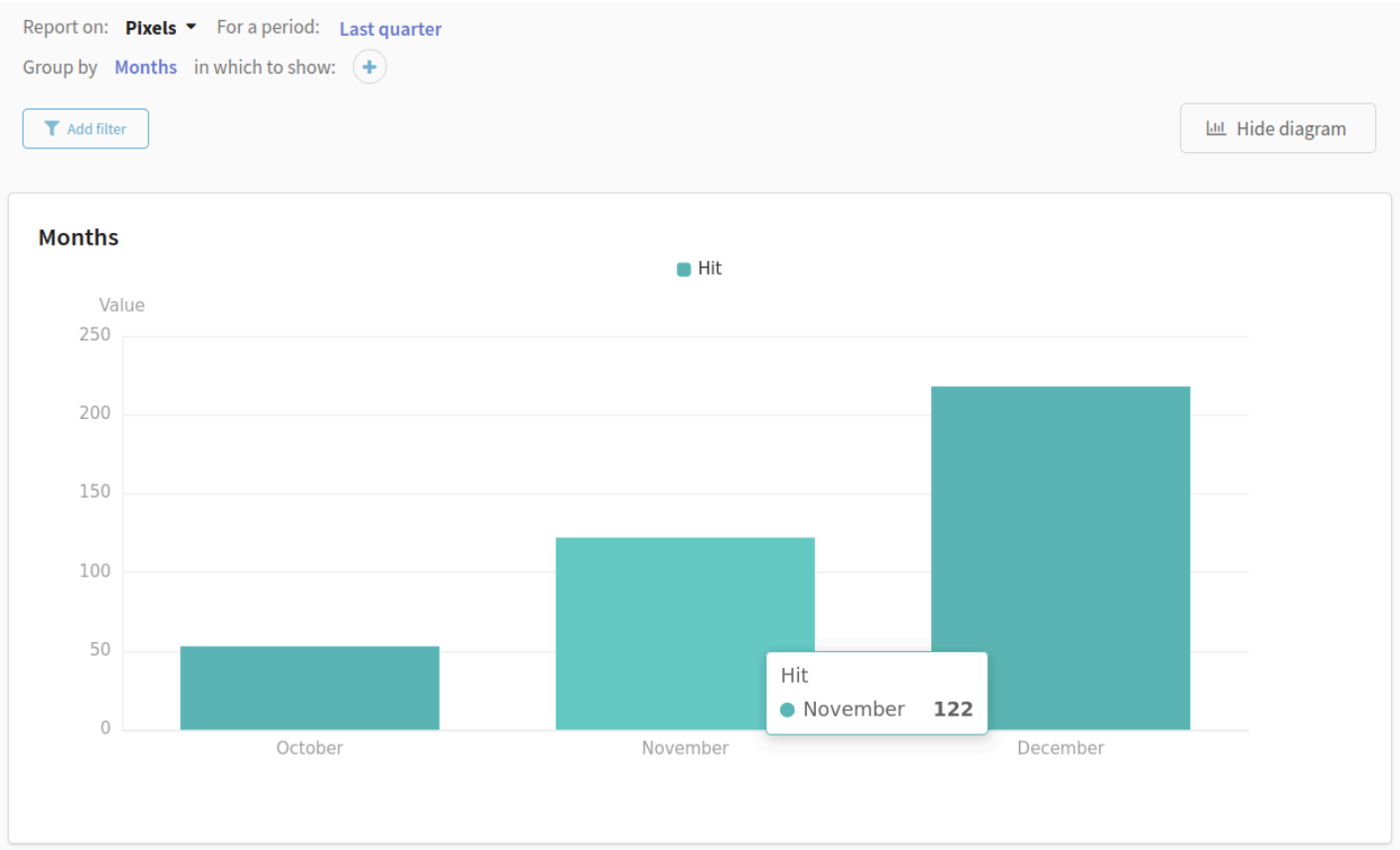
The report includes a table with data and a corresponding diagram. The chart can be collapsed by clicking on the Hide diagram button.

Diagram

The chart displays data based on grouping. When the report is grouped by days, weeks and quarters, the diagram shows the dynamics of changes in the metrics for the selected period:



In other cases, the diagram shows the ratio of quantitative or percentage indicators in the selected period:



When you hover over the chart, you can see the exact figures.

The header contains the metrics displayed on the chart. Any column of the table can be displayed on the diagram, but the maximum number of indicators is 7. Click the metric to hide it.

Data table

The table contains quantitative and percentage indicators by traffic sources. Description of metrics can be found below.

Metrics are grouped based on the report settings. Depending on the details of the grouping, the table structure may have nested elements.

For example, if you set `"Group by days, in which to to show Pixels, in which to show* Pixel goals"` **, the table will display the following data:**

Name	Hit	Uniq. hit	Goal value	Value per hit
Total	16	16	2 800	1 325
2022-10-14	1	1	50	50
2022-10-11	15	15	2 750	1 275
#18 Test pixel	2	2	150	75
#17 Online store pixel	6	6	2 200	1 100
view	2	2	200	100
payment	3	3	1 500	500
card	1	1	500	500
#16 Website pixel	4	4	400	100

To add a new metric to the table, hover over the column name and click

. Select a metric from the list. If you want to remove a column, click

.

To move a column to another part of the table, you need to hover over the column name, hold down the left mouse button and drag the column to the desired position.

Name	Send	Delivered	Delivery rate	Uniq. content clicked	Content CTR
Total	8 303 211	7 287 553	87,77 %	2 150 570	29,51 %
2022-11-11	3 270	746	22,81 %	368	49,33 %

Other options:

— switch quantitative metrics to percentages;

— display a column on a chart.

Comparing data

The comparison mode allows you to see data on several objects at the same time. For example, you can compare the performance of two templates or evaluate the change in mailing metrics from period to period.

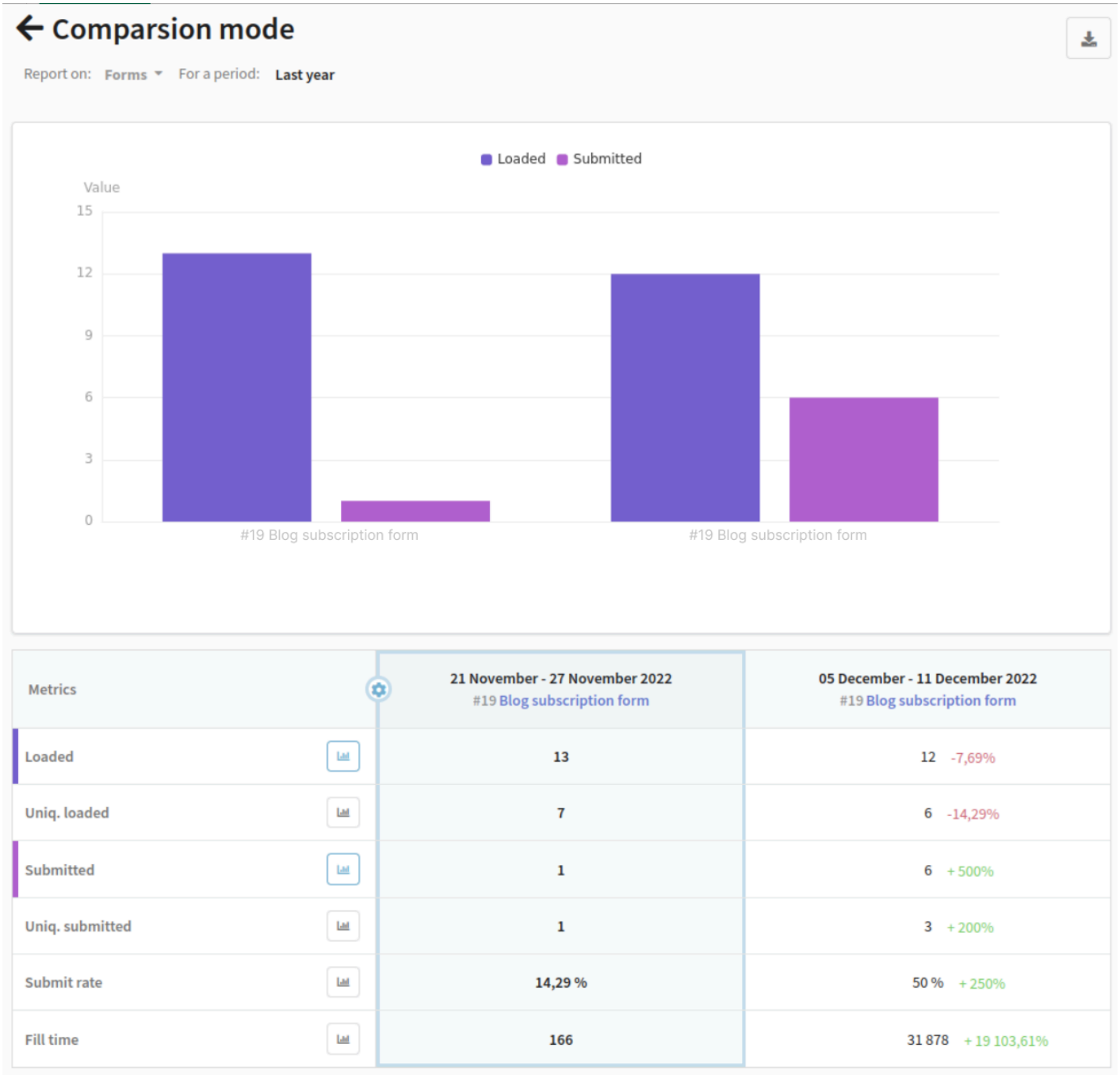
To switch to the comparison mode, select the objects in the table that you want to compare. Click “Compare”. It is possible to compare objects at different nesting levels.

Name	Compare	Loaded	Uniq. loaded	Submitted	Uniq. submitted	Submit rate
Total		265	155	42	36	23,23 %
26 December - 01 January 2022		11	9	2	2	22,22 %
19 December - 25 December 2022		12	8	2	1	12,5 %
12 December - 18 December 2022		5	3	1	1	33,33 %
05 December - 11 December 2022		13	7	7	4	57,14 %
#23 Webinar registration		1	1	1	1	100 %
#19 Blog subscription form		12	6	6	3	50 %
21 November - 27 November 2022		17	9	1	1	11,11 %
#23 Webinar registration		4	2	0	0	0 %
#19 Blog subscription form		13	7	1	1	14,29 %

The comparison mode also contains a chart and a table with metrics. In the diagram, the metrics for each object are grouped together.

The first object in the table is the comparison column. Other objects are compared with this object. The percentage increase or decrease of the metric is calculated relative to this object. To change the comparison column, click

and select a different object.



Metrics	21 November - 27 November 2022 #19 Blog subscription form	05 December - 11 December 2022 #19 Blog subscription form
Loaded	13	12 -7,69%
Uniq. loaded	7	6 -14,29%
Submitted	1	6 + 500%
Uniq. submitted	1	3 + 200%
Submit rate	14,29 %	50 % + 250%
Fill time	166	31 878 + 19 103,61%

To go back to the general mode, click ← Comparison mode.

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Report metrics

Report by	Metric	Description
Channels	**Opened**	**The number of messages opened by profiles.**
	Uniq. opened	**The number of unique message opens.** **If the same profile opens a message multiple times, only one unique open is counted.**
	Unsubscribed	**The number of unsubscribes from resources in all channels.**
	Subscribed	**The number of subscriptions to resources in all channels.**
	Confirmed	**The number of clicks on the confirmation link (link with the `confirm-link` attribute).**
	Hit	**The number of clicks on a link within a message.**
	Uniq. hit	**The number of unique clicks on a link in messages.** **If the same profile opens a message multiple times, only one unique hit is counted.**

Report by	Metric	Description
Pixels	**Hit**	**The number of goals registered by pixels.**
	Uniq. hit	**The number of unique goal registrations.** **If a target is registered multiple times for the same profile, only one unique hit is recorded.**
	Goal value	**The sum of the values of all goals registered by the pixels.**
	Value per hit	**(Goal value / Hit) * 100%**
Forms	**Loaded**	**The number of times the form page was loaded.**
	Uniq. loaded	**The number of times the form page was loaded by a profile.** **If the same profile loads the form multiple times, only one unique load is logged.**
	Submitted	**The number of successfully submitted forms.**
	Uniq. submitted	**The number of unique submissions.** **If one profile submits the form multiple times, one unique posting is recorded.**
	Submit rate	**(Uniq. submitted / Uniq. loaded) * 100%**
	Fill time	**Duration of interaction with forms (in seconds). The time spent by each profile filling out the form is summed up.**
Pop-ups	**Hits to targeting**	**The number of times the conditions for the popup to appear, specified in the trigger, were met. However, the popup might not be shown to the user due to tag limits specified in the "Activation settings" and "Limits" blocks.**
	Shows	**The number of times the popup has been shown to users.**
	Uniq. shows	**If a popup is shown to a user multiple times within the same session, only one unique show is recorded.**
	Uniq. CTA	**The number of clicks as long as there are no fills, or the number of fills if there have been any.**
	Closed	**The number of clicks on the close button (closeButton).**
	Uniq. closed	**If a user closes a popup multiple times within the same session, one unique closure is recorded.**

Cohorts report

Description

A cohort in Altcraft analytics is a group of subscriber profiles from a specific database imported at the same time period. This type of panel data empowers you with over time customer behavior statistics. You can use it to define how different groups of subscribers - from different lead providers for instance - interact with your marketing services and thus measure the real impact of lead sources.

> TIP
>
> **You can predict efficiency of future or recent marketing actions comparing cohorts and actions dates.**

Subscribers are divided into groups, which behavior is analyzed, in this report.

Comparing different subscriber cohorts you can retrieve actionable data on your lead engagement mailings to predict future impact and improve your customer engagement strategy.

Another option is to analyze activity rate of customer profiles from one database, imported at different time ranges, to get insights on how minor marketing changes affect over time profile activity - especially changes that have no instant impact but affect longitudinal statistics.

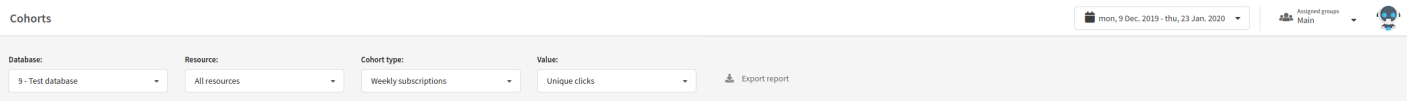
Managing cohorts

Diagram

> TIP
>
> **Start working with cohorts from selecting time period, database, resource, cohort type, and its value.**

Cohorts report can track changes in subscribers groups by next values:

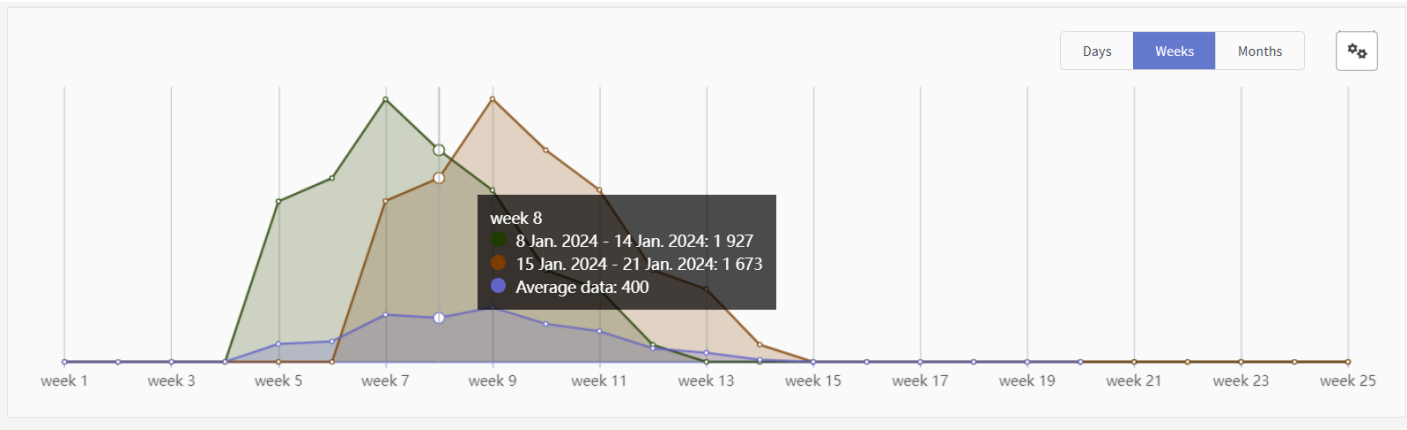
- **Unique clicks**
- **Unique opens**
- **Unique clicks to unique opens**
- **Unique opens to sends**
- **Unique clicks to sends**



Cohort consists of subscribers within certain period of time (week or month).

Cohorts in diagram can be grouped by days, weeks or months.

When you place cursor on the diagram, list of appropriate information will appear.



> TIP
>
> **Cohorts report can be downloaded in XLS or CSV formats.**

Table

The following information is displayed in the table: amount of actions selected in Value menu, and average result.

Cohorts / Intervals	week 5	week 6	week 7	week 8
1 Jan. 2024 - 7 Jan. 2024	0	0	0	0
8 Jan. 2024 - 14 Jan. 2024	1 461	1 672	2 390	1 927
15 Jan. 2024 - 21 Jan. 2024	0	0	1 464	1 673
22 Jan. 2024 - 28 Jan. 2024	3	1	3	0
29 Jan. 2024 - 4 Feb. 2024	0	0	0	0

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Lifetime report

Description

Customer lifetime report is an accurate tool to analyze customer cohorts lifetime. Lifetime here indicates client message activities like opens or link clicks.

Graph columns represent average subscriber activity time for the specified subscribers cohort.

- **A maximum clicks per day amount for a cohort is retrieved.**
- **Then all other days results are divided by this maximum number. Thus relative value for each day is retrieved.**
- **Then the relative results are summed to form an average customer lifetime, measured in days.**
- **If a day's result is less than 5% of a cohort maximum it will be skipped.**

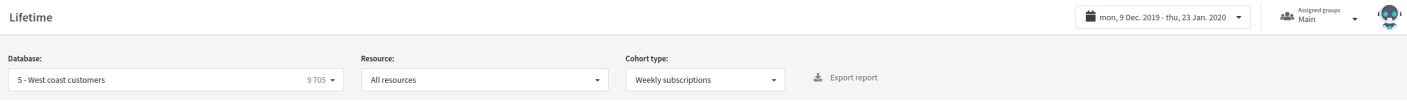
Cohort activity values in lifetime report table represent a relation between a specific time period and the period of maximum activity.

General settings

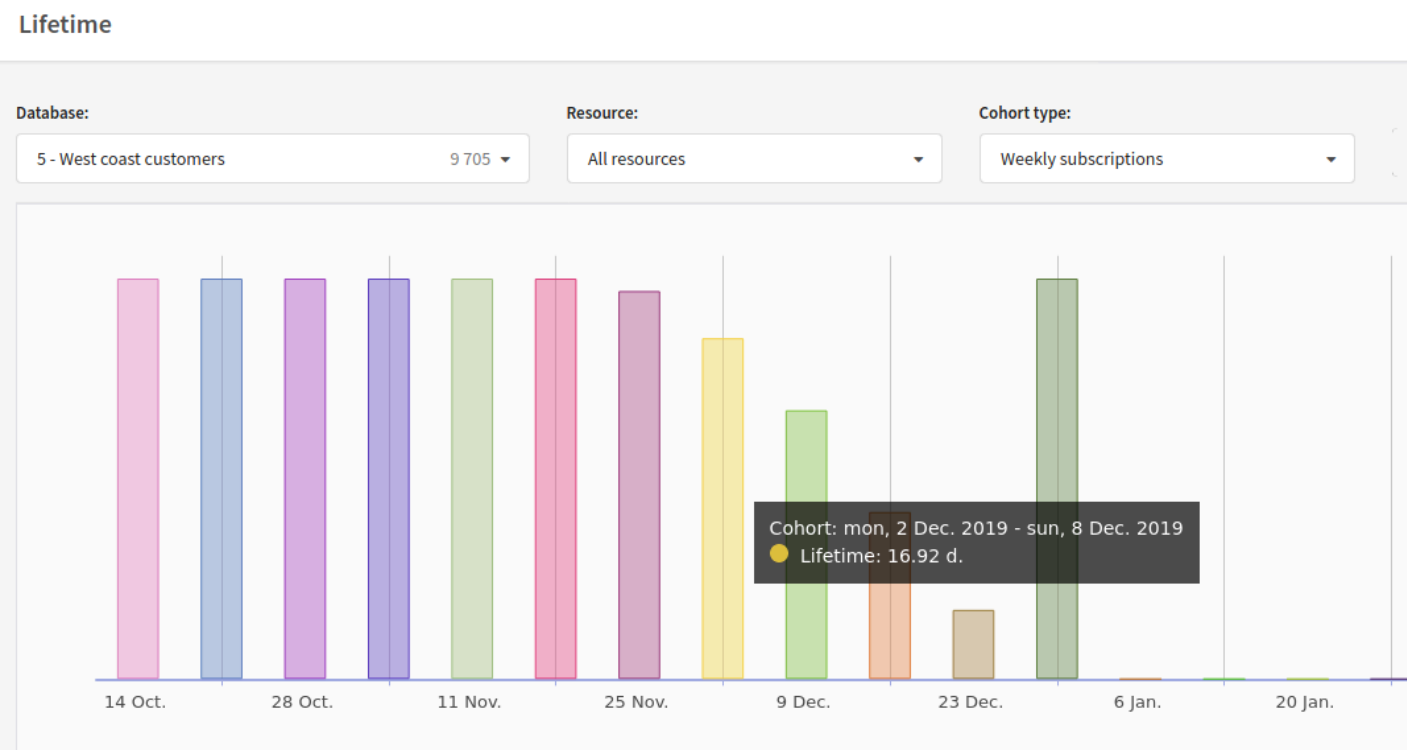
To calculate certain cohort lifetime, select database first.

After this, you can select resource. Also, you can select no resource, and lifetime report will be counted for all resources.

You can also select cohort size - monthly or weekly.



In cohorts columns, you can see sequence of decreasing clicks amount by each week and decreasing interest to sent messages.



INFO

To the right of the settings is the Export report button, with which you can download data in XLSX or CSV format.

Table

You can see ratio of clicks amount in one cohort to general amount of subscribers in cohort in the table.

Cohorts / Intervals	week 1	week 2	week 3	week 4	week 5	week 6
9 Dec. 2019 - 15 Dec. 2019	39.17%	100.00%	84.73%	0%	0%	0%
16 Dec. 2019 - 22 Dec. 2019	39.17%	100.00%	0%	0%	0%	0%
23 Dec. 2019 - 29 Dec. 2019	100.00%	0%	0%	0%	0%	0%
30 Dec. 2019 - 5 Jan. 2020	39.17%	100.00%	84.73%	36.28%	0%	0%
6 Jan. 2020 - 12 Jan. 2020	0%	0%	0%	0%	0%	0%
13 Jan. 2020 - 19 Jan. 2020	0%	0%	0%	0%	0%	0%

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

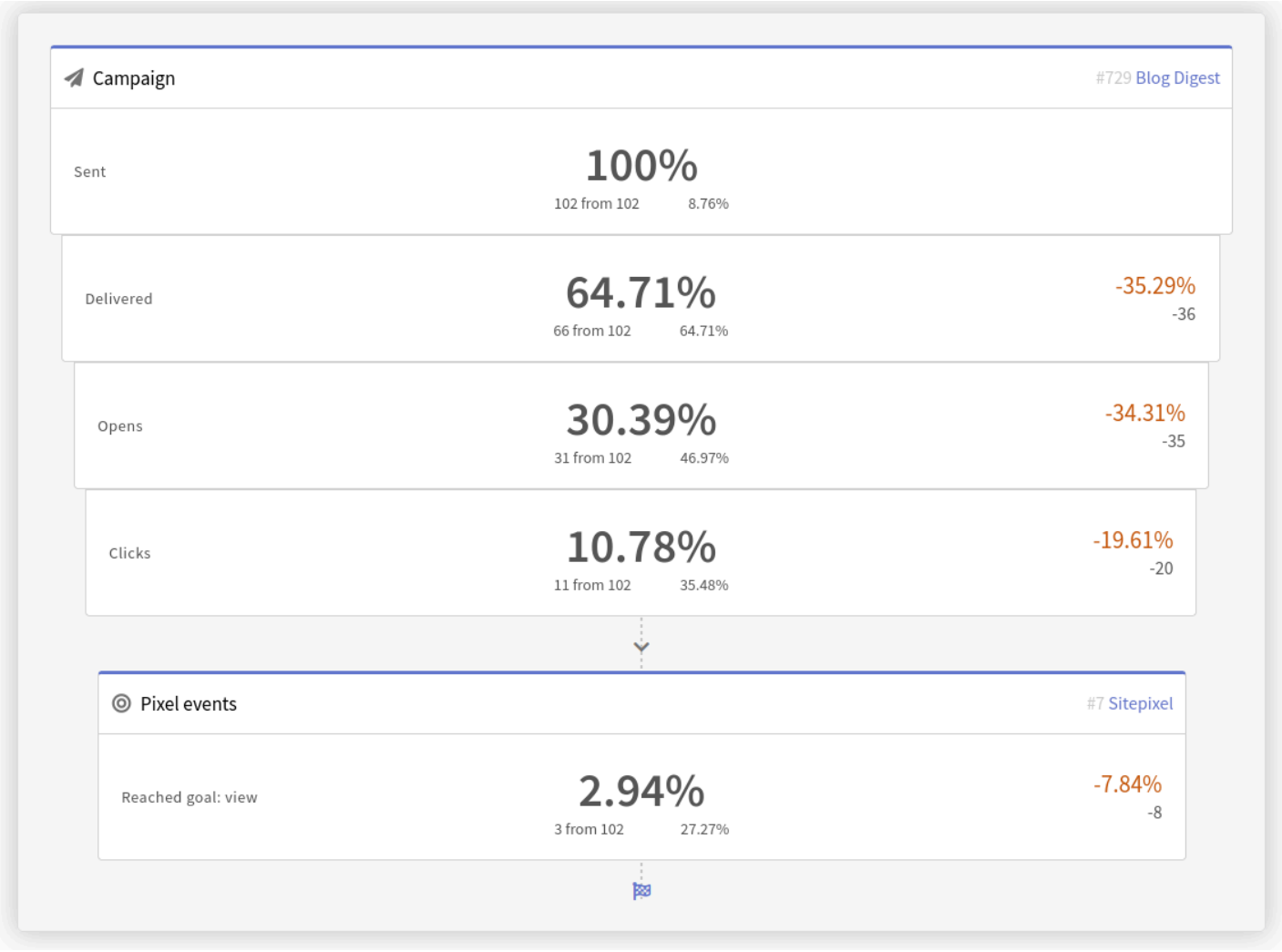
Funnels report

Description

Conversion funnel is a term that describes customers quantity changes as they go through the whole transformation process from website visitors to buyers or subscribers.

From the start number, only some part of the audience will go to registration form, and less of them will become real buyers. Graphic display of this process is similar to the funnel.

Using a variety of available conversion funnel steps you won't miss a part of your customers' journey in need of improvement. You can also detect steps where conversion rate is high and perform appropriate actions.



What can be conversion funnel used for?

Conversion funnels can become your trusted providers of actionable marketing analytics. Each funnel step shows a a number of profiles engaged in a certain activity (profile import, message reads and link clicks, web page visits or purchases) in relation to the previous step. Analyzing conversion funnel can help you to understand reasons for unsubscribing and test key elements of the previous page.Using a variety of available conversion funnel steps you won't miss a part of your customers' journey in need of improvement. You can as well detect steps where conversion rate is high and act accordingly and precisely upon this data.

Calculation of the conversion period

Since the funnel report collects data from different sources, the calculation can take a long time in some cases. To optimize the process of working with the report, the date periods for it are calculated and stored in the platform.

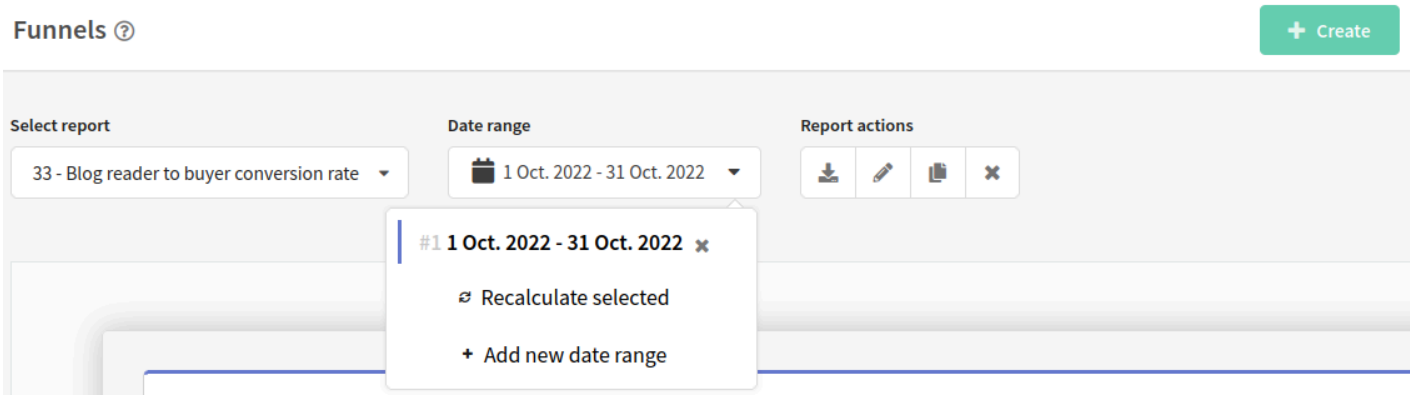
In the menu, select a ready-made report and the time period you are interested in. You can also add a new reportdate range or recalculate the current one.

The following buttons can be found on the right side:

****Export report** —**

Edit report — modify the report;

Clone report — duplicate the report



How to create a funnel?

To add a new conversion report, click Create in the top right corner of the page.

General settings

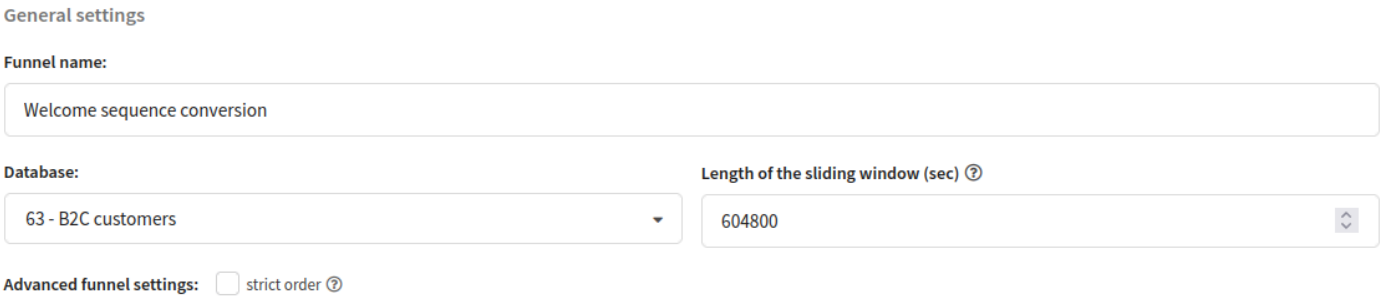
Enter the name and description of the new funnel, and select the database that stores the data of the customer audience you are interested in.

Next, specify the length of the sliding window. This is the time period between the first and last conditions in the funnel. The period is specified in seconds and defaults to 604800 seconds (7 days).

For example, you leave the default length of the sliding window (604800 seconds) and add the following steps to the funnel: *Template #1 delivered —> Template #1 opened —> Achievement of the "purchase" goal*. In this case, the first condition will be met when the message is delivered to the profile, and the last condition will be met when the profile activates the pixel by his actions. The profile will get to the last step of the funnel only if no more than 7 days pass from the moment the message is delivered to the moment the pixel is activated. At the same time, the order of actions specified in the funnel should be preserved.

TIP

30 days — 2592000 seconds
14 days — 1209600 seconds
1 day — 86400 seconds
1 hour — 3600 seconds



Additionally, you can enable strict order in the funnel. This setting does not allow other events to intervene in the given sequence.

For example, you are building a funnel: *Mailing #1 delivered → Mailing #1 opened → Achievement of the "purchase" goal.* If between the second and third steps the platform recorded another event in the profile history, the profile will not proceed to the third step.

Use a strict order if it is important to you that the profile perform a strict sequence of actions in a short period of time.

Building a funnel

The following blocks can be used in custom order in funnels:

- **Information by segment**
- **Information by template actions**
- **Information by mailing actions**
- **Information by goals reached**
- **Information by subscriptions**

CAUTION

The funnel counts the number of profiles, not the number of events. For example, if the platform recorded two profile clicks on a template, the funnel will only show one profile at that step.

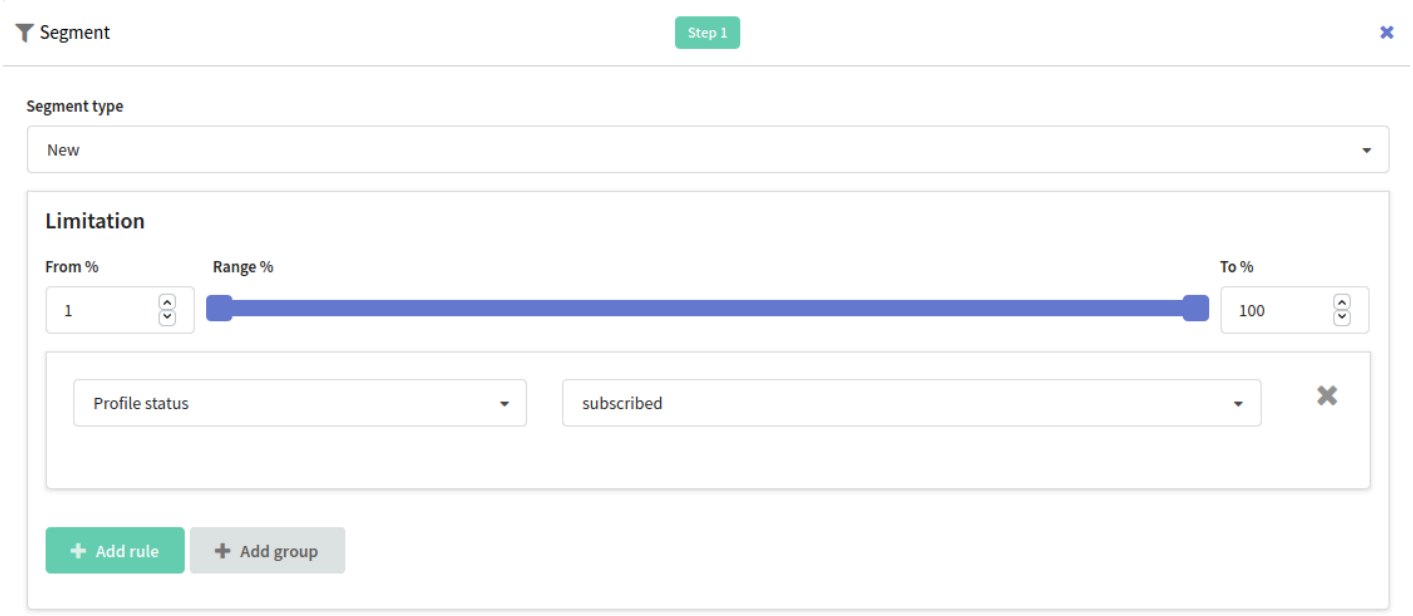
Add a step: segment

A segment as a funnel condition can only be added in the first step. This allows you to narrow the selection of profiles observed in the funnel.

To get information about segment, you need to select segment type (new or existing) —> select amount of profiles in the selection, which can be fixed number or range —> add rule or rules group:

- **"AND" condition — subscribers that fit all rules and rules groups in query, will be selected.**
- **"OR" condition — subscribers that fit at least one rule or rules group, will be selected.**

—> find necessary segment. After this, you can add one more rule or rules group, or go to the next step.



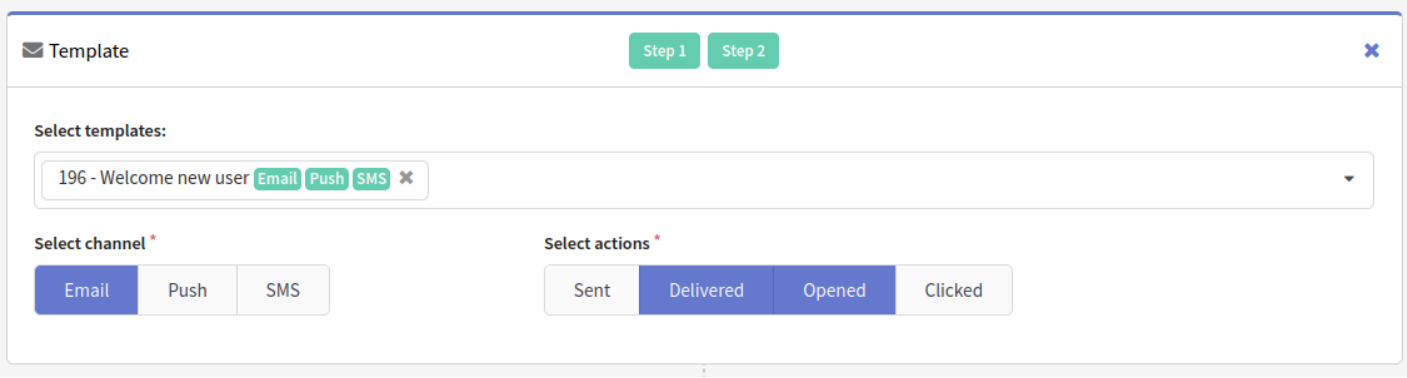
CAUTION

“Segment” step does not use selected time period for selection. It works only as filter.

Add a step: template

This block shows amount of profiles which performed selected actions in selected actions.

To get information about actions in the template, select “Add a step: template” from the options —> select the template —> select communication channel or channels (email; push; sms) —> select action or actions in message, that you want to get information about. If you select several actions, amount of steps will be increased.



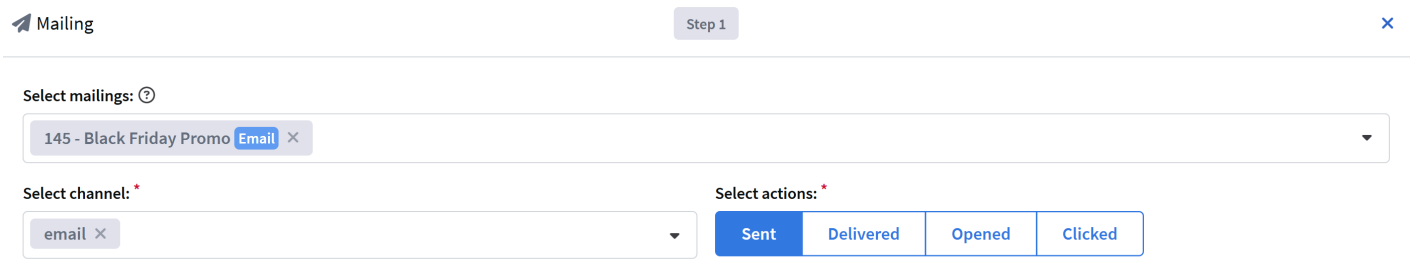
CAUTION

If a click event is registered in the SMS channel but there is no delivery event, the delivery event will be automatically generated — 100 milliseconds before the click event.

Add a step: mailings

This blocks allows you to count profiles by amount of certain actions by selected channels in the mailing.

To get information by mailings, select the mailing, then select action or actions in message, that you want to get information about. If you select several actions, amount of steps will be increased.



CAUTION

If a click event is registered in the SMS channel but there is no delivery event, the delivery event will be automatically generated — 100 milliseconds before the click event.

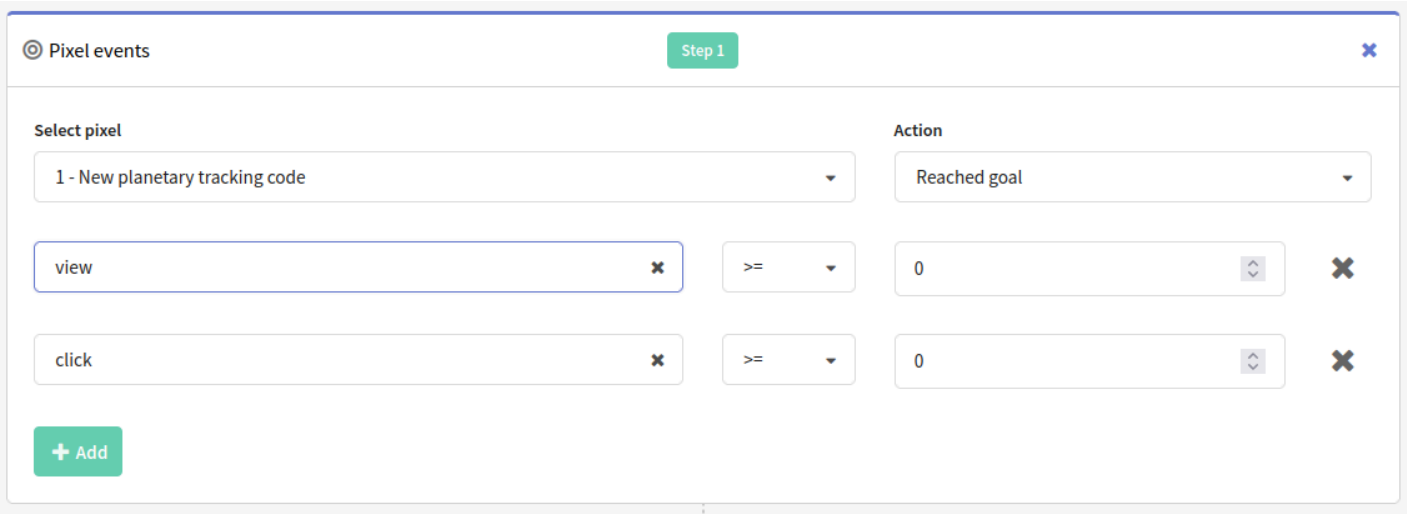
Add a step: goal reached

This block shows amount of subscribers that reached pixel goal or performed target actions. Available events:

- **pixel visit — any event of a selected pixel;**
- **reached goal — an event that a client achieves one or more goals.**

To get information about the number of subscriber who reached a certain goal, you need to select a pixel for a certain event, select a goal and its value. Next, specify the pixel value that must be fixed for the profile to get to this funnel step. The following comparison operators are available:

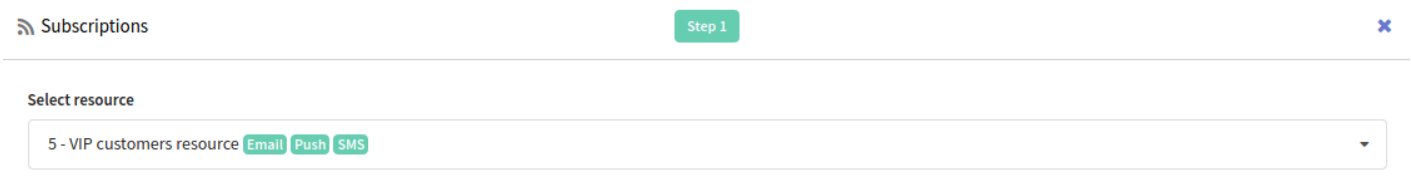
- **more or equal**
- **less or equal**
- **more**
- **less**
- **equals**



Add a step: subscriptions

This block shows information about profiles who subscribed to any channel of a certain resource. Information about subscriptions gives subscriptions quantity.

To get information about completed subscriptions amount, select the resource.



INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Goals report

Description

Goals report are a report of how your clients have completed targeted actions and the value of taking them. In other words, goals report displays amount of pixels activated by website visitors depending on goals defined.

Goals can be attached to customer profiles using such Altcraft tools as:

- **Tracking pixel scripts hosted on site pages and other resources**
- **Node “Goal registration” in scenario workflow**
- **Goal Registration API**

> INFO
>
> **The report automatically adds up the value of actions with positive and negative values and displays it as a “Profit” indicator.**

Managing report

At the top of the page, select the period for generating the report.

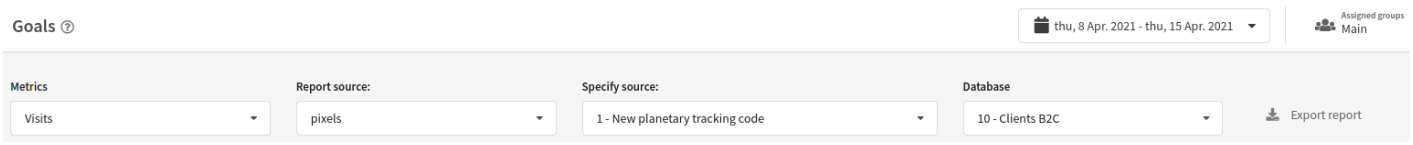
In goals report, you can select Report source and Metrics.

There are two available metrics:

- **visits — how many goal actions the user performed;**
- **profit — the sum of the action value.**

The report sources:

- **pixels — analytics for pixels for which goals are registered;**
- **goals — analytics for goals registered for different pixels.**



You can specify the source: select a pixel or goal to analyze. When choosing a specific pixel or goal, you can also specify Profiles Database — for well-known clients:

> INFO
>
> **You can download the resulting report in MS Excel and CSV format by clicking the Export report button.**

Profit dynamics and goals achievement

The goal report graph shows the total number of visits as well as the total profit from the actions performed.

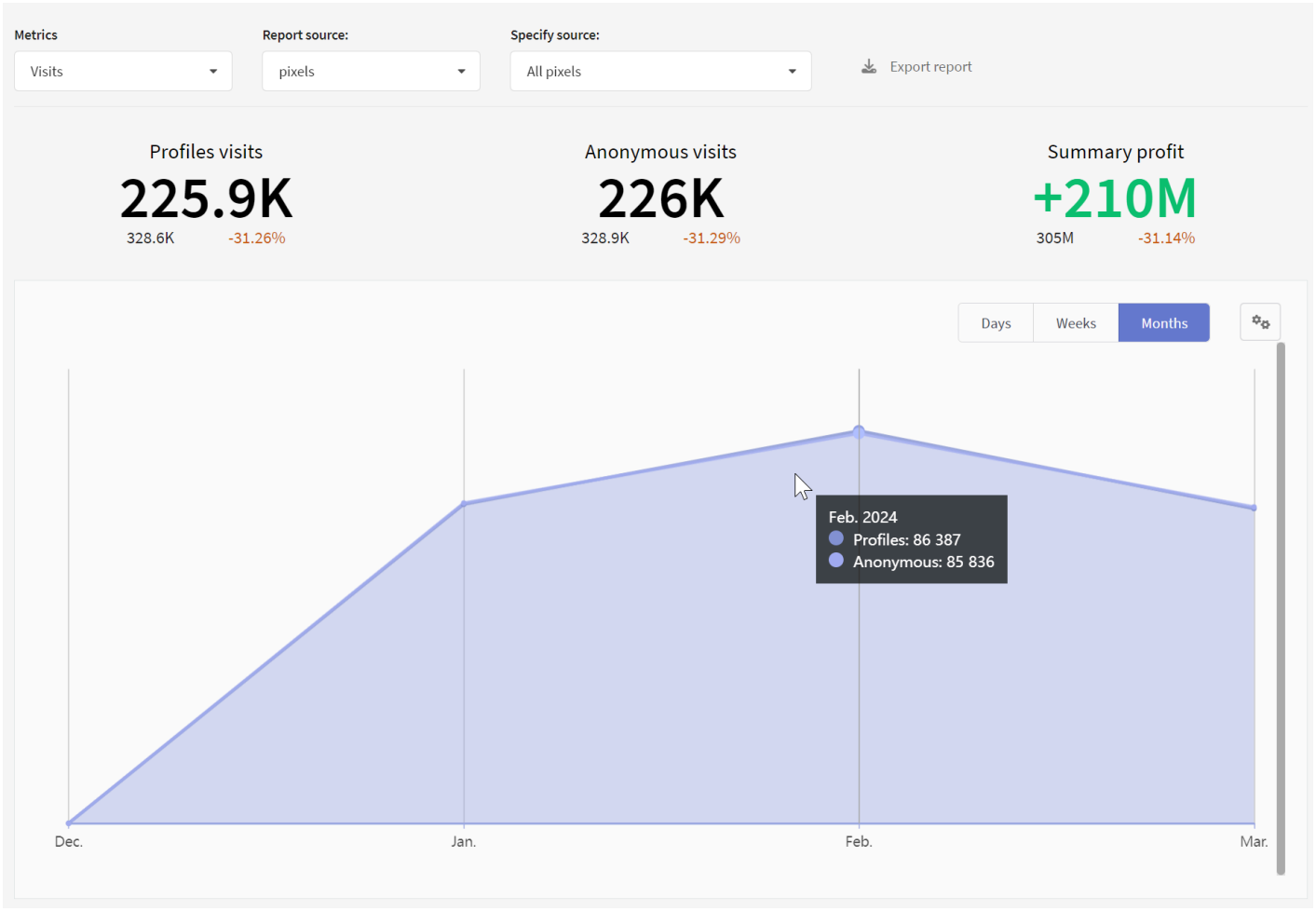
Profile visits give information about pixel activation by subscribers from databases. Anonymous visits give information about customers not from databases.

> INFO
>
> **Goals can be registered for clients whose contact and personal information you do not yet have. Such events are called anonymous. Altcraft distinguishes them from each other based on connection data (cookies). If a client later registers using the same browser, their data is added to the already existing profile.**

Using pixel, you can register certain income or another value at the moment of page visit or completing a goal. Summary number of anonymous and profiles visits is summary profit.

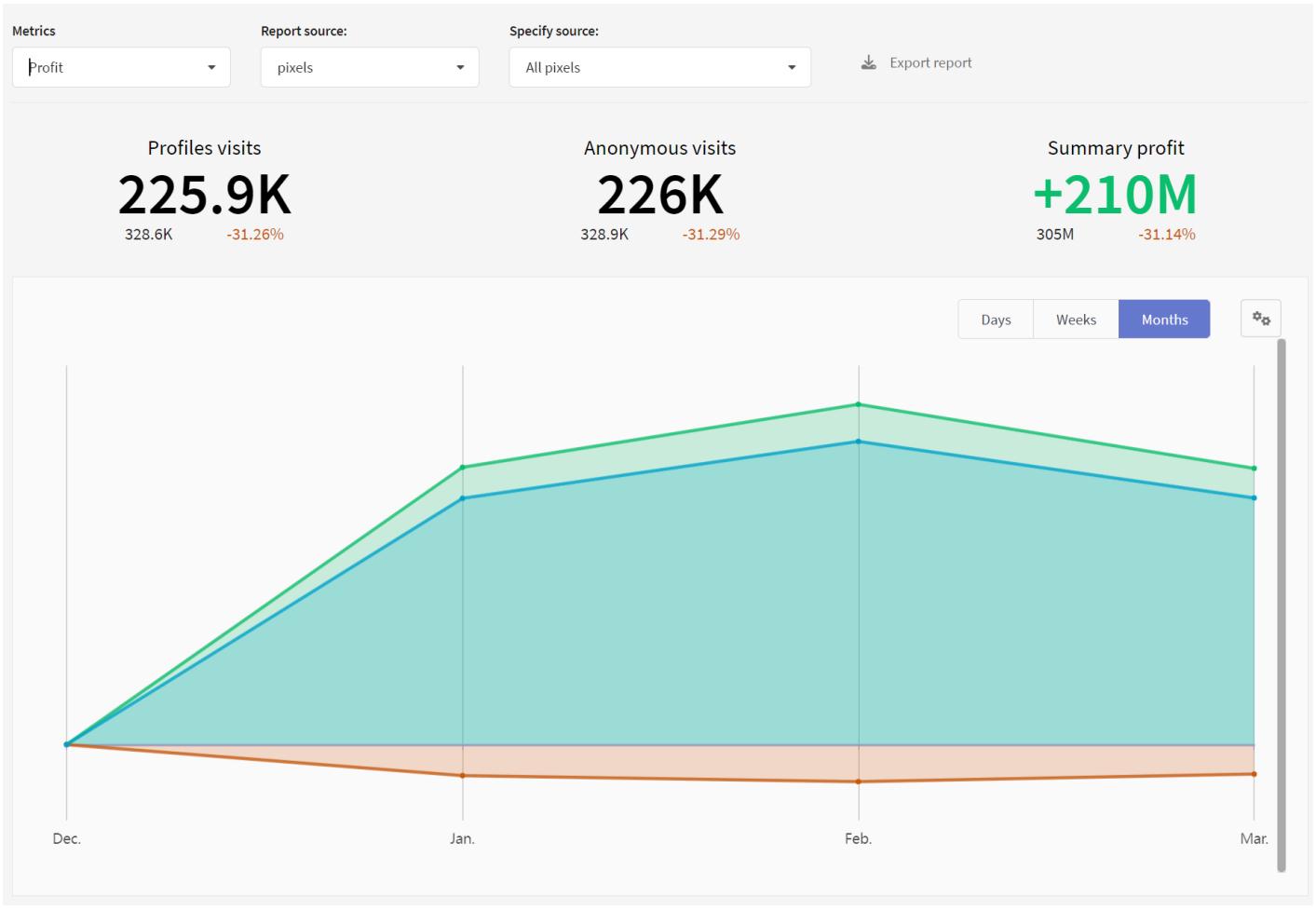
Visits metric

If you select Visits metric, on the graph you can see such information as the number of profiles visits and the number of anonymous visits.



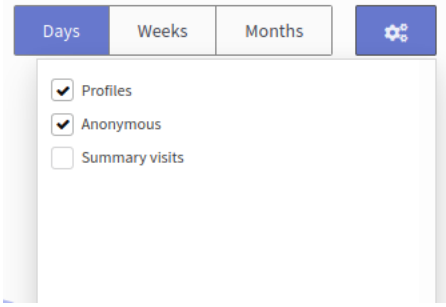
Profit Metric

If you select Profit metric, on the graph you can see such information as summary income, summary expenses and summary profit.



The graph is built by days, weeks or months of the report period. Using the button

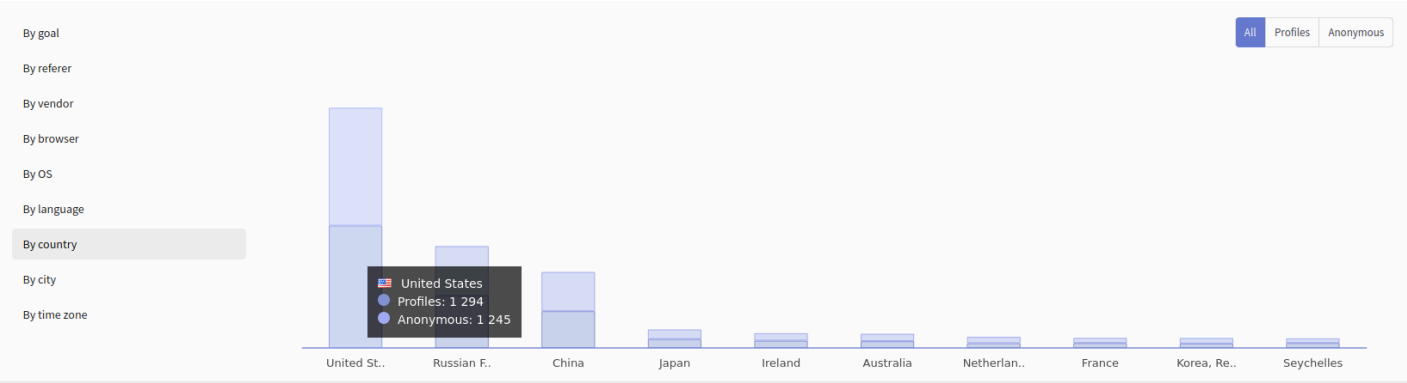
, you can remove unnecessary indicators from the chart or, conversely, add them.



Analytical report slices

When registering a goal action, Altcraft collects information about the device used by the client, as well as about the network connection and referer. Based on this data, you can additionally group the resulting audience:

- **By goal**
- **By referer**
- **By vendor**
- **By browser**
- **By OS**
- **By language**
- **By country**
- **By city**
- **By time zone**



Report indicators

The table at the bottom of the page contains a summary of the report based on the settings, slicers, and metrics you selected. It is this data that will be uploaded during export. Click

to add columns to the table or hide unnecessary ones:

Profiles	Anonymous	Profiles income	Anonymous income	Profiles expenses	Anonymous expenses	Profiles profit	Anonymous profit	Summary income
68	74	11 411.2	13 017.5	-7 801.9	-3 447.5	3 609.3	9 570	24 428.7
26	24	4 404.2	3 604.3	-3 400.7	-1 223.5	1 003.5	2 380.8	8 008.5
21	28	3 803.4	2 605.3	-1 600.3	-2 201.1	2 203.1	404.2	6 408.7
35	27	4 003	5 206.6	-2 801.4	-1 600.3	1 201.6	3 606.3	9 209.6
27	33	2 805.4	9 005.5	-3 046.3	-1 600.8	-240.9	7 404.7	11 810.9
346	367	66 658.3	56 667.3	-26 699.4	-29 770.1	39 958.9	26 897.2	123 325.6
39	40	6 607.8	5 206.1	-2 224	-3 601.3	4 383.8	1 604.8	11 813.9
30	34	5 604.8	5 006	-1 400.7	-2 600.8	4 204.1	2 405.2	10 610.8
31	40	4 003.5	6 607.8	-2 424.1	-3 401.2	1 579.4	3 206.6	10 611.3
15	32	3 600.8	4 205.1	-1 000.5	-1 600.8	2 600.3	2 604.3	7 805.9
2 779	**2 837**	**476 868.2**	**450 325.4**	**-233 764**	**-227 302.1**	**243 104.2**	**223 023.3**	**927 193.6**

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Audience growth report

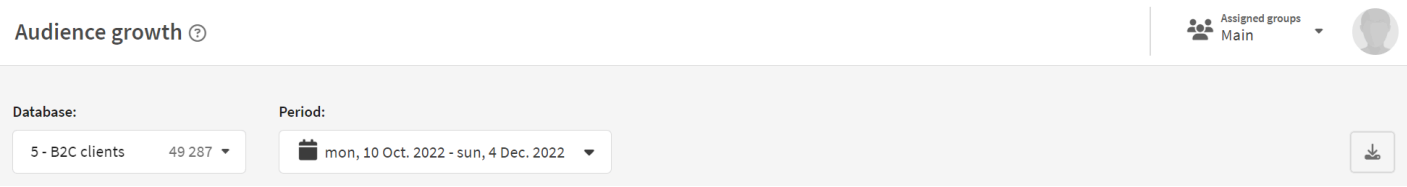
Description

The audience growth report allows you to track the increase in the number of profiles in your database. This report only records events of new subscriber profile additions and does not account for unsubscribes or deletions.

Report settings

At the top of the page, select the period for generating the report.

By default, the report is generated for all available profile databases. You can use the dropdown menu to select a specific client database:

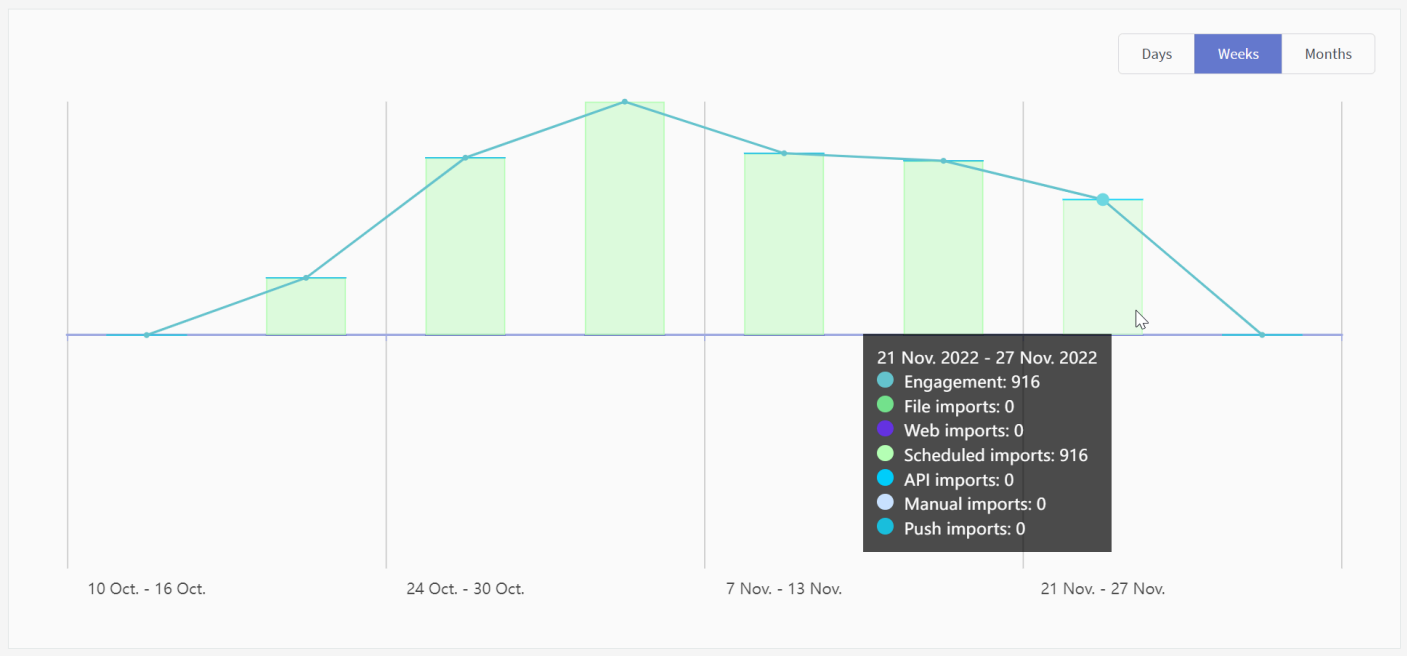


TIP

You can download report in MS Excel or CSV formats.

Growth graph

The visual part of the report displays the dynamics of audience growth. The data can be presented in the form of charts, grouped by days, weeks, or months of the selected period.



Audience dynamics metrics

Audience growth is represented by the following import metrics:

- **Engagements — net increase in the number of profiles in the database (excluding unsubscribes or other factors that reduce the number of profiles).**
- **File imports — adding multiple profiles from a table file.**
- **Web imports — creating new profiles at the moment of subscribing through the form or pop-up published on the website.**
- **Scheduled imports — creating of new profiles on a schedule from an external SQL database.**
- **API imports — creating new profiles via API.**
- **Manual import — creating profiles through the platform interface.**
- **Push imports — creating anonymous and personal profiles at the moment of subscribing to push channel.**

Date	Engagement	File imports	Web imports	Scheduled imports	API imports
28 Nov. 2022 - 4 Dec. 2022	+1	0	1	0	0
21 Nov. 2022 - 27 Nov. 2022	+916	0	0	916	0
14 Nov. 2022 - 20 Nov. 2022	+1 178	0	0	1 178	0
7 Nov. 2022 - 13 Nov. 2022	+1 229	0	0	1 229	0
31 Oct. 2022 - 6 Nov. 2022	+1 578	0	0	1 578	0
24 Oct. 2022 - 30 Oct. 2022	+1 199	0	0	1 199	0
17 Oct. 2022 - 23 Oct. 2022	+387	0	0	387	0
10 Oct. 2022 - 16 Oct. 2022	0	0	0	0	0
Totals:	6 488	0	1	6 487	0

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

TIP

Audience growth report displays the increase in the number of profiles in the database. If you need to track changes in the number of subscriptions to a resource, use Traffic report and apply the filter.

Click map report

Description

Click map is a report on your customers' clicks on links in a sent Email message. The click map displays only unique clicks on all links of an Email message, including unsubscribe links. The click map helps to determine the quality of the template content and identify the most successful location and design of links.

INFO

Click map is available only for Email messages. This feature is not supported for other sending channels (SMS, Push).

TIP

It should be noted that the report is based on a template, not on mailings. If the template was used in several mailings, the data will be summarized.

Statistics Collection Features

When analyzing click map data, consider:

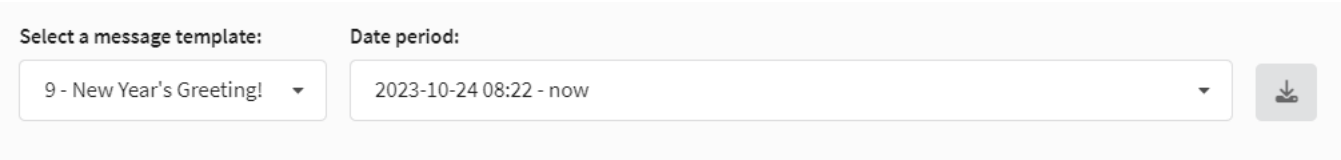
- **Total unique clicks (displayed at the top) - unique clicks across the entire message for the selected period**
- **Clicks on individual links (in the table) - unique clicks for each specific link**
- **Click uniqueness is determined within 24 hours - repeated clicks from the same user within 24 hours are not counted**
- **All clicks are counted, including unsubscribe link clicks that may have been performed through email client buttons**
- **Dynamic links - for personalized URLs, statistics are collected by the final domain, not by the original link template**

CAUTION

Numbers in overall statistics and detailed table may differ for several reasons: different uniqueness calculation periods, counting all types of clicks (including unsubscribe links), dynamic link processing features, and platform data processing specifics.

Managing click map

To start working on click map report, you need to select the template that you want to analyze. If you have changed the message, for example, for different mailings, select the required version from the drop-down menu Date period:



TIP

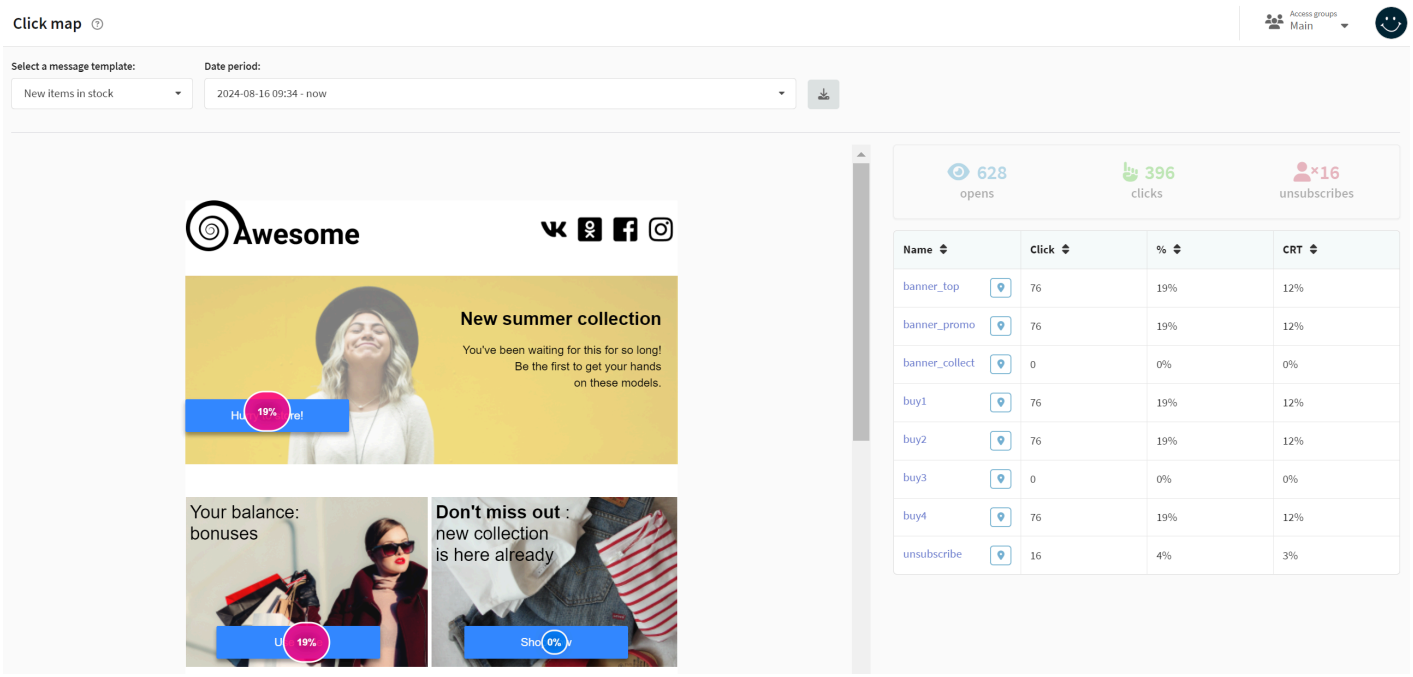
You can download the resulting report in MS Excel and CSV format by clicking the Export report button.

Click map report displays only unique clicks for all links in email message, including unsubscription links.

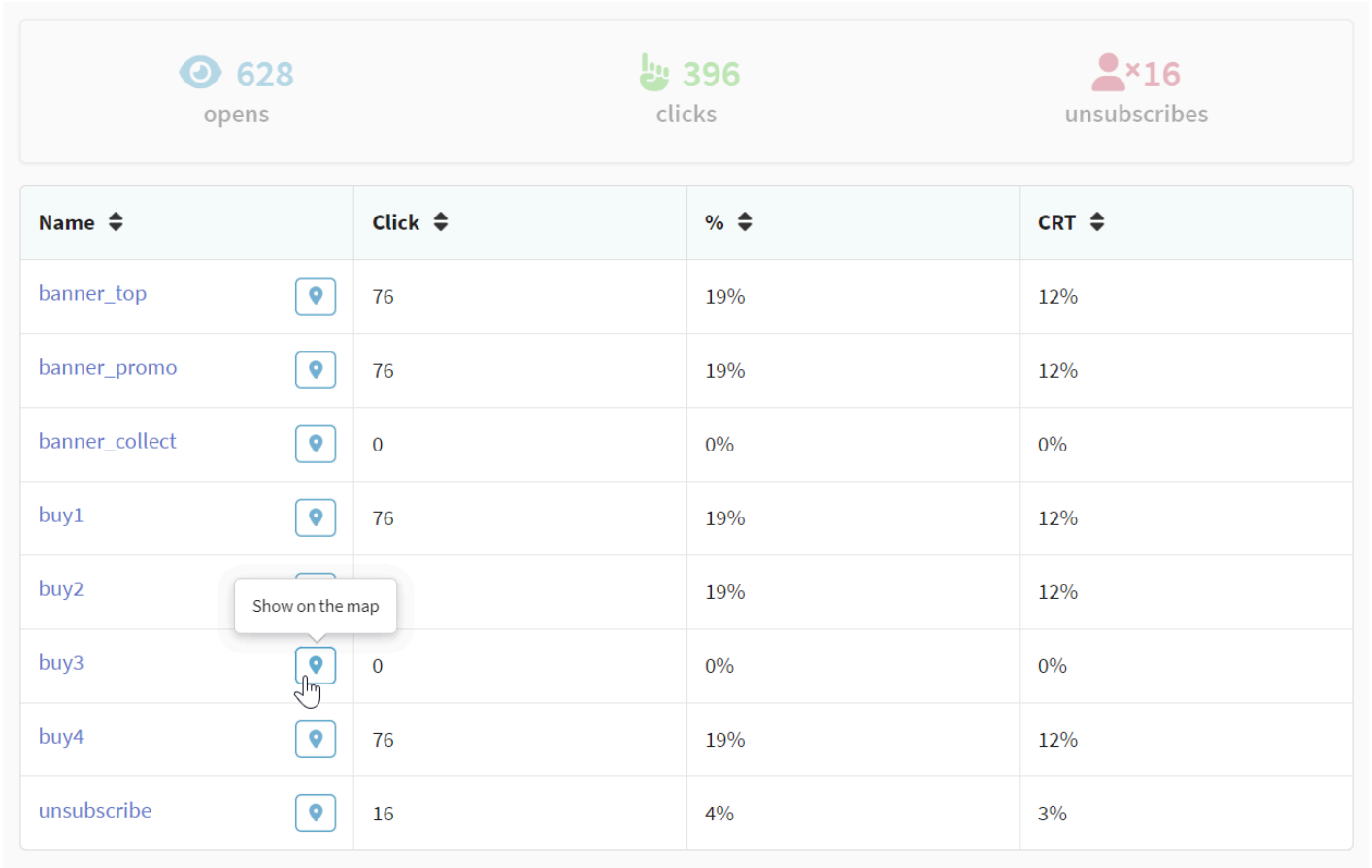
Additional information is displayed on the right:

- **The number of unique openings of the template**
- **The total number of unique clicks**
- **The number of unsubscribed customers**

Below is a click chart and a table with the number of clicks on a particular link, percentage of total clicks and CTR for each individual link of the message:



There is a geo-location button next to each link. When you click on it, you are transported to the area of the letter where the link is located.

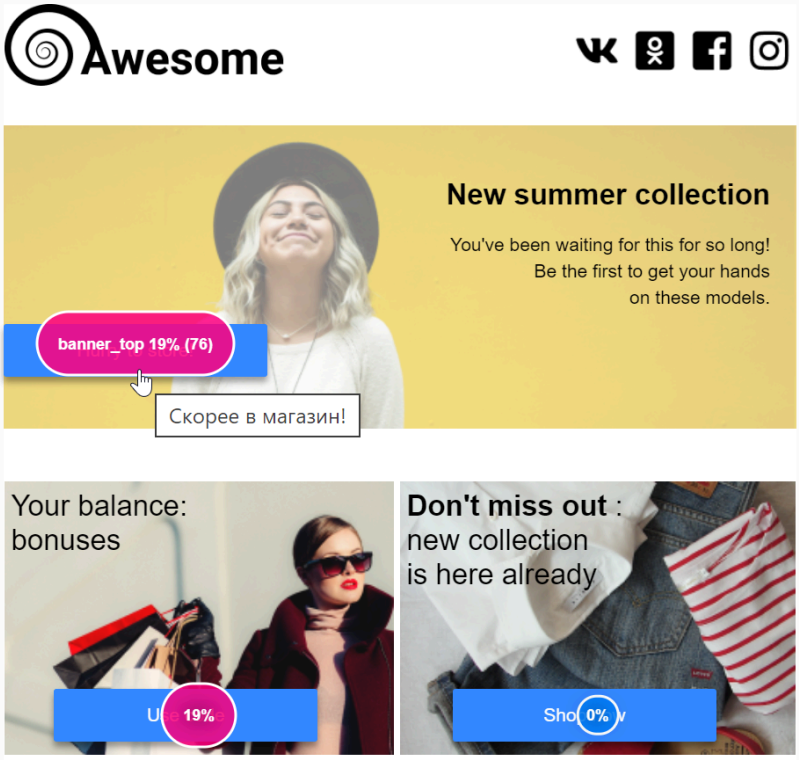


This makes the search for links more convenient, especially if there are a lot of them in the template or you are analyzing a large letter with links scattered around.

Click map

Select a message template: New items in stock

Date period: 2024-08-16 09:34 - now



INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Loyalty programs report

The loyalty program report shows key loyalty program metrics: points usage, transactions, purchases, and member activity. The report gives a clear view of customer engagement and overall program effectiveness.

To open the report, go to Analytics → Loyalty in the main menu.

Main report metrics

TIP: What is NQP

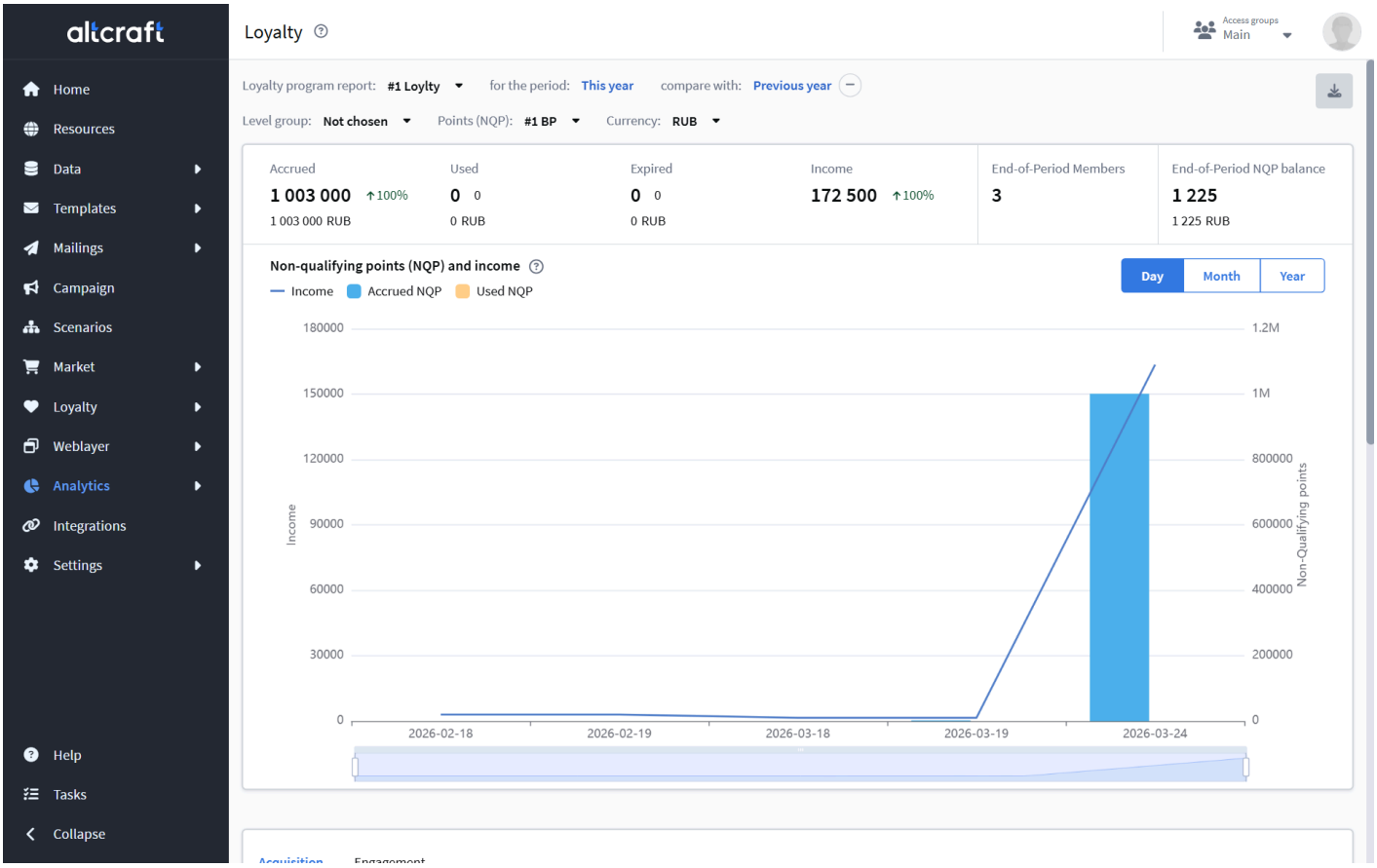
One of the key metrics used in analytics is NQP points. NQP are points that do not participate in loyalty tier calculations but affect member activity. In other words, these are bonus points that a member earns and spends.

Select a loyalty program, the reporting period, and a comparison period. You can also choose a tiers group, the points type, and the currency.

Main report metrics:

- **Accrued — amount of NQP accrued to members during the period;**
- **Redeemed — amount of NQP redeemed by members during the period;**
- **Expired — amount of NQP that expired during the period;**
- **Revenue — total value of orders placed by loyalty members during the period;**
- **Members at period end — number of loyalty program members (recalculated at the end of the day);**
- **NQP balance at period end — total amount of NQP held by loyalty program members (recalculated at the end of the day).**

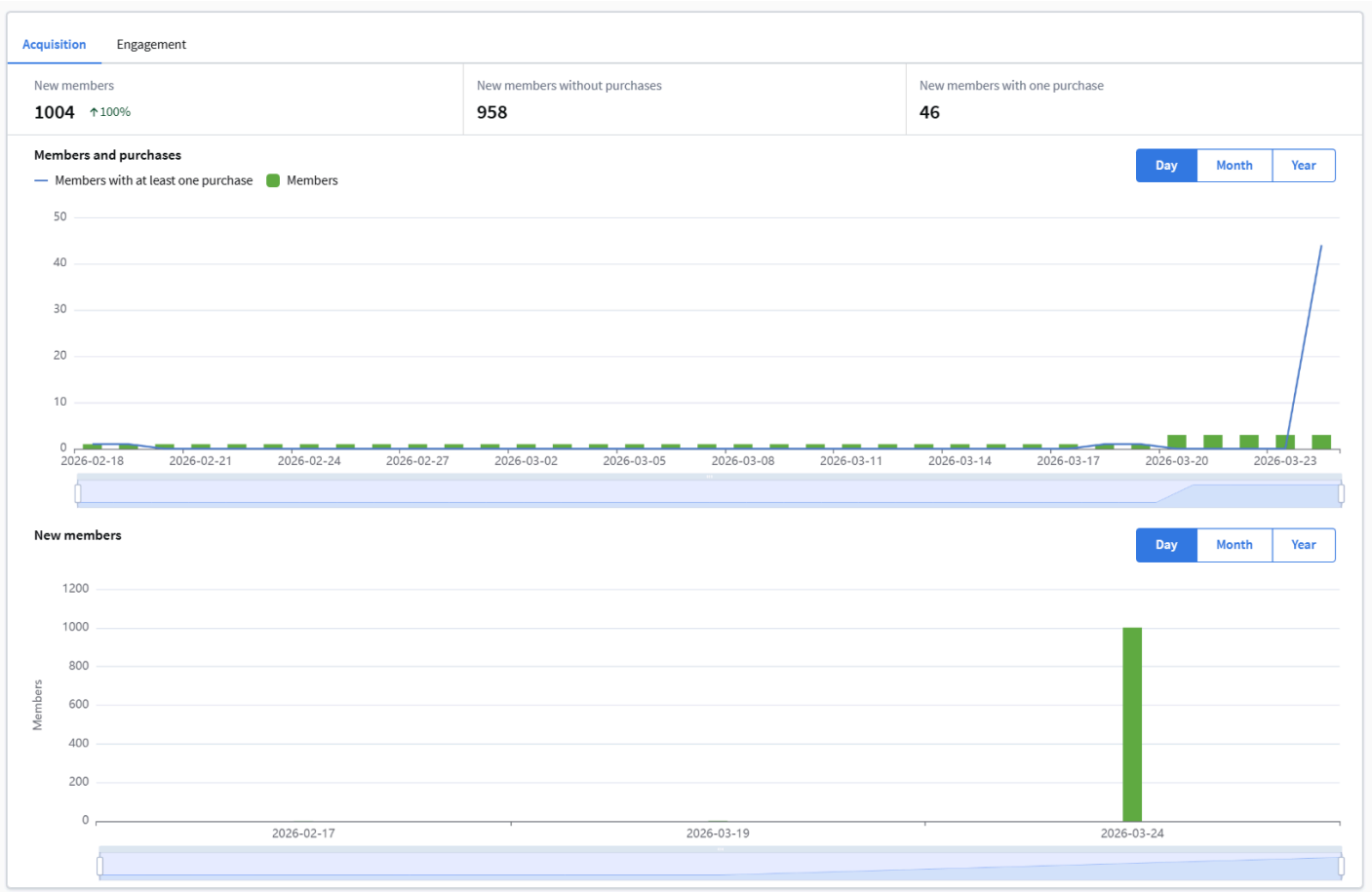
Below the metrics, charts are displayed: member distribution by tiers and a chart of revenue, accrued, and redeemed NQP.



Acquisition

This section contains data on acquiring new loyalty program members:

- **Number of new members, including members without purchases and members with at least one purchase;**
- **New members chart.**



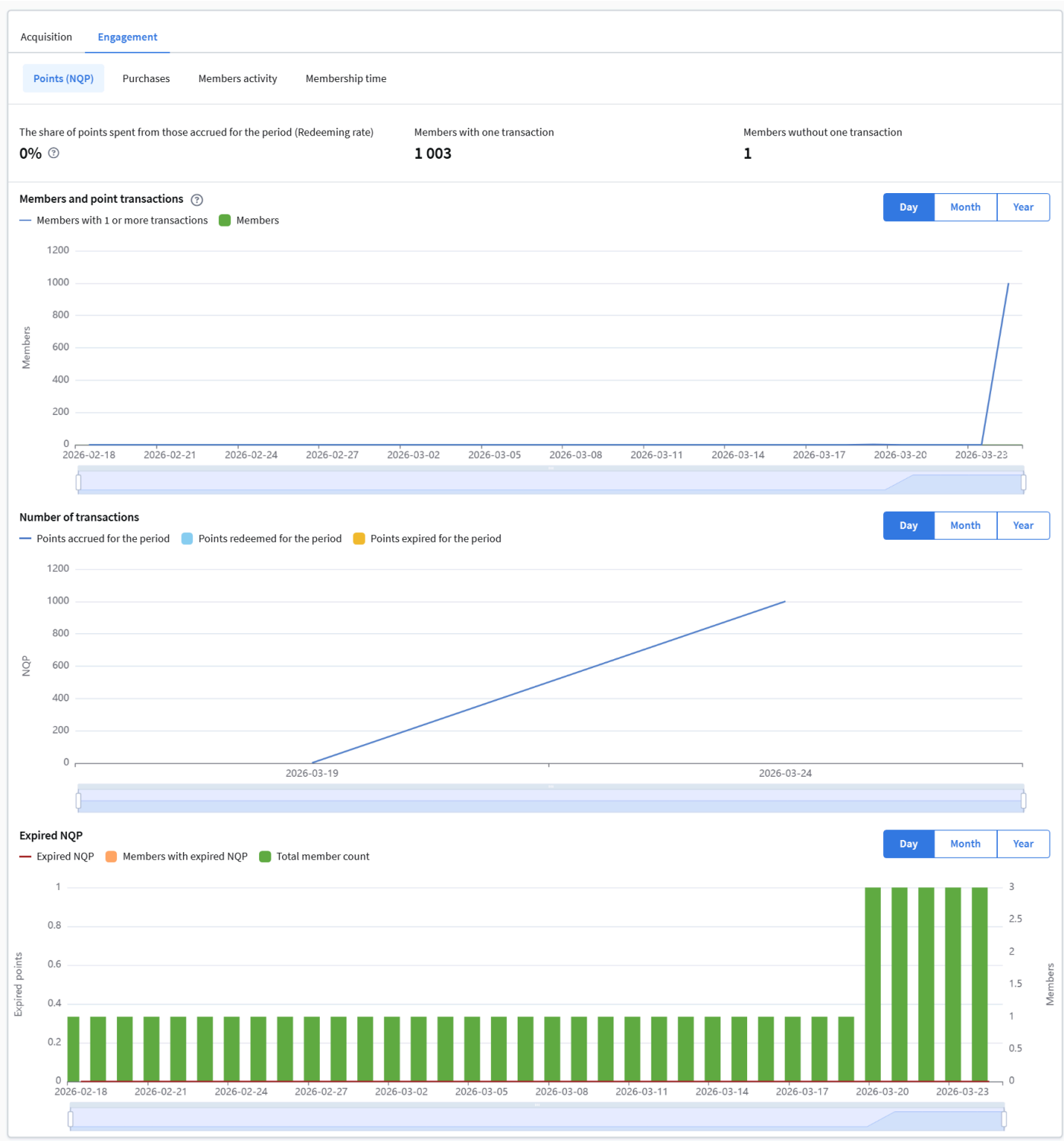
Engagement

This section shows member engagement based on activity. It includes the Points, Purchases, Member activity, and Membership duration tabs.

Points (NQP)

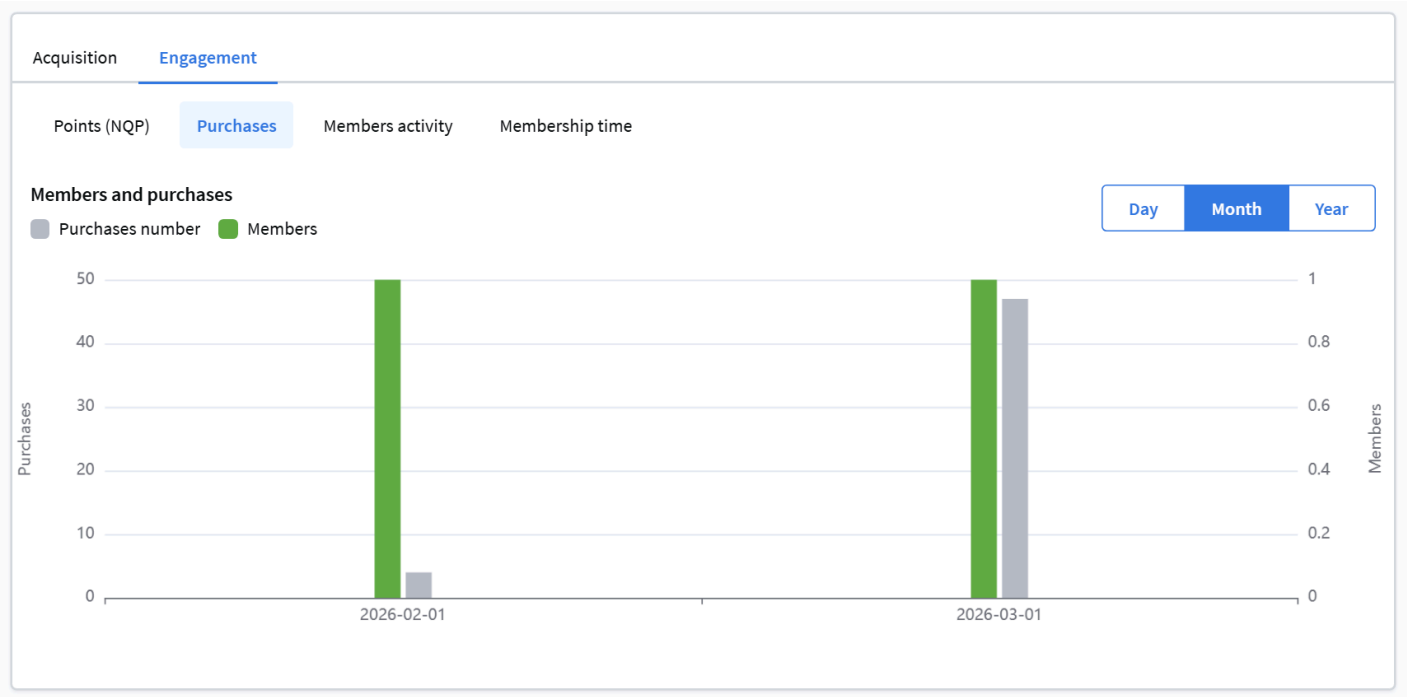
This tab contains:

- **Share of redeemed points relative to accrued points (for the period), including new members with and without transactions;**
- **Charts for program members and their transactions;**
- **Expired NQP chart.**



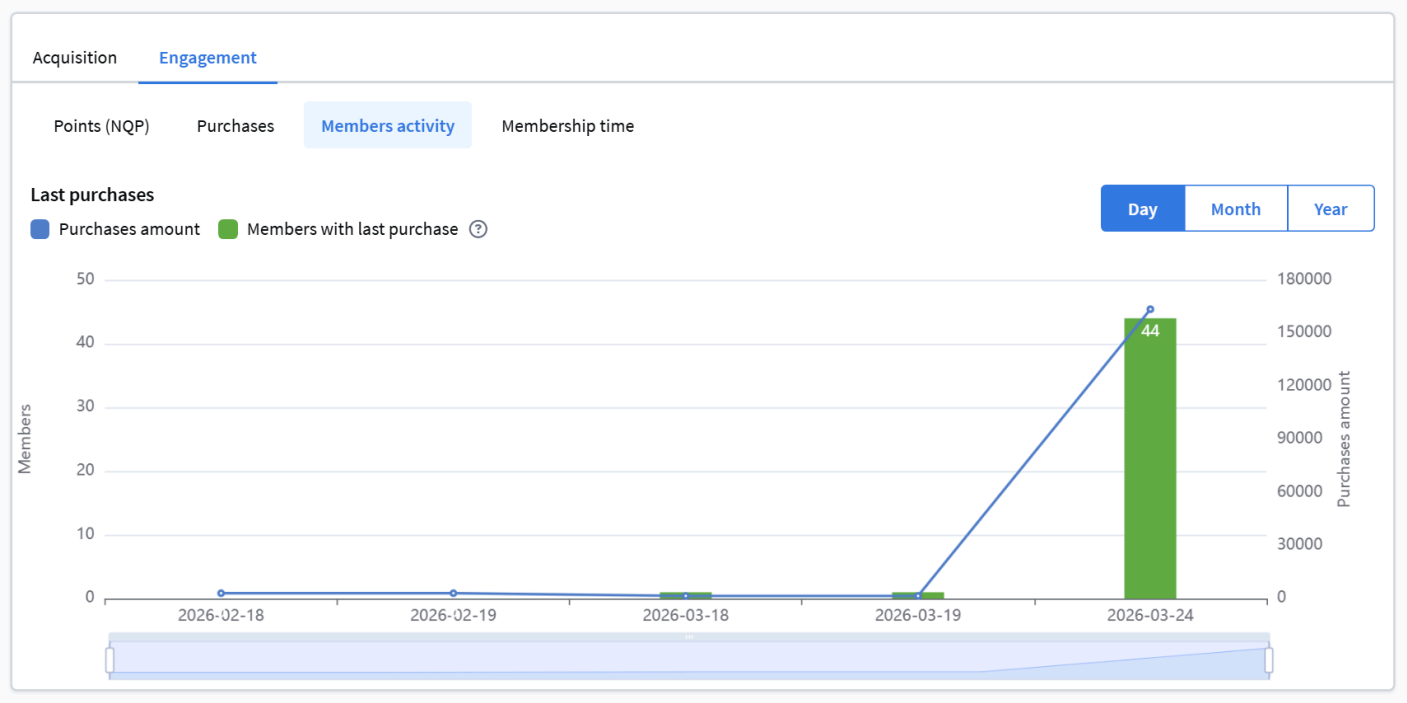
Purchases

This section displays purchase data for loyalty program members.



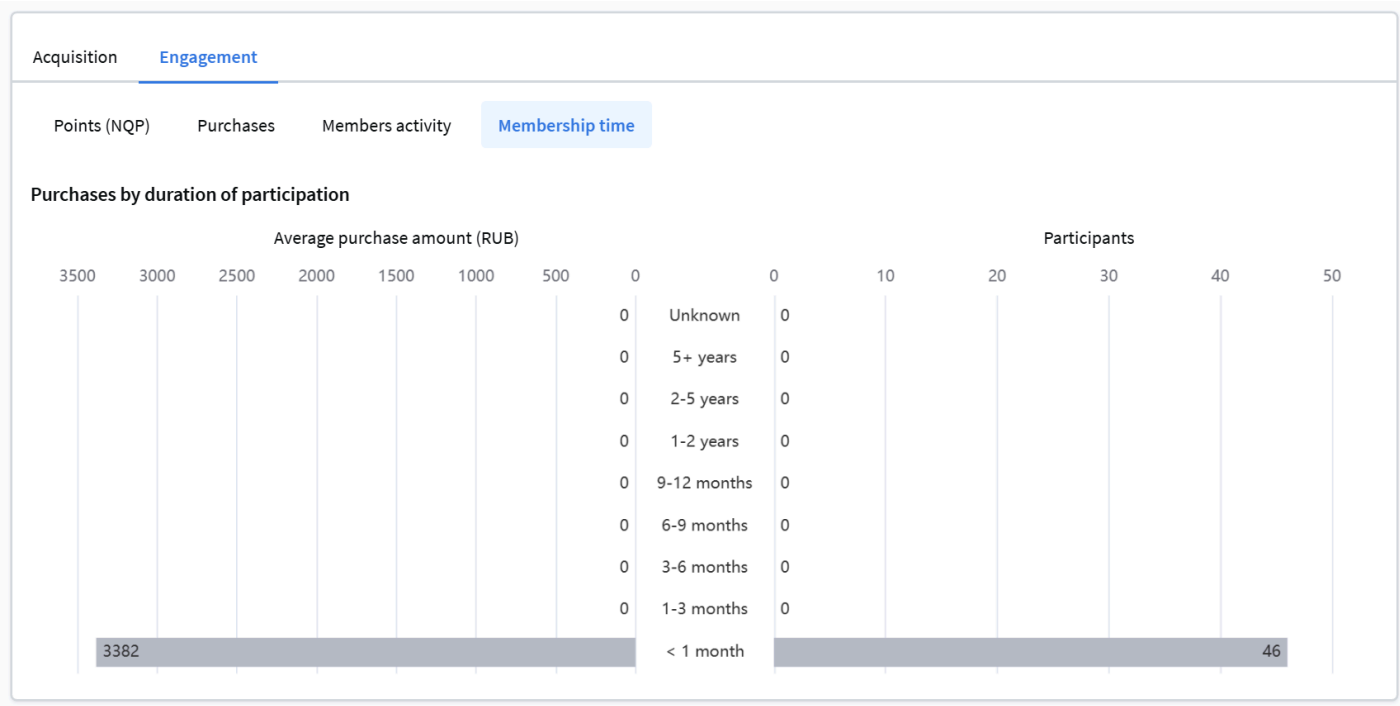
Member activity

This chart shows purchase totals by month and members whose last purchase occurred in the corresponding month.



Membership duration

This section contains a histogram with data on members, their average purchase amount, and the length of their participation in the loyalty program.



Report export

To export the report, click the corresponding icon in the upper-right corner of the screen. Enter a report name and click Export.

An export task for the loyalty program report will be created. You can download it from the Tasks section.

Bounces report

Hard and Soft bounces

In such cases, when the message cannot be delivered to the recipient, the mail server sends back an error message. There are two types of these errors: Hard bounce and Soft bounce.

Hard bounces are server class errors (5xx). Such errors occur in cases, when sending to the mail server is impossible, for example, the email does not exist. If the server returns a Hard bounce error, the platform changes the profile status to Hard bounced and adds the email address to the global supression list. As long as the address is in the global supression list, you will not be able to send a message to it. You can remove an address from the global supression list only in the administration panel in the [Supression list **] section .**

> CAUTION
>
> **Email addresses with Hard bounce errors are collected in the supression list located in the administration panel. This should not be confused with the Global suppress list in the user panel.**

Soft bounces are sending client errors (4xx). Unlike Hard bounce, Soft bounce does not affect customer subscription statuses. If a Soft bounce is received, the platform tries to resend the message according to resend rules. If the message cannot be delivered, the platform will record a non-delivery event.

The reasons of Soft bounce can be:

- **Subscriber inbox is full**
- **Letter attachments are too big for subscriber mail server**
- **Email service is temporarily unavailable**

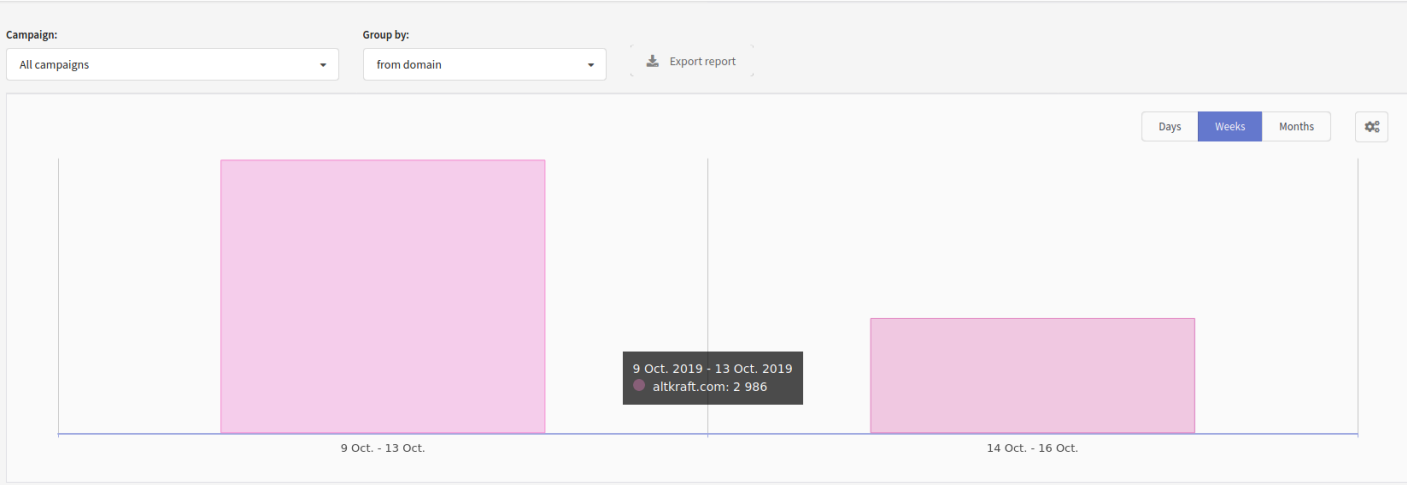
This report will allow you to track the number of Email messages that didn't get delivered to recipients the first time. In other words, soft bounces are combined in this report.

Report settings

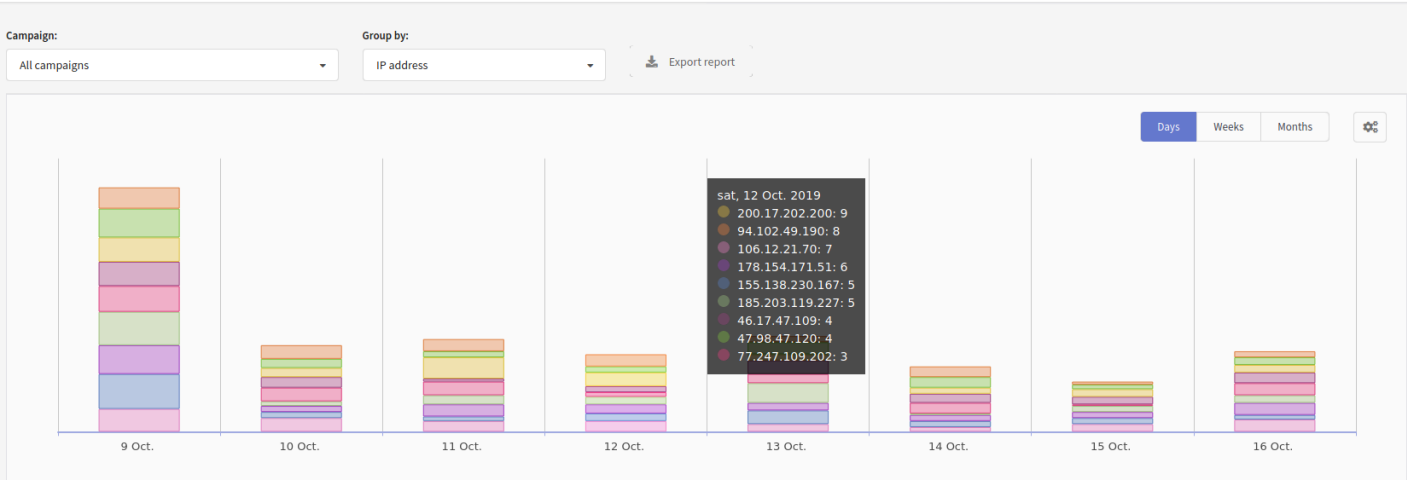
To start working on bounces report, you need to select time period and mailing (or all mailings).

Also, you can use grouping by from domain or IP address that was used for mailing.

Grouping by from domain:



Grouping by IP address



INFO

You can download the resulting report in MS Excel and CSV format by clicking the Export report button.

Table

The following information is displayed in the table:

- **email domain that was used for sending mailings;**
- **bounce text (reply from provider about customer email temporary unavailable);**
- **amount - number of message bounces if one message was sent several times.**

Group by	Bounce text	To domain	Amount
95.163.255.37	Your message was temporarily def..	wikipedia.org	4
94.102.49.190	This response is sent when the m..	ft.com	3
106.12.105.45	Your message was temporarily def..	ameblo.jp	3
172.104.108.109	Your message was temporarily def..	vk.com	3
185.234.219.238	This kind of response is sent ba..	ustream.tv	3
186.43.201.151	Your message was temporarily def..	com.com	3
134.209.163.72	Your message was not delivered b..	paginegialle.it	3
106.12.105.45	Your message was temporarily def..	vinaora.com	3
62.173.138.240	This response is sent when the m..	tamu.edu	3
200.17.202.200	This response is sent when the m..	tuttocitta.it	3
Totals:			**1 478**

To view the full error message, hover over the bounce text. A list of common soft bounce reasons and their descriptions is provided in the table below:

Bounce text	Description
`421 Message source blocked temporarily. One or more sender authentication rules matched.`	**Messages from this sender are temporarily blocked due to a failure to meet email authentication rules. This can occur if the sender did not pass SPF, DKIM, or DMARC verification. The sender's From-domain authentication settings should be checked.**
`421 Service Temporarily Unavailable`	**The server is temporarily unavailable, potentially due to overload or technical maintenance.**
`451 Please try again later`	**A temporary issue on the server side, likely due to overload or maintenance.**
`451 <<EMAIL>>: Recipient address rejected: Intentional policy rejection, please try again later`	**The recipient's email address was rejected based on server policy. This may happen due to incorrect settings on the recipient's side or if the sender was previously marked as spam.**
`450 <<EMAIL>>: Sender address rejected: Greylisted for 120 seconds! Please, see <URL>`	**The sender's address is temporarily rejected due to greylisting. The server temporarily refuses messages from the sender and expects the sender to retry after the specified time. The platform will automatically retry after 120 seconds. If the issue persists after retrying, the sender's settings should be checked.**

INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Undeliveries report

Description

Undeliveries report displays information about number of Email, SMS, Push, Telegram Bot, MAX Bot, Viber, WhatsApp* messages that could not reach subscribers.

Most recent undelivery reasons are:

Channel	Reasons
Email	**Maximum number of attempts to send message has been reached** **Message has not been sent until end of its lifetime (24 hours)** **Message has been removed from messages queue by user** **Message has been sent to mailbox which does not exists**
SMS	**Study your provider's documentation**
Push	**Invalid push token in a resource subscription**
Telegram Bot	**Invalid chat_id.**
Viber	**Study your provider's documentation (the reasons may vary)**
MAX Bot	**Invalid user_id**
WhatsApp*	**Study your provider's documentation (the reasons may vary)**
Notify	**Study your provider's documentation (the reasons may vary)**

In some cases it's impossible to determine a reason of undelivery. These messages delivery error marked as undefined.

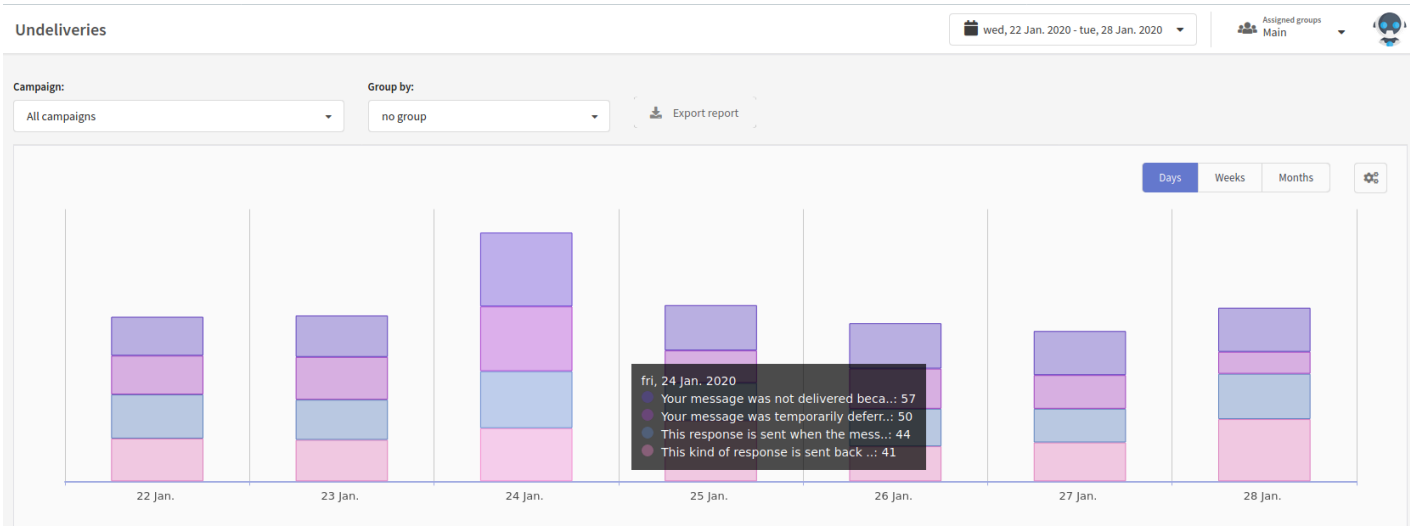
INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Report settings

At the top of the page, select the period for the report.

You can group information by one or all mailings, also you can use grouping by from domain or by IP address which was used for sending.



INFO

You can download the resulting report in MS Excel and CSV format by clicking the Export report button.

Table

The following information is displayed in the table:

- **Group by — selected mailing (or "all mailings");**
- **To domain — receiving email domain;**
- **Bounce text — message from email provider about temporary unavailability of the recipient mailbox;**
- **Amount — number of replies.**

Group by	To domain	Bounce text	Amount
All campaigns	freewebs.com	Your message was not delivered b..	4
All campaigns	aol.com	Your message was not delivered b..	4
All campaigns	rambler.ru	Your message was not delivered b..	3
All campaigns	furl.net	Your message was temporarily def..	3
All campaigns	scientificamerican.com	Your message was temporarily def..	3
All campaigns	smh.com.au	This response is sent when the m..	3
All campaigns	fotki.com	This kind of response is sent ba..	3
All campaigns	nps.gov	This response is sent when the m..	3
All campaigns	odnoklassniki.ru	This kind of response is sent ba..	3
All campaigns	merriam-webster.com	This response is sent when the m..	3
Totals:			**955**

To view the full error text, hover over the bounce message. Common delivery errors and their descriptions are listed in the table below:

Bounce text	Description
`500 dial tcp <IP4:PORT>-><IP4:PORT>: i/o timeout`	**Network issue, failed to connect to the receiving server.**
`500 lookup <DOMAIN> on <IP4:PORT>: no such host for domain:<DOMAIN>`	**The domain does not exist.**
`540 <<EMAIL>>: recipient address rejected: Inactive`	**The recipient address exists, but it has been deactivated (e.g., disabled or temporarily suspended). This address may be reactivated later.**
`550 relay not permitted`	**Incorrect forwarding configuration on the recipient's mailbox.**
`550 No such user!`	**The email address does not exist on the server (either the account was deleted or never existed).**
`550 Unrouteable address`	**Unroutable address (e.g., if the recipient's domain lacks A/PTR records).**
`550 Message was not accepted -- invalid mailbox. Local mailbox <EMAIL> is unavailable: user is terminated`	**The user's mailbox has been deleted.**
`550 Message was not accepted -- invalid mailbox. Local mailbox <EMAIL> is unavailable: account is disabled`	**The user's mailbox is disabled.**
`552 5.2.2 <<EMAIL>>: user is over quota`	**The recipient's mailbox is full.**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

Report on global control groups

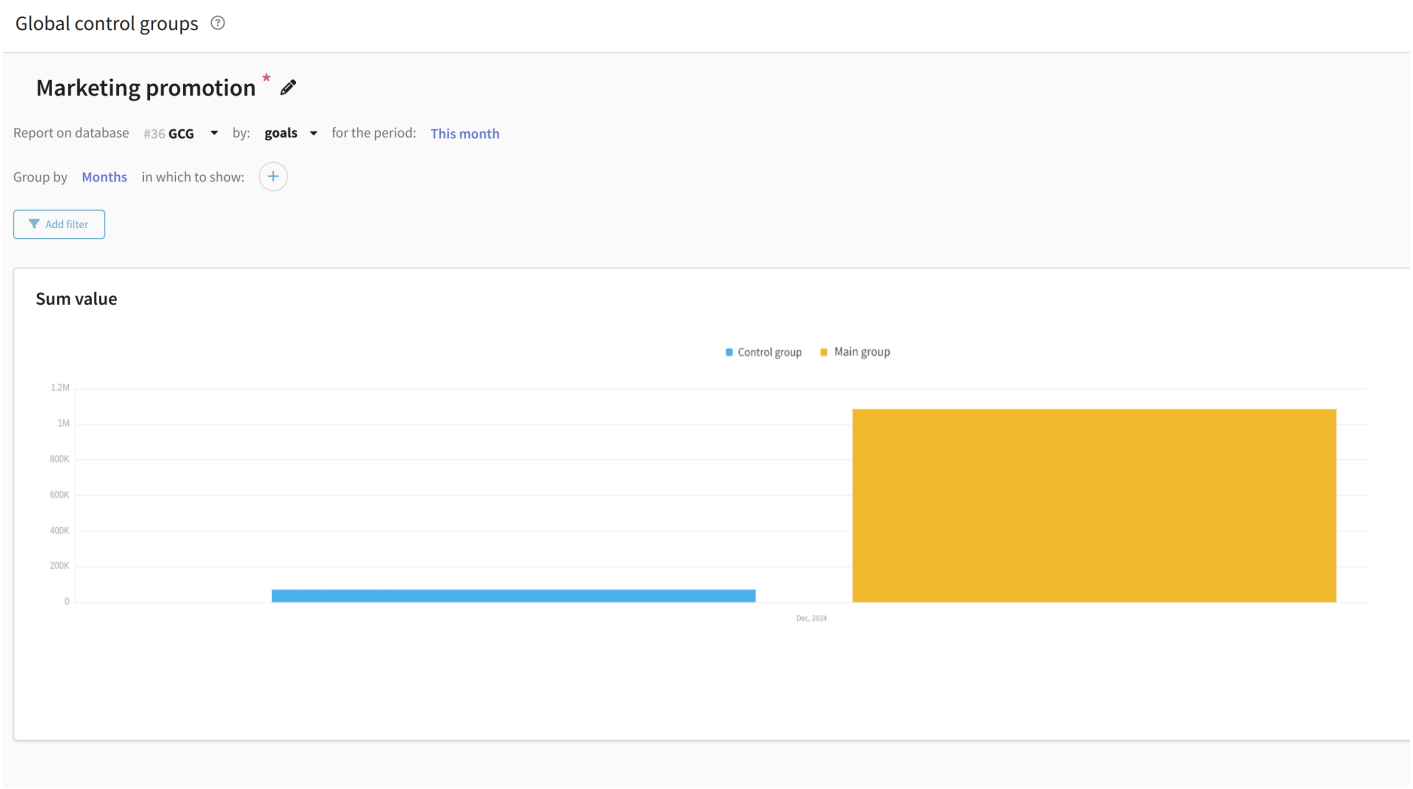
Global control groups (GCG) - a part of the customer base that temporarily does not receive marketing communications in order to analyze the effectiveness of marketing strategies. You can learn more about control groups on this page.

The Global Control Groups report is available on the corresponding tab of the Analytics section:

TIP

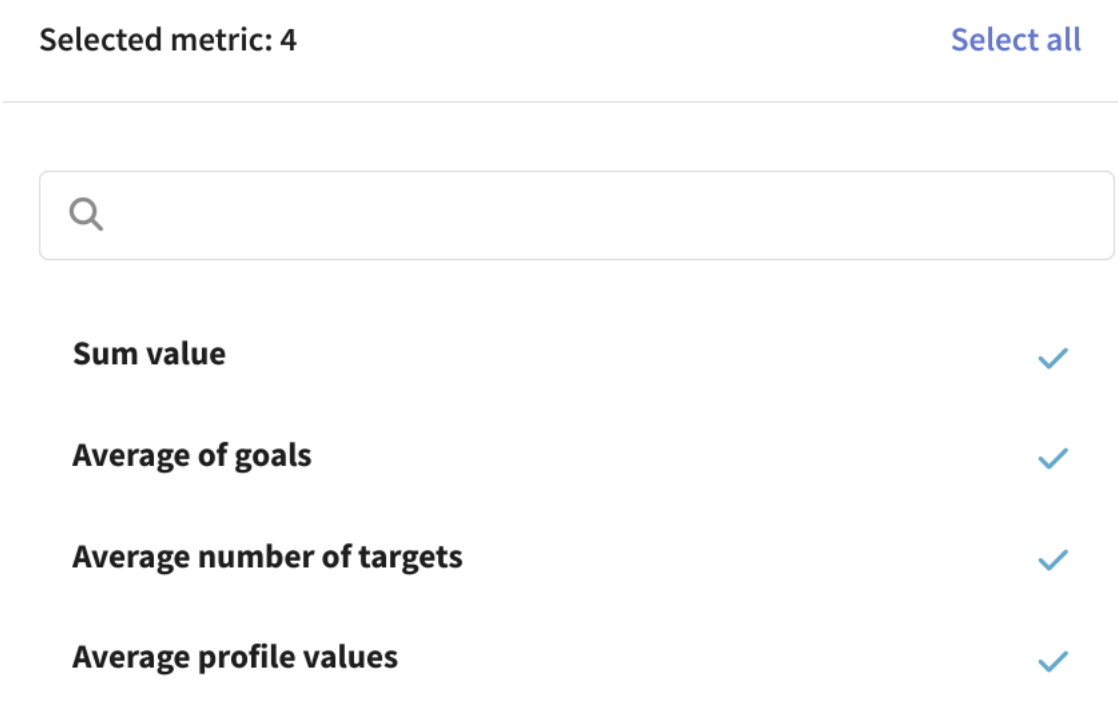
Global control group analytics are based on metrics collected from either Market data or pixel targets. This requires you to configure them in the platform beforehand. You can read more about the Market here, and this section contains information about tracking pixels.

The report presents a comparison of data for the control and main groups. Information can be retrieved by market and by targets. For market metrics, you can select sales channel, and for targets, a selection of pixels and pixel targets is available. A graph based on the specified metrics of both groups will be displayed on the screen:



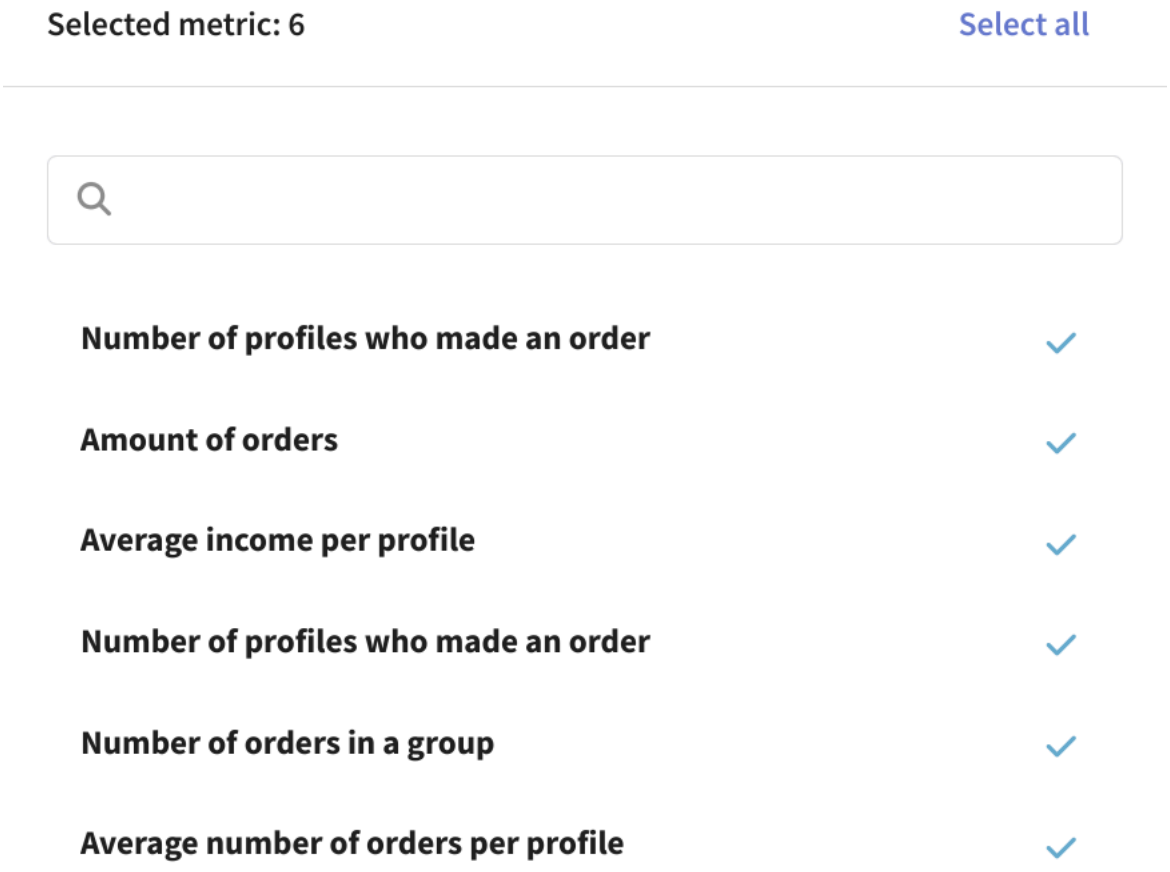
In the table below you can see the values of the metrics. The following metrics are available for targets:

- **Sum Value**
- **Average value of goals**
- **Average number of goals**
- **Average profile value**

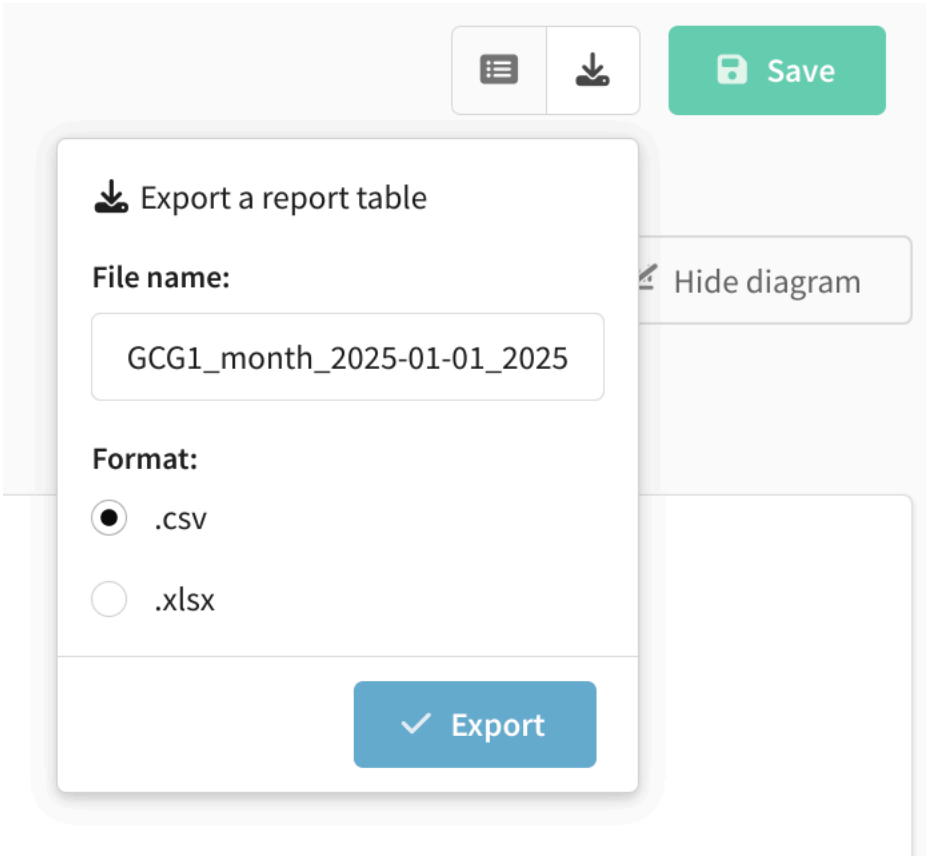


For market, metrics such as:

- **Number of profiles that have ordered**
- **Amount of orders**
- **Average revenue per profile**
- **Conversion to customers**
- **Number of orders per group**
- **Average number of orders per profile**



You can save the report or upload it in any convenient format by clicking on the corresponding button at the top of the screen:



The saved report will be available in the report list available next to the upload and save buttons:

Channel reports

Saved reports

Search...

GCG

Close

Summary report

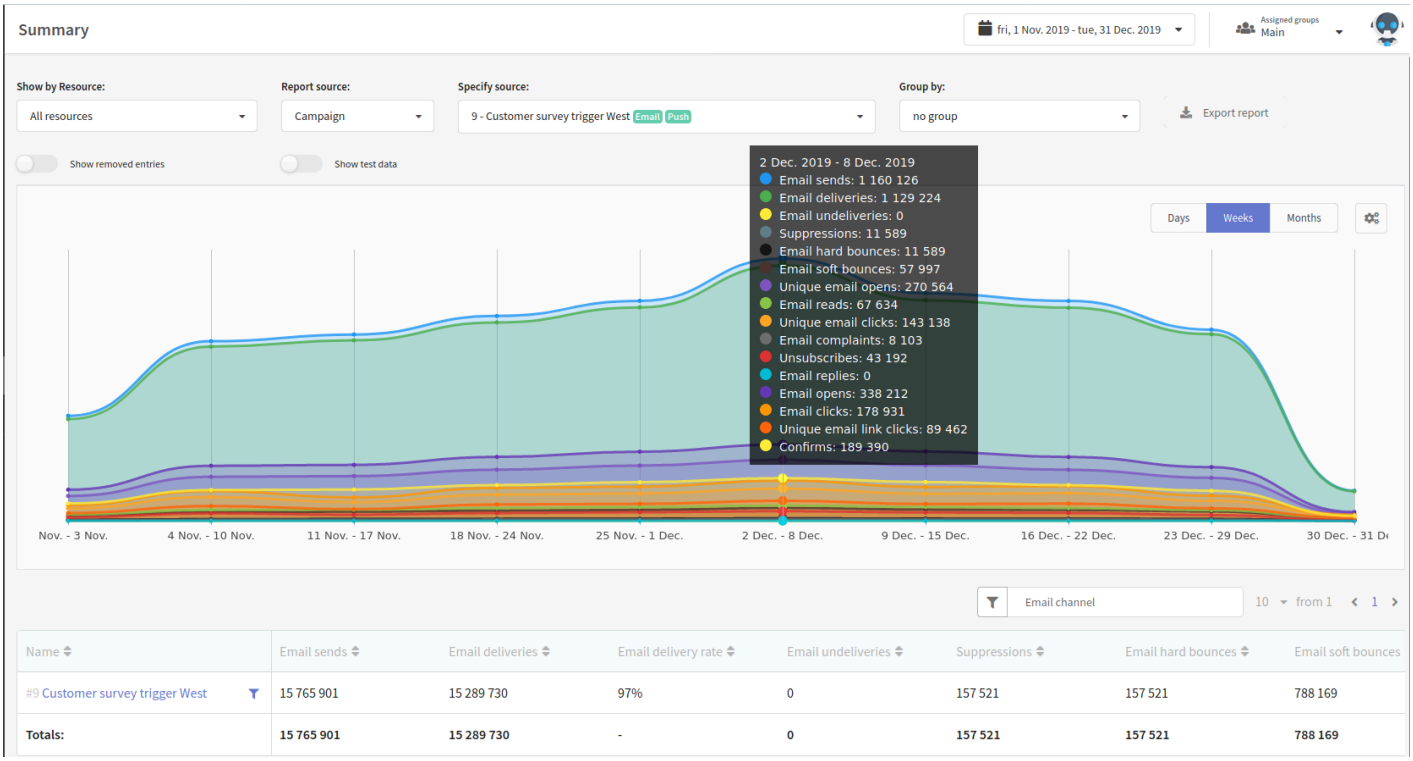
CAUTION

The summary report is no longer supported. It is not available in versions 65 and higher.

Description

A summary report is the basic tool for detailed marketing mailing analytics. It gathers mailing statistics from the moment when messages are sent, including absolute numbers of opens, reads or clicks as well as relative parameters like delivery rate, open rate or click-through rate.

Moreover the summary report then merges selected mailings to provide you with a summary of your marketing efficiency, represented as an interactive graph, where you can see your over-time messaging dynamics and get insights on possible communication improvements.



INFO

The Altcraft platform allows an inaccuracy of 0.02% when generating reports.

Managing summary report

You can start working on summary report from resource selection in Show by Resource menu (1), or display report by all resources.

Next, you can select Report source(2). The following options are available:

- **Mailing**
- **Scenario**
- **Database**
- **Segment**
- **Message template**
- **Resource**
- **Domain group**
- **Domain**
- **Vendor**
- **Day**
- **Week**
- **Month**
- **Environment**

- **Sender**

Using Specify source menu (3), you can build diagram using information from the menu 2, except domain, day, week, month, environment:

Using next Group by menu (4), you can group report by:

- **Domain**
- **Domain group**
- **Database**
- **Segment**
- **Message**
- **Resource**
- **Push provider**
- **Vendor**
- **By day**
- **By week**
- **By month**
- **Environment**
- **Sender**

Next, you can select time period for the report.

You can download configured report in XLS or CSV formats.

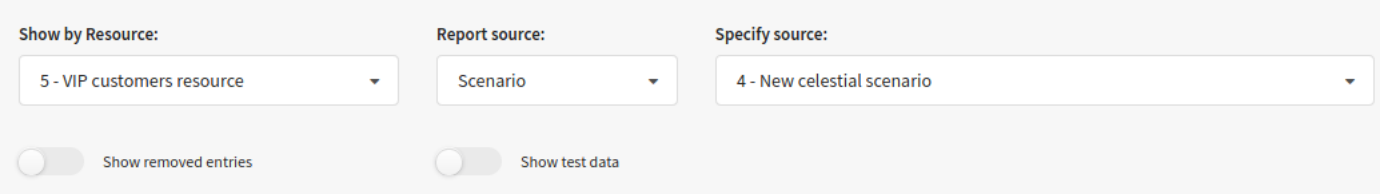
Also, there are two options — Show removed entries and Show test data.

Show removed entries — includes removed information into the report

Show test — includes test mailings into the report

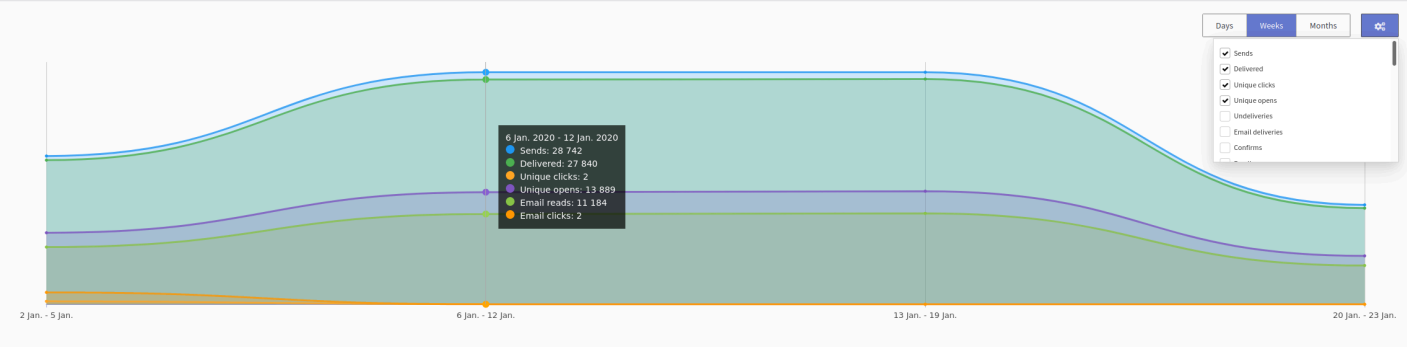
Summary report by scenarios triggers

Select report by Scenario to see detailed statistics by all mailings within automation scenario.

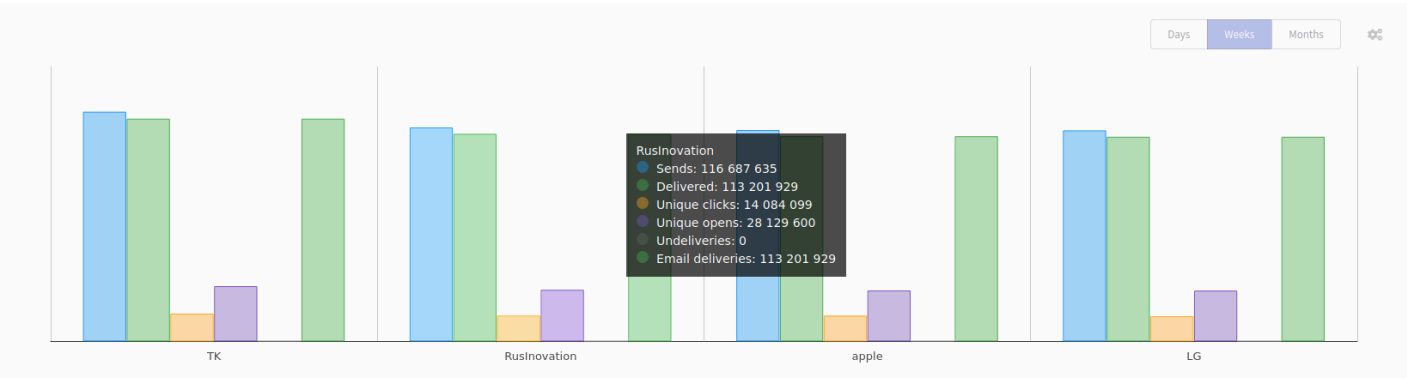


Diagram

You can select information for display in the diagram by “gear” buttons. Also, you can see diagram for day, week, or month.



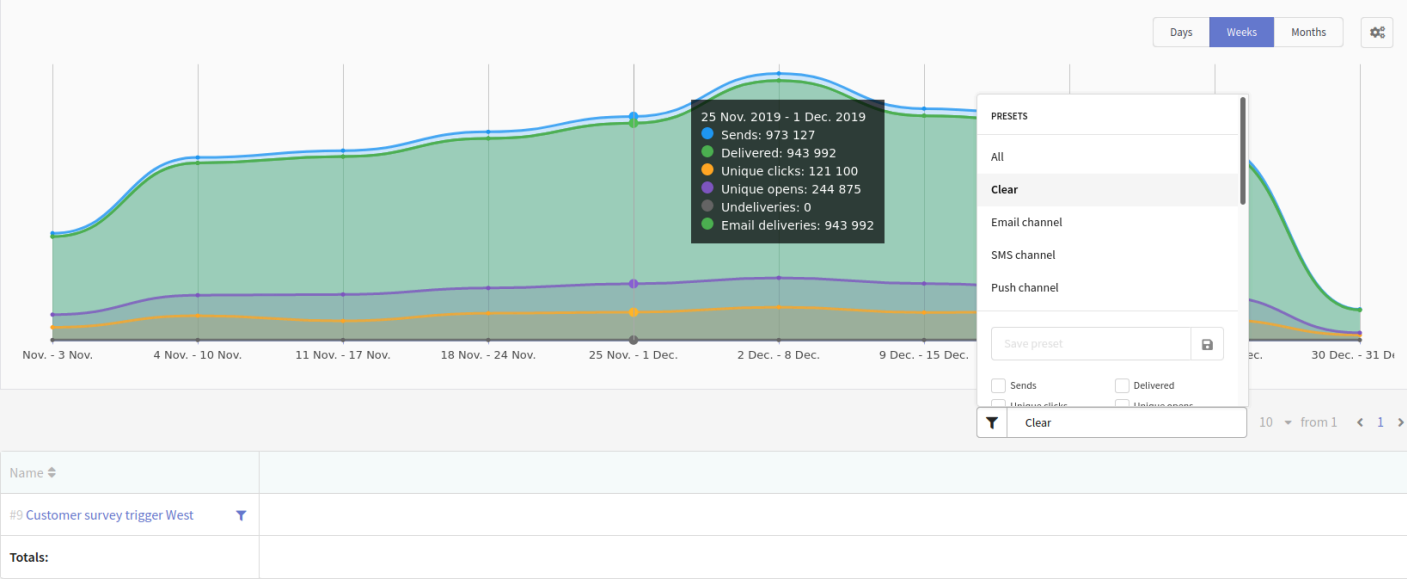
For *domain groups, domains, vendors and environment*, diagram is displayed in columns.



Performance indicators

You can select values for display in table in Select column settings menu. You can use presets or select values manually. Numbers in the table show amount of actions.

If you need more information about values of the summary report, you can read [“Summary report metrics”](#).



TIP

You can filter data in the table by tag, which allows to improve search.

Summary report metrics

Channel	Indicator	API name	Description	Formula
All	Sends	sends_total	Total number of sent messages in all the channels	sends + sends_push + sends_sms
All	Deliveries	delivs_total	Total number of delivered messages in all the channels	delivs + delivs_push + delivs_sms
All	Unique clicks	uclicks_total	Total number of unique clicks in messages in all the channels	sends + sends_push + sends_sms
All	Unique opens	uopens_total	Total number of unique opened messages in all the channels	uopens + uopens_push
All	Undeliveries	undelivs_total	Total number of unique undelivered messages in all channels	undelivs + undelivs_sms + undelivs_push
All	Offence	offence	Number of communications policy violations	—
Email	Email sends	sends	Number of sent emails	—
Email	Email deliveries	delivs	Number of delivered emails	—
Email	Email opens	opens	Number of opened emails	—
Email	Email clicks	clicks	Number of clicks in emails	—
Email	Unique email opens	uopens	Number of unique email opening	—
Email	Unique email clicks	uclicks	Number of unique clicks on any clickable item of the email	—
Email	Unique email link clicks	ulclicks	Uniqueclick on each link of each email by the same profile	—
Email	Email reads	reads	Number of read emails. Email read event is logged if the recipient has opened the message and left it open for at least 12 seconds.	—
Email	Email replies	replies	Number of emailreplies	—
Email	Email complaint	complains	Number of emails followed by spam complaints (FBL)	—
Email	Email hard bounces	hbounces	Number of emails that have failed to deliver because of recipient'sinvalid address or a sudden mistake during the process of sending	—
Email	Email soft bounces	sbounces	Number of emails that has failed to deliver because the mailbox is full or the server is temporarily down	—
Email	Confirms	confirms	Number of confirmed emails	—
Email	Email unsubscribe	unsubs	Number of unsubscribers. Recipient opted out by clicking a message link (`{unsubscribe}` or `{globalunsubscribe}`).	—
Email	Suppression list	suppresses	Number of emails that hasn't been delivered because the recipient's address is in the suppression list	—
Email	Email undeliveries	undelivs	Number of undelivered emails. It consists of soft and hard bounces and undefined delivery errors	—
Email	% Email delivery rate	delivery_rate	The percentage of the emails that are successfullydelivered	(delivs / sends) * 100
Email	% Email open rate	u_open_rate	The percentage of the recipients who opened an email	(uopens / delivs) * 100
Email	% Email CTR	u_click_rate	The percentage of unique recipients who clicked on a link after receiving the email	(ulclicks / delivs) * 100
Email	% Email CTOR	u_open_click_rate	The percentage of unique recipients who clicked on a link after opening the email	(ulclicks / uopens) * 100
Email	% Email read rate	email_read_rate	The percentage of recipients who read the email, i.e. those who opened the email and did not close it for at least 12 seconds.	(reads / delivs) * 100
Email	% Email reply rate	email_reply_rate	The percentage of the recipients who replied to an email	(replies / delivs) * 100
Email	% Email complaint rate	complain_rate	The percentage of the emails that are followed by spamcomplaints	(complains / delivs) * 100
Email	% Email hard bounce rate	hb_rate	The percentage of emails that have failed to deliver because of recipient'sinvalid address or a sudden mistake during the process of sending	(hbounces / sends) * 100
Email	% Email soft bounce rate	sb_rate	The percentage of emails that has failed to deliver because the mailbox is full or the server is temporarily down	(sbounces / sends) * 100
Email	% Email unsubscribe rate	email_unsub_rate	The percentage of people who unsubscribed from emails	(unsubs / delivs) * 100
SMS	SMS sends	sends_sms	The number of sent SMS	—
SMS	SMS deliveries	delivs_sms	The number of delivered SMS	—
SMS	SMS clicks	clicks_sms	The number of clicks in SMS	—
SMS	Unique SMS clicks	uclicks_sms	The number of unique clicks in SMS	—
SMS	% SMS delivery rate	sms_deliv_rate	The percentage of SMS that has been successfully delivered	(delivs_sms / sends_sms) * 100
SMS	SMS undeliveries	undelivs_sms	The number of undelivered SMS	—
SMS	% SMS click-though rate	sms_ctr	The percentage of unique clicks in the SMS in relation to the total number of delivered SMS	(uclicks_sms / delivs_sms) * 100
SMS	SMS unsubscribes	unsubs_sms	Number of unsubscribers. Recipient opted out by clicking a message link `{unsubscribe}` or `{globalunsubscribe}` .	—
Push	Push sends	sends_push	The number of sent push notifications	—
Push	Push opens	opens_push	The number of opened push notifications	—
Push	Unique push opens	uopens_push	The number of opened push notifications	—
Push	Push deliveries	delivs_push	The number of delivered push notifications	—
Push	Push undeliveries	undelivs_push	The number of undelivered push notifications	—
Push	Push clicks	clicks_push	The number of clicks on any clickable element of the push notification	—
Push	Uniquepush clicks	uclicks_push	The number of clicks on any clickable element of the push notification	—
Push	% Push open rate	push_open_rate	The percentage of opened push notifications	(uopens_push / delivs_push) * 100

Channel	Indicator	API name	Description	Formula
Push	% Push send rate	push_deliv_rate	The percentage of push notifications that has been successfully delivered	(delivs_push / sends_push) * 100
Push	% Push click-though rate	push_ctr	The percentage of clicks in the push notification in relation to the push openings	(uclicks_push / uopens_push) * 100
WhatsApp*	Delivered	whatsapp_deliv	Number of messages delivered via WhatsApp*	—
WhatsApp*	undelivered	whatsapp_undeliv	Number of messages send via WhatsApp*	—
WhatsApp*	Send	whatsapp_send	Number of messages send via WhatsApp*	—
WhatsApp*	WhatsApp* subscription	whatsapp_subscribe	The number of new subscriptions to the WhatsApp* channel	—

INFO

Unique event

The uniqueness of any event is limited by the time specified in the platform configuration (`UNIQUE_ACTIONS_TTL_MIN` parameter). By default it is 1440 minutes. It means that a repeated event in a day will be counted as unique again. Check with your platform administrator for the uniqueness value set.

All indicators of unique events are marked in the table.

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.*

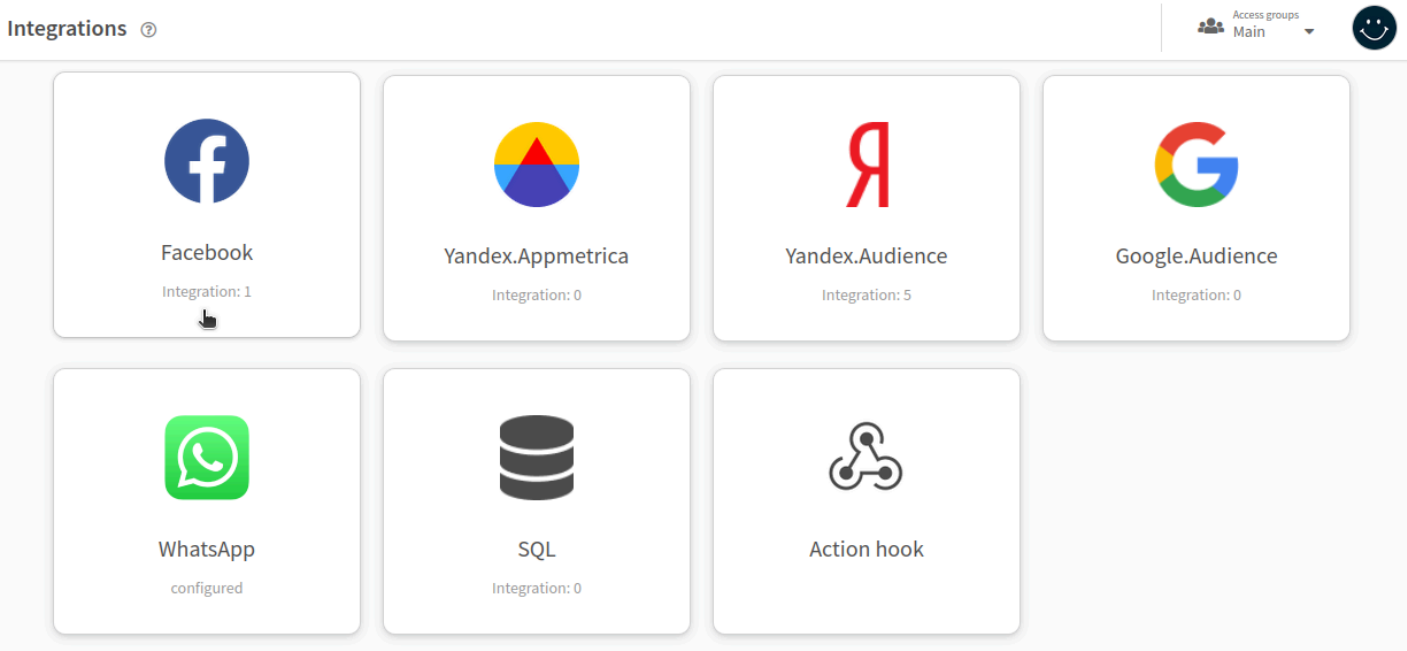
Facebook Ads Manager™

Description

Facebook* integration allows you to synchronize the data of static segments in Altcraft Marketing Platform with Facebook* custom audiences. You need to create a static segment in Altcraft MP for each Facebook* custom audience.

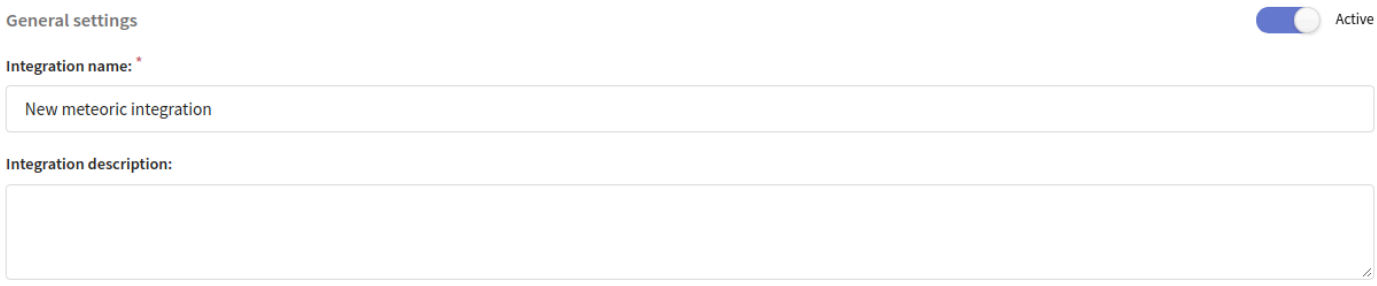
Click "Integrations in the navigation menu to add a synchronization. Then click + Create" at the top of the page and select the integration type — Facebook*.

> **TIP**
>
> **It's necessary to set up a connection in Altcraft Platform to integrate with Facebook* Ads Manager™*. You can find detailed instructions there.**

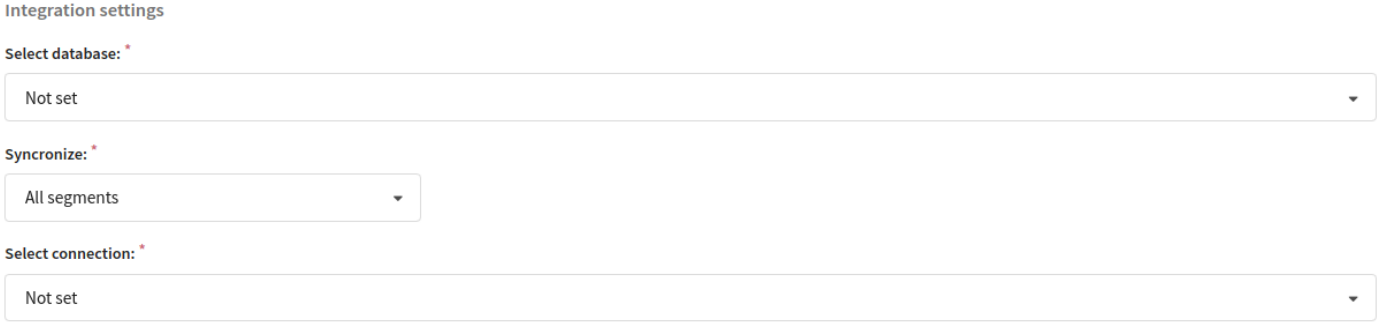


Integration settings

In the tab "General settings" enter the name and description. There is also a switch with which you can activate or stop the integration:



Set the integration scope and choosethe Facebook* connection that you configured earlier.



Type of data for profile search

Now it's time to select the type of data for profile search:

The type of data for profile search is information that Altcraft MP provides to Facebook*™. Facebook*™ will look for a match between the received data and the data of its users. Once a match is found, that user will be added to your custom audience.

Data type for search	Description
Contact data	**Email and phone number (including from subscriptions)**
Geo data	**Country, city**
Personal data	**First name, last name, birth date, gender**
External id	**Matching on the selected profile field**

By default, the search for profiles is configured by contact information.

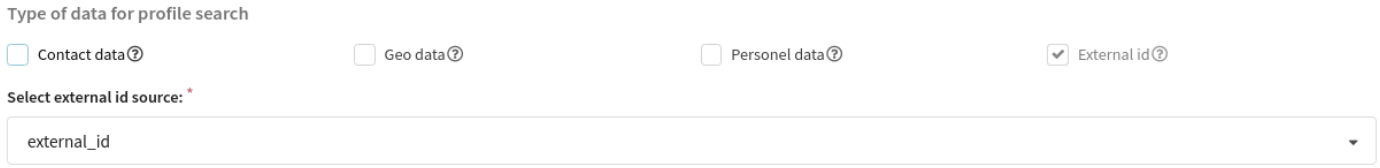
An identifier is required to find a profile. The identifier can be both contact data and an external id. That's why, you cannot select only geo data or personal data as the type of data for searching for profiles — one of two fields must be marked with them: contact data or an external id.

TIP

Sometimes, multiple profiles are created for a single subscriber imported from the Altcraft static segment in a Facebook* Custom Audience. This behaviour is caused by the fact that in Altcraft MP this profile was subscribed to several resources and each subscription was transferred to Facebook*™ separately.

External ID

If you select external id as a type of data for profile search, below you will get a new field in which you need to select the external id source.



The external id must be entered not only in Altcraft MP but also in Facebook*™.

The external id in Altcraft is a custom field in the profile database. Add this field to the database and then select it as the source of the external id.

External ID on Facebook*™ is a unique string identifier that you pass to Facebook* along with other user data. After that, Facebook* finds the user and binds this id to him. Read more about External ID on Facebook*™ here.

Facebook™ integration statistics

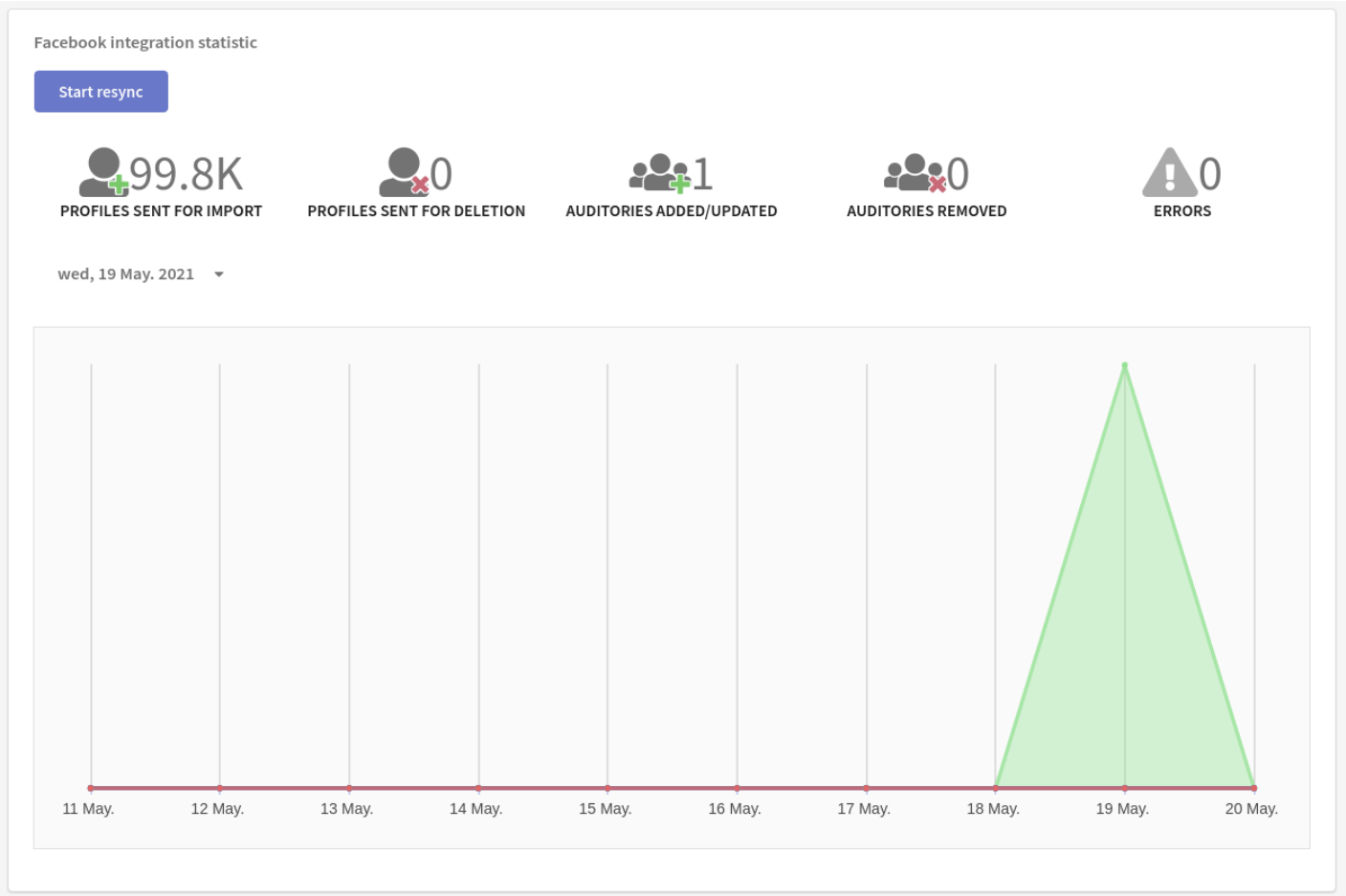
TIP: Integration schedule

It is not possible to configure a resynchronization schedule for this integration. It starts automatically when new profiles enter the segment linked to the integration.

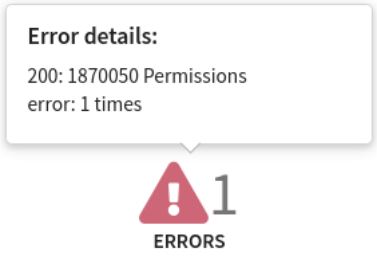
This block displays information about the number of successful operations for the specified time. You can see how many:

- **profiles sent for import**
- **profiles sent for deletion**
- **auditories added/updated**
- **auditories removed**

Use the "Start Resync" button to re-send the segment data to Facebook*™.



By the way, here you can find errors that occurred during synchronization. Click on the corresponding icon to view the error details:

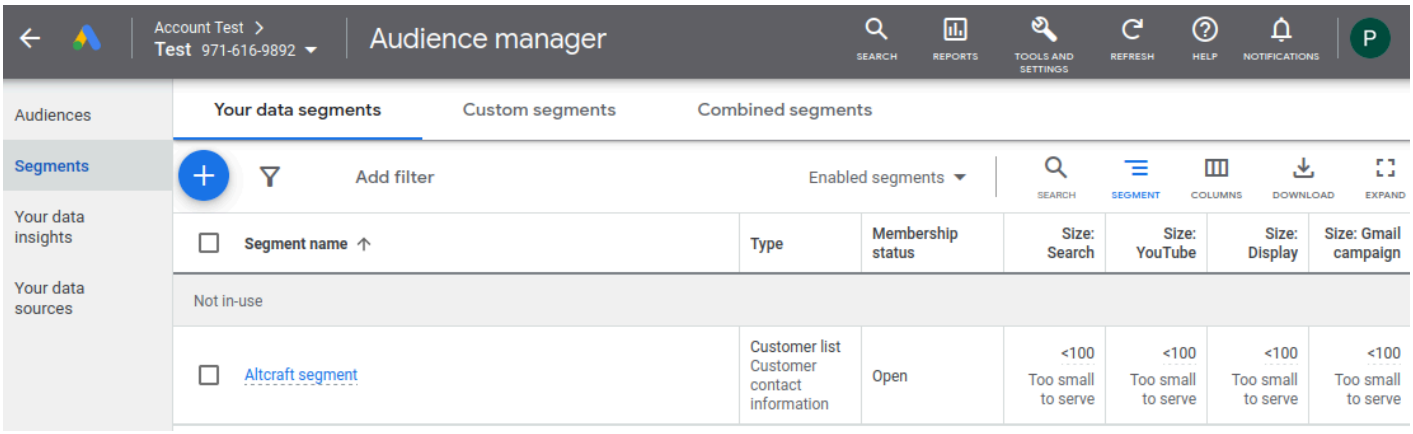


The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

Google Ads Audiences

Description

Integration with Google Ads allows you to synchronize customer data from Altcraft Platform static segments with audiences in Google Ads. After exporting the segments to your ad account, you can use the data you receive to advertise on Google Search, Gmail, YouTube, and other platforms of its advertising network.

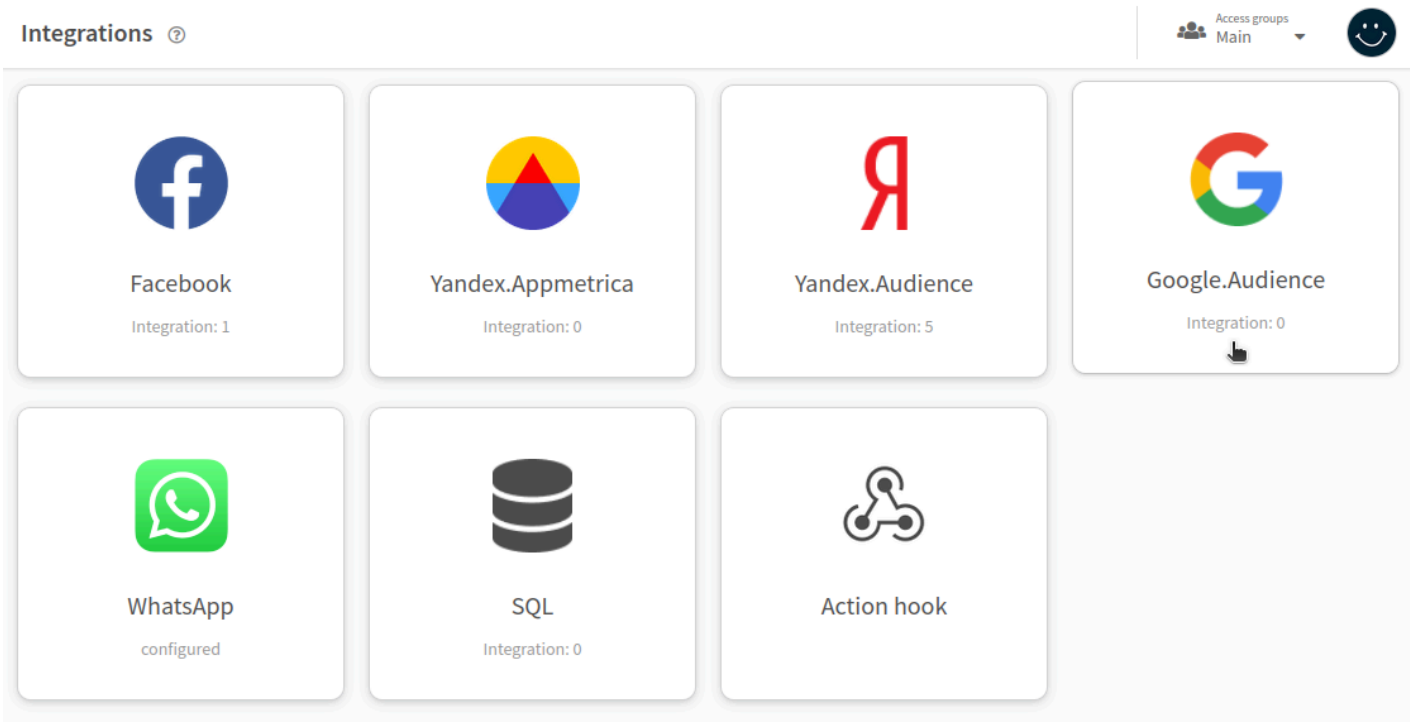


To create an integration, you need a Google Ads account and the appropriate administrator credentials.

To get started, go to the Integrations section of the main menu. Select Google.Audiences as the integration type. To add a new integration, click + Create on the top panel.

TIP

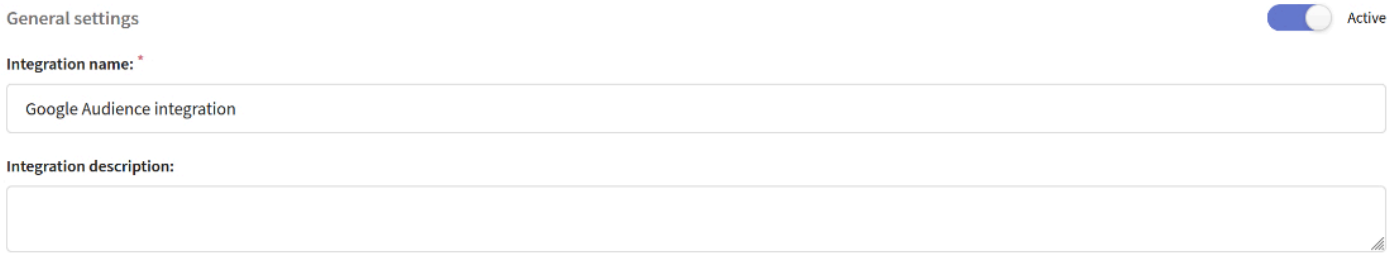
To integrate with Google Ads audiences, you need to set up a connection in Altcraft Platform. You can find detailed instructions in this article.



Settings

General settings

On the “General settings” tab, enter a name and description. There is also a toggle you can use to activate or stop the integration:

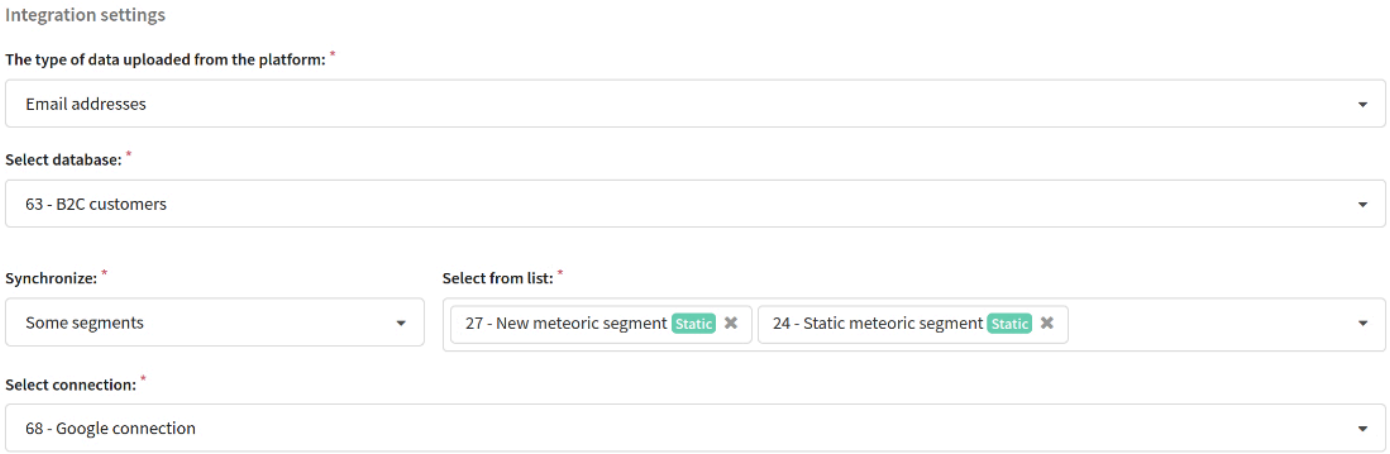


Integration settings

Below, select the type of data exported from the platform:

- **Email addresses**
- **Advertising IDs (there must be an additional field in the database that stores this ID)**
- **Phone numbers**

Set the scope of the integration. Select the connection to Google Ads you configured earlier.

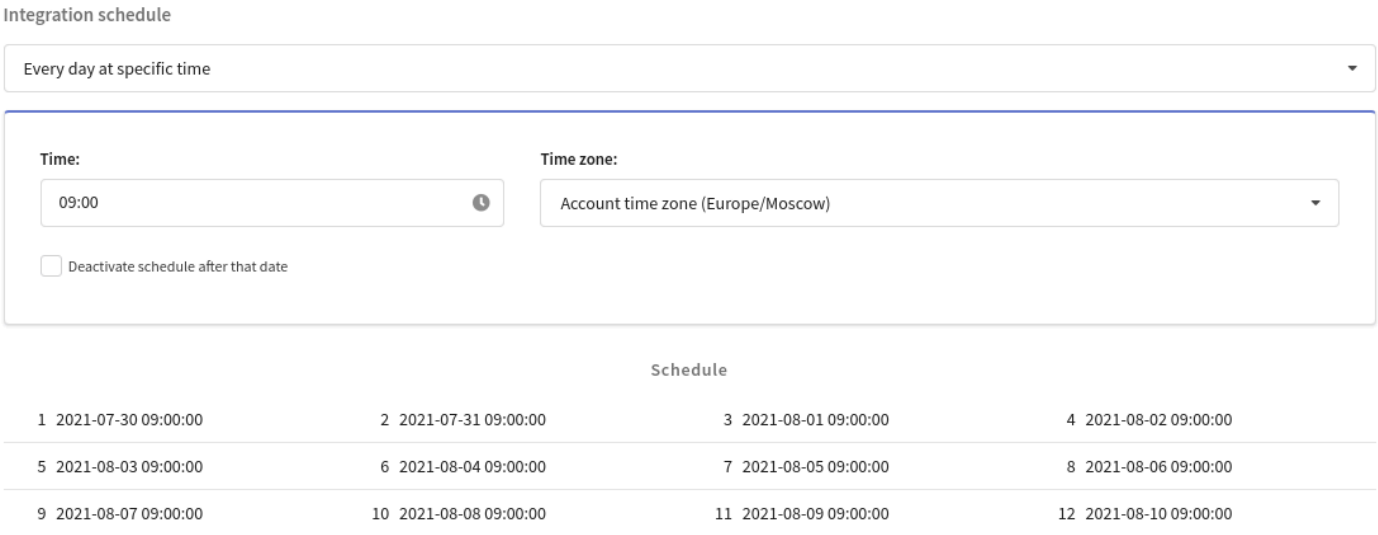


TIP

Please note that Google Ads cannot serve ads for a segment that contains less than 100 profiles.

Integration schedule

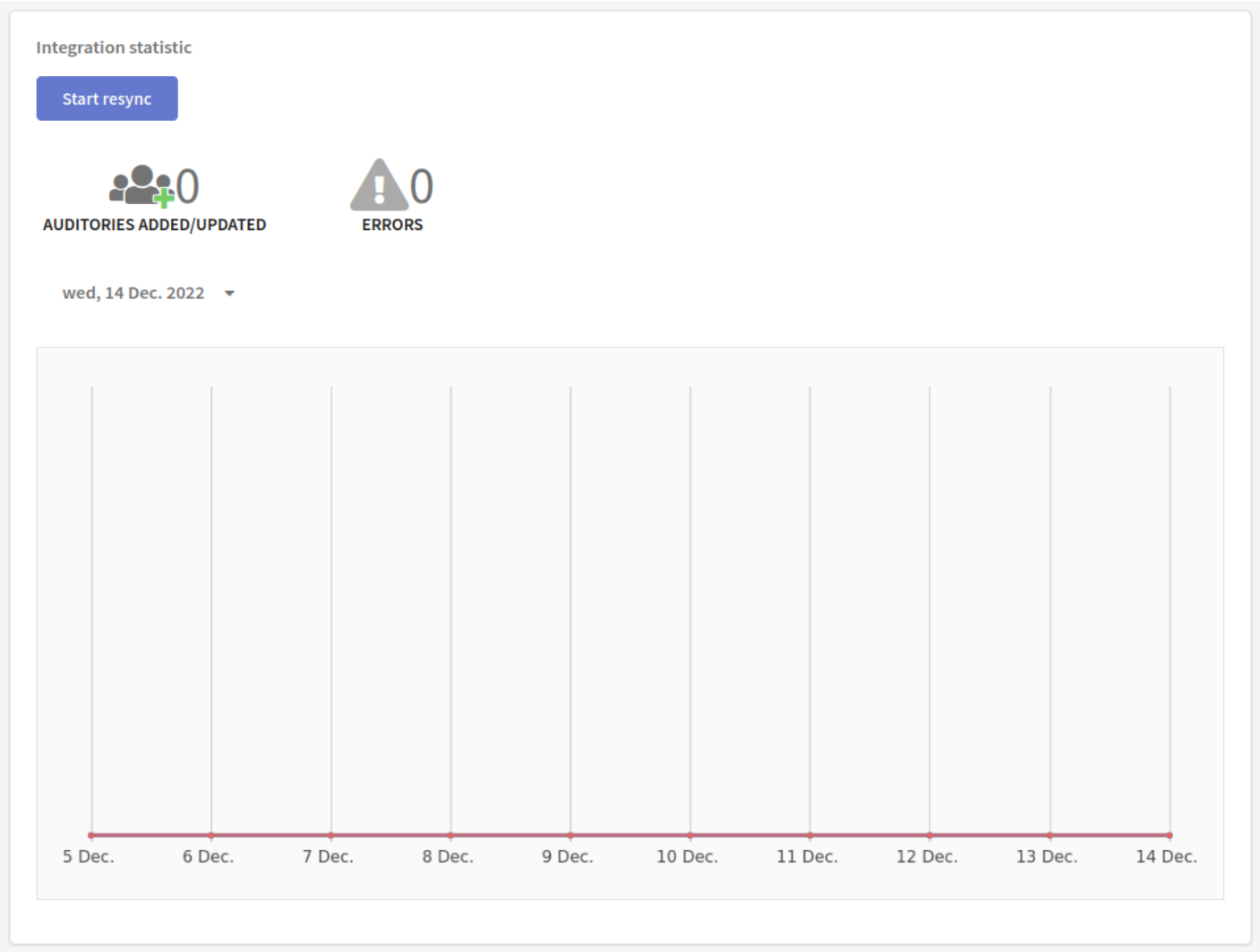
Set how often you want to update segment data. Read more about how to set up a schedule in the platform in this article.



Sync statistics

At the bottom of the page there is a statistics block. It displays information about the number of audiences that have been successfully added or updated. If errors occur during synchronization, you can view the details by clicking on the corresponding icon.

Use the "Start resync" button to send new segment data to Google Audiences.



The table at the bottom of the page shows information for each segment: the name of the Altcraft segment and the Google Ads segment it syncs with, as well as audience size and last sync status. You can view a detailed history of all syncs by clicking on the icon

.

Integration with the MAX messenger

CAUTION

When using the integration, take into account the internal rules of the MAX messenger.

TIP: MAX connection

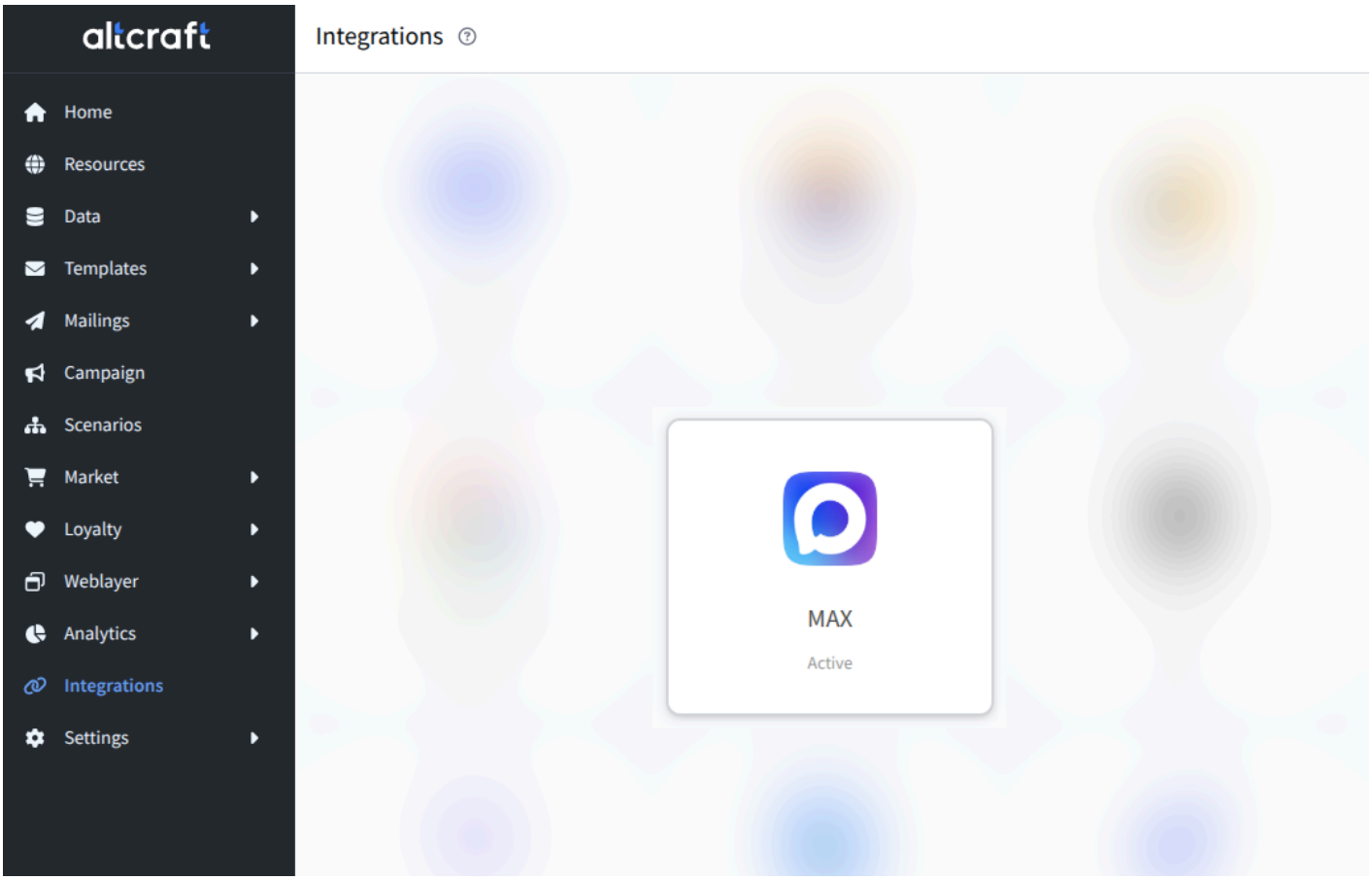
To create an integration, you must first configure a connection to the service. Instructions for connecting a MAX bot are available on this page.

Integration with MAX enables using the messenger as a communication channel for sending messages via a bot. The Altcraft platform provides two types of MAX integration:

- **bot integration — MAX Bot**
- **chat (group) integration — MAX Group**

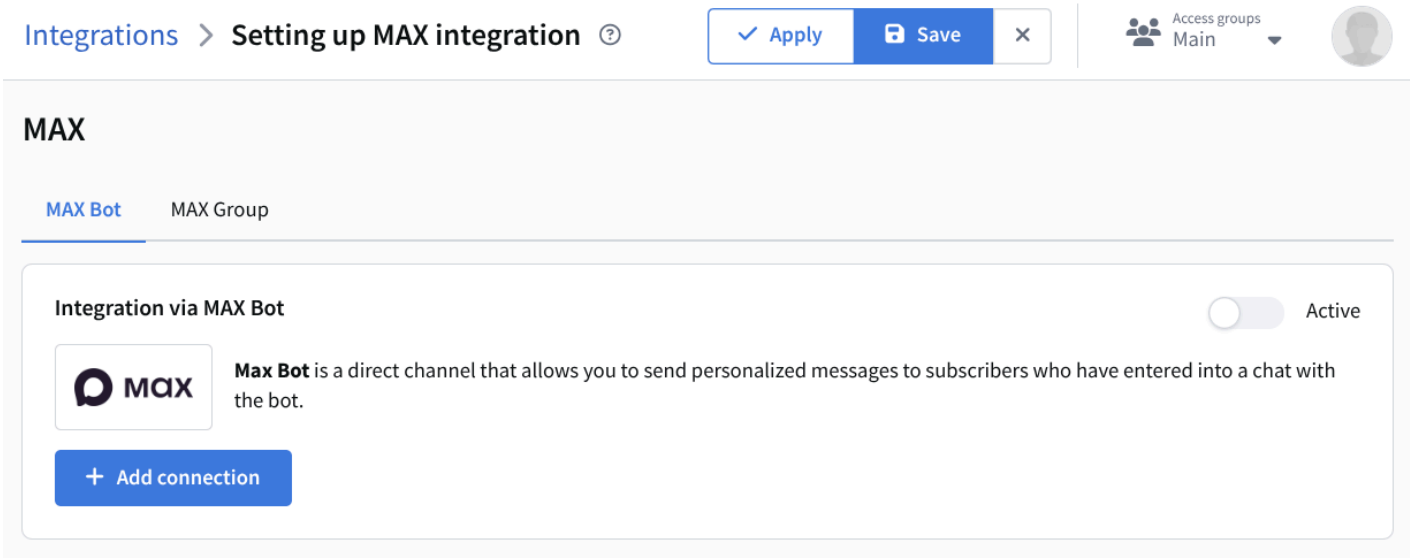
These integrations control the operation of the corresponding communication channels.

To configure it, open the Integrations section and select the integration type — MAX:

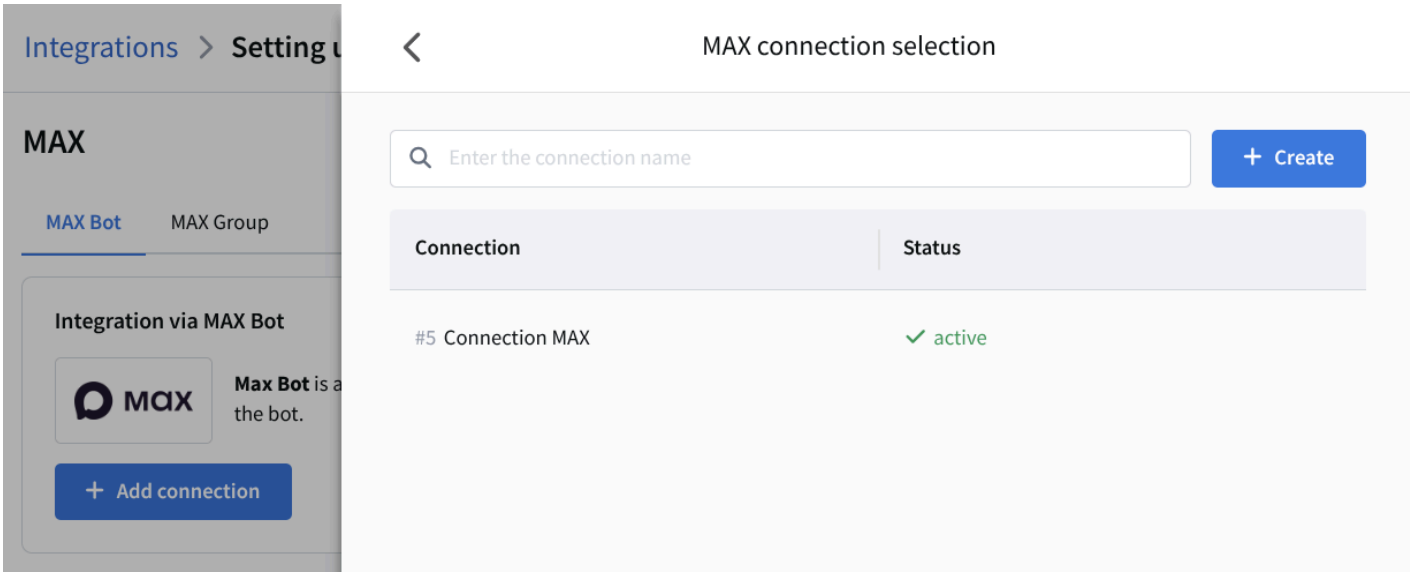


MAX Bot integration

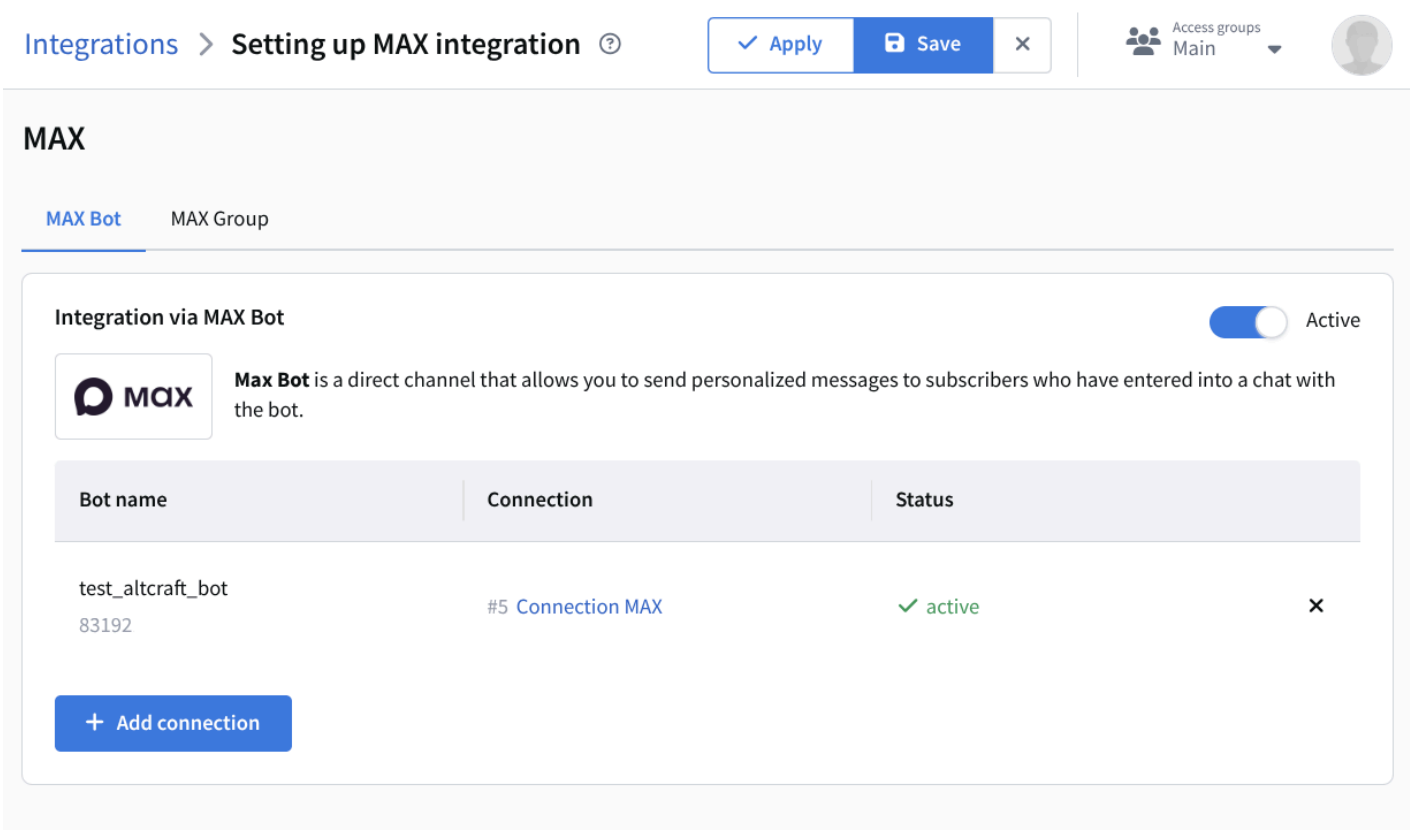
Open the MAX Bot tab. Click Add connection:



A list of created MAX bot connections will appear. Select the required connection from the list or create a new one:



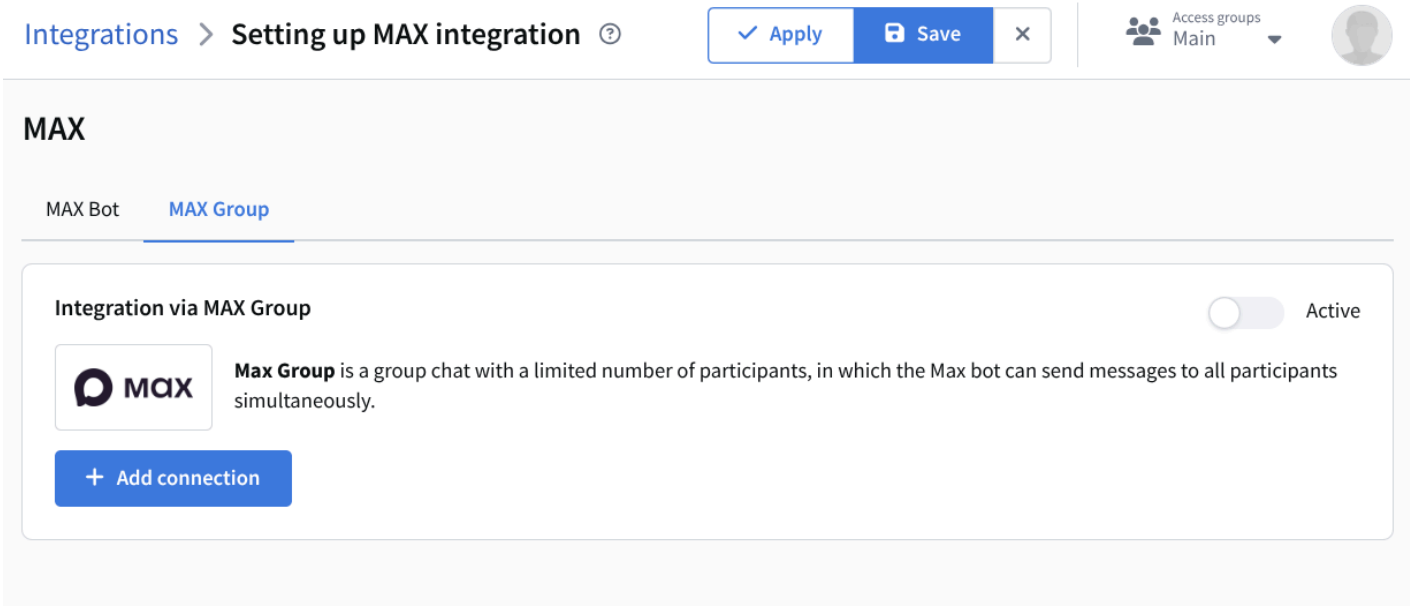
Enable the integration using the switch at the top of the screen and click Save:



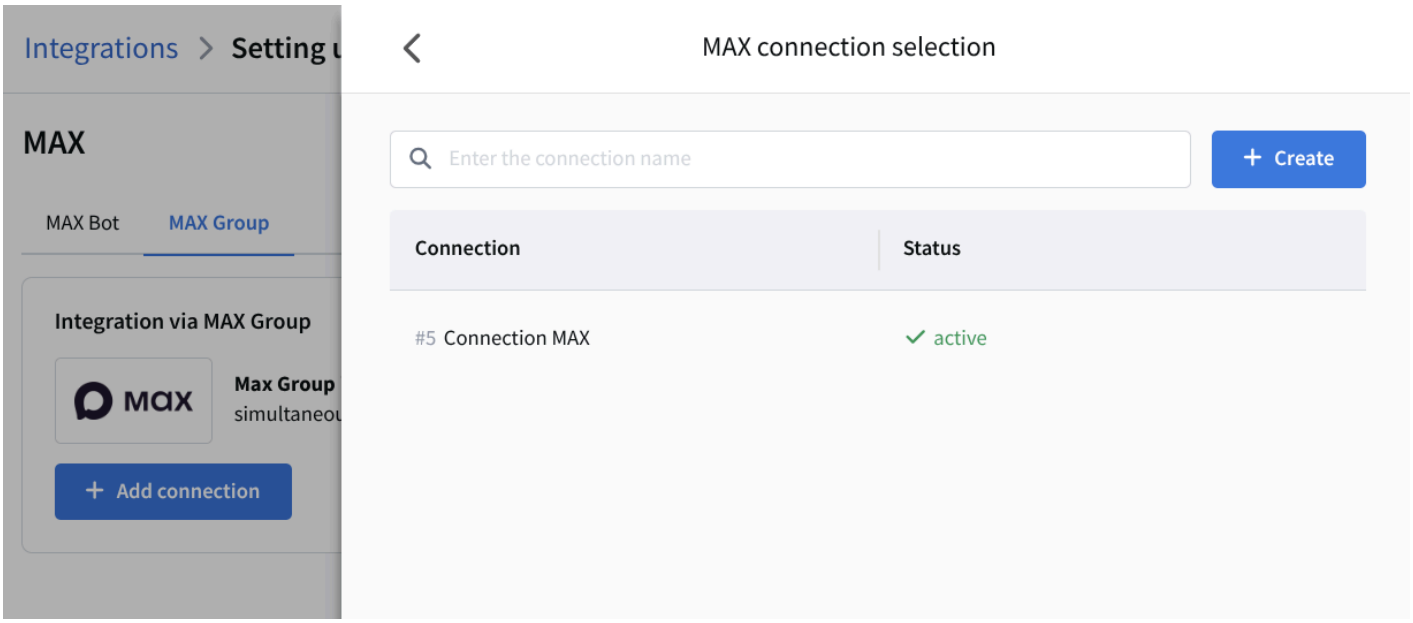
The integration is ready to use. The next step is configuring a resource for sending Campaigns via the MAX bot.

MAX Group integration

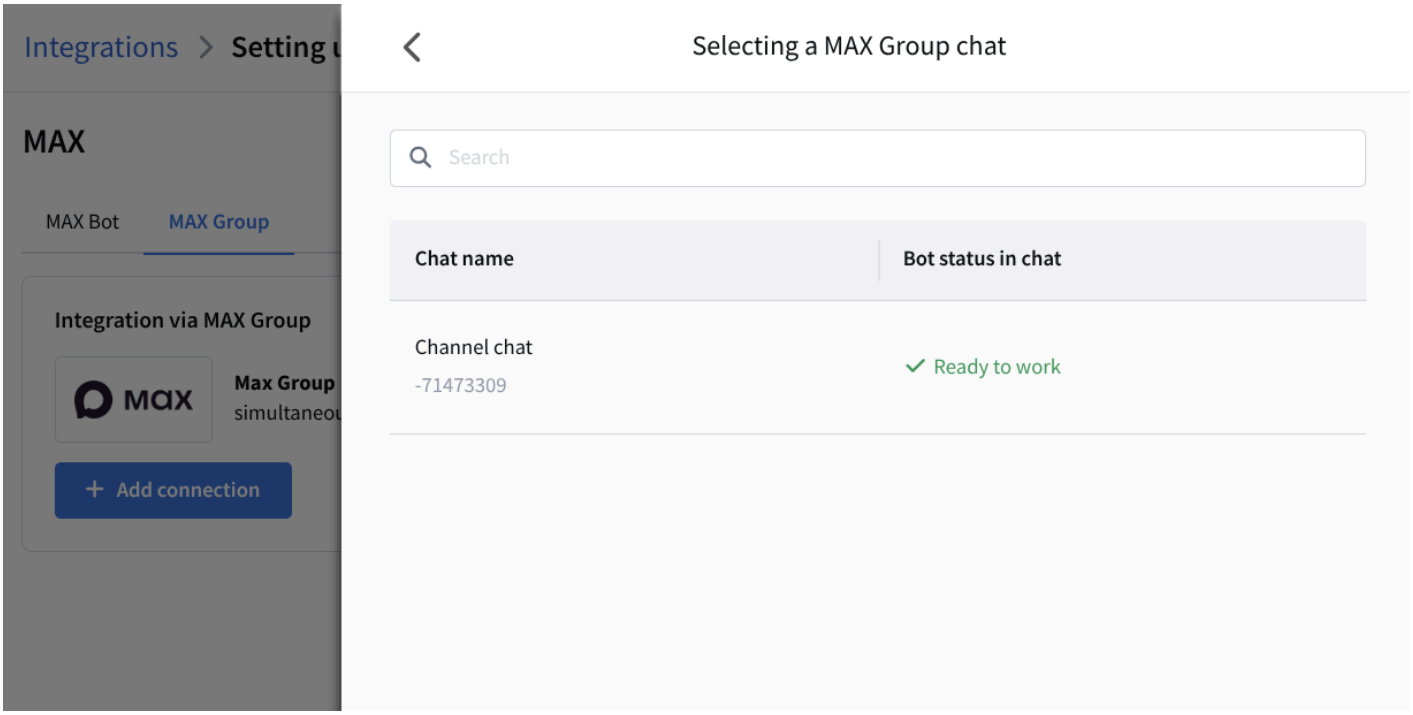
Open the MAX Group tab. Click Add connection:



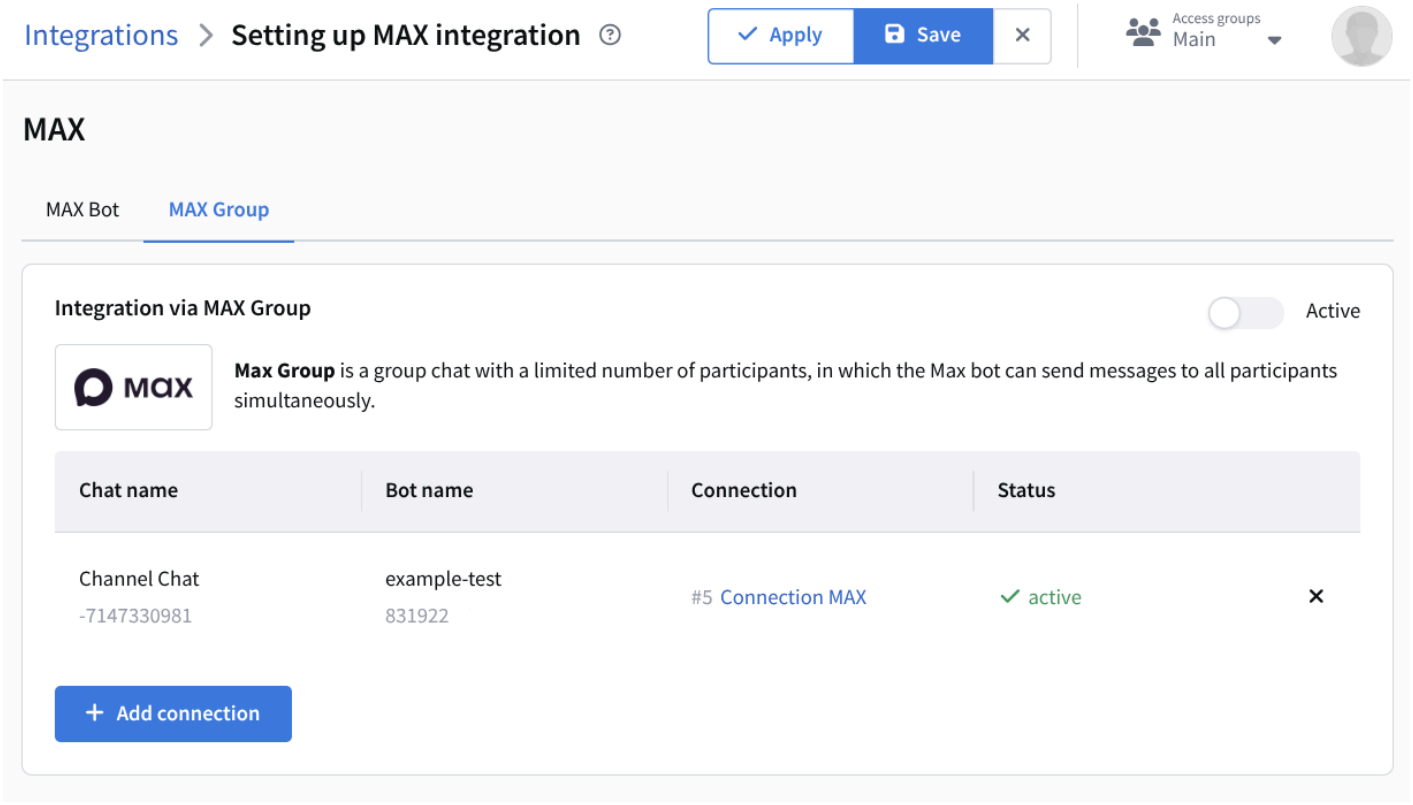
A list of created MAX bot connections will appear. Select the required connection from the list or create a new one:



Next, select the chats where messages will be sent via the MAX bot:



Enable the integration using the switch at the top of the screen and click Save:



The integration is ready to use. The next step is configuring a resource for sending Campaigns to MAX messenger chats.

Yandex.Audience™

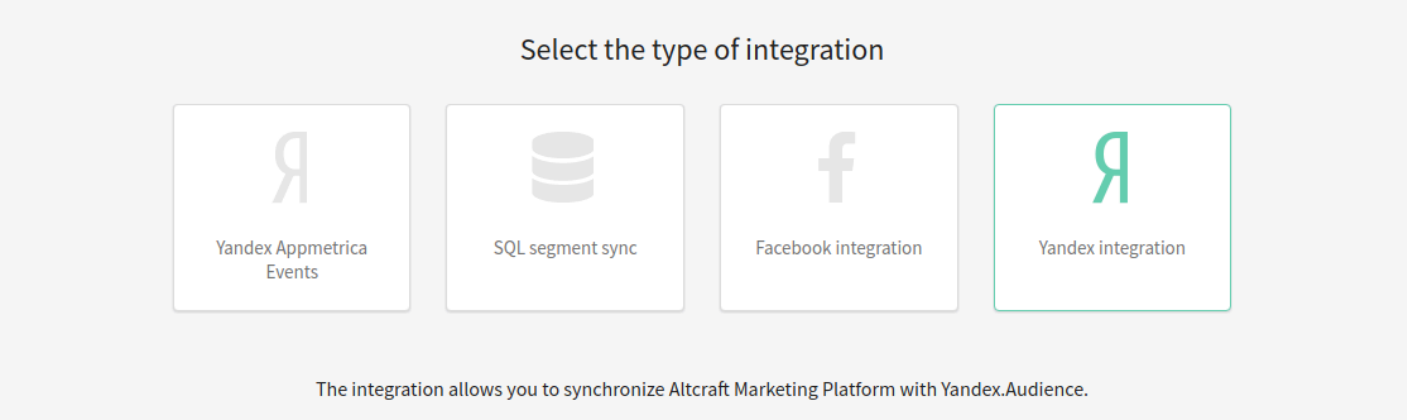
Description

Yandex.Audience™ integration allows you to synchronize the data of static segments in Altcraft Marketing Platform with segments in Yandex.Audience. In other words, you export segments from the platform to Yandex.Audience™. Then this service, using the received profile data, finds the right people and launches targeted advertising for them.

Click “Integrations” in the navigation menu to add a synchronization. Then click + Create at the top of the page and select the integration type — Yandex Integration.

TIP

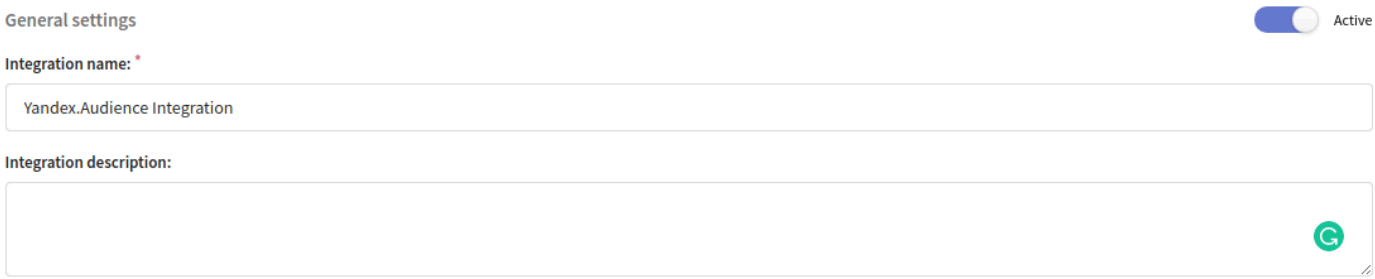
It’s necessary to set up a connection in Altcrfat MP to integrate with Yandex.Audience™. You can find detailed instructions in the article.



Integration settings

General settings

In the tab “General settings” enter the name and description of the integration. There is also a switch you can use to activate or stop the integration:

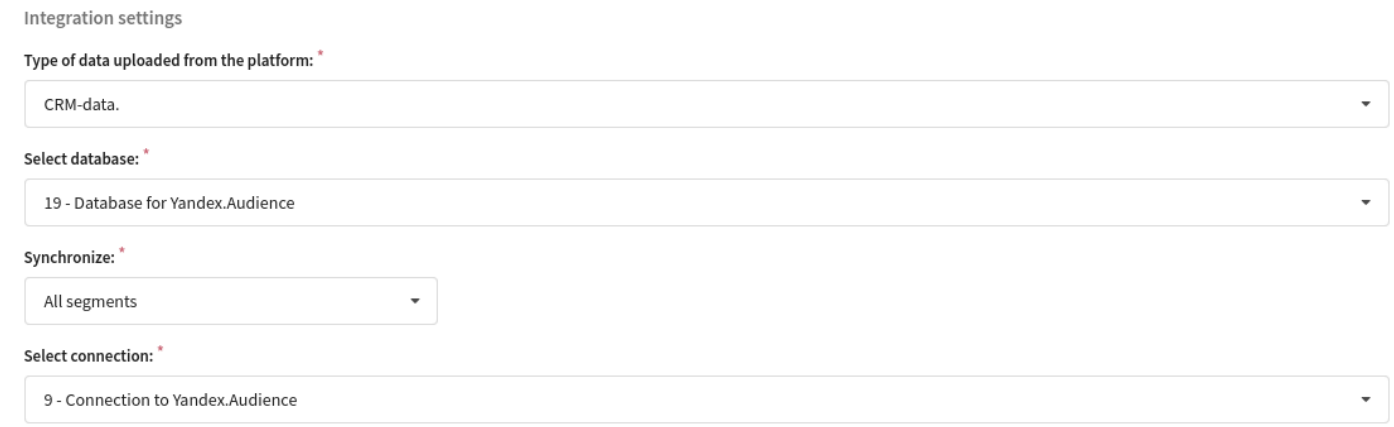


Integration settings

Choose the type of data uploaded from the platform:

- **CRM-data: email and phone number.**
- **Mobile device identifiers: data is taken from profile subscriptions. Yandex.Audience™ accepts identifier types such as AppMetrica with Google AID and AppMetrica with iOS IFA.**
- **MAC addresses: the database must have a custom field for the mac address of the profile's device.**

Set the scope of the integration. Select the connection to Yandex.Audience™ you configured earlier.

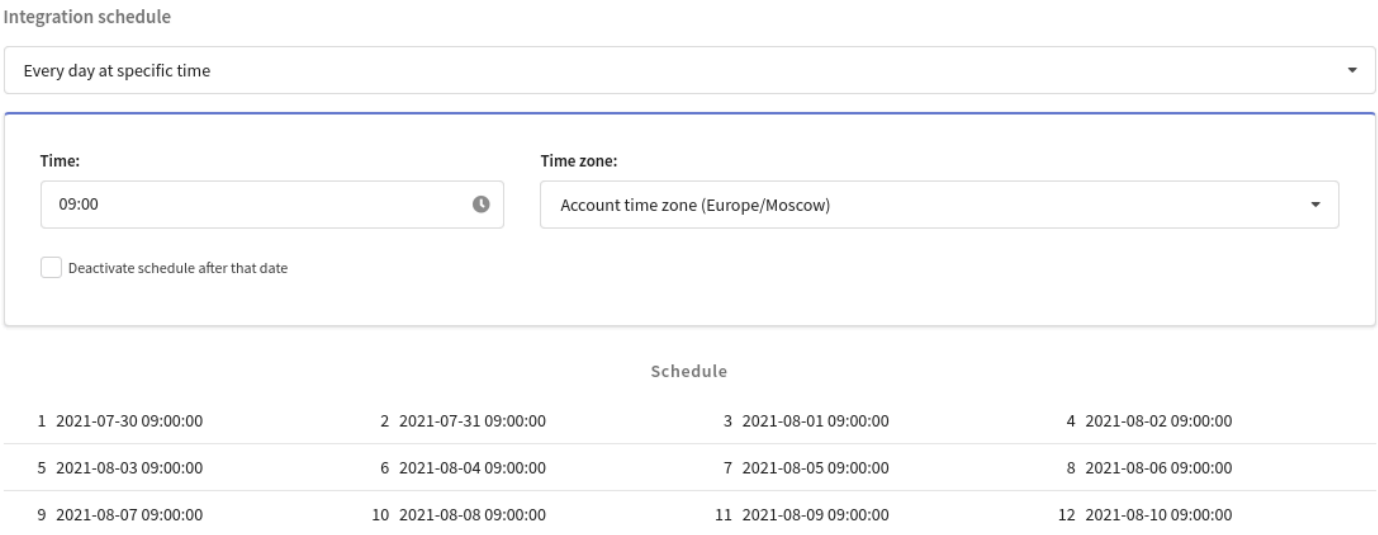


TIP

Please note that you cannot build a segment in Yandex.Audience™ that contains less than 100 profiles.

Integration schedule

Set how often you want to update segment data. For more information on how to set up a schedule in the platform, read this section.



Schedule

1 2021-07-30 09:00:00	2 2021-07-31 09:00:00	3 2021-08-01 09:00:00	4 2021-08-02 09:00:00
5 2021-08-03 09:00:00	6 2021-08-04 09:00:00	7 2021-08-05 09:00:00	8 2021-08-06 09:00:00
9 2021-08-07 09:00:00	10 2021-08-08 09:00:00	11 2021-08-09 09:00:00	12 2021-08-10 09:00:00

Sync statistics

This block displays information about the number of successful operations for the specified time. You can see how many:

- **profiles imported**
- **profiles removed**
- **segments imported**
- **segments removed**

Use the “Start Resync” button to re-send the segment data to Yandex.Audience.

Sync statistics

Start resync

38 PROFILES IMPORTED

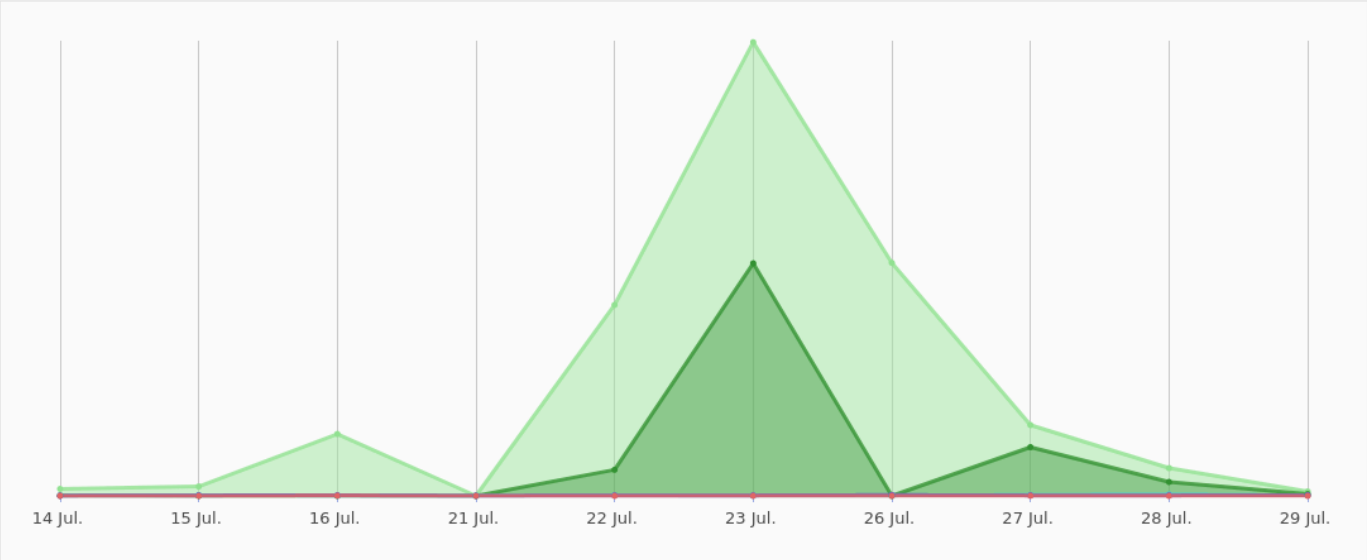
16 PROFILES REMOVED

4 SEGMENTS IMPORTED

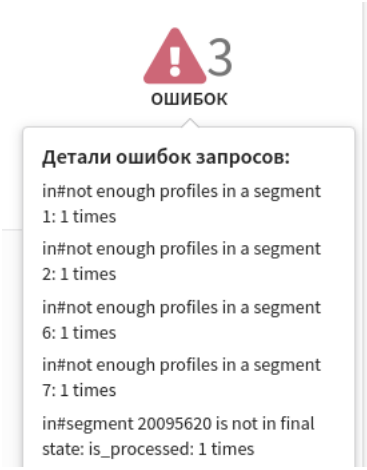
6 SEGMENTS REMOVED

0 ERRORS

Today



By the way, here you can find errors that occurred during synchronization. Click on the corresponding icon to view the error details:



The table shows the current statuses of the segments in which they are in Yandex.Audience™ at the moment:

Auditories information

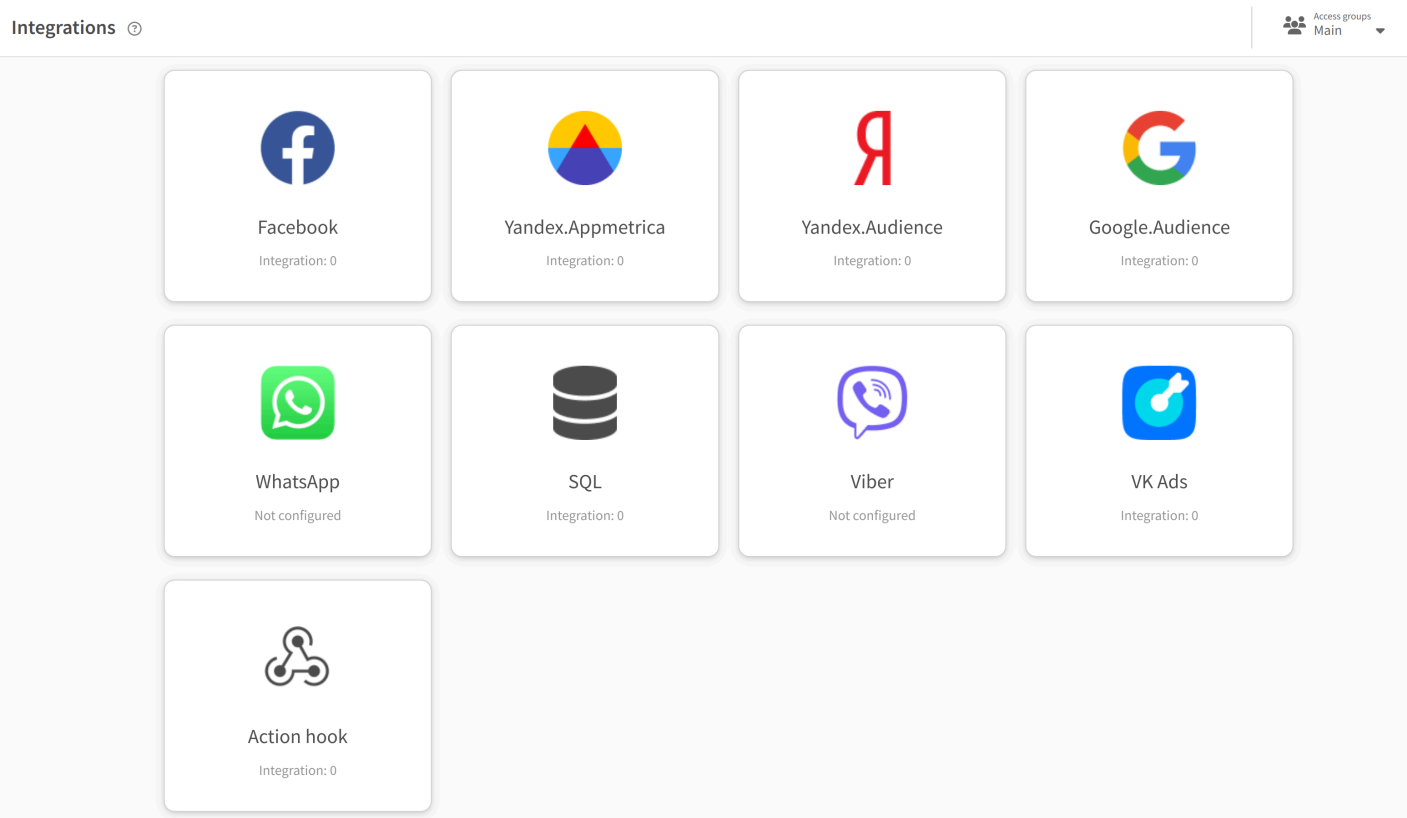
Segment	Auditory	Amount	Status	Last sync
#91 seg 1 ya.Au	seg 1 ya.Au (20054702)	772	Ready	2021-07-21 08:59:21
#92 seg 2 ya.Au	seg 2 ya.Au (20054703)	806	Ready	2021-07-21 08:59:22
#93 seg 3 ya.Au	seg 3 ya.Au (20054704)	877	Ready	2021-07-21 08:59:23

VK Ads™

Integration with VK Ads™ allows you to synchronize data from Altcraft Platform static segments with audiences in the Vkontakte advertising account. In other words, you transfer segments from the platform to VK Ads™ so that this service, using the received profile data, can then find the specific groups and launch targeted advertising for them.

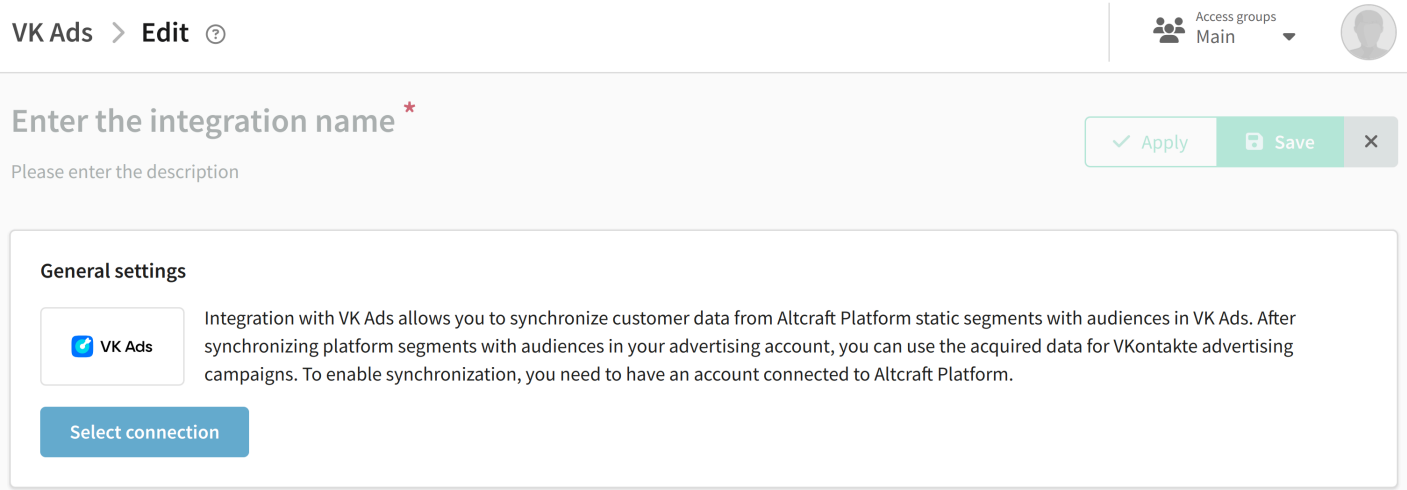
> **TIP**
>
> **To integrate with VK Ads™, you need to configure a connection in the Altcraft platform. You will find detailed instructions here.**

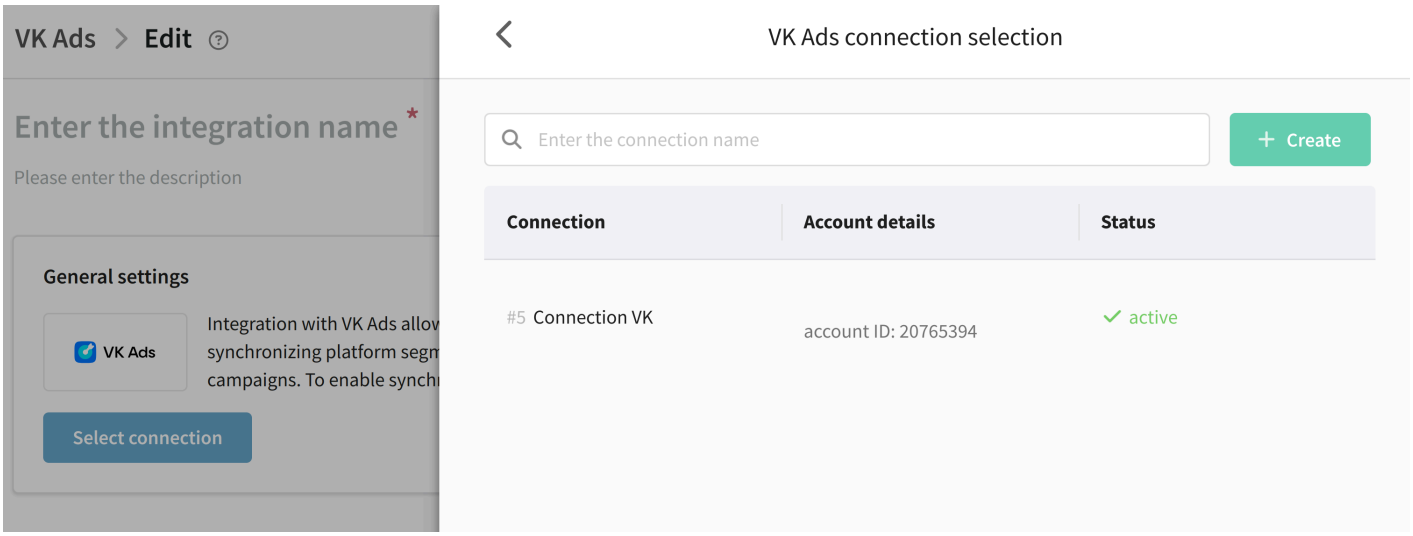
To add synchronization, click "Integrations" in the main menu. Select the type of integration — VK Ads, and then at the top of the page click the Create button.



Settings

The integration creation window will open. Click on the Select connection button to view existing connections:

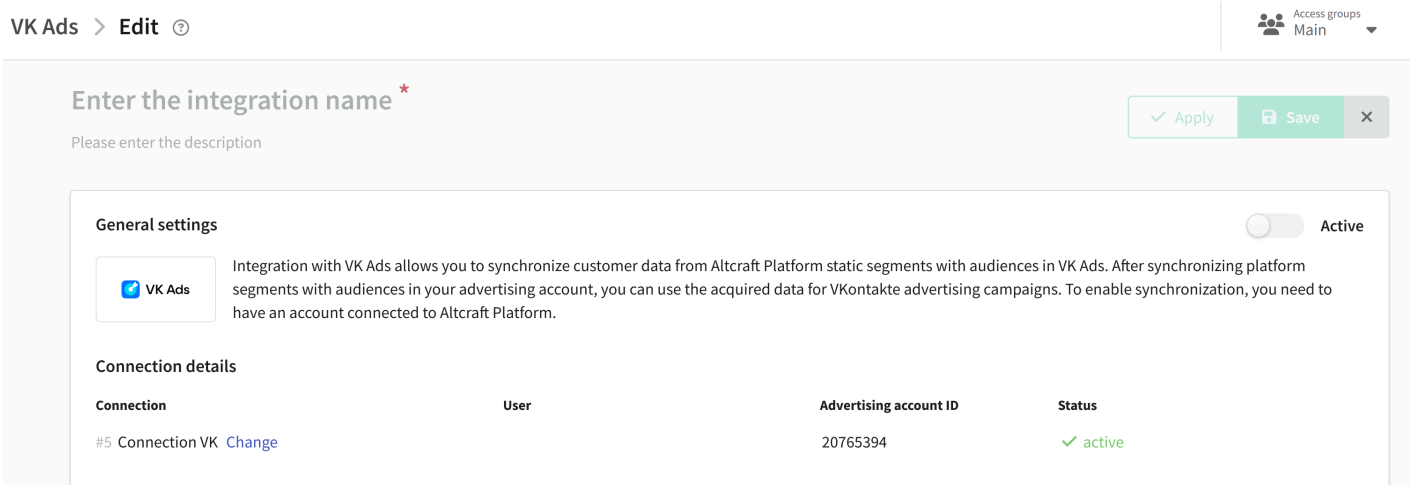




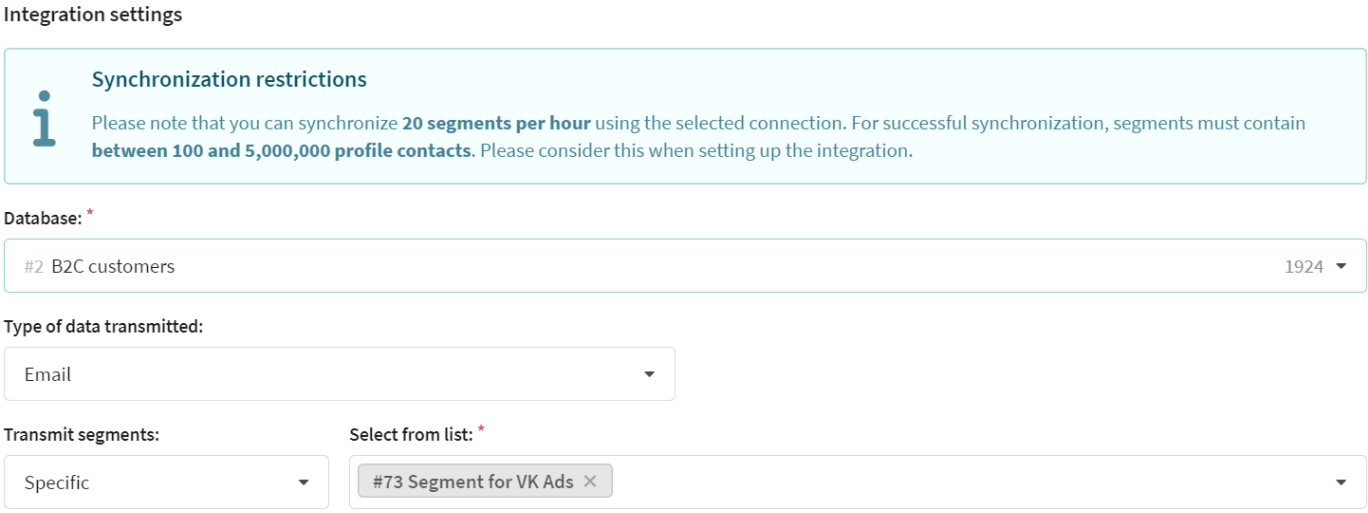
From this menu you can create a new connection. You can find detailed instructions for creating a connection [in this article](#).

General settings

On the General settings tab, enter a Name and Description. There is also a switch here that you can use to activate or stop the integration.



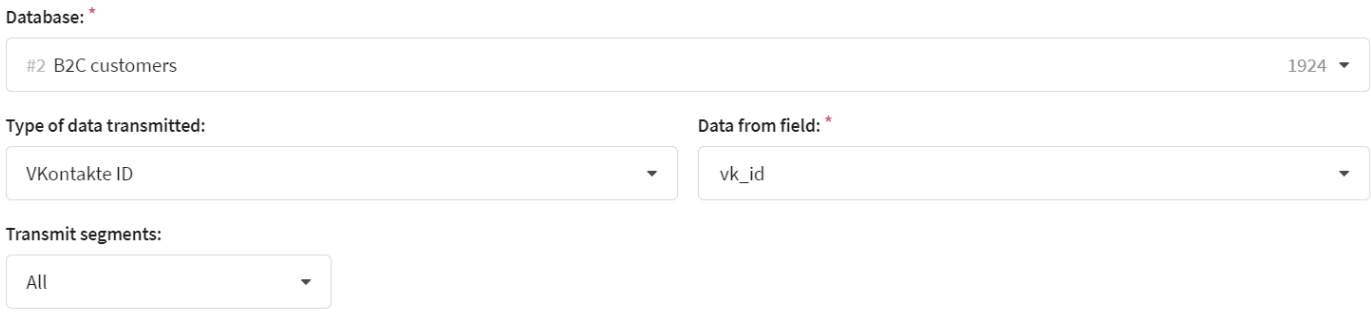
Integration settings



In this section, you can configure one Database and up to 20 Segments. It's impossible to transfer a segment which contains less than 100 profiles. Then select type of data transmitted. VK Ads™ supports the integration of the following types of data:

- **VK ID**
- **email**
- **OK ID**
- **Phone numbers**
- **Google Advertising ID**
- **Device IDFA**

For each type, except for email and phone number, you have to create additional field in database.

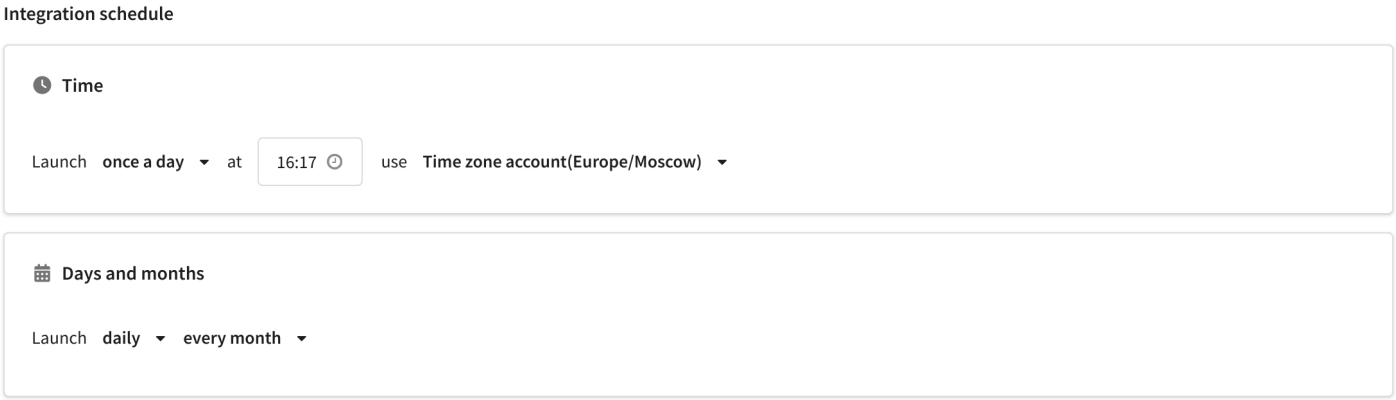


CAUTION

The integration will transfer data only from the selected database. If your segment affects several databases, then only those profiles that are included in the database you initially specified will be imported.

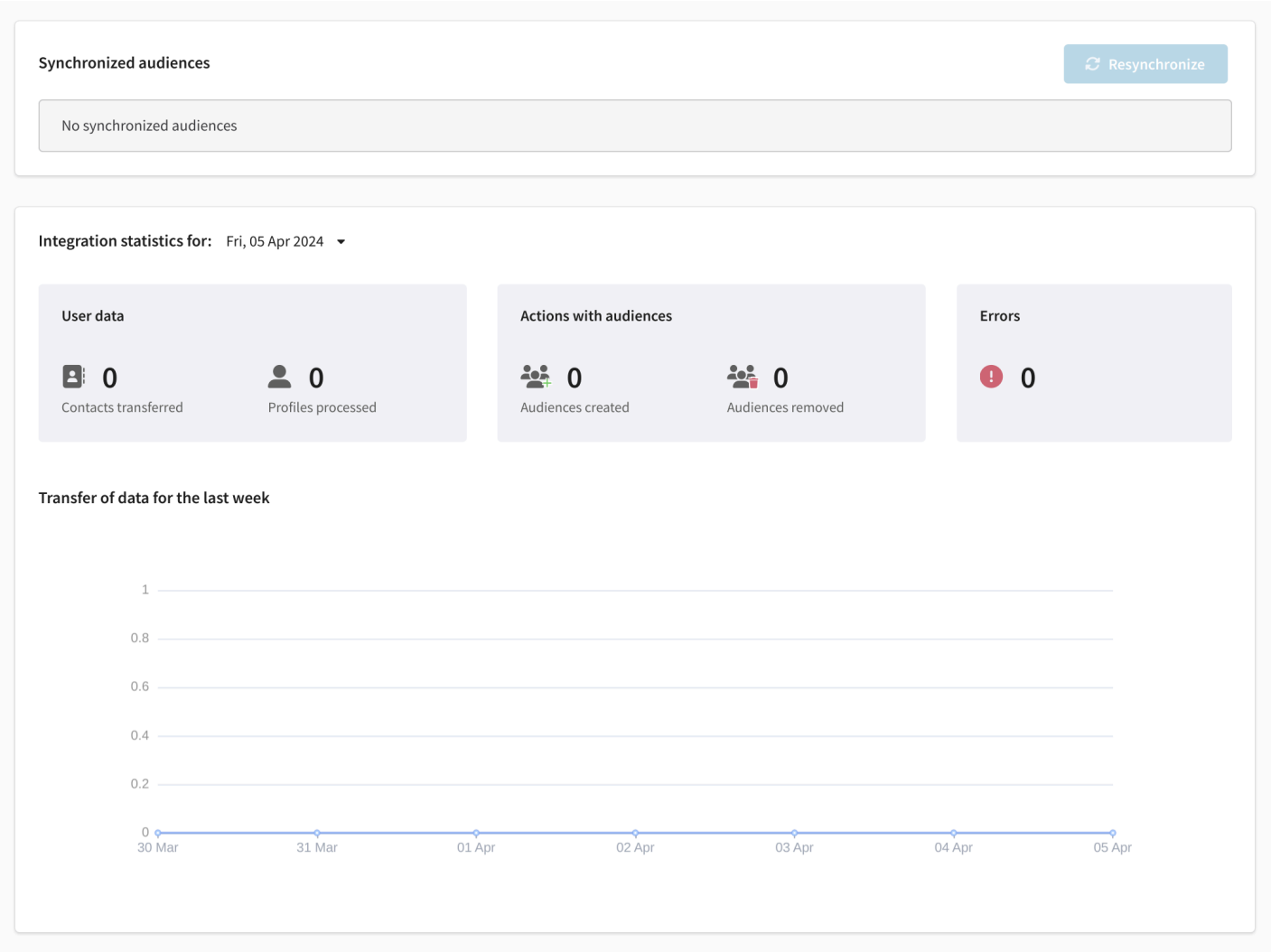
Integration schedule

Here you can configure how often the integration will occur. Read more about how to set up a schedule in Altcraft Platform in this article.

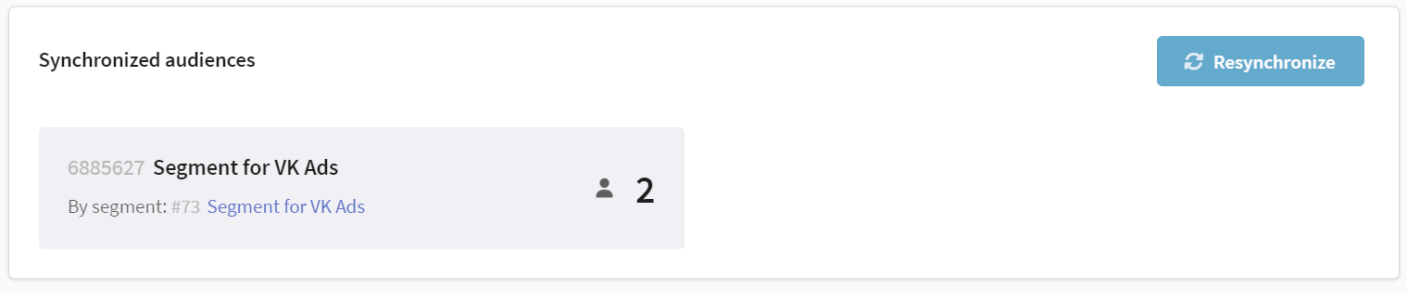


Synchronization and statistics

After you save the integration, the Synchronized audiences and Integration statistics sections will appear on the settings page:



A list of audiences transferred to VK Ads™ will appear in the Synchronized audiences section. You can also synchronize audiences by clicking the Resynchronize button. After clicking it, all ever synchronized audiences will be deleted and all segments that were within the scope of integration will be re-uploaded.



TIP

The integration transfers only those profiles that have appeared since its launch. For the most complete transfer of audiences, we recommend running Resynchronize before starting the integration

In the Integration statistics section you can see detailed information about synchronizations over the last week, what actions were performed with them and errors that occurred during synchronization.

Integration statistics for: Fri, 24 May 2024

User data

1926
Contacts transferred

1924
Profiles processed

Actions with audiences

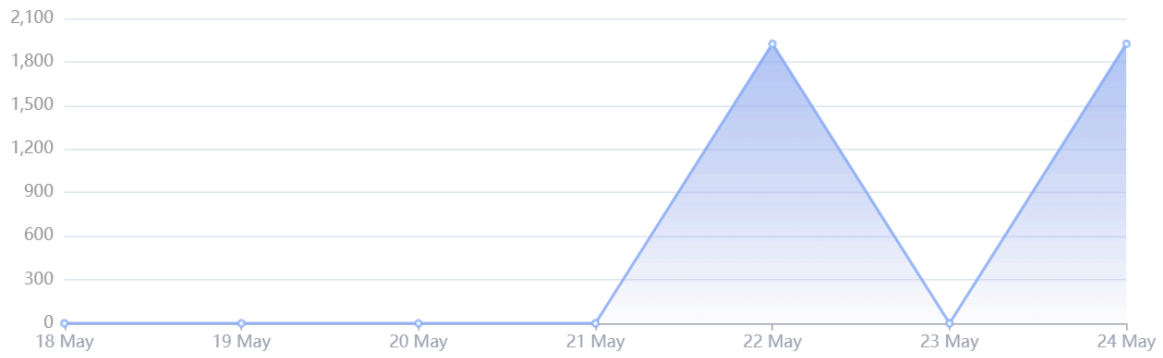
1
Audiences created

0
Audiences removed

Errors

0

Transfer of data for the last week



TIP

Contacts transferred tab counts every transferred communication channel of profile. For example, if a profile has a phone number, email and VK ID, then 3 contacts will be included in the statistics, not 1.

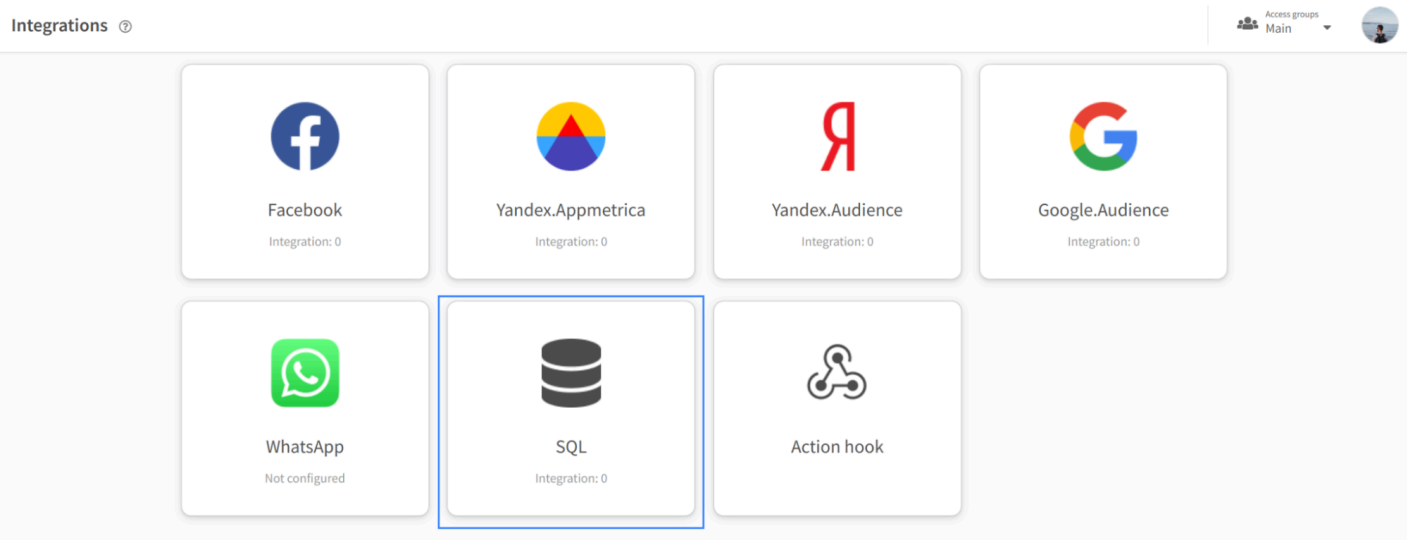
Static segment synchronization

Description

Data from static segments can be synchronized with external SQL data tables. This integration is utilized when you need to maintain up-to-date information about customer groups in external systems. Data updates in the external system occur based on specific events.

- **Adding profiles to a static segment.**
- **Removing profiles from a static segment.**
- **Clearing (removing all profiles from) a static segment.**
- **Creating or updating a static segment.**
- **Deleting a static segment.**

To add synchronization, select Integrations in the left menu. Then click on SQL.



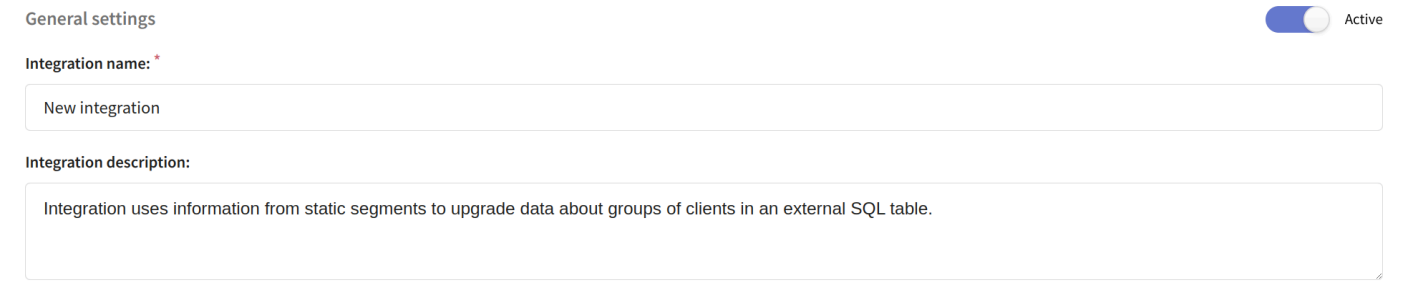
Now click on + Create on the top of the page.

TIP

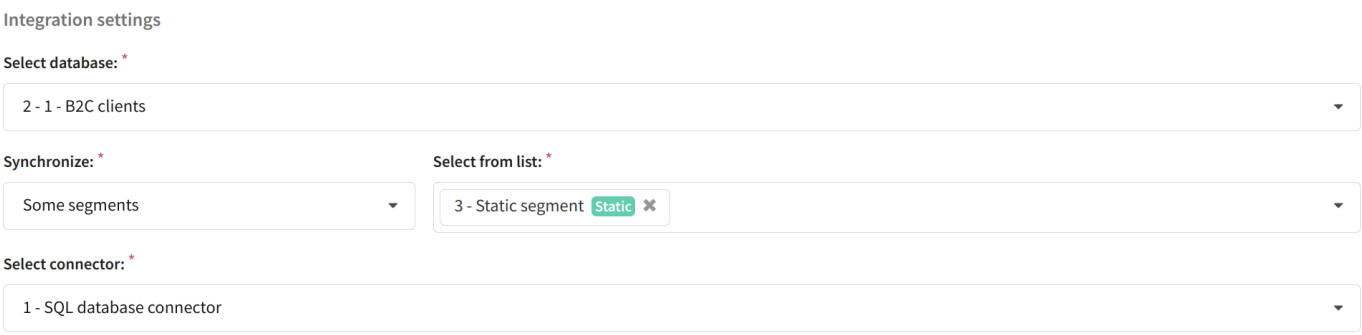
To synchronize within the Altcraft Administrative Panel, you need to create a connector to an external SQL storage. You can find instructions on adding a connector here.

Integration settings

In the main integration settings, enter the Integration name and Integration description. You'll also find a toggle switch here that allows you to activate or pause the integration.



Now set the integration scope. Below in the same section, select the SQL connector for the storage with the table where the data will be exported to.



SQL queries for event synchronization

Below, you need to configure SQL queries that will add or remove data from an external table based on events related to the selected static segments.

TIP

Queries can involve additional profile data and logical expressions. The matching of profile fields and data table columns is determined by the WHERE operator.

Adding and removing profiles, segment clearing

CAUTION

Queries for adding and removing profiles are mandatory for integration.

Events of adding or removing customer profiles from a static segment can be exported individually or in batches. When using batches, the number of added profiles (Batch Size) and a timeout (Timeout) are set. After the timeout, synchronization will be triggered even if the required number of profiles hasn't accumulated. Default timeout value — 60 seconds.

Example of data table structure

INFO

`segId` **— static segment identifier**

`clientId` **— unique client identifier in the company's data center**

`updated` **— time of profile addition or removal**

```
mysegment

+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| segId    | int(11)      | YES  |     | NULL    |       |
| clientId | varchar(255) | YES  |     | NULL    |       |
| updated  | datetime     | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

Example of a batch profile addition query

INFO

`{segment_id}` **— segment identifier in the platform**

`{$lead.CustomID}` **— custom (additional) profile field.**

The batch is configured using the following cycle:

```
{for $index, $lead = $batch}
    ({segment_id}, "{$lead.CustomID}", NOW()){autocomma} // Here fields for export are specified
  {end}
```

Altcraft variables and functions

```
INSERT INTO mysegment (segId, clientId, updated)
  VALUES
  {for $index, $lead = $batch}
    ({segment_id}, "{$lead.CustomID}", NOW()){autocomma}
  {end}
  ON DUPLICATE KEY UPDATE
    updated = NOW();
```

Example of a batch profile removal query

INFO

`{segment_id}` **— segment identifier in the platform**

`{$lead.CustomID}` **— custom (additional) profile field.**

Altcraft variables and functions.

```
DELETE FROM mysegment
WHERE segment_id = {segment_id}
AND clientId IN ({batch_in "CustomID"})
```

Example of a query to remove all profiles from a segment

INFO

`{segment_id}` **— segment identifier in the platform**

Altcraft variables and functions.

```
DELETE FROM mysegment
WHERE segment_id = {segment_id}
```

Segment Addition/Update and Deletion

Additionally, you can configure the export of events for adding a new segment, updating existing data, and segment deletion. Segment data is exported to a separate SQL table:

Example of data table structure

INFO

`name` — **name of the static segment**

`segment_id` — **unique ID of the segment in Altcraft**

`updated` — **event date**

```
segments

+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| name        | varchar(255) | YES  |     | NULL    |       |
| segment_id  | int(11)      | YES  |     | NULL    |       |
| updated     | datetime     | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
```

Example of a query for adding/updating a segment

```
INSERT INTO segments (segment_id,name,updated)
  VALUES
     ({segment_id}, "{segment_name}", NOW())
  ON DUPLICATE KEY UPDATE
    name = "{segment_name}",
    updated = NOW();
```

Example of a query for deleting a segment

INFO

`{segment_id}` — **segment identifier in the platform**

Altcraft variables and functions.

```
DELETE FROM segments
WHERE segment_id = {segment_id}
LIMIT 1
```

INFO

When transferring profile data to an external database using variables, single quotes `'` will be escaped. The escape symbols used depend on the type of database:

Database	Escape symbol	Example
MySQL, MSSQL, PostgreSQL, Oracle	`'` **(single quote)**	`value''s`
ClickHouse	`\` **(backslash)**	`value\'s`

Synchronization Statistics

In the Sync statistics section, information about the number of successful SQL queries within the specified time will be displayed, along with synchronization execution errors.

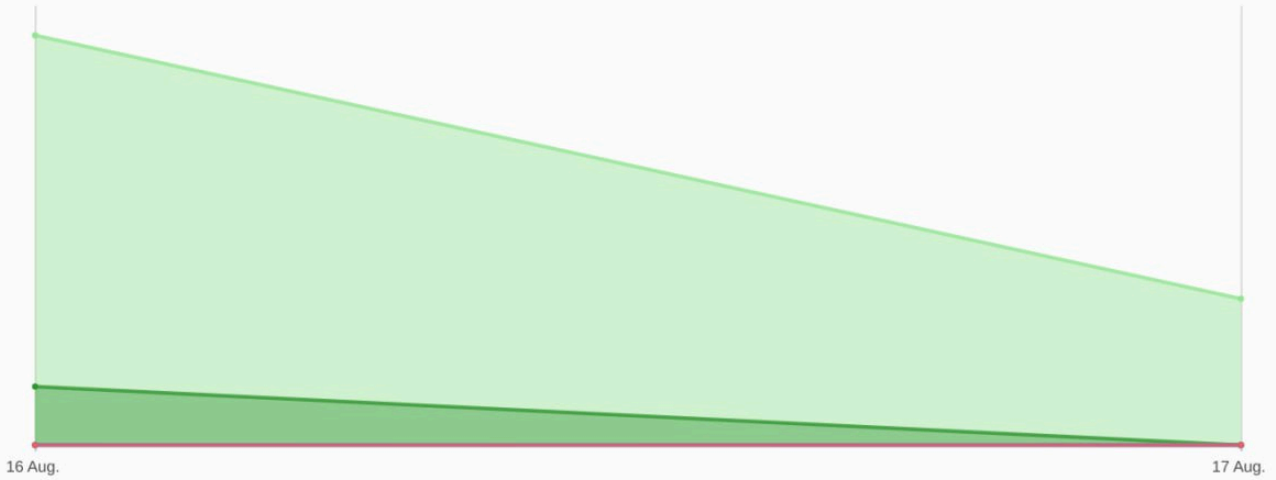
Using the Start resync button, you can recalculate segment data for export.

Sync statistics

Start resync

14K	2K	3	1	0
PROFILES IMPORTED	PROFILES REMOVED	SEGMENTS IMPORTED	SEGMENTS REMOVED	ERRORS

wed, 16 Aug. 2023



Auditories information

Segment	Auditory	Amount	Status	Last sync
#154 Segment 1	Segment 1 (31646481)	unknown	Processing	2023-08-17 10:27:12

Yandex AppMetrica™

Description

Yandex AppMetrica™ is an analytics platform for mobile applications. It also manages mobile push notifications using Google Firebase SDK.

Web-site: AppMetrica: https://appmetrica.yandex.ru

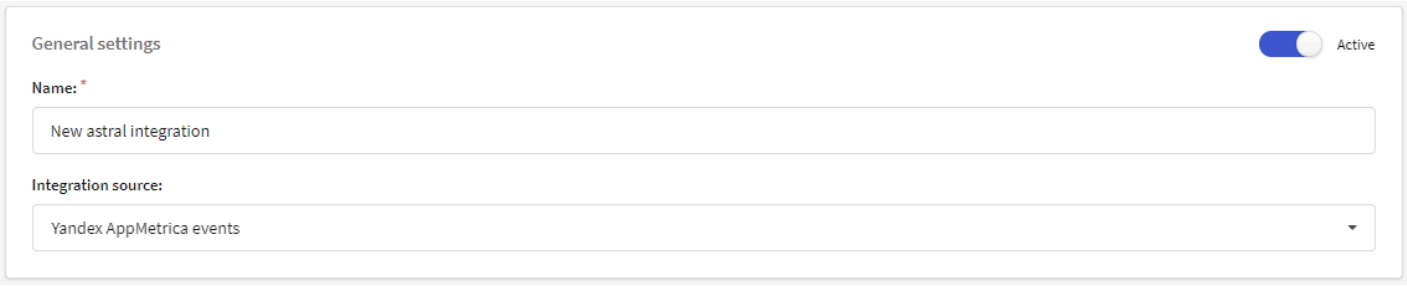
Altcraft MP captures AppMetrica™ installations and events information from Logs API, that can be used for:

- **goals and their values registration**
- **profiles import and update**

To start using AppMetrica data in Altcraft, go to Integrations in the main menu.

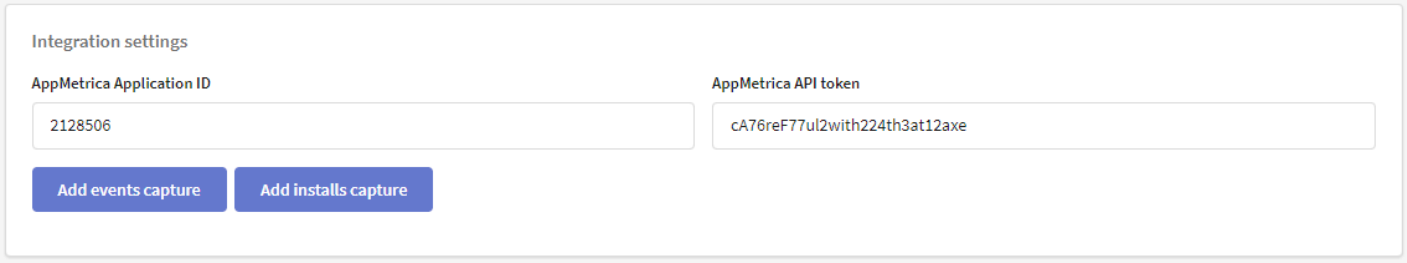
Accessing AppMetrica API

Create a new integration. In General settings give it a custom Name and select AppMetrica events from Integration source dropdown. You can activate or stop the integration by a special button in this section:



In Integration settings enter your application AppMetrica credentials:

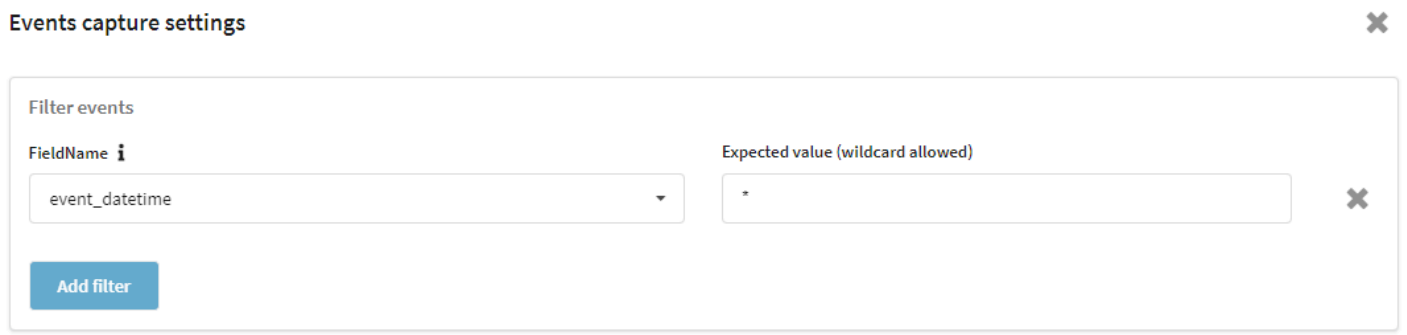
- **Application ID — unique application ID in AppMetrica.**
- **AppMetrica API token — data access token. You can retrieve it in AppMetrica application settings.**



After that you can add hooks for 2 types of AppMetrica™ events:

- **Events — captures all events in installed applications.**
- **Installations — captures new installations of your application.**

You can filter events to be captured by adding rules based on system or custom event data. In filters, you can use a search template * (wildcard). For instance, *@gmail.com will capture all profiles with a google mailbox:



Yandex AppMetrica data

Integration supports all data fields from AppMetrica Logs API. Field structure is available in AppMetrica documentation:

- **Events — https://appmetrica.yandex.ru/docs/mobile-api/logs/ref/events.html**
- **Installations — https://appmetrica.yandex.ru/docs/mobile-api/logs/ref/installations.html**

INFO

Business data can be retrieved from your Application in the `event_json` parameter in AppMetrica. To use custom data in Altcraft integration enter the FieldName manually, providing path to the data you need. For example:

- ****event_json.email** - if your customer's email is sent to AppMetrica as a root property in "event_json": ```bash "event_json" : {"email" : "example@example.com"} ```**
- **event_json.contacts.email - if the email is a part of "contacts" object: ```bash "event_json" : {"contacts": {"email" : "example@example.com", "phone" : "+12345678910"} } ```**

When using AppMetrica user interface you can configure up to 5 levels of `event_json` object nesting, when using AppMetrica API- 10. Altcraft supports the maximum value you set.

Custom event data in AppMetrica documentation: https://appmetrica.yandex.ru/docs/mobile-events/concepts/events.html

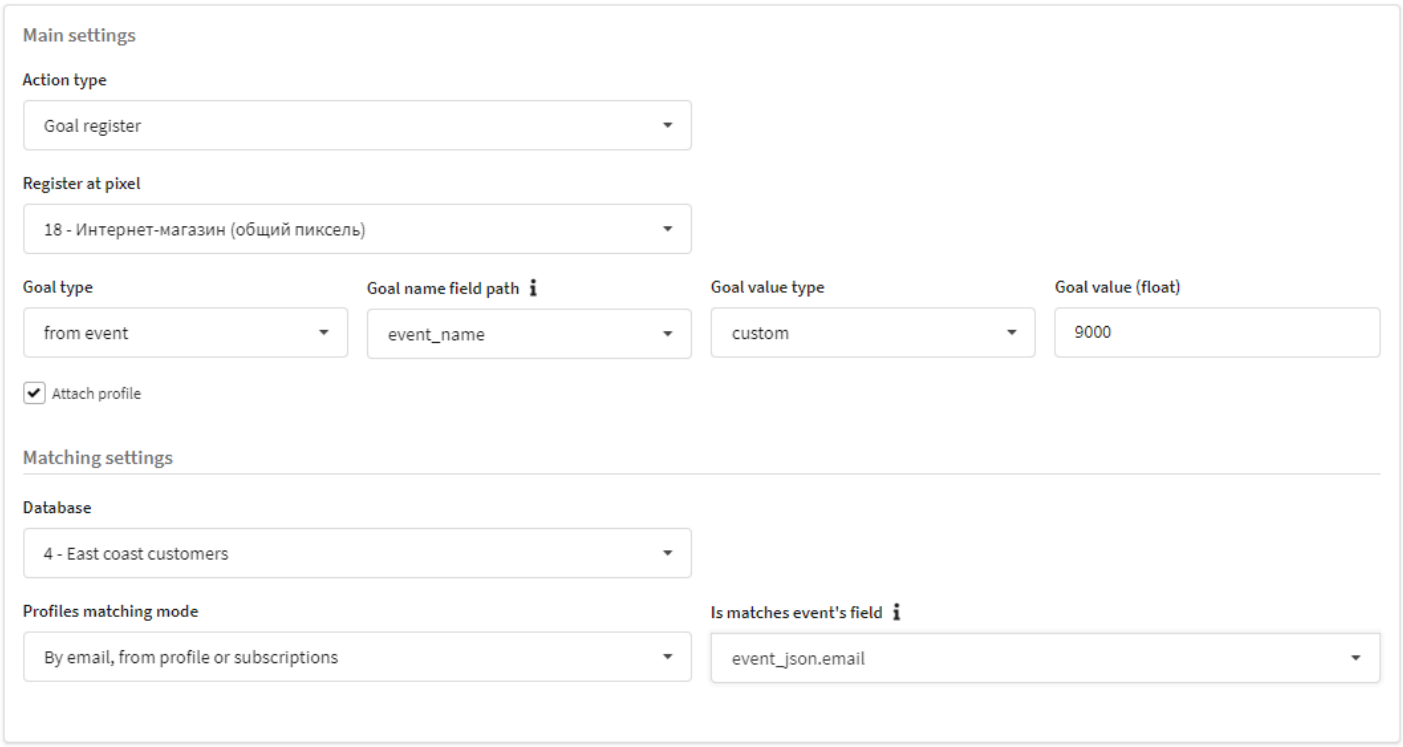
Goal registration

An event, captured by AppMetrica can record a goal and its value.

Goals are organized by Altcraft Pixels. Recording the same goals for different pixels will allow you to compare two or more UI variants or customer stories. Select the pixel you need from a dropdown.

Both goal and value data can be taken from AppMetrica data or entered manually.

- **Custom goals and values are static and recorded as is at the moment the event is captured.**
- **AppMetrica data is dynamically changed from profile to profile.**



Each goal can be recorded to a known and existing customer profile or to an anonymous profile (by Cookie data)/

- **Anonymous goals are used in goals reports to measure common effectiveness.**
- **Profile goals and values can be used in reports, for RFM and LTV segmentation and scoring. Goals reaching is recorded in profile action history.**

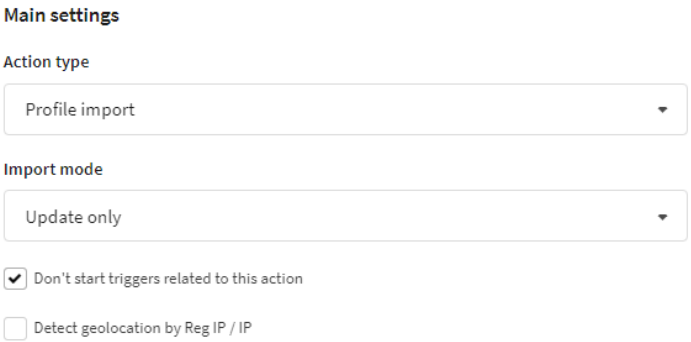
To record the goal for a specific profile, activate option Attach profile, select a database and a matching mode to search the profile:

- **By email from profile — email field from main profile data.**
- **By email, from profile or subscriptions — main profile email or any email from profile subscriptions.**
- **By AppMetrica ID — appmetrica_device_id property. When profiles are imported from AppMerica, this data is recorded automatically. For other imports — add this field manually to search profiles.**
- **By custom field — by a value of database custom field. Select the field from a dropdown and specify the corresponding property in AppMetrica data.**

Profile import

The second feature of AppMetrica integration - allows to import and update Altcraft customer profiles. Select the corresponding Action type in Main settings. There are 3 import modes:

- **Add new leads and update existing ones**
- **Add new leads, do not update existing ones**
- **Update only**

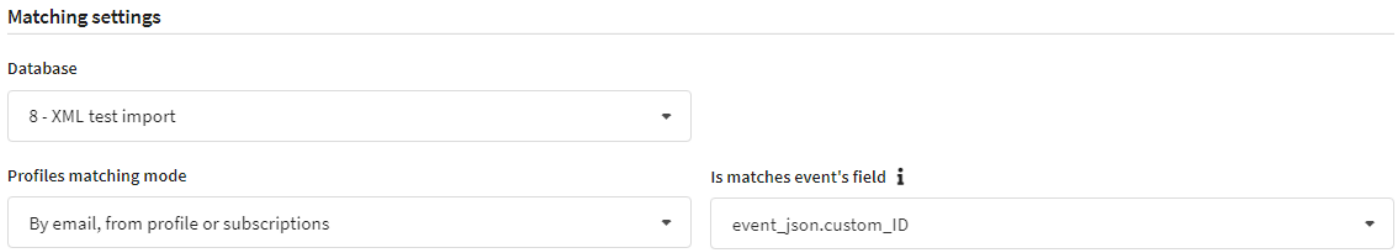


Profile matching

Select a database and a matching mode to search the profile. If a record is found - the profile will be updated, if not - a new customer record will be created.

- **By email from profile — email field from main profile data.**
- **By email, from profile or subscriptions — main profile email or any email from profile subscriptions.**
- **By AppMetrica ID — appmetrica_device_id property. When profiles are imported from AppMerica, this data is recorded automatically. For other imports - add this field manually to search profiles.**
- **By custom field — by a value of database custom field. Select the field from a dropdown and specify the corresponding property in AppMetrica data.**

> CAUTION
>
> **By AppMetrica ID is recommended if you have customers with multiple devices running your app. Do not use it along with email matching modes if you want no duplicated client profiles!**



Auto-subscription

While importing or updating customer profiles you can automatically subscribe them to Email, SMS or Push. Before you start make sure:

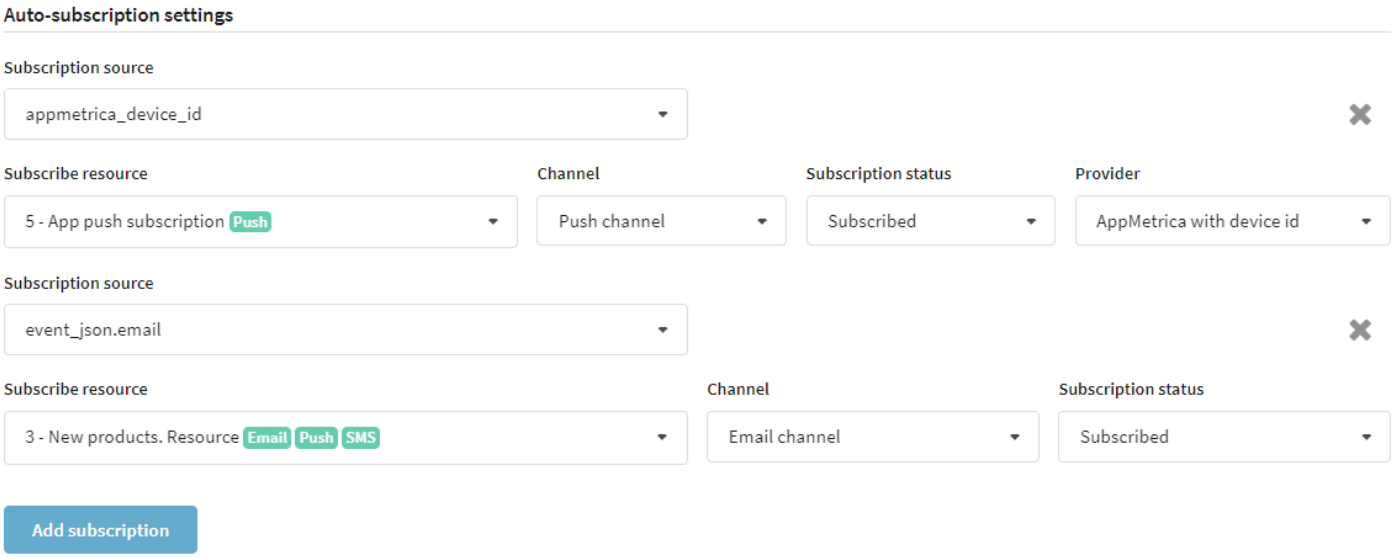
1. **You have at least one resource with a communication channel configured.**
2. **Database you selected is added to the resources you subscribe profiles to.**

After these prerequisites are made, Add subscription button will be available in the integration form:**

For every subscription record specify subscription source — this can be any of AppMetrica data fields, most commonly data from event_json. AppMetrica Push subscriptions use standard data fields:

Push provider	Event field
Yandex AppMetrica with device id	**appmetrica_device_id**
Yandex AppMetrica with iOS IFA	**ios_ifa**
Yandex AppMetrica with Google AID	**google_aid**

For other communication channels, you would like to send contact data as custom information. Custom event data in AppMetrica documentation: https://appmetrica.yandex.ru/docs/mobile-events/concepts/events.html



You can also set the status for every subscription you add or modify:

- **When importing new profiles using Double Opt-in, set status Unconfirmed**
- **When updating profile details you can change status based on customer actions. For example, if a customer wants to Suspend a subscription or Unsubscribe.**

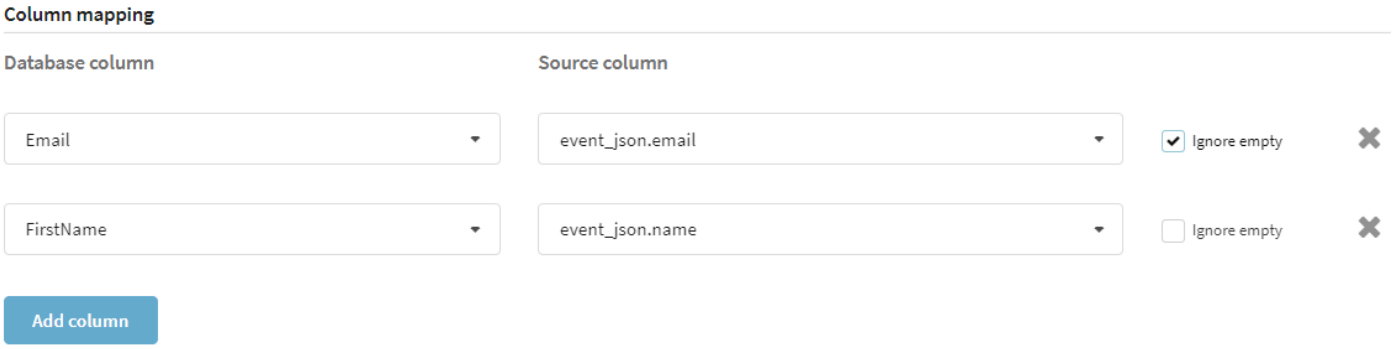
Column mapping

For any new profile, a unique ID will be assigned. If matching is set to a custom valueand a value is not found - it will be recorded for a new profile. If email is used for matching - appmetrica_device_id will be nevertheless recorded as a system parameter.

Set column mapping for Altcraft database default and custom fields based on AppMetrica events data. Custom AppMetrica data is provided within event_json object.

CAUTION

Ignore empty option will skip empty values, importing profiles with missing data. If the field is mandatory for import (email for default) and this option is not enabled - an import error will occur.



Tilda™

Tilda profile import integration

In Altcraft Platform, you can transfer the data of a user who filled out a form on your website designed on Tilda. To do this, you need to use the Webhooks tool. You can read more about Tilda Webhooks in this article.

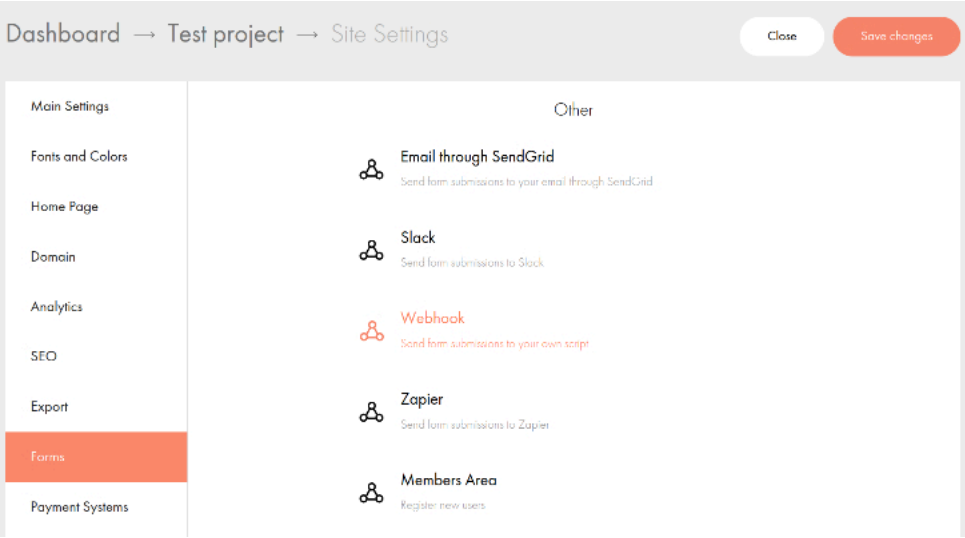
> NOTE
>
> **Please note that integration with Tilda allows you to integrate only user subscriptions to email and SMS resources.**

For this integration you will need the following data.

- **Altcraft service instance URL. For example, `https://ru.altcraft.com`.**
- **API token: tokens are created by platform master users.**
- **Database ID and Resource ID. If there is no need to subscribe to a resource, you need only the database ID.**

Step 1: Create a Tilda webhook

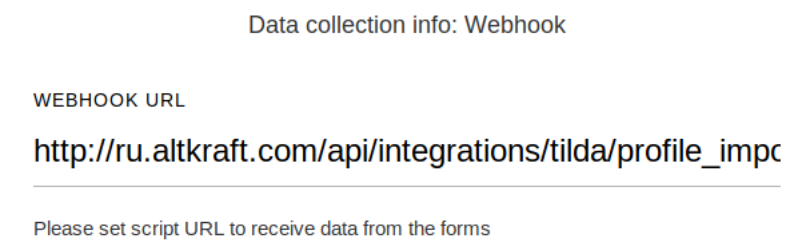
Go to the section: Dashboard —> Your project —> Site settings —> Forms. At the very bottom you will find Other —> Webhook.



Fill in the WEBHOOK URL field as follows:

```
<your.domain>/api/integrations/tilda/profile_import/?token=<api_token>
```

Here you need to replace <your.domain> with the URL of the Altcraft platform service instance and <api_token> with the token provided to you:

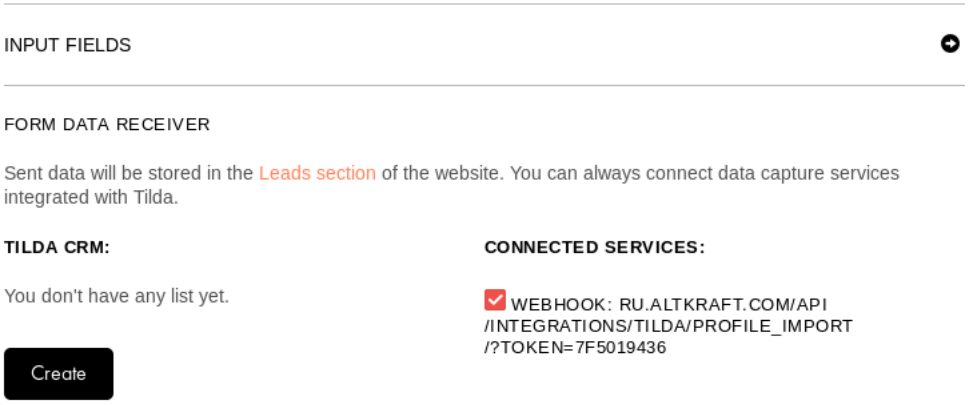


Click “Add”. After that, you will be prompted to assign this data collection service to all forms on the website. Skip this step by clicking the Close button. So, Webhook is created.

Step 2. Set up a Tilda form

Open the site page editing window and find the form from which you want to transfer data to Altcraft. Then open the “Content” tab:

Find the FORM DATA RECEIVER section and enable your Webhook:



There are two types of form in Tilda:

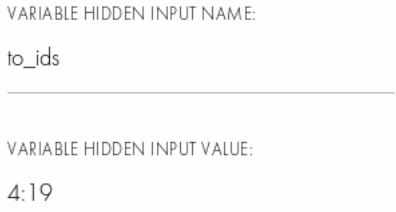
- **a simple form with a button,**
- **form with multiple fields.**

The settings for each of them are slightly different.

Forms with one input

For a single input field form you will need block “Options” to set variable names.

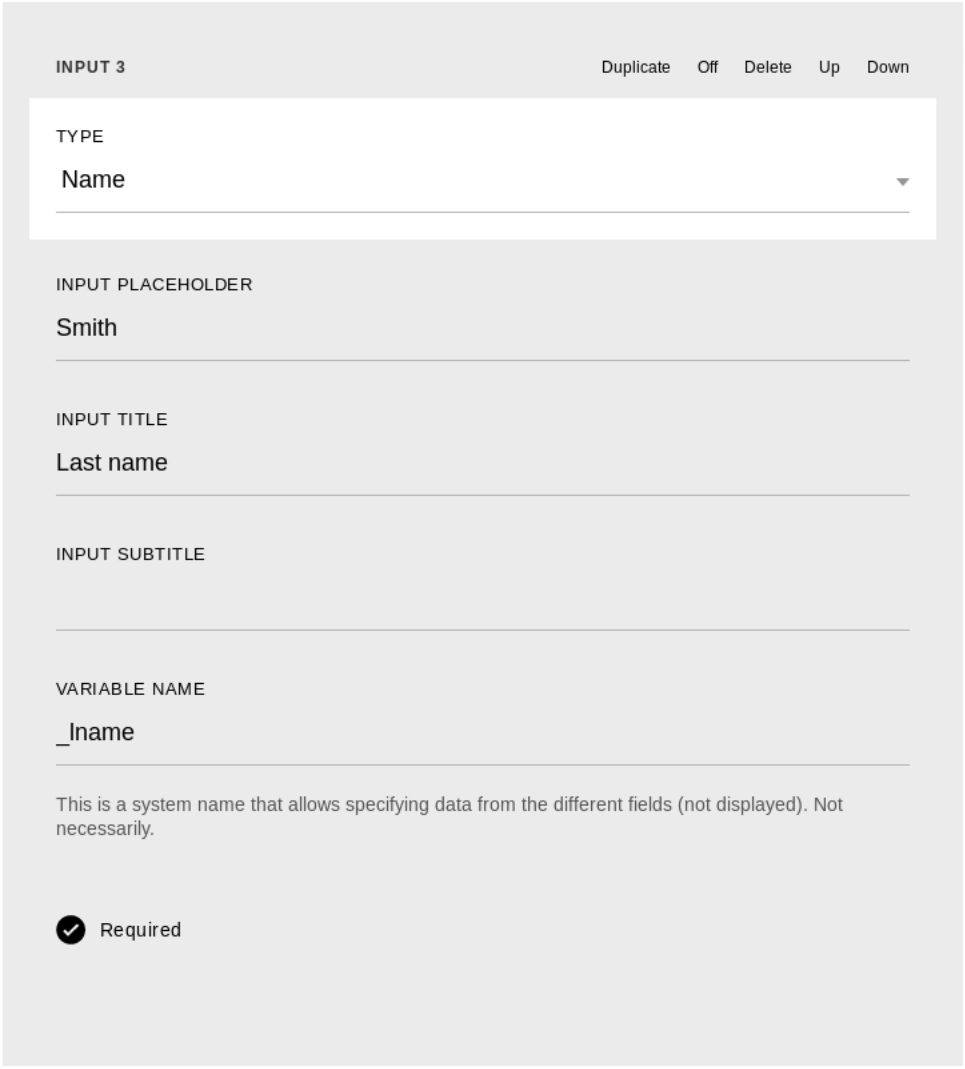
Field	Description
Variable name	**If you want this field to get into Altcraft, then its name must correspond to the same field in the database. The variable name must be email or phone.** **The name is not case sensitive, for example, if the Altcraft database has an “email” field, and Tilda’s field is called “EMAIL”, the data from the form will be submitted successfully.**
Variable name for a hidden field	**Name it** `to_ids`
Variable value for a hidden field	**Enter a colon-separated database ID and resource ID. If you do not need to subscribe to the resource, then you need only the database ID. In the screenshot below, in the example,** `4` **is the database ID,** `19` **is the resource ID.**



Form with several inputs

If you are working with a form that contains multiple fields, you need to open the “Content tab. There, expand the INPUT FIELDS” tab.

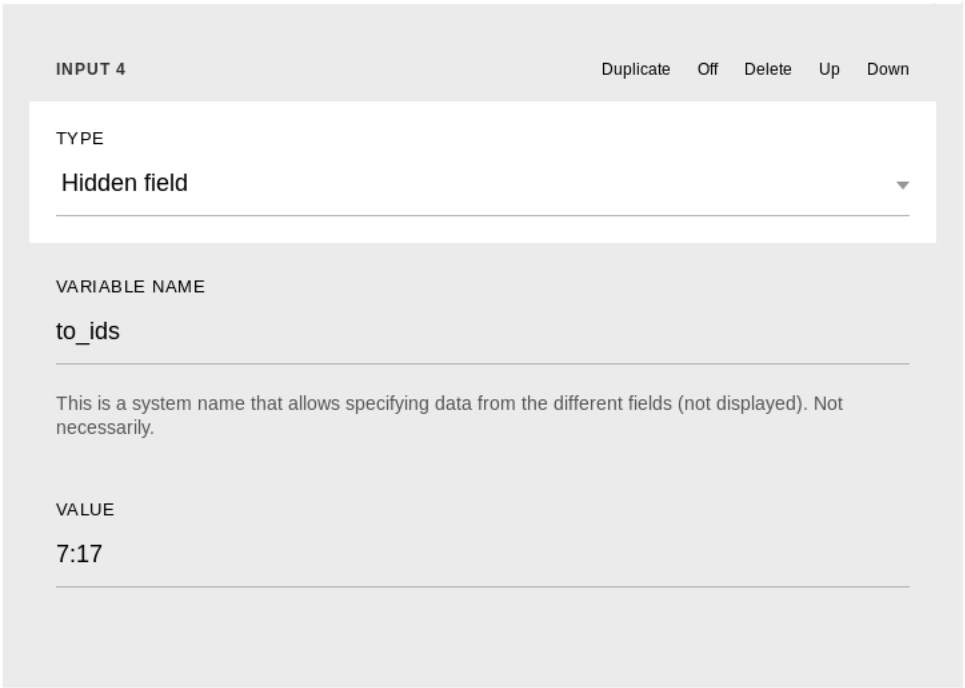
Pay attention to the names of all the variables that the form contains. If you want a field to get into Altcraft, its name must correspond to the same field in the database.



The contact data fields should be named email or phones.The variable name is case insensitive, e.g. EMAIL will also work.

After setting up the visible form fields, add another field. Type is hidden. Name it to_ids . Enter a colon-separated database ID and resource ID. If you do not want to subscribe to the resource, enter only the database ID.

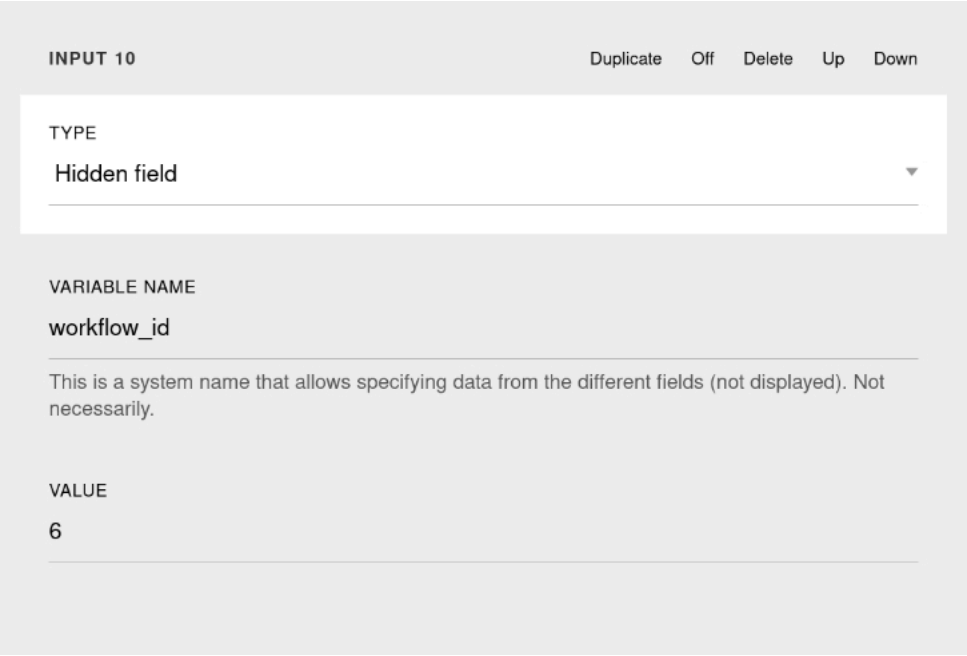
In the example in the screenshot below, 7 is the database identifier, 17 is the resource identifier.



Integration with Tilda allows not only importing new profiles into Altcraft databases, but also immediately start a trigger event. For example, you can run a trigger mailing right after importing a new profile to send them a welcome email, or add the profile to a welcome scenario.

To do this, you need to add additional hidden fields in the form:

Action after import	Variable name	Value
Launch a trigger mailing	**trigger_id**	**Trigger mailing ID**
Add a profile to a scenario	**workflow_id**	**Scenario ID**



Passing a subscription status

When you import a profile from a multi-field form, you can pass a specific subscription status to the resource. It is this status that will be displayed in the digital profile of the subscriber in Altcraft Platform.

To pass profile status, you need to add one more hidden field in your form settings. You must specify status_sub as the variable name, and choose one of the statuses reserved by Altcraft Platform as the value.

Available statuses:

Status	Status_sub value
Subscribed	**0**
Unsubscribed	**1**
Complaint	**2**
Hardbounced	**3**
Unconfirmed	**4**
Suspended	**5**
Invalid	**6**

Step 3. Test a form

After you set up the fields for the form, publish the page. Then test the page by submitting test data to the form.

TIP

Tilda won't update the form right away, so it's best to test it a couple of minutes after posting.

Next, open Your project —> Leads.

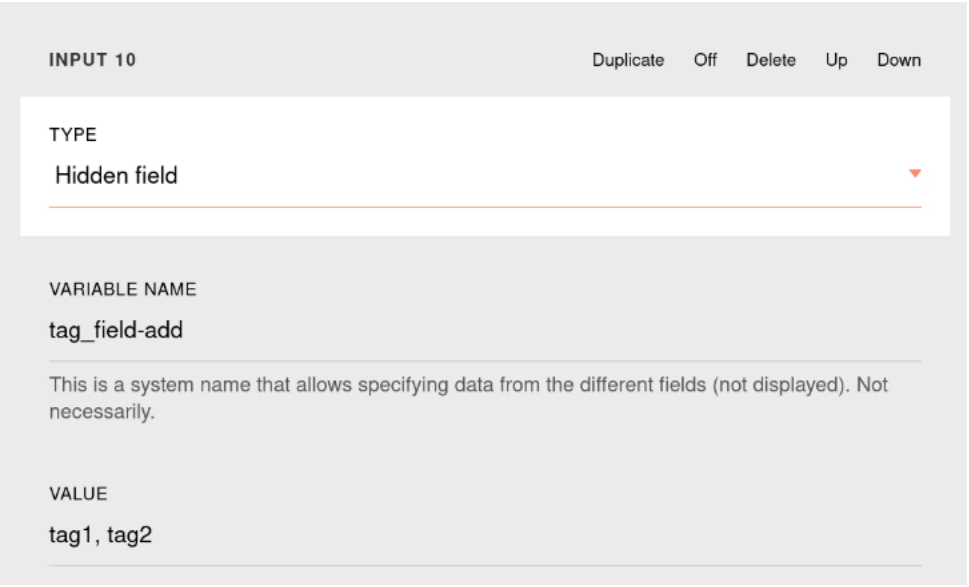
The screenshot shows an example of a successfully sent application. If you see an error, you need to check the following points:

- **The URL in the Webhook is correct and contains the correct API Key**
- **The database exists and it is selected in the resource you are using to subscribe.**
- **The form has an email or phone field**

Extra options

You can set different variable parameters as hidden form fields:

- **Profile matching mode by any of available. Default matching is Email.**
- **You can use `db_id` and `resource_id` instead of `to_ids` variable**
- **You can set a custom value for `_vendor` variable. By default it will send form ID.**
- **When updating a profile field with the “tags” type in the database, the field is completely overwritten. To add new values to existing ones, add the `-add` to the variable name when setting up a field in Tilda. For example, to update a tag_field, use the variable name `tag_field-add`.**



Lpgenerator™

Description

Imports or updates customer profile.

At first create an input form for customer profiles data fields in the landing page editor. It can contain any database field inputs.

Request URL

Method: POST

`https://example.com/api/v1.1/integrations/lpgenerator/profile_import`

Request parameters

Lpgenerator default fields. You cannot redefine field values.

Parameter	Type	Example	Required	Description
id	**string**	**“777777”**		
meta_data	**JSON object**	**“meta_data” : {...}**	**Да**	**Metadata like IP and browser info**
utm_data	**JSON object**	**“utm_data” : {...}**	**Да**	**Predefined utm data (can be empty)**

Hidden fields

Parameter	Type	Example	Required	Description
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**API token**
db_id	**string**	**“1”**	**Yes**	**Database ID**
matching	**string**	**“phone”**	**No**	**Profiles matching mode: email/phone/custom By default – email**
field_name	**string**	**“custom_ID”**	**No**	**Sets a custom field for profile matching.**
resource_id	**string**	**“3”**	**No**	**Defines subscription resource ID**

Profile fields

Parameter	Type	Example	Required	Description
lead_data	**JSON object**	**“lead_data” : {...}**		**Altcraft database fields**

`db_id` , `token` , `resource_id` , `matching` , `field_name` **must be predefined in LPGenerator web hook settings**

Use Altcraft API URL as LPGenerator web hook URL: `<your.domain>/api/integrations/lpgenerator/profile_import`

API tokens are created by Altcraft platform master users.

Request examples

Email matching profile import:

```
{
        "id": 77777777,
        "lead_data": {
                "email": "qwerty@dron.ff",
        },
        "meta_data": {
                "HTTP_USER_AGENT": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99
Safari/537.36",
                "LANGUAGE": "en-US",
                "form_name": "form 1",
                "from": "Page 107570 (Variant 1)",
                "page_id": 1642169,
                "user_ip": "96.123.123.123",
                "variant_id": 1935651
        },
        "utm_data": {}
    "db_id": "2",
    "resource_id": "2",
    "matching": "email",
    "token": "abcdefghijklmnqrstuvwxyz"
}
```

Profile fields update with a resource subscription (custom field profile matching by `custom_ID` field):

```
{
        "id": 27773744,
        "lead_data": {
                "email": "ytrewq@dron.ff",
        "custom_ID": "Bagdasaryan"
        },
        "meta_data": {
                "HTTP_USER_AGENT": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99
Safari/537.36",
                "LANGUAGE": "en-US",
                "form_name": "form 1",
                "from": "Page 207571",
                "page_id": 1642169,
                "user_ip": "96.123.123.123",
                "variant_id": 1935651
        },
        "utm_data": {}
    "db_id": "2",
    "resource_id": "2",
    "matching": "custom",
    "field_name": "custom_ID",
    "token": "abcdefghijklmnqrstuvwxyz"
}
```


Response example


```
{
  "error": 0,
  "error_text": "Successful operation"
}
```


Response parameters

Parameter	Type	Description
error	**int**	**Error code**
error_text	**string**	**Error text**

WhatsApp

Description

Use WhatsApp* integration to automate communication with subscribers in the messaging app. By setting up this channel, you will be able to:

- **send personalized messages to subscribers in the app;**
- **segment the audience based on user behavior;**
- **use WhatsApp* channel to set up scenarios;**
- **collect statistics and analyze the received data.**

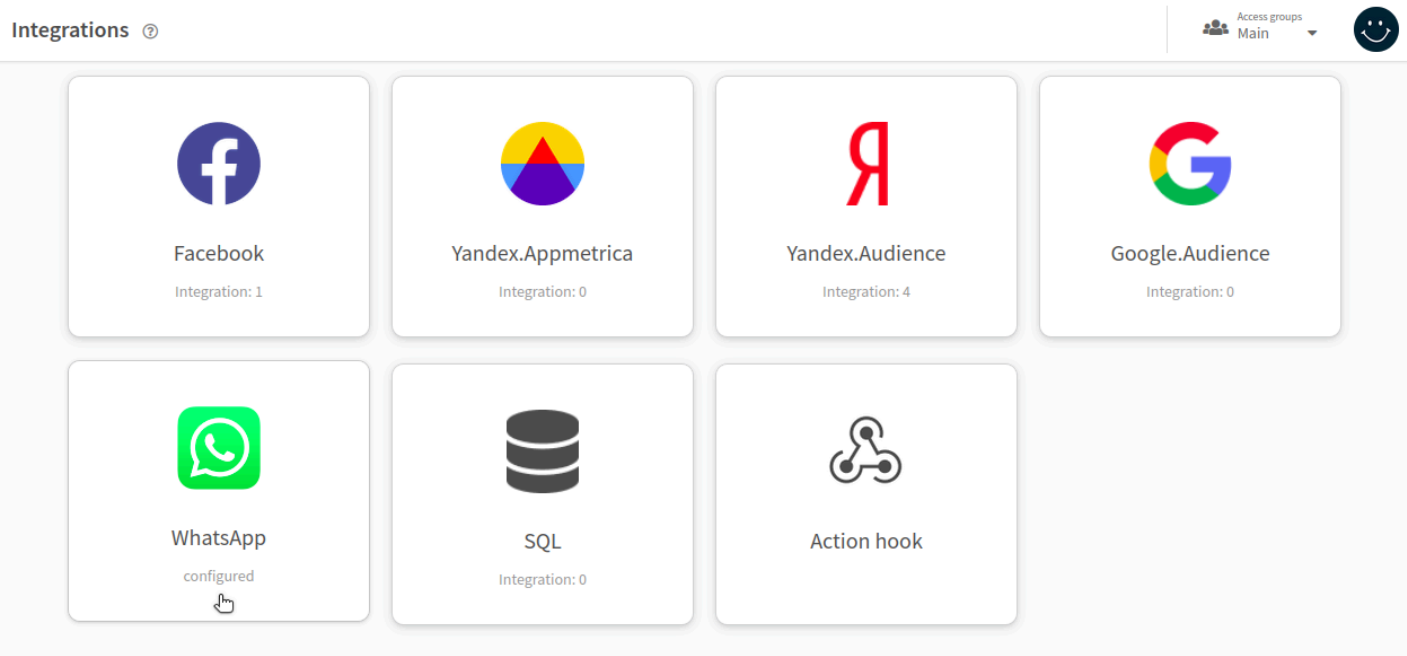
> CAUTION
>
> **In order to initiate a WhatsApp* message to a person, you must first receive opt-in permission confirming that they wish to receive future messages from you on WhatsApp*. Learn more.**

Businesses use the WhatsApp* Business API to communicate with customers in the app. API access is provided through official Meta providers. In Altcraft Platform, you can set up integration through such providers as Edna and 360dialog. In order to link the provider's account to Altcraft Platform, you need to create a connection through Edna or through 360dialog.

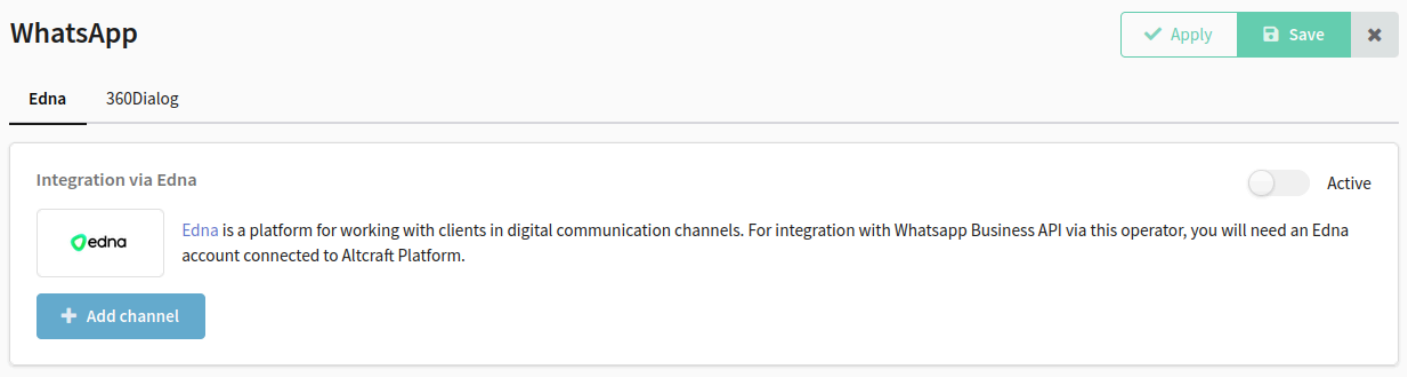
> TIP
>
> **In addition to the services of the provider, you also have to pay for messages sent to users. You can find out more about the cost of WhatsApp* messages here.**

Creating an integration

Go to the Integrations section and select the type of integration — *WhatsApp.



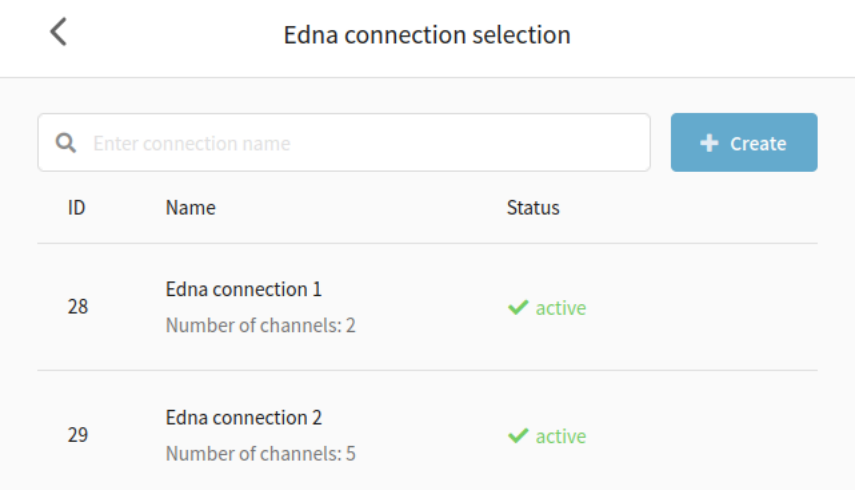
Go to your provider's tab:



Edna

Activate the integration using the toggle in the top right corner. Click “Add channel”.

A window will appear on the right where you first need to select an available connection:



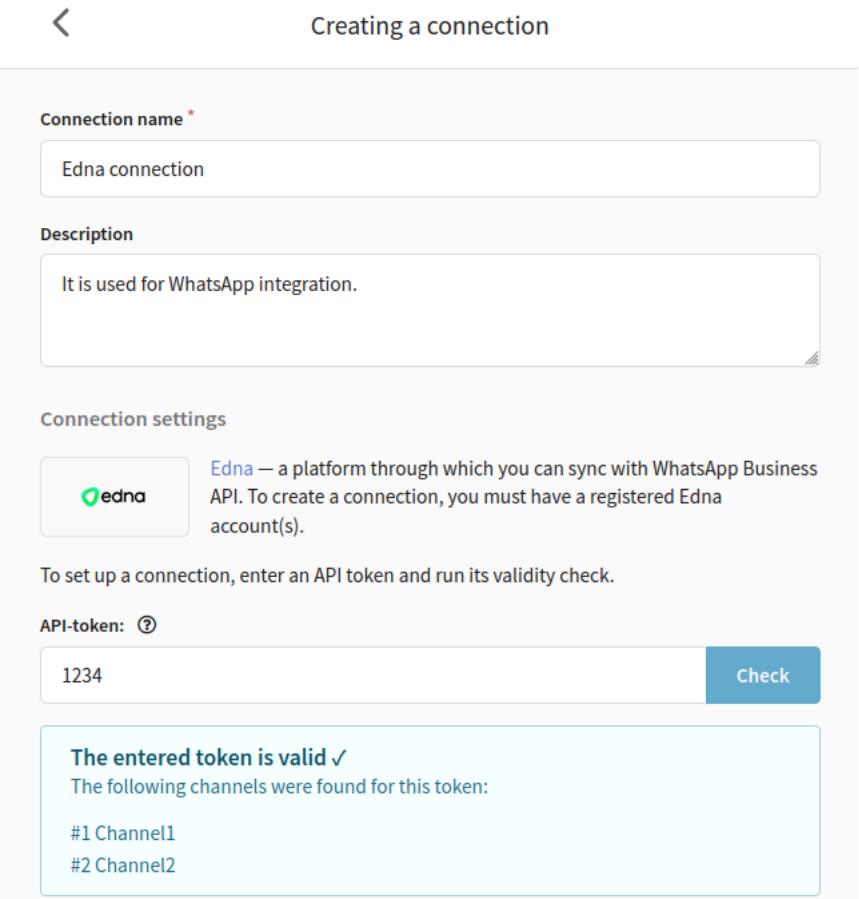
I don’t have any connections available

If you haven’t set up a connection to Edna yet, you can create one without leaving the current page.

Click “Create”. Enter a name and description for the connection. Next, specify the API token generated in your Edna’s personal account and check its validity. If the token passed the verification, click the Create button in the lower right corner.

TIP

You can read more about creating a connection to Edna in this guide.



Next, select an active channel. The channel should be in the “Correct” status.

After that, specify the cascade. The edna cascade is a scenario that sets an order in which the system sends the messages through several channels if the message was not delivered to the previous channel.

CAUTION

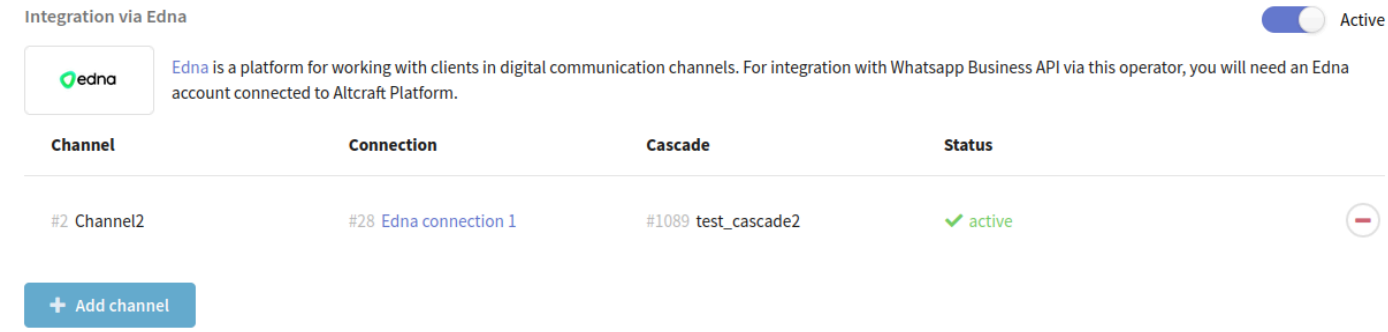
Some cascades may not be available in the list. Such cascades are configured for several channels, and the platform considers them invalid. You can only select cascades that use one channel.

If needed, you can enable event capture in Edna to send message status data to Altcraft Platform. When this option is enabled, a Callback URL will be automatically added to your Edna account after the integration is saved. This URL is used to transmit WhatsApp events to Altcraft. If the option is disabled, delivery and read analytics will not be available in the platform.

Once you've set up the channel and cascade, click "Add" in the bottom right corner.

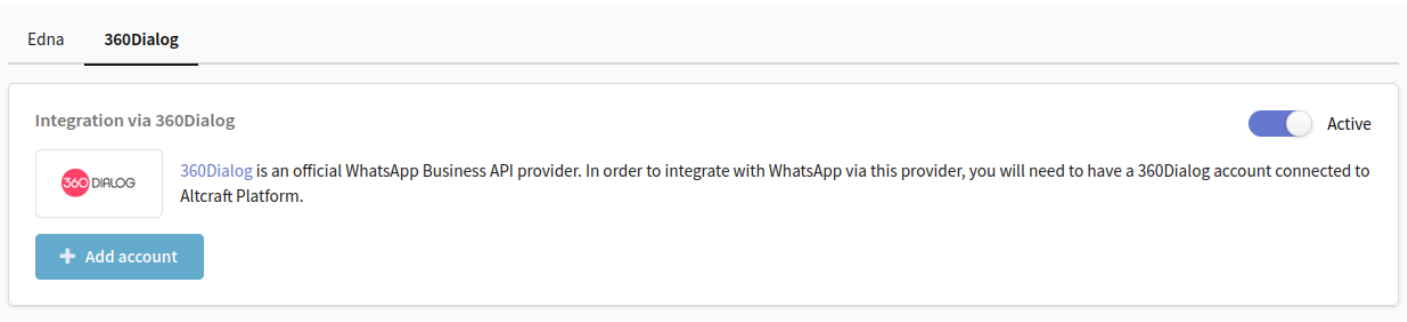
Details for the added channel will be displayed in the integration. You can add multiple channels if necessary. To remove a channel from integration, click

.

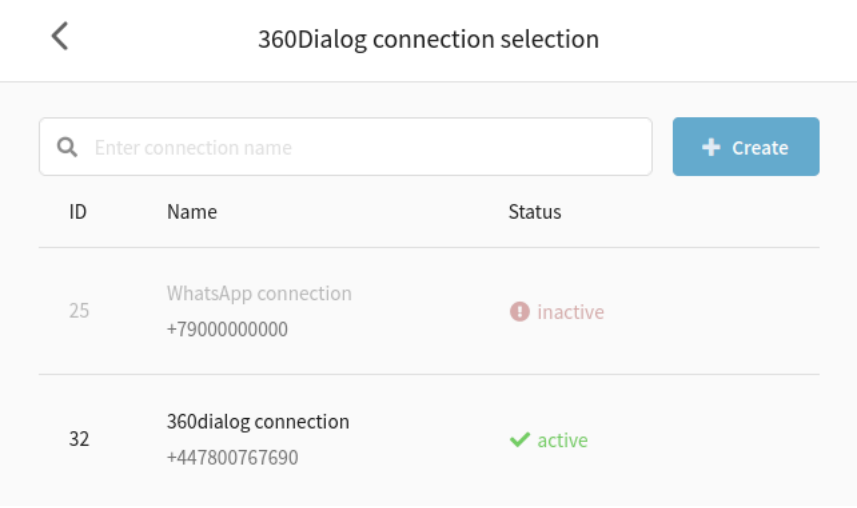


360dialog

Activate the integration using the toggle in the top right corner. Click Add account.



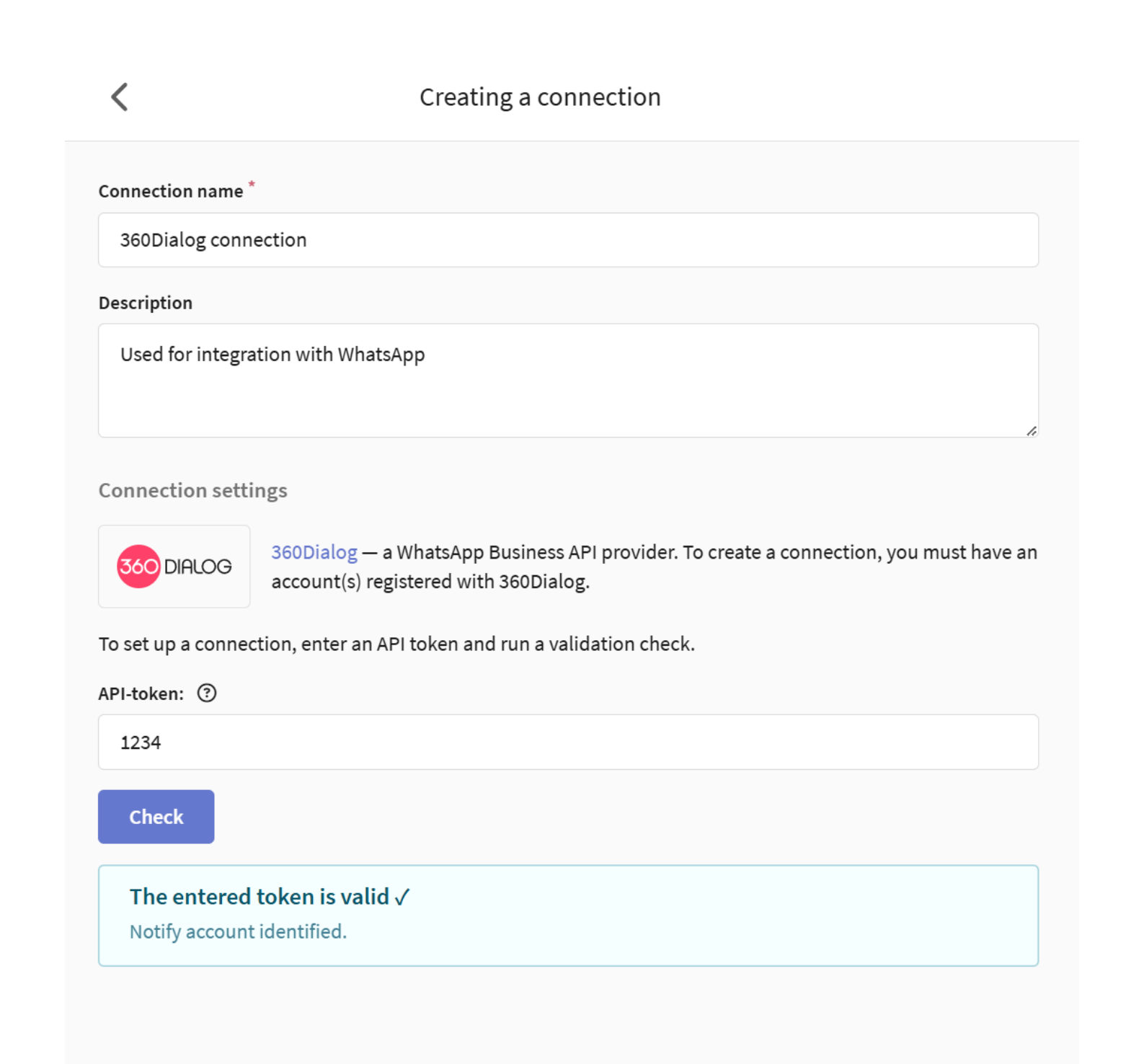
A window will appear on the right where you first need to select an available connection:



No available connections

If you have not yet configured a connection to 360dialog, you can create one directly in this window.

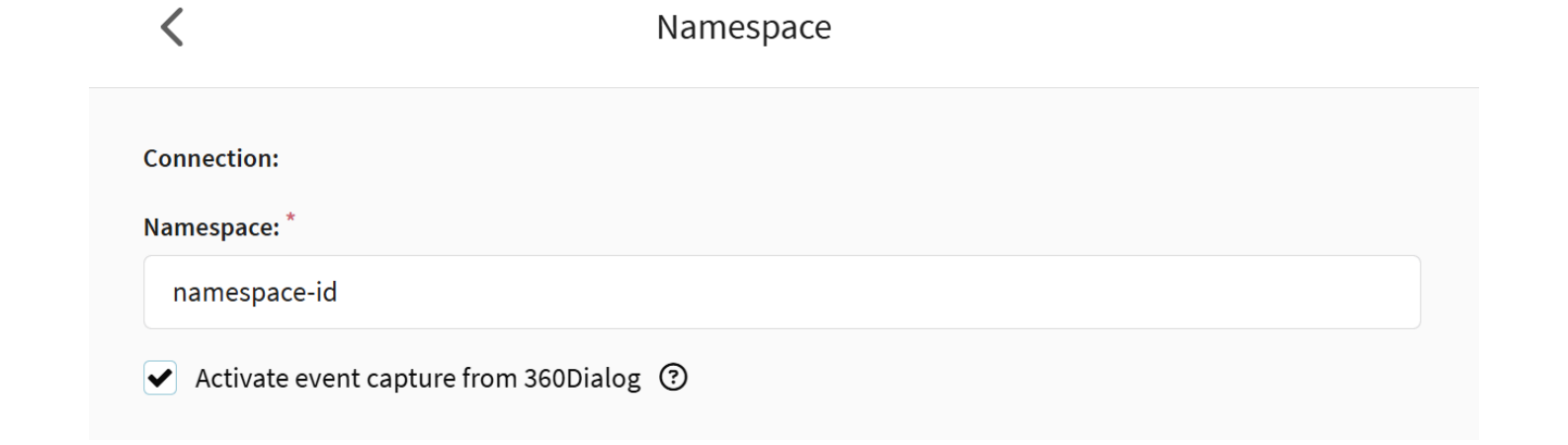
Click Create. Enter a name and description for the connection. Next, enter the API token generated in your 360dialog account and run a validation check. If the token passes the validation, click the Create button in the lower right corner.



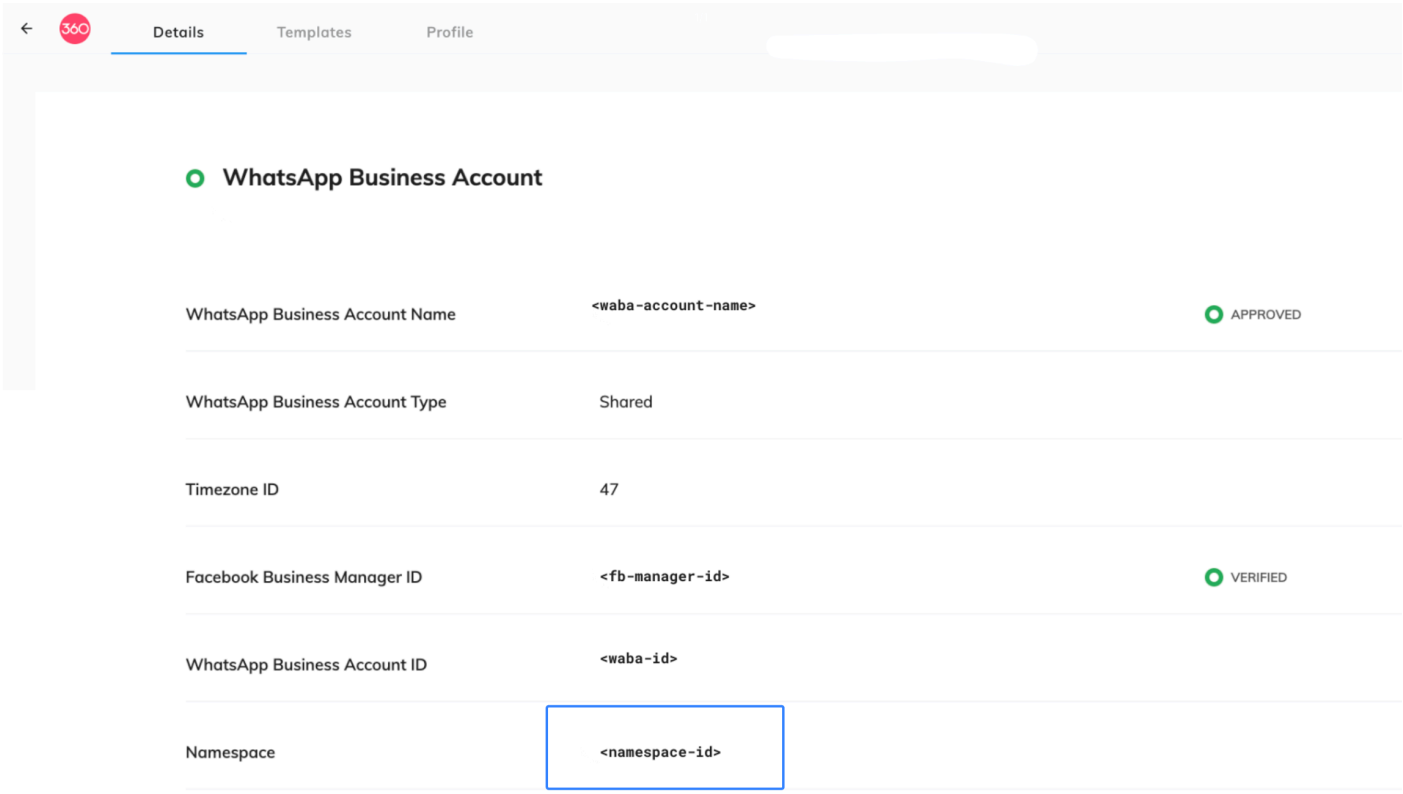
TIP

You can read more about creating a connection here.

Next, enter the namespace — a unique string used to identify all templates created within a specific WhatsApp Business Account.



You can find the Namespace in your 360dialog account:



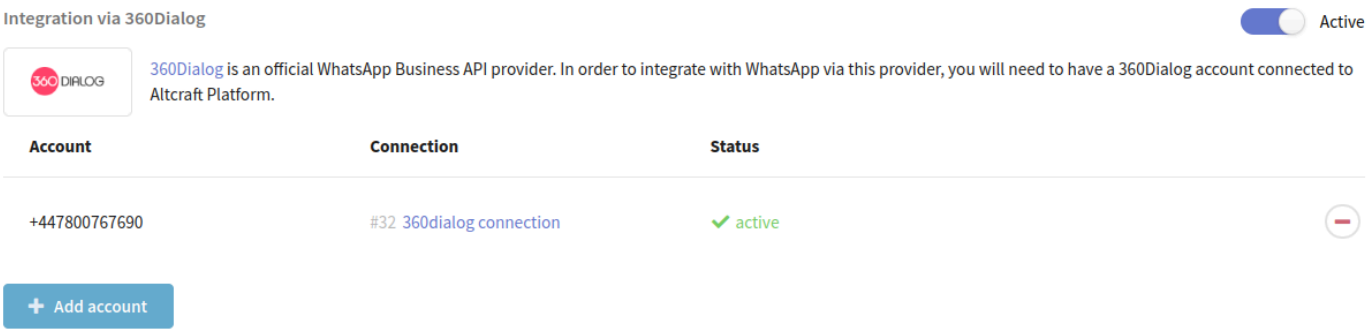
If needed, you can enable event capture in 360dialog to send message status data to Altcraft Platform. When this option is enabled, a Callback URL will be automatically added to your 360dialog account after the integration is saved. This URL is used to transmit WhatsApp events to Altcraft. If the option is disabled, delivery and read analytics will not be available in the platform.

Click Add in the lower right corner.

> TIP
>
> **You can read more about creating a connection to 360Dialog in this guide.**

You will see information on the added account in the table. You can add multiple accounts if needed. To remove an account from the integration, click

.



Click the Save button in the upper right corner to save your integration.

Once you have successfully set up the integration, you can use it to add a WhatsApp* channel to a resource.

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

Viber*™ integration

Description

Use Viber*™ integration to automate communication with subscribers in the messaging app. By setting up this channel, you will be able to:

- **send personalized messages to subscribers in the app;**
- **segment the audience based on user behavior;**
- **use this channel to set up scenarios;**
- **collect statistics and analyze the received data.**

The integration can be implemented through the providers Devino Telecom, SMS Traffic, MTS OmniChannel or Rapporto. Before setting up the integration, you need to:

1. Register an account with any provider. 2. Create a connection in order to link your provider account with Altcraft Platform.

- **Instructions for Devino Telecom**
- **Instructions for SMS Traffic**
- **Instructions for MTS OmniChannel**
- **Instructions for Rapporto**

INFO

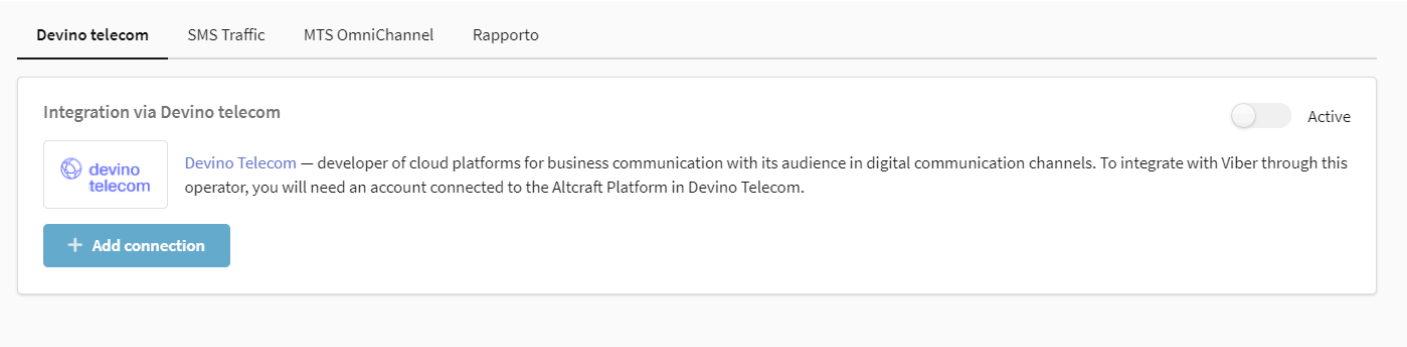
The cost of messages is determined by providers. Altcraft does not charge additional fees for sending messages to this messenger*™.

Creating an integration

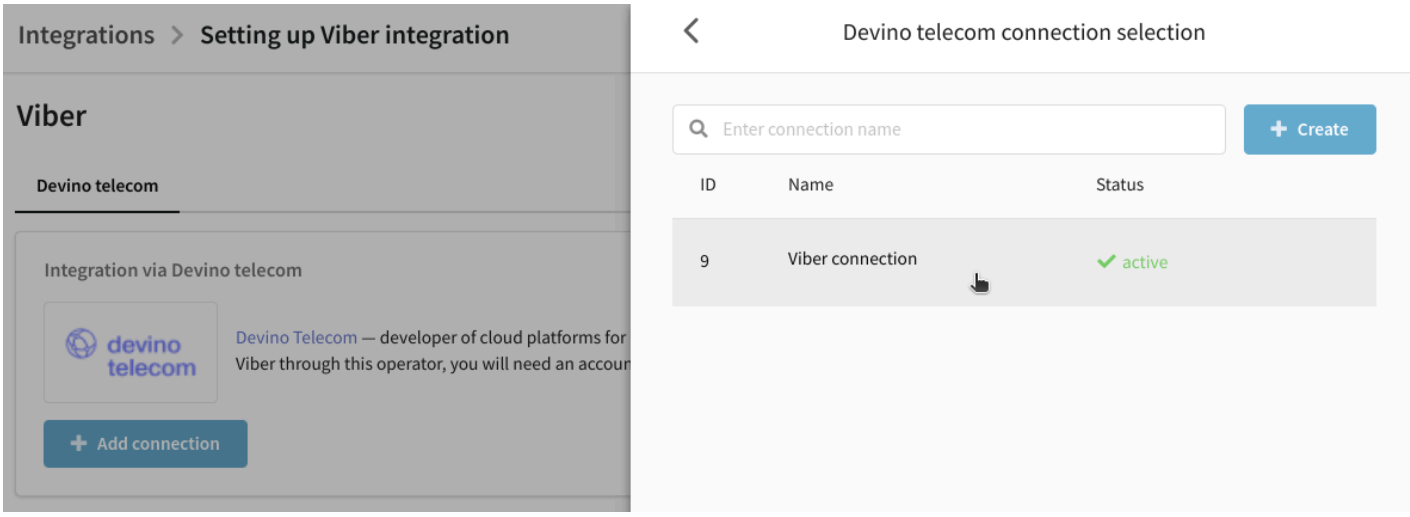
Go to the Integrations section and select the required type of integration.

Devino

Activate the integration using the switch in the upper right corner. Click on “Add Connection”:



A window will appear on the right where you need to select an available connection:



No connections available

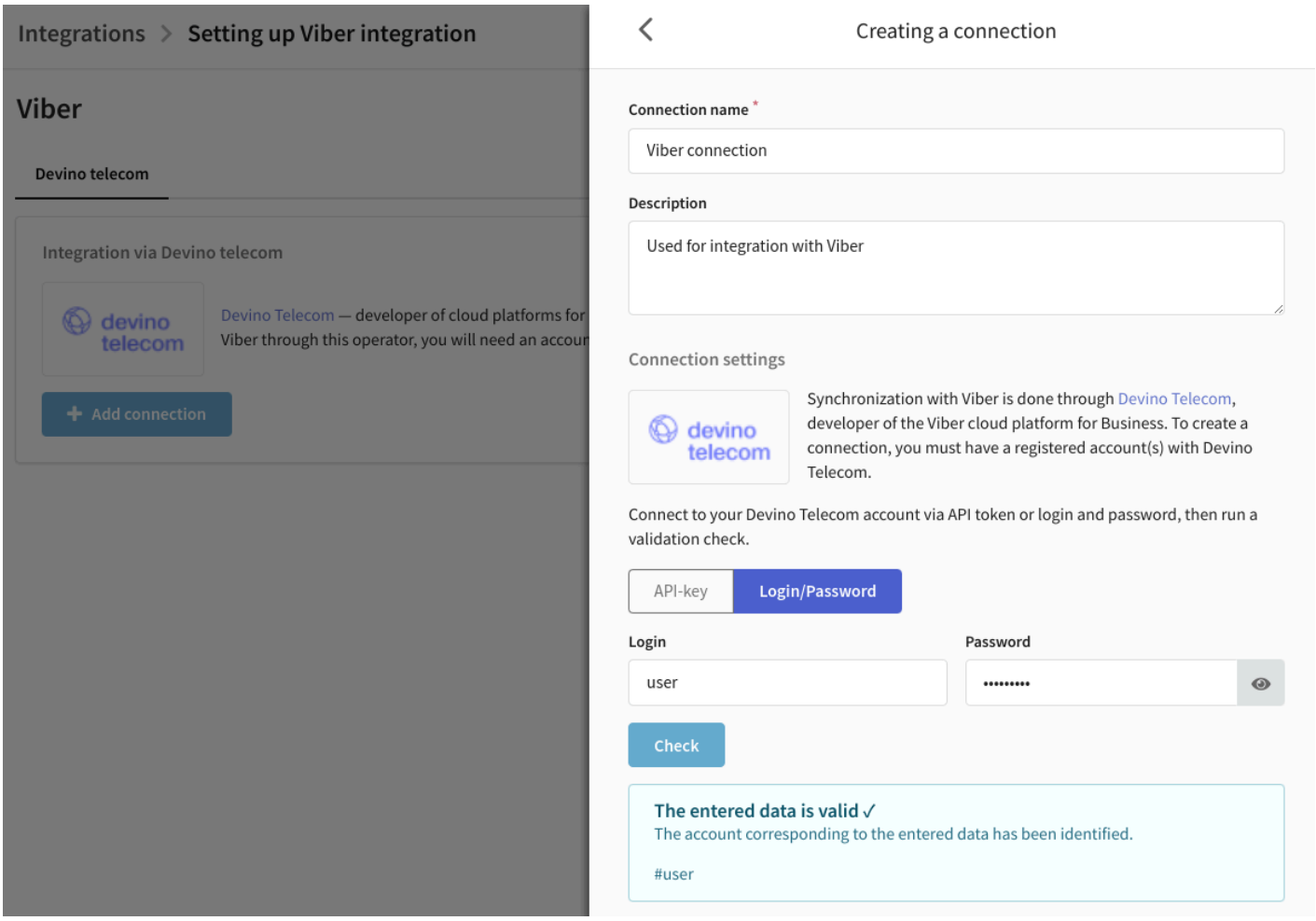
If you have not yet set up a connection to your provider, you can create one directly in this window.

1. **Press Create.**
2. **Enter the name and a description of the connection.**
3. **Enter the authorization data (two authorization options are available for Devino).**

4. **Press Check to run a validation check on the entered data.**
5. **If the entered data has been verified, click the ** “Create”** button in the lower right corner.**

TIP

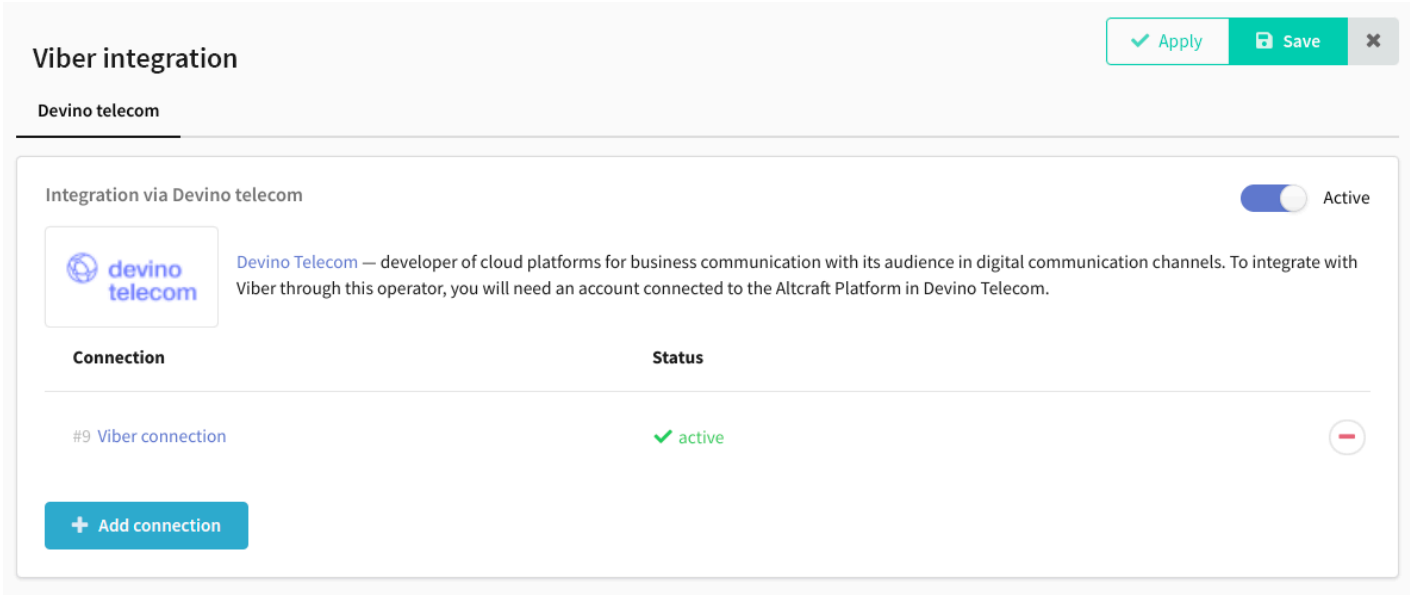
The documentation portal has published detailed instructions for creating a connection via Devino Telecom.



Information about the added connection will be displayed in the integration. Make sure that the connection status is “active”. Otherwise, check the status of your account in the provider’s personal account. Keep in mind that the account may be blocked if the subscription fee is not paid.

If necessary, you can add multiple accounts. To remove an account from integration, click

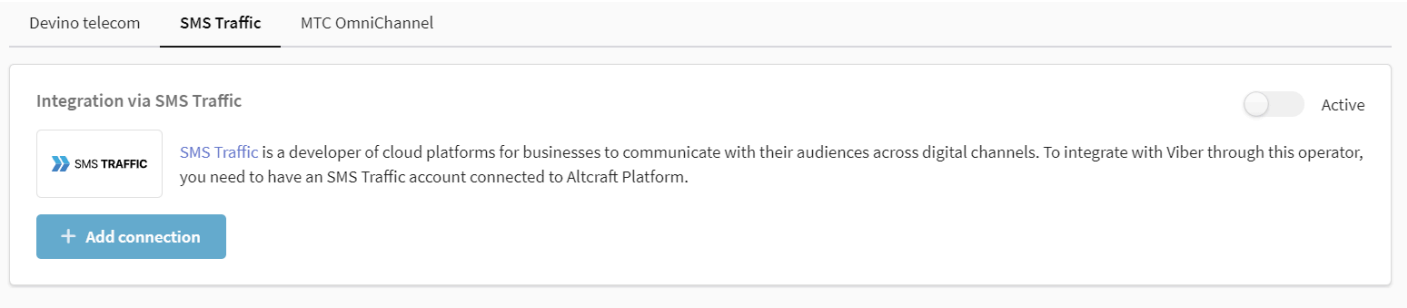
.



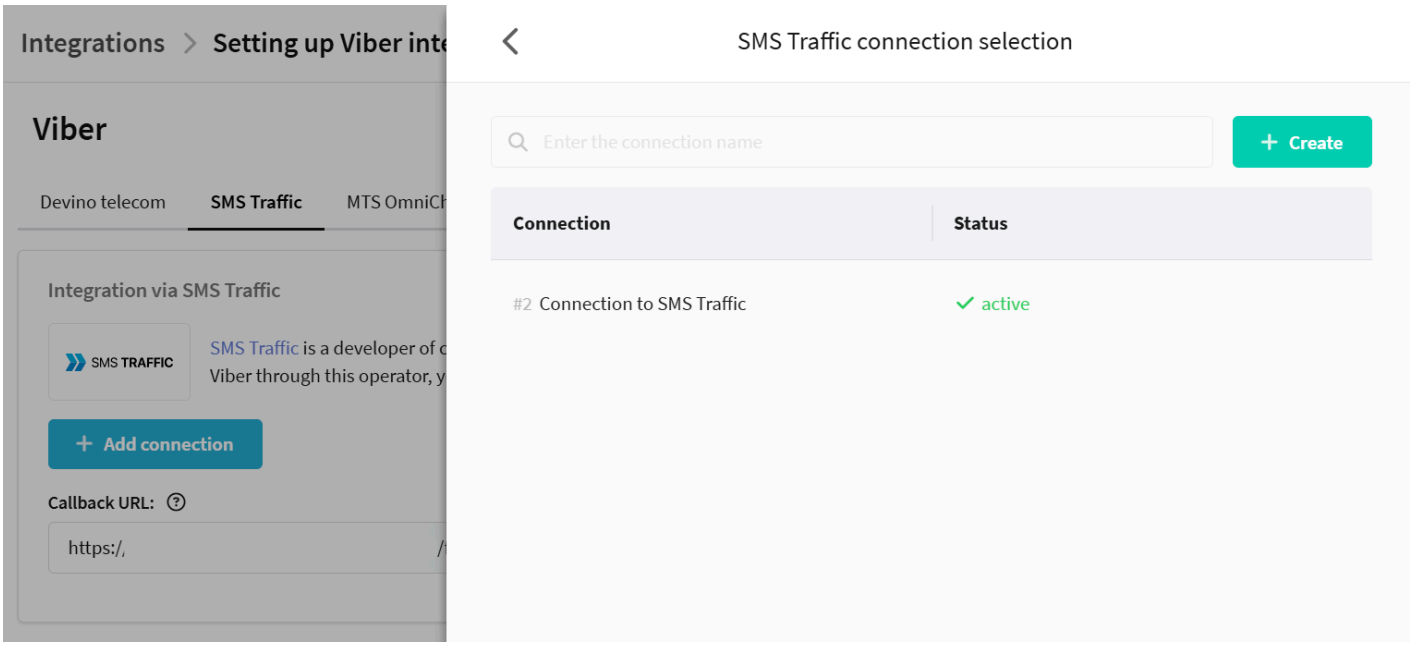
Save the integration using the buttons in the upper right corner.

SMSTraffic

Activate the integration using the switch in the upper right corner. Click on “Add Connection”:



A window will appear on the right where you need to select an available connection:



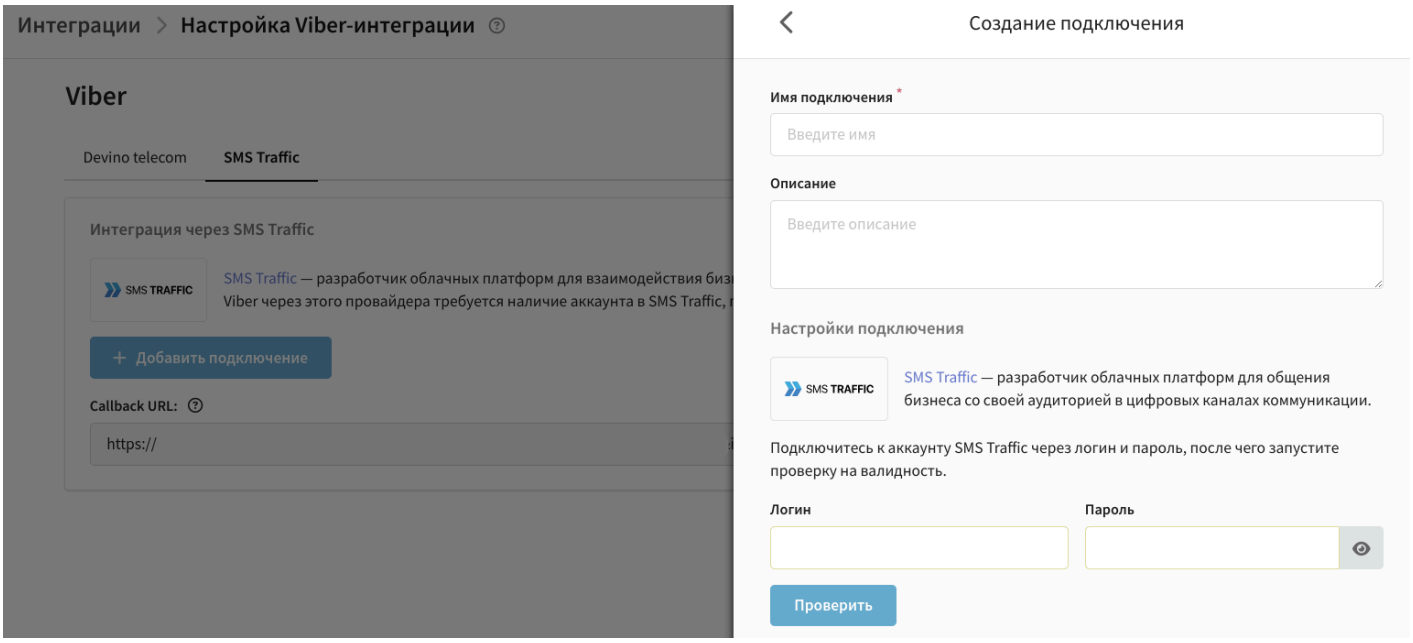
No connections available

If you have not yet set up a connection to your provider, you can create one directly in this window.

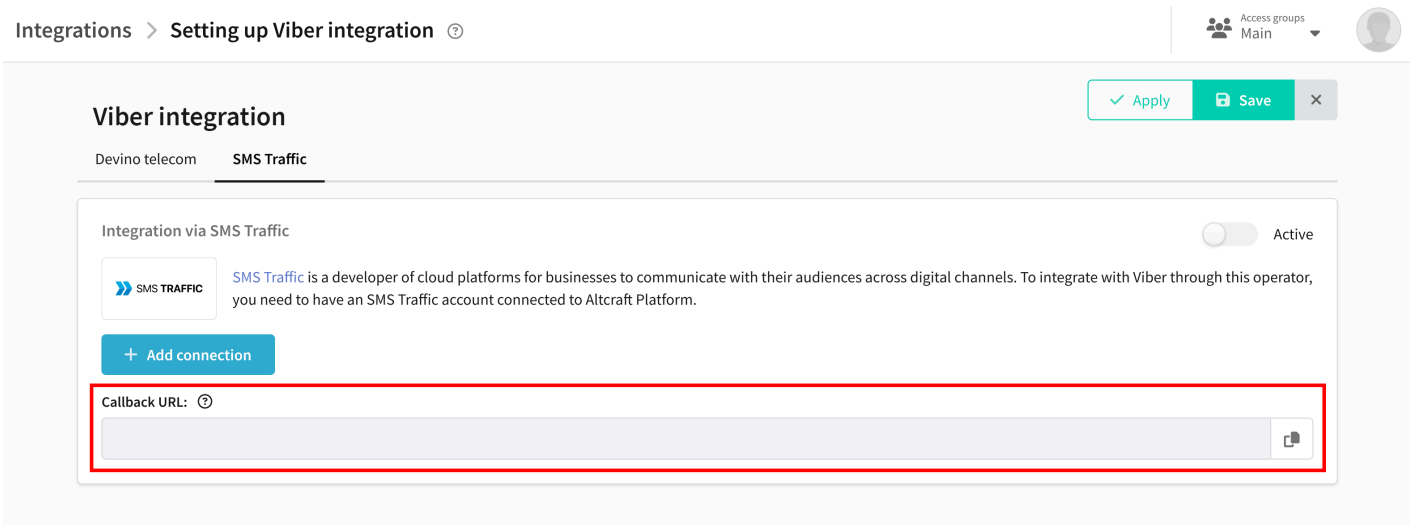
1. **Press Create.**
2. **Enter the name and a description of the connection.**
3. **Enter the authorization data.**
4. **Press Check to run a validation check on the entered data.**
5. **If the entered data has been verified, click the ** “Create”** button in the lower right corner.**

TIP

The documentation portal has published detailed instructions for creating a connection via SMSTraffic.



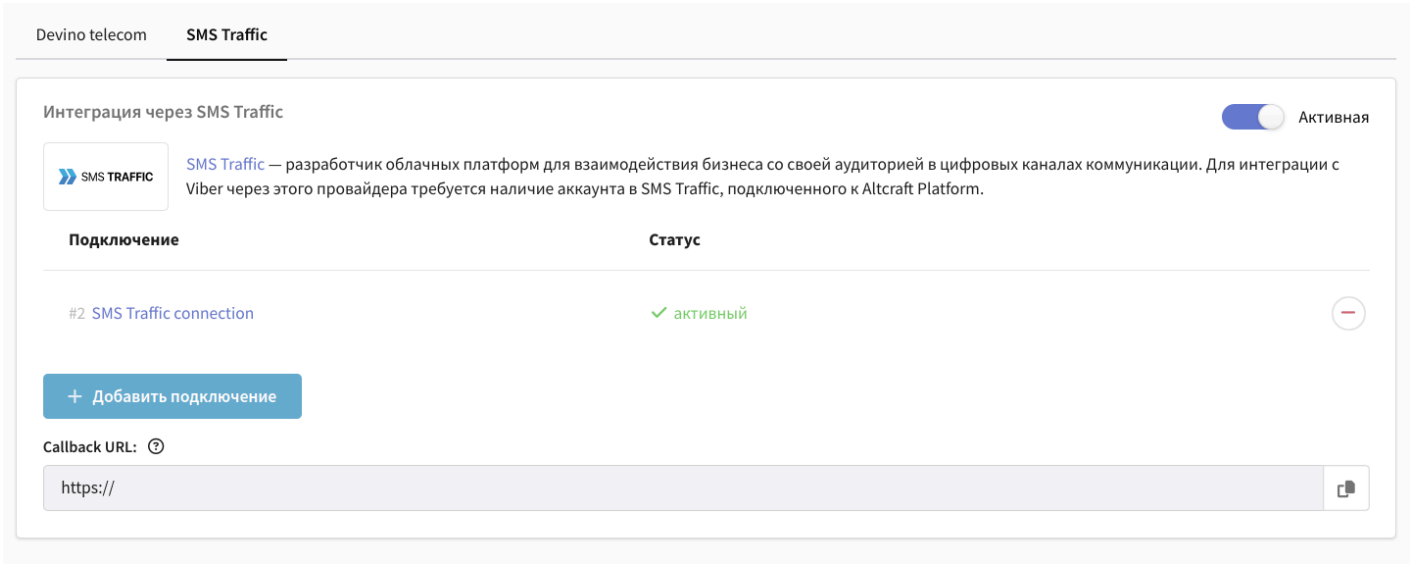
For the integration to work correctly, you also need to send to [technical support] your Callback URL specified in the integration.



Information about the added connection will be displayed in the integration. Make sure that the connection status is “active”. Otherwise, check the status of your account in the provider’s personal account. Keep in mind that the account may be blocked if the subscription fee is not paid.

If necessary, you can add multiple accounts. To remove an account from integration, click

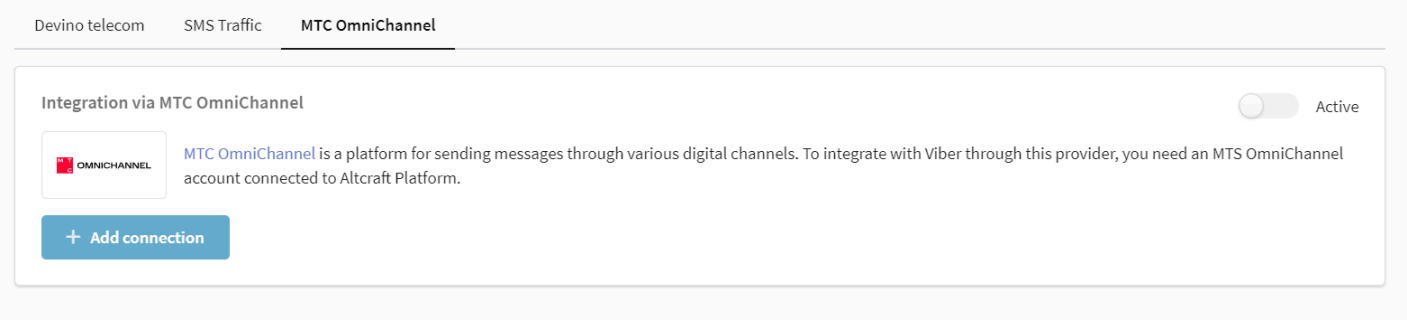
.



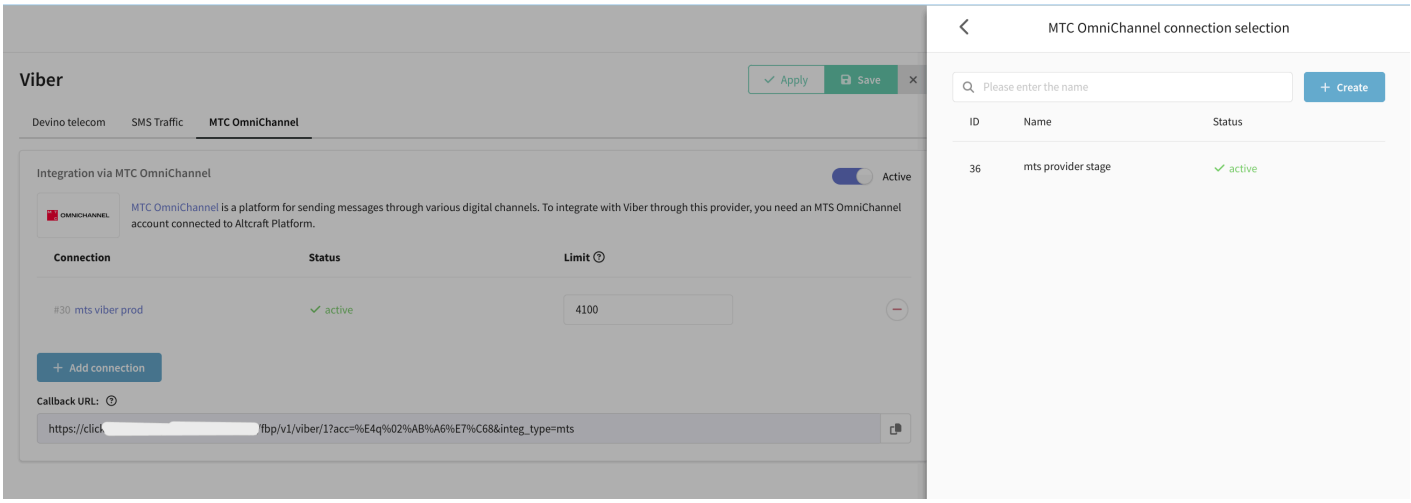
Save the integration using the buttons in the upper right corner.

MTC OmniChannel

Activate the integration using the radio button in the upper right corner. Click on Add Connection:



A window will appear on the right where you need to select an available connection:



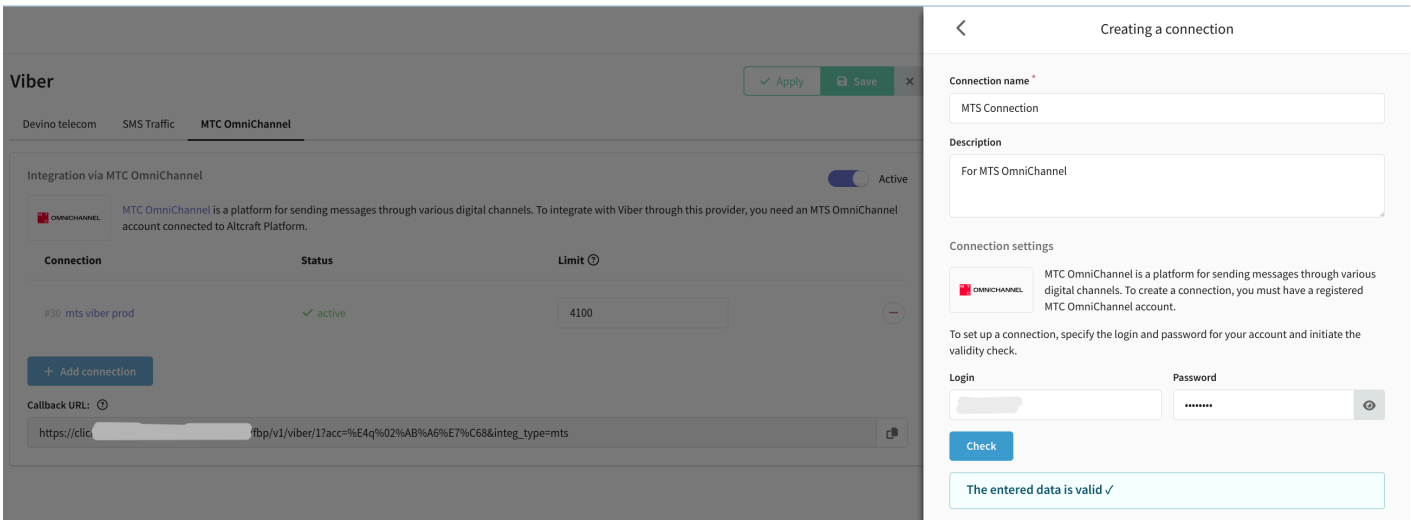
No connections available

If you have not yet set up a connection to the provider, you can create one directly in this window.

1. **Press Create.**
2. **Enter the name and a description of the connection.**
3. **Enter the authorization data.**
4. **Press Check to run a validation check on the entered data.**
5. **If the entered data has been verified, click the Create button in the lower right corner.**

TIP

The documentation portal has published detailed instructions for creating a connection via MTS OmniChannel.

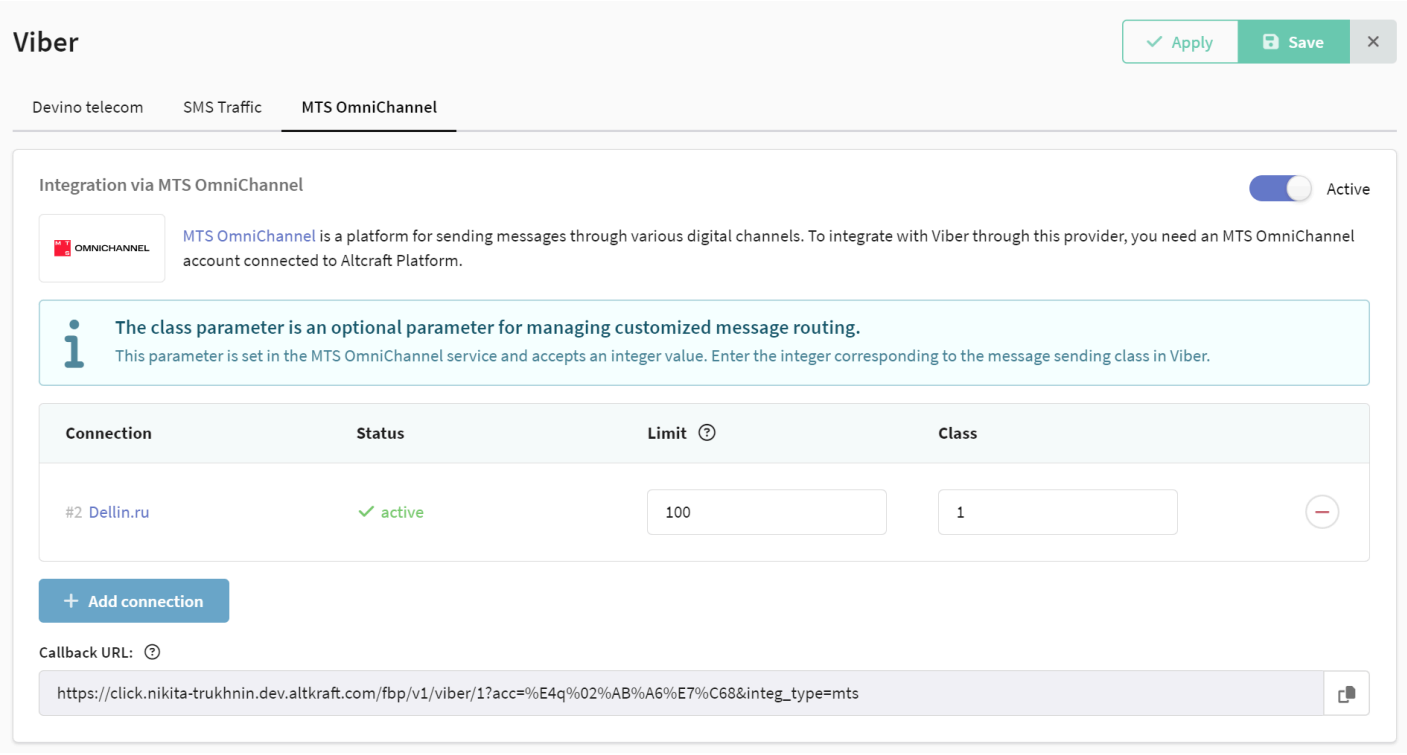


For the integration to work correctly, you will also need to report to tech support your Callback URL specified in the integration.

Information about the added connection will be displayed in the integration. Make sure that the connection status is “active”. Otherwise, check the status of your account in the provider’s personal account. Keep in mind that the account may be blocked if the subscription fee is not paid.

If necessary, you can add multiple accounts. To remove an account from integration, click

.



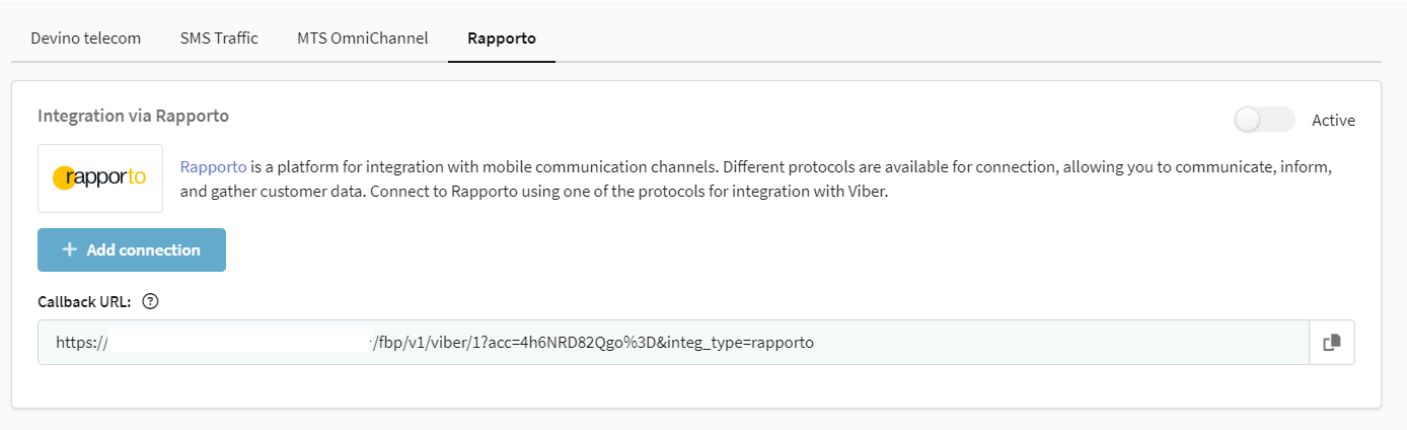
Specify the limit of messages sent per second (0 by default - no limit). Check with your MTC OmniChannel manager to find out your limit.

In the class field, specify the message routing parameter. This parameter is set in the MTS OmniChannel service and accepts an integer value. Enter the integer corresponding to the message sending class in Viber*™.

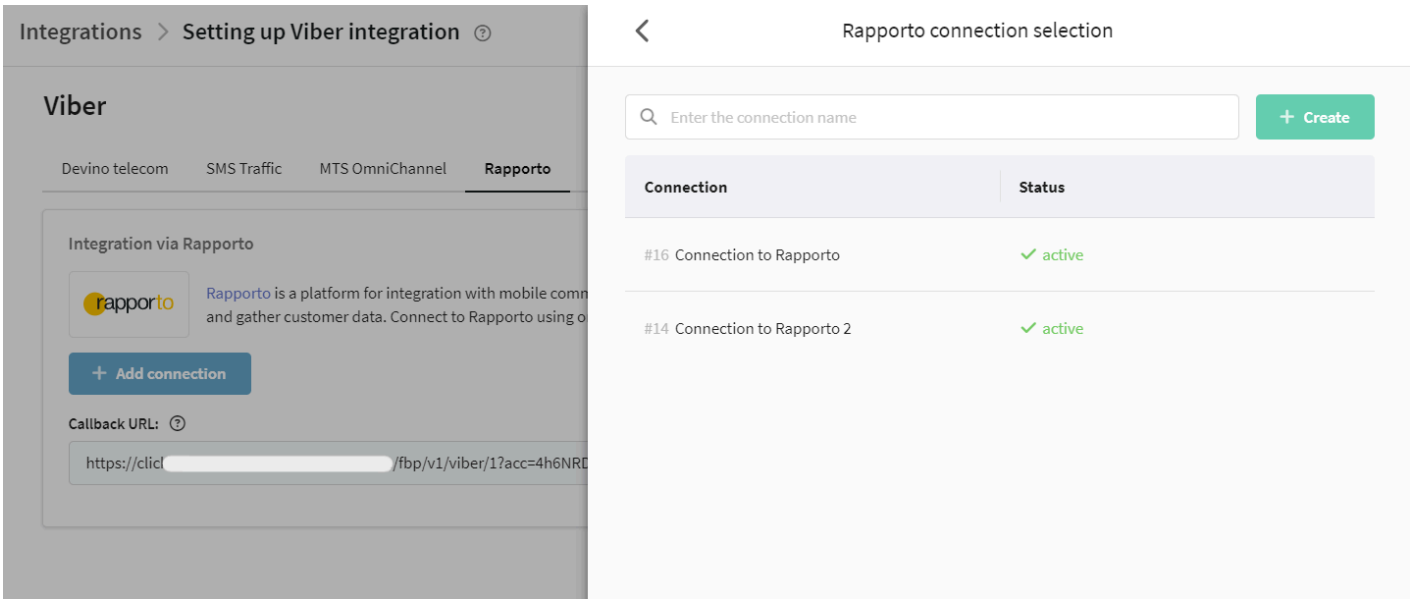
Save the integration using the buttons in the upper right corner. Save the integration using the buttons in the upper right corner.

Rapporto

Activate the integration using the button in the upper right corner. Click on Add Connection:



A window will appear on the right where you need to select an available connection:



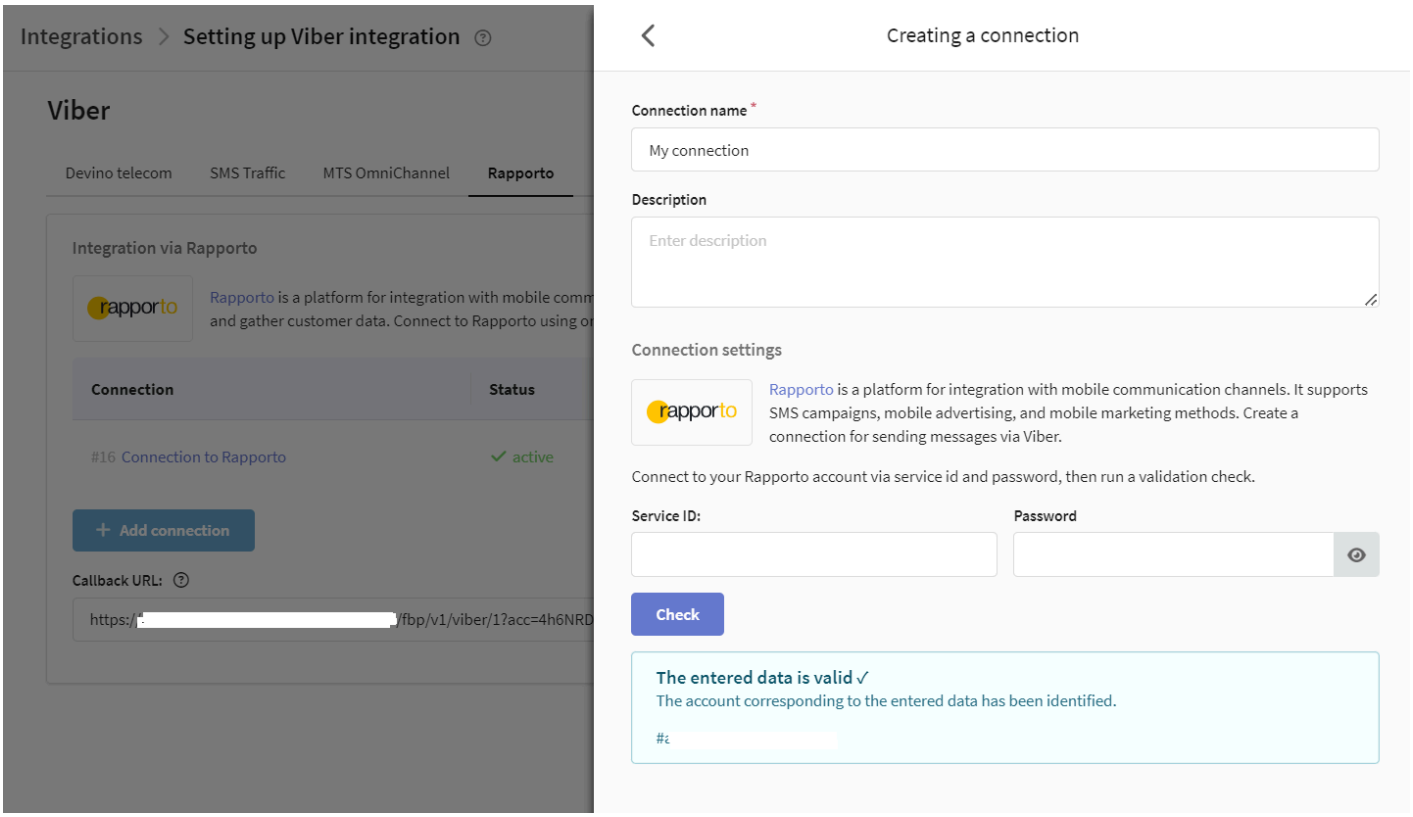
No connections available

If you have not yet set up a connection to the provider, you can create one directly in this window.

1. **Press Create.**
2. **Enter the name and a description of the connection.**
3. **Enter the authorization data.**
4. **Press Check to run a validation check on the entered data.**
5. **If the entered data has been verified, click the Create button in the lower right corner.**

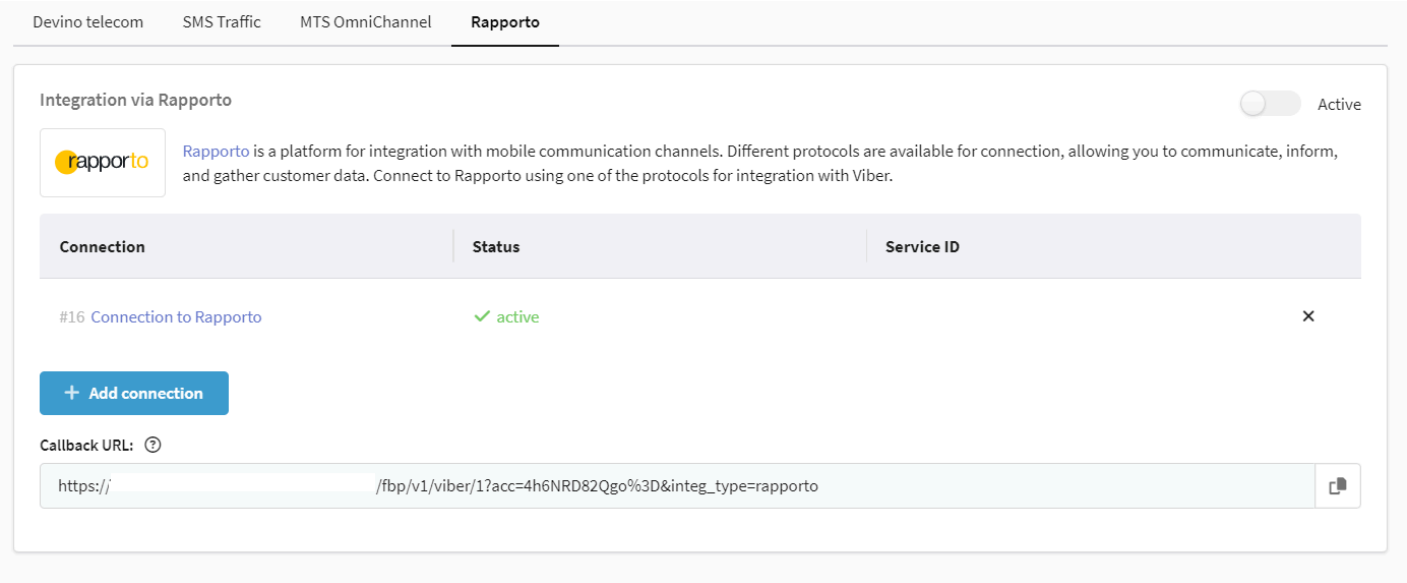
TIP

The documentation portal has published detailed instructions for creating a connection via Rapporto.



For the integration to work correctly, you will also need to report to tech support your Callback URL specified in the integration.

If needed, you can add multiple accounts. To remove an account from the integration, click on the cross.



Save the integration using the buttons in the upper right corner.

Finishing creating an intergration

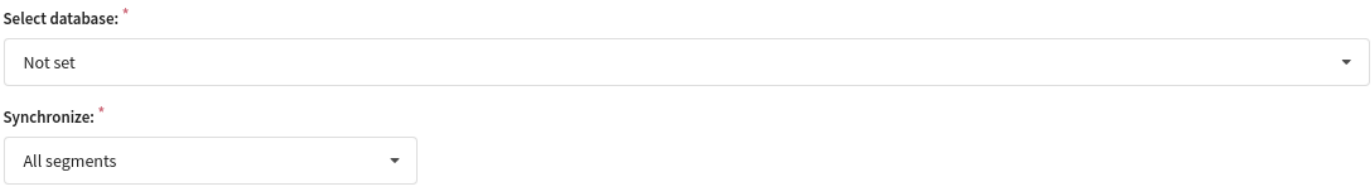
Once you have successfully set up the integration, you can use it to add a new channel to your resource.

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Integration scope

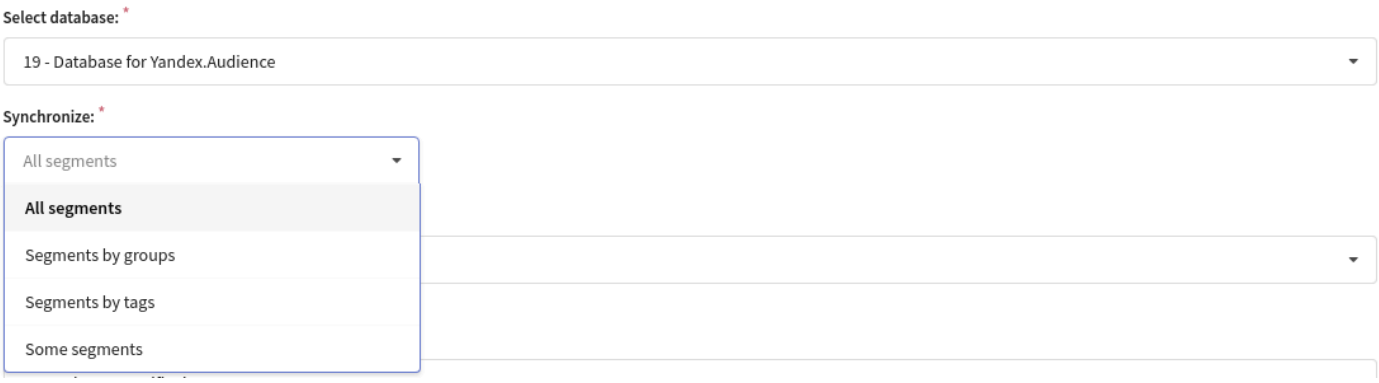
Integration scope is the scope within which the profiles whose data you want to export to external systems are located. It is limited to the database of profiles and static segments.

Setting up the scope is required for all integrations that synchronize with static segments of the Altcraft Marketing platform.



Segments can be grouped as follows:

- **All segments — the data of all static segments associated with the selected database will be exported.**
- **Segments by groups — the data of the segments for which a certain access group is set will be exported. Select the required groups from the menu on the right.**
- **Segments by tags — the data of the segments for which certain custom labels (tags) have been added will be exported. Tags can be added in the main segment settings. Select the desired labels from the menu on the right.**
- **Some segments — in the menu on the right, manually select the required segments.**



Data Transmitted During Synchronization

To synchronize segments based on audiences from external sources (Google Audience, Yandex Audiences™, VK Ads™), hashed profile data is transmitted to the corresponding systems.

This article provides a list of data transmitted during audience synchronization. This information may be required by platform users upon request from their information security department.

Yandex.Audiences™

Authorization	
Application ID	**Client ID from Yandex.OAuth**
Application Secret	**Client Secret from Yandex.OAuth**
Transmitted Data	
email	**Email address. Transmitted in hashed form (MD5)**
phone	**Phone number. Transmitted in hashed form (MD5)**
externalIDValue	**Segment ID in Yandex.Audiences™. Transmitted in hashed form (MD5)**
PushData.SubscriptionID	**Push subscription ID. Transmitted in hashed form (MD5)**

TIP: Data Hashing

Data hashing can be disabled by setting the configuration parameter `YAAUDIENCES_SYNC_DO_HASHING` to `false`.

Google Audience

Authorization	
Application ID	**Client ID**
Secret Key	**Client secret**
Token	**Developer token**
Transmitted Data	
email	**Email. Transmitted in hashed form (SHA256)**
phone	**Phone. Transmitted in hashed form (SHA256)**
Advertising Identifiers	**IDFA / Google Advertising ID and other advertising IDs**
Segment Data	**Segment ID and segment name**

VK Ads™

Authorization	
Client Identifier	**client_id**
Secret Key	**client_secret**
Transmitted Data	
vk	**VK user identifier**
email	**Email address. Transmitted in hashed form (MD5)**
phone	**Phone number. Transmitted in hashed form (MD5)**
ok	**Odnoklassniki user identifier**
advertising_id	**Google Advertising ID**
idfa	**Apple mobile device identifier**

Facebook Ads Manager*™

Authorization	
Application ID	**App ID**
Application Secret Key	**App Secret**
Transmitted Data	
em	**Email. Transmitted in hashed form (SHA256) after normalization**
ph	**Phone. Transmitted in hashed form (SHA256) after normalization**
ct	**City. Transmitted in hashed form (SHA256) after normalization**
country	**Country. Transmitted in hashed form (SHA256) (ISO 3166-1 alpha-2)**
fn	**First name. Transmitted in hashed form (SHA256) after normalization**
ln	**Last name. Transmitted in hashed form (SHA256) after normalization**
ge	**Gender. Transmitted in hashed form (SHA256), `m`/`f`**
db	**Date of birth. Transmitted in hashed form (SHA256) in YYYYMMDD format**
external_id	**Transmitted without hashing (recommended to send in original format)**

Meta, the organization that owns Instagram, Facebook and WhatsApp, is recognized as extremist and banned in the Russian Federation.

Notify

INFO

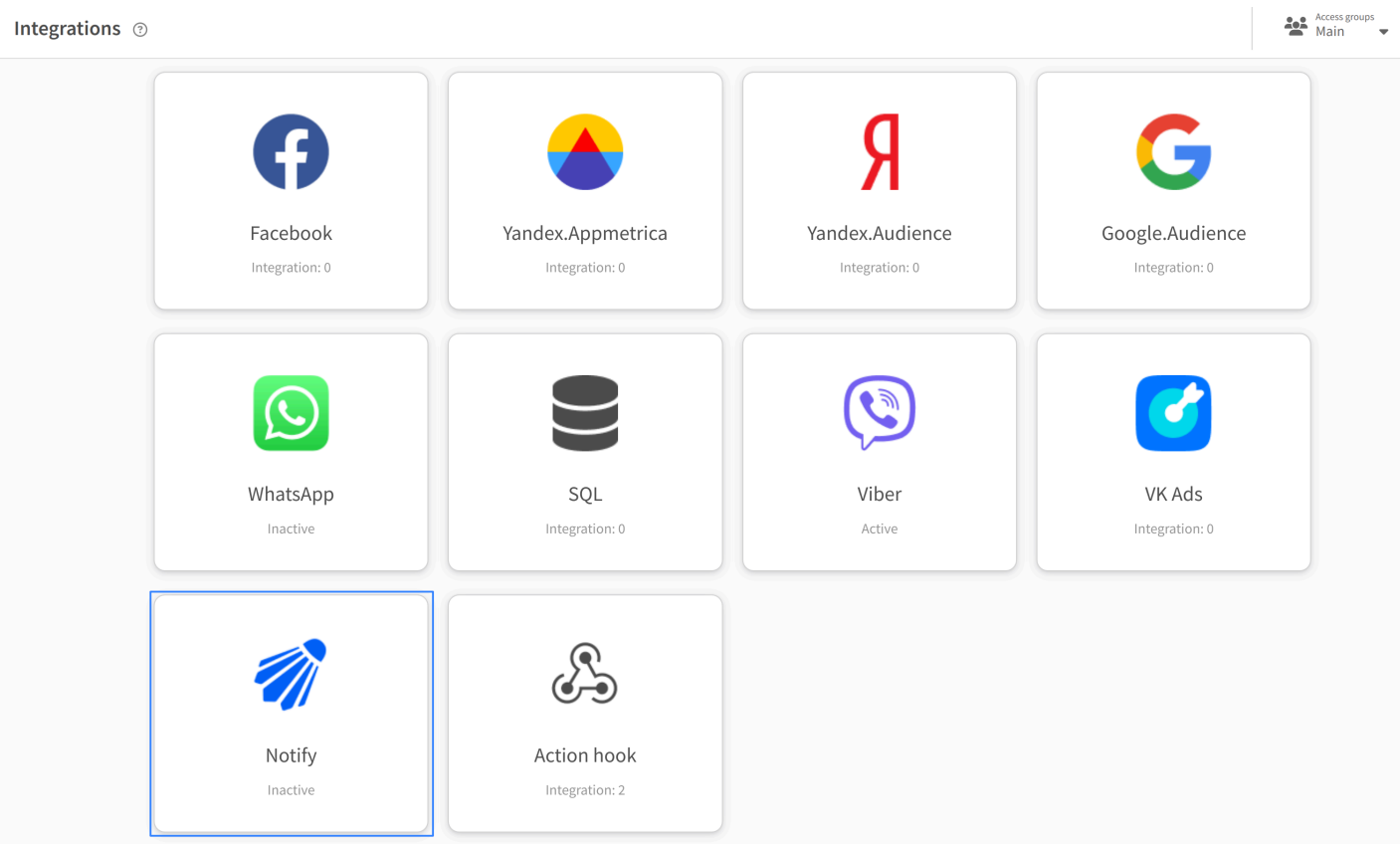
According to the service policy, new access to the Notify service personal account is currently not provided.

Support for current users is provided by MAX Business tech support.

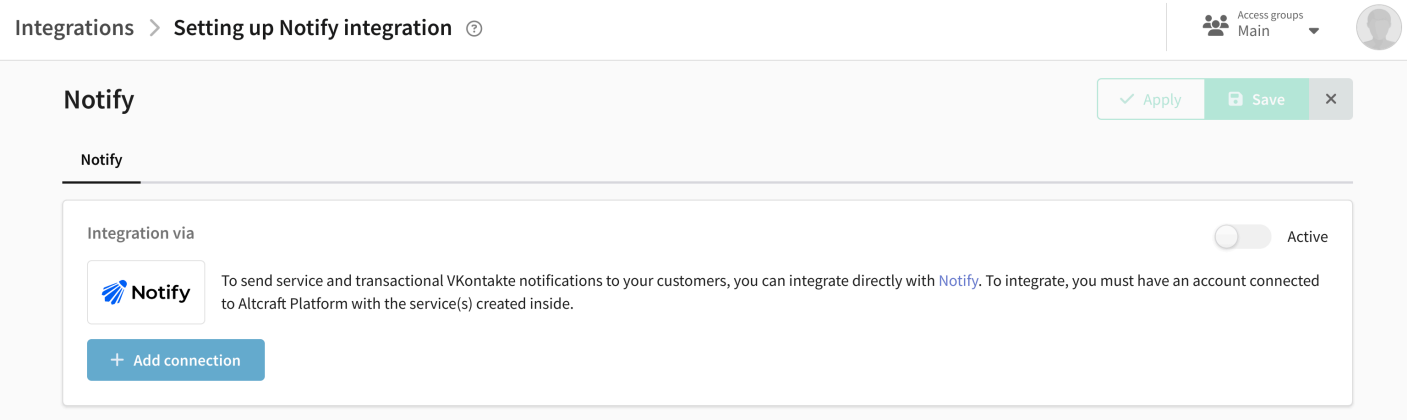
Integration with Notify will allow you to use the service as a communication channel for sending transactional and service notifications. To create an integration, you need to pre-configure a connection to the service. Instructions on how to connect to Notify are available on this page.

Integration settings

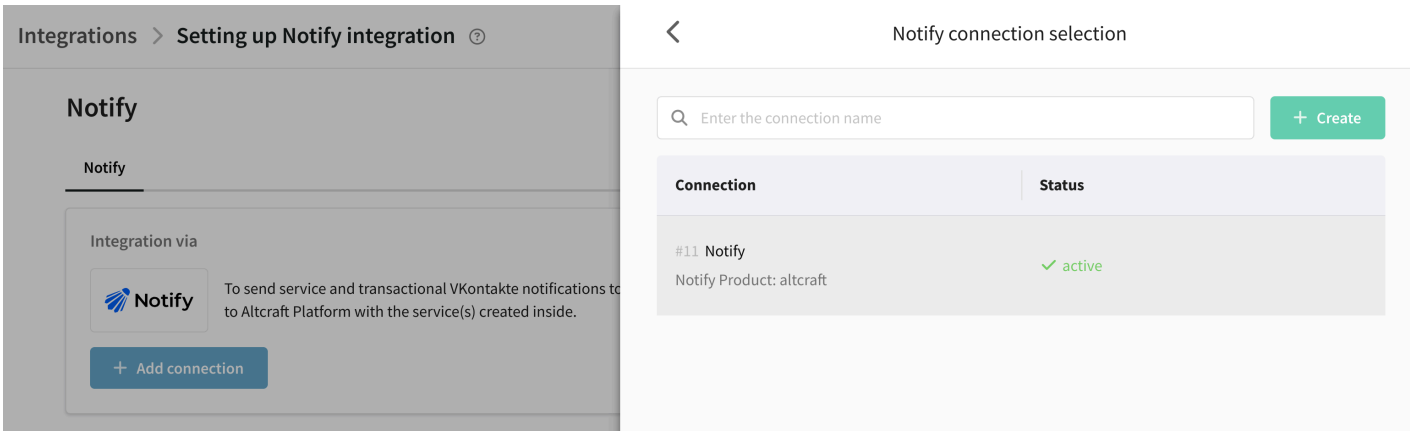
Open the Integrations section and select the integration type - Notify:



Click Add Connection:



Select an available connection to Notify:



No available connections

If you haven’t set up a connection to Notify yet, you can create one right in this window.

Click Create. Enter a name and description for the connection. Next, enter the Application and Signature key parameters and click Check to verify that your data is correct. If the data has been verified, click the Create button in the lower right corner:

Creating a connection

Connection name *

Enter name

Description

Enter description

Connection settings

Notify is a tool that allows you to send service notifications to customers in the social networks VKontakte and Odnoklassniki, thus establishing contact with customers, increasing their loyalty to the brand.

To set up the connection, enter the Application and Signature key, and then run the verification to identify your Notify account.

Application *

Enter the application name

Signature key *

Enter the signature key

Check

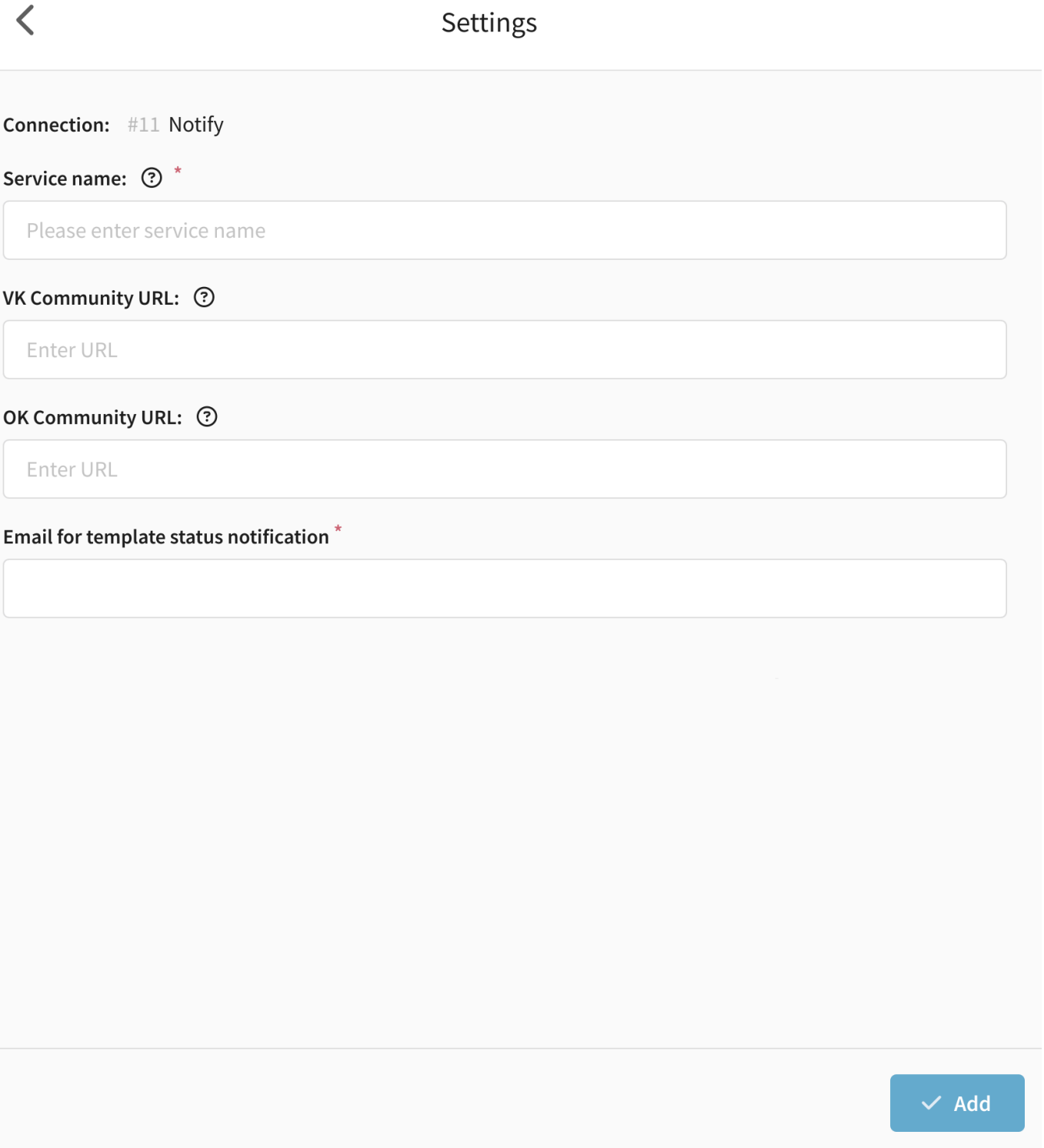
Create

TIP

You can read more about creating a connection ih this article.

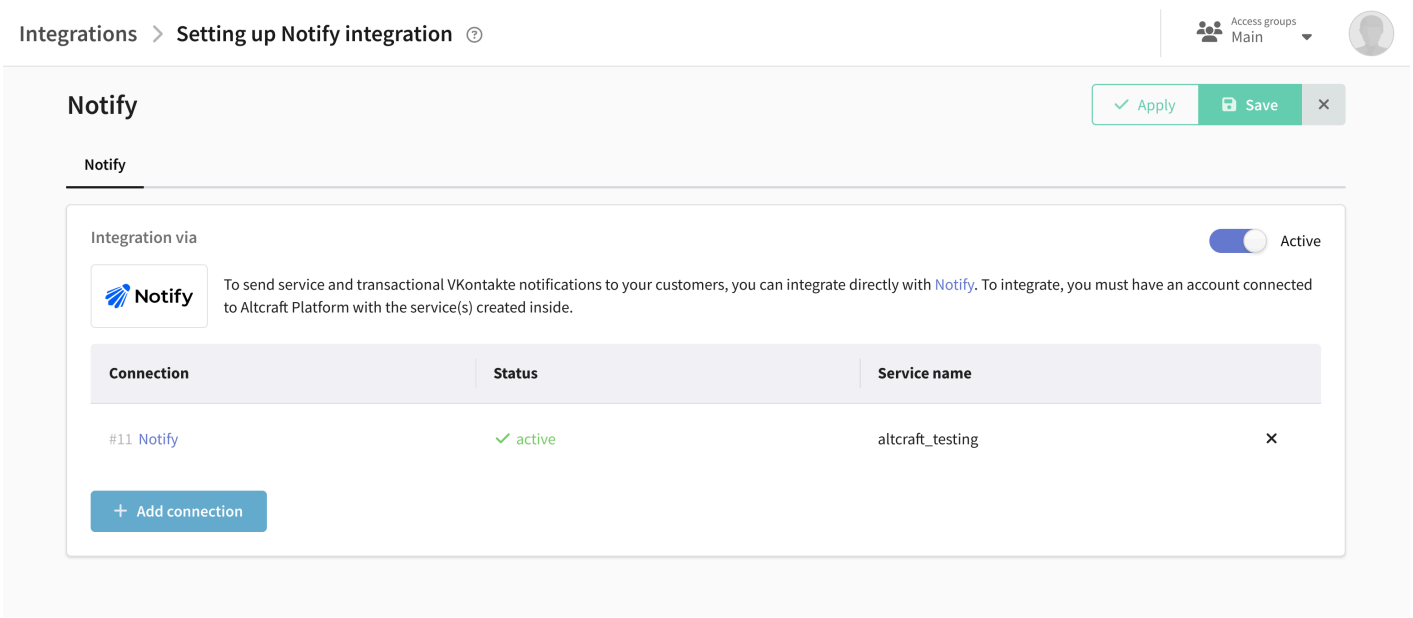
Next, specify integration parameters, for correct work they must exactly match the corresponding parameters from Notify account:

- **Service name - the name of the service for mailing, mandatory field.**
- **VK Community URL - link to the VK community on behalf of which the mailing will be made.**
- **OK Community URL - link to the community in OK on behalf of which the mailing will be sent.**
- **Email for template status notification - e-mail address where reports on template statuses will be sent. Required field.**



After filling in the fields, click the Add button. If the entered data is incorrect, the platform will ask you to verify its correctness. If the data is correct, the parameters will be saved.

Activate the integration using the toggle and save it.



.

After customizing the integration, you can use it to add a Notify feed to resource.

Altcraft action hooks

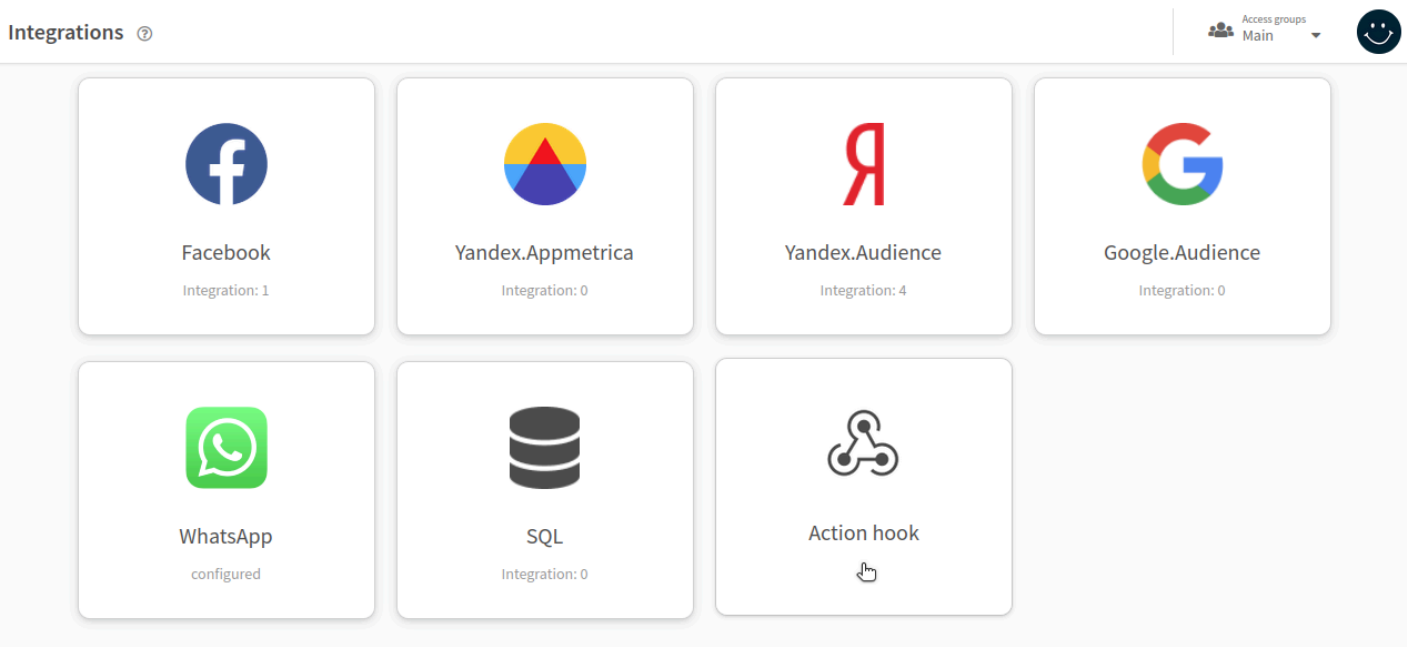
Overview

Action Hooks is a mechanism for notifying external systems of platform events. These can range from system actions related to message delivery or importing subscriber profiles, to customer actions tracked by Altcraft platform pixels on your site or application.

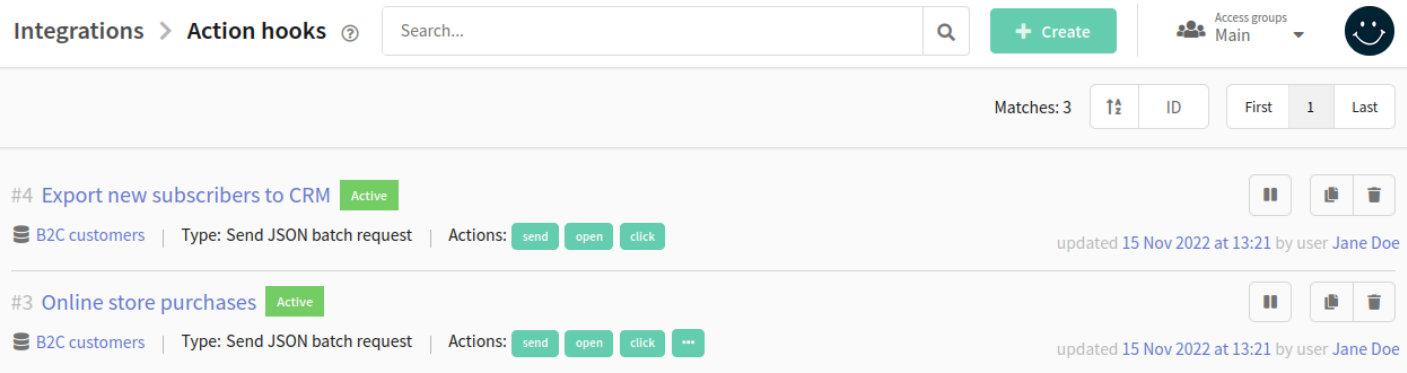
Action Hooks can work with all databases or any specific one. Captured events can be transmitted batch in JSON format over HTTP or sent to the broker queue asynchronously.

Configurable by master user user for a specific database or for all databases at once.

From the main menu, select Integrations → Event Capture to get started:

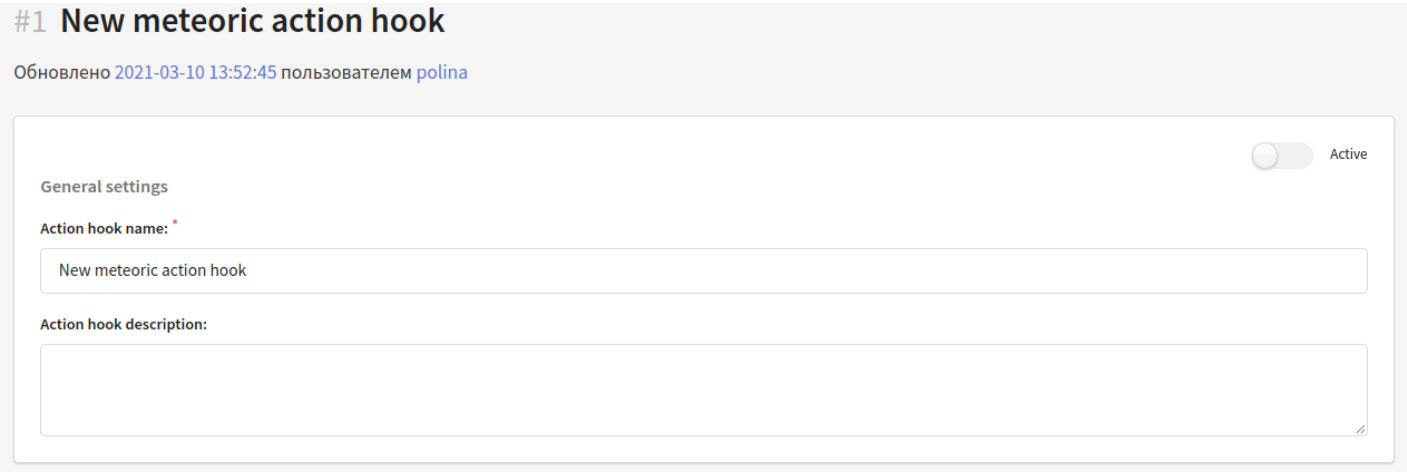


You can activate or stop existing action hooks, clone or delete them using the context menu. If you want to edit a hook, click on its title. To add a new one, click + Create at the top of the page:



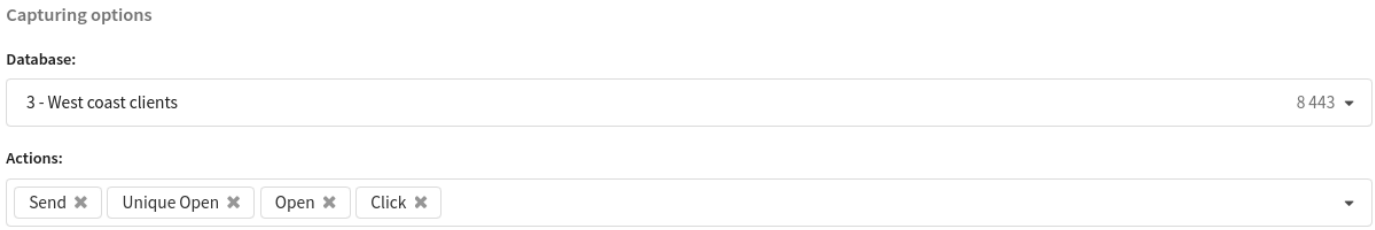
Action hook general settings

In General settings enter a name for the new hook. Here you will also find a switch to activate the event capturing process:



Capturing options

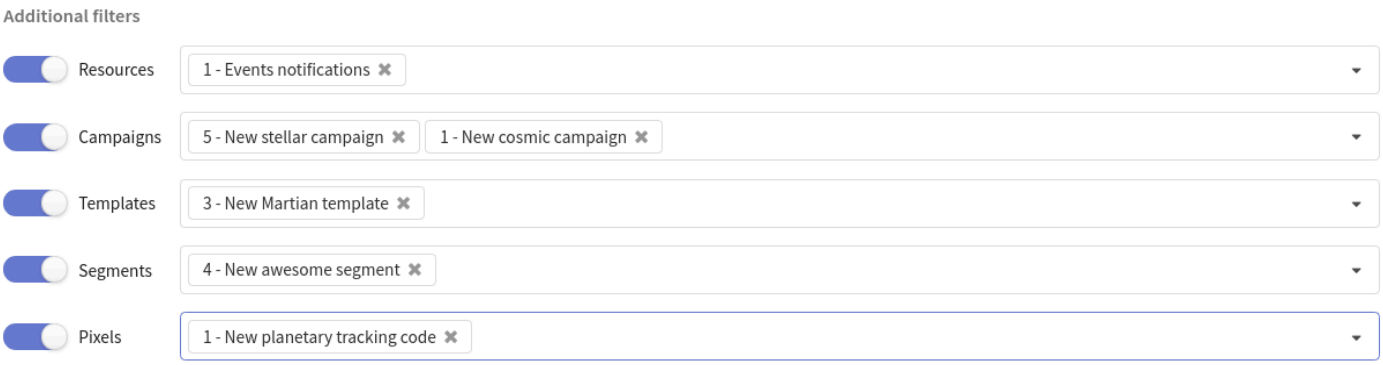
Action hook can work with all [Profile Databases] at once or with one specific one. In the Capture Settings block, select the desired option from the Database drop-down menu:



In the Actions field, select from the drop-down menu the events to be sent to external systems. Event types, their designations in the menu and in the hook request, and descriptions can be found in [this article].

By default, Action hook logs all events of the types you select. You can limit the events to get a narrower selection in the Additional Filters: block.

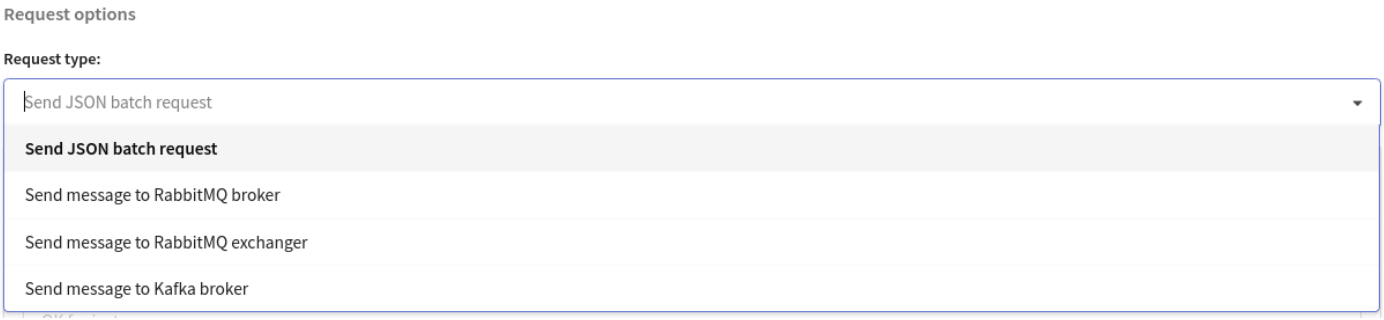
- **Resources - events related to communications within certain [resources](/en/user-guide/profiles-and-databases/66427) will be captured.**
- **Mailing - events from specific marketing [mailings](/en/user-guide/campaigns/) will be captured.**
- **Templates - events from mailings using specific [templates](/en/user-guide/message-templates/managing-templates/) will be captured.**
- **Segments - events related to subscribers entering certain [segments](/en/user-guide/segmentation/) will be captured.**
- **Pixels - actions of subscribers on external resources, registered by certain pixels, will be captured.**



Request options

The following request types are available:

1. **Send a JSON batche request. [Read more].**
2. **Send a message to the RabbitMQ queue. [Read More].**
3. **Send a message to the RabbitMQ exchange. [Read More].**
4. **Send a message to Kafka broker. [Read more].**



INFO

Retry

If for some reason an error occurs during sending a message (e.g. network failure), the message will be sent again after a certain period of time. The number of repeated sends and their frequency are set in the system configuration file “main.json”:

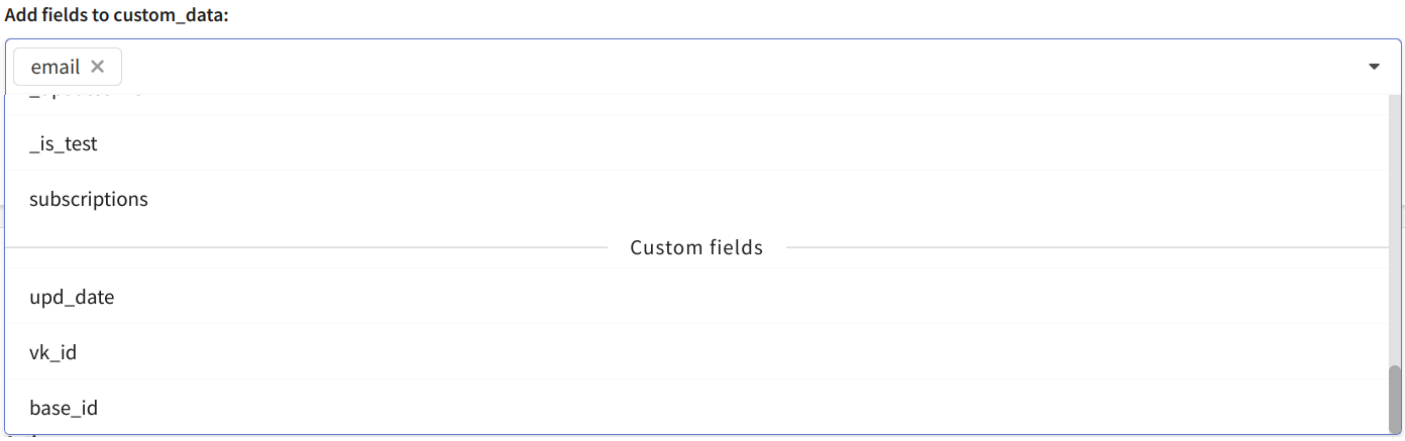
`"HOOK_RETRY_TIMES"` **: set maximum number of attempts (default 4)**

`"HOOK_RETRY_PERIOD_SEC"` **: periods between attempts, in seconds (default 15)**

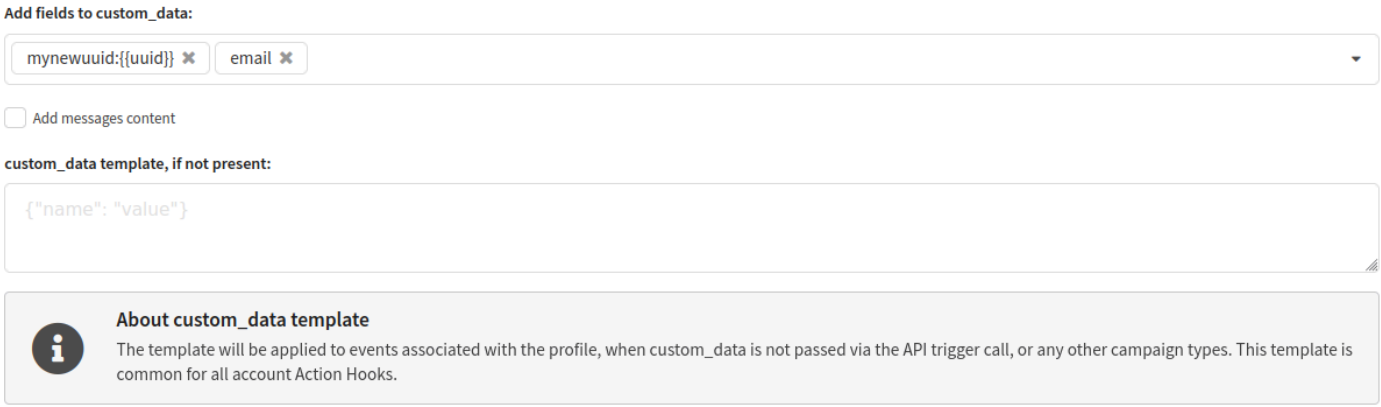
Transmit additional event data in the capture

Client profile data in capture

Along with information about the captured event, you can pass values of standard and additional profile fields. The set of available additional fields depends on the selected database. Select from the drop-down list the profile fields you want to Add fields to custom_data:



The template custom_data can be set below. The platform will use the value from this template in the capture if no data is passed in the `custom_data` **field in the API request to start the trigger.**



Additional data in the hook

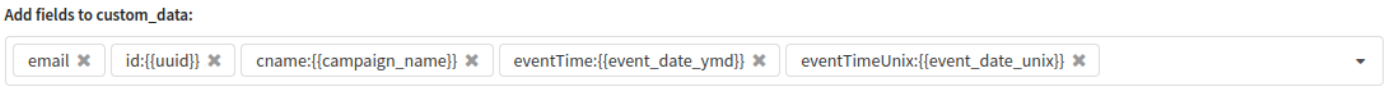
In addition to profile data, you can transmit any data in `mykey:myvalue` **format to the** `custom_data` **object. To do this, manually enter the key and value and then press Enter ↵.**

Template options available:

Parameter	Description
`{{uuid}}`	**Generates new UUIDv4**

Parameter	Description
`{{event_date}}`	**Event date in the format “2006-01-25T15:04:05.999999Z”**
`{{event_date_ymd}}`	**Event date in the format “2006-01-25”**
`{{event_date_rfc3339}}`	**Event date in the format RFC 3339**
`{{event_date_unix}}`	**Event date in unix time, in string**
`{{event_date_unixnano}}`	**Event date in unix time (nanoseconds), in string**
`{{campaign_name}}`	**Retrieves the mailing name if it is relevant to the event, otherwise empty**
`{{campaign_subid}}`	**Retrieves the mailing SubID if it is relevant to the event, otherwise empty**
`{{database_name}}`	**Retrieves the database name if it is relevant to the event, otherwise empty**
`{{segment_name}}`	**Retrieves the segment name if it is relevant to the event, otherwise empty**
`{{message_name}}`	**Retrieves the template name if it is relevant to the event, otherwise empty**
`{{message_subject}}`	**Retrieves the template saubject if it is relevant to the event, otherwise empty**
`{{resource_name}}`	**Retrieves the resource name if it is relevant to the event, otherwise empty**

For example, you add fields to `custom_data`:



The data will be transferred in this way:

```
"custom_data": {
        "cname": "Black Friday",
        "email": "test@example.com",
        "eventTime": "2021-03-17",
        "eventTimeUnix": "1615988820",
        "id": "dc876134-87a4-4437-aabb-d2fc8a3d4213",
    }
```

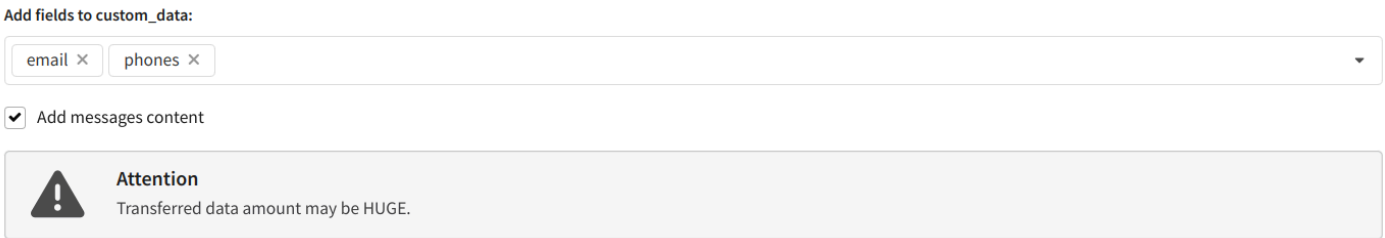
The `custom_data` field can contain any arbitrary data. Fields passed inside it do not have to exist in the profile database and are used only within the event itself. In the event, they appear in the following format:

```
"custom_data": {
    "test_field": "test"
}
```

It is important to note that fields in `custom_data` are not directly tied to the structure of the profile database. When configuring event capture, it may seem that these fields must exist in the database because they are included in the API request, but this is not required. The “Add fields to custom_data” function is used to append additional data—for example, to include values from a profile or other sources - even if those fields were not originally present in the request.

Message content in hook

In order to transmit the content of all messages along with events, activate the option Add messages content at the bottom of the page.



The content will be passed to the `"content"` field. The field structure for each channel is described in [this article](#).

CAUTION

Please note that this will store all sent messages in storage for a fixed amount of time. This will require additional disk space. The default retention time is 72 hours; it is set by the global setting `HOOK_CONTENT_CLEANUP_HOURS` in the main.json file. After this time, events will be transmitted without any content associated with them.

Exporting Historical Events for a Period

This function allows you to export to an integration the data about events that have already occurred on the platform. It is useful for data recovery if event capturing was not configured in time or if a failure occurred.

NOTE

The following event types are not available for historical export: `campaign_launch`, `profile_merge`, `profile_split`.

Export Setup

To start exporting historical events, enable and save the event capture with the desired settings. Then, in the control block, select the option Capture historical events for a period, specify the start and end dates, and click Start:

INFO: Note

The period for exporting historical events is limited: the minimum date is 01-01- 1970, and the maximum is the previous day. The current date cannot be selected.

To verify the export operation, use the testing tool — the system will generate an example event of the selected type.

Managing the Process

After starting the export, you can monitor progress and manage execution:

- **Suspend — temporarily stop the export**
- **Resume — continue the export**
- **Abort — completely cancel the export**

Execution Statuses

- **In Progress —** `The export of historical events for the selected period is in progress`
- **Paused —** `The export process of historical events has been paused`
- **Completed —** `The export of historical events for the period has been completed`
- **Error —** `An error occurred during the export of historical events`

NOTE

If you deactivate event capture during the export of historical events, the process will be paused until reactivation.

Action hooks event types

Interface name	API name	Description
Email channel events		
Send	**send**	**Email is sent to a recipient.**
Reply	**reply**	**Subscriber's reply to received email.**
Unique Open	**uopen**	**Unique email message opening.**
Open	**open**	**Any email opening.**
Unique Click	**uclick**	**Unique email link click. It creates open and uopen events when *open* event is not captured.**
Click	**click**	**Any email link click.**
Confirm	**confirm**	**Confirmation link click in a regular message (creates click and uclick, if it is the first click for this message).**
Confirm Subscription	**confirmsub**	**Confirmation link click in a double opt-in message.**
Unsubscribe	**unsub**	**Recipient opted out by clicking a message link `{unsubscribe}` or `{globalunsubscribe}`.**
Complain	**complain**	**A recipient complained or reported spam.**
SoftBounce	**sbounce**	**Temporary delivery problems provider message. Soft bounce.**
HardBounce	**hbounce**	**Invalid address or similar delivery problems message. Hard bounce.**
Suppression	**suppress**	**Email was not sent: recipient address is in suppression list.**
Delivered	**deliv**	**Email delivered to a recipient.**
Undelivered	**undeliv**	**Email could not be delivered.**
Read Message	**read**	**Message remained open for 8 seconds or longer.**
Glanced Message	**glanced**	**Message remained open for less than 4 seconds.**
Skimmed Message	**skimmed**	**Message remained open for 4 to 8 seconds.**
Subscriber Profile Import Events		
Manual Import	**import_manual**	**Manual profile import via platform interface.**
File Import	**import_file**	**From file profile import via platform interface.**
API Import	**import_api**	**API profile import.**
SMS channel events		
Send SMS	**send_sms**	**SMS sent to a recipient.**
Click SMS	**click_sms**	**Any SMS link click.**
Unique Click SMS	**uclick_sms**	**Unique SMS link click.**
Delivered SMS	**deliv_sms**	**SMS delivered to a recipient.**
Undelivered SMS	**undeliv_sms**	**SMS could not be delivered.**
Unsubscribe SMS	**unsub_sms**	**Recipient opted out by clicking a message link `{unsubscribe}` or `{globalunsubscribe}`.**
Push channel events		
Unique Open Push	**uopen_push**	**Unique push open.**
Open Push	**open_push**	**Any push open.**
Send Push	**send_push**	**Push sent to a recipient.**
Click Push	**click_push**	**Any push link click.**
Unique Click Push	**uclick_push**	**Unique push link click.**
Delivered Push	**deliv_push**	**Push delivered to a recipient.**
Undelivered Push	**undeliv_push**	**Push could not be delivered to a recipient.**
SoftBounce Push	**sbounce_push**	**Temporary delivery failure (network issues, push service unavailability for any reason). This event means that the platform will continue attempting to deliver the message.**
Hardbounce Push	**hbounce_push**	**Permanent delivery failure (Invalid subscription id). After receiving such a response, the platform will not attempt to deliver the message again.**
Custom (API) opt out		
API unsubscribe	**unsub_api**	**API profile opt out.**
Mailing launch		
Mailing launch	**campaign_launch**	**Mailing launch. It has a custom hook message structure. Not available for [historical events export.](/user-guide/integrations/action-hooks/37849833#export)**
Mailing errors		
General: Error	**camp_error**	**Errors that occurred during the campaign.**
The events that are captured by Pixels in the external resources		
Pixel Open	**pixel_open**	**Pixel script activation. You can find out more about the parameters of this request [there](/en/developer-guide/goals/2195843).**
Promo code events		
Promocode Attach	**promocode_attach**	**The subscriber is assigned a promotional code.**
Promocode Detach	**promocode_detach**	**The promo code is unlinked from the subscriber's profile.**
Promocode Activate	**promocode_activate**	**The promo code is activated.**
Events related to the automatic creation of a static segment during import		
Static Segment Added	**segs_add**	**The profile entered the static segment.**
Static Segment Removed	**segs_remove**	**The profile exited the static segment.**
Events of merging multiple profiles or splitting one profile		
Profile merge	**profile_merge**	**Multiple profiles are merged into a new or existing profile through an API request. Not available for [historical events export.](/user-guide/integrations/action-hooks/37849833#export)**
Profile split	**profile_split**	**A profile is divided into several new or existing profiles through an API request. Not available for [historical events export.](/user-guide/integrations/action-hooks/37849833#export)**

MAX Bot channel events		
MAX bot delivery	**max_bot_deliv**	**The message was delivered to the chat with the MAX bot.**
MAX bot undelivery	**max_bot_undeliv**	**The message was not delivered to the chat with the MAX bot.**
MAX bot send	**max_bot_send**	**The message was sent for delivery to the chat with the MAX bot.**
MAX bot click	**max_bot_click**	**Non-unique click on a link in a message in the chat with the MAX bot.**
MAX bot unique click	**max_bot_uclick**	**Unique click on a link in a message in the chat with the MAX bot.**
MAX bot subscription	**max_bot_subscribe**	**The profile subscribed to the MAX bot.**
MAX bot unsubscribe	**max_bot_unsub**	**The profile unsubscribed from the MAX Bot channel.**
MAX bot complaint	**max_bot_complain**	**The subscriber complained about the message.**
MAX bot sending policy violation	**max_bot_offence**	**The limit on the number of messages was exceeded.**
MAX Group channel events		
MAX group delivery	**max_group_deliv**	**The message was delivered to the MAX channel.**
MAX group undelivery	**max_group_undeliv**	**The message was not delivered to the MAX channel.**
MAX group send	**max_group_send**	**The message was sent for delivery to the MAX channel.**
MAX group click	**max_group_click**	**Non-unique click on a link in a message sent to the MAX channel.**
MAX group unique click	**max_group_uclick**	**Unique click on a link in a message sent to the MAX channel.**
MAX group subscription	**max_group_subscribe**	**The profile subscribed to the MAX Group channel.**
MAX group unsubscribe	**max_group_unsub**	**The profile unsubscribed from the MAX Group channel.**
MAX group complaint	**max_group_complain**	**The subscriber complained about the message.**
MAX group sending policy violation	**max_group_offence**	**The limit on the number of messages was exceeded.**
Telegram Bot events		
Telegram bot send	**telegram_bot_send**	**Message has been sent for delivery to a subscriber in a chat with a Telegram bot.**
Telegram bot delivered	**telegram_bot_deliv**	**The message has been delivered to a chat with a Telegram bot.**
Telegram bot click	**telegram_bot_click**	**A non-unique click on a link in a message in a chat with a Telegram bot.**
Telegram bot unique click	**telegram_bot_uclick**	**A unique click on a link in a message.**
Telegram bot undelivered	**telegram_bot_undeliv**	**The message has not been delivered to a chat with a Telegram bot.**
Telegram bot subscription	**telegram_bot_subscribe**	**A profile has subscribed to a bot by initiating a dialogue.**
Telegram Group events		
Telegram group send	**telegram_group_send**	**Message has been sent for delivery to a subscriber in a chat with a Telegram bot.**
Telegram group delivered	**telegram_group_deliv**	**Message has been delivered in a Telegram group.**
Telegram group click	**telegram_group_click**	**Click on the link in the message in a Telegram group.**
Telegram group undelivered	**telegram_group_undeliv**	**Message has not been delivered to a Telegram group.**
WhatsApp* events		
WhatsApp* send	**whatsapp_send**	**Message has been sent for delivery to a subscriber in a WhatsApp* chat.**
WhatsApp* deliv	**whatsapp_deliv**	**Message has been delivered in a WhatsApp* chat.**
WhatsApp* click	**whatsapp_click**	**A non-unique click on a link in a message in a chat.**
WhatsApp* unique click	**whatsapp_uclick**	**A unique click on a link in a message.**
WhatsApp* undeliv	**whatsapp_undeliv**	**Message has not been delivered in a WhatsApp* chat.**
WhatsApp* read	**whatsapp_read**	**Message has been read by a user.**
WhatsApp* subscription	**whatsapp_subscribe**	**Profile has been subscribed to a WhatsApp* channel.**
WhatsApp* unsubscribe	**whatsapp_unsub**	**Profile has been unsubscribed from a WhatsApp* channel.**
Viber events		
Viber send	**viber_send**	**Message has been sent for delivery to a subscriber in a Viber chat.**
Viber deliv	**viber_deliv**	**Message has been delivered in a Viber chat.**
Viber click	**viber_click**	**Click on a link in the message. If the recipient clicks on the same link multiple times, each click event is recorded.**
Viber unique click	**viber_uclick**	**Unique click on a link in the message. If the recipient clicks on any link at least once, one unique click event is recorded.**
Viber undeliv	**viber_undeliv**	**Message was rejected by Viber or Devino.Online.**
Viber read	**viber_read**	**Message was read by the recipient.**
Viber subscription	**viber_subscribe**	**Profile subscribed to the Viber channel, i.e., agreed to receive messages from the business account.**
Viber unsubscribe	**viber_unsub**	**Profile unsubscribed from the Viber channel, i.e., blocked receiving messages from the business account.**
Notify channel events		
Notify delivered	**notify_deliv**	**Message has been delivered.**
Notify undelivered	**notify_undeliv**	**Message has not been delivered.**
Notify send	**notify_send**	**Message has been sent.**
Notify click	**notify_click**	**Click on a link in the message. If the recipient clicks on the same link multiple times, each click event is recorded.**
Notify unique click	**notify_uclick**	**Unique click on a link in the message. If the recipient clicks on any link at least once, one unique click event is recorded.**
Notify subscription	**notify_subscribe**	**Profile has been subscribed to a Notify channel.**
Notify read	**notify_read**	**Message has been read by a user.**
Notify unsubscribe	**notify_unsub**	**Profile has been unsubscribed from a Notify channel.**
Notify complain	**notify_complain**	**A recipient complained or reported spam.**
Pop-up events		
Popup: Post	**popup_post**	**The popup form has been completed and submitted.**
Popup: Click	**popup_click**	**Click on any popup element except the "Close" button.**

Popup: Error	**popup_error**	**Any error in the popup.**
Popup: Bounce	**popup_bounce**	**While submitting the popup form, technical problems occurred (for example, a server failure).**
Popup: Close	**popup_close**	**Click on the "Close" button in the popup.**
Popup: Open	**popup_open**	**The popup is shown to the user.**
Popup: Custom	**popup_custom**	**An event configured on the client side was registered.**
Loyalty program events		
Loyalty: Registration	**loyalty_register**	**Profile registration in the loyalty program.**
Loyalty: Unregistration	**loyalty_unregister**	**Profile exclusion from the loyalty program.**
Loyalty: Level Up	**level_up**	**Loyalty level upgrade.**
Loyalty: Level Down	**level_down**	**Loyalty level downgrade.**
Loyalty: Points Accrual	**points_accrual**	**Loyalty points accrual.**
Loyalty: Points Activation	**points_activate**	**Loyalty points activation.**
Loyalty: Points Expiration	**points_expire**	**Loyalty points expiration.**
Loyalty: Points Redeem	**points_redeem**	**Loyalty points redemption.**
Loyalty: Cancel Accrual	**points_cancel_accrual**	**Loyalty points accrual cancellation.**
Loyalty: Cancel Redemption	**points_cancel_redeem**	**Loyalty points redemption cancellation.**
Loyalty: Points Reset	**points_q_reset**	**Loyalty points reset.**
Campaign events		
Campaign: joined audience	**sc_add**	**Profile added to the campaign audience**
Campaign: failed to join	**sc_gcg_suppress**	**Profile not added to the campaign audience because it is in the global control group**
Campaign: joined test group	**sc_test_group_assign**	**Profile assigned to the test group**
Campaign: joined control group	**sc_control_group_assign**	**Profile assigned to the control group**
Campaign: left the audience	**sc_remove**	**Profile removed from the campaign audience**
Added to Campaign Audience	**sc_add**	**The profile has been added to the campaign audience.**
Assigned to Test Group	**sc_test_group_assign**	**The profile has been assigned to a test group.**
Assigned to Control Group	**sc_control_group_assign**	**The profile has been assigned to a control group.**
Removed from Campaign Audience	**sc_remove**	**The profile has been removed from the campaign audience.**
Suppressed by GCG from Campaign	**sc_gcg_suppress**	**The profile was not added to the campaign audience due to being in the GCG (Global Control Group).**
GCG events		
GCG: Add	**gcg_add**	**Registered GCG entry event.**
GCG: Remove	**gcg_remove**	**Registered GCG exit event.**

****The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.***

Action Hook Message Structure

General structure

General hook message structure for most event types:

Parameter	Type	Required	Example	Description
event_type	**string**	**Yes**	**“click”**	**Event name as in the event types table**
event_date	**string**	**Yes**	**“2018-03-04T13:13:19.034315796Z”**	**Event timestamp in RFC 3339 format**
profile_id	**string**	**No**	**“abcdefgh12345678”**	**Recipient profile ID**
_xxh	**string**	**No**	**“ad1850e0d1d9481e”**	**Subscriber email hash used for identification in Altcraft**
_md5	**string**	**No**	**“145bb30f7ddf1184cb560bc4b7705646”**	**MD5 hash of the subscriber’s email**
list_id	**int**	**No**	**111**	**Database ID where the event occurred**
account_id	**int**	**Yes**	**23**	**Account ID where the event occurred**
sends_trying_num	**int**	**No**	**2**	**Number of message sending attempts (present only if the number of attempts is greater than 0)**
is_test	**bool**	**Yes**	**false**	**Whether the event was triggered by a test send**
is_static	**bool**	**No**	**false**	**Whether dynamic substitutions were used in the sent content**
subscription_hash_id	**string**	**No**	**“0a4b910312345”**	**Subscription hash**
custom_data	**JSON object**	**No**	`{“alice@example.org”}`	**Subscriber data set configured in the Action Hook editing interface**
event_data	**JSON object**	**Yes**	`{“browser”: “Firefox”}`	**Data set that depends on the event type (there are two types: tracking events and events that occur during delivery)**
subscription	**array**	**No**	`[ { “resource_id”: 123, “channel”: “email”, “email”: “user@example.com”, “phone”: “+79991234567”, “subscription_id”: “abc123”, “provider”: “androidfirebase”, “priority”: 1, “status”: “subscribed”, “ccsub”: { “custom_key”: “custom_value” } } ]`	**Array with subscription details including information about the resource, communication channel, recipient, and subscription status. The structure is described below.**

INFO: Note

When exporting historical events, message content is not transferred. Events are sent in the same format as in real time, and `null` is passed for deleted profiles.

event_data structure

Campaign events

For events of the following types:

- **sc_add — profile added to the campaign audience**
- **sc_gcg_suppress — profile not added to the campaign audience because it is in the global control group**
- **sc_test_group_assign — profile assigned to the test group**
- **sc_control_group_assign — profile assigned to the control group**
- **sc_remove — profile removed from the campaign audience**

Parameter	Type	Required	Example	Description
scampaign_id	**int**	**No**	**1**	**Campaign ID in Altcraft**
segment_id	**int**	**No**	**5**	**ID of the segment related to the campaign audience**
prev_st	**int**	**No**	**1**	**Group ID: 1 — test group, 2 — global control group**

Tracking events

For the following event types:

- **Email: uopen, open, uclick, click, confirm, confirmsub, unsub, read, glanced, skimmed**
- **SMS: uclick_sms, click_sms, unsub_sms**
- **Push: uopen_push, open_push, click_push, uclick_push**

Parameter	Type	Required	Example	For which channel	Description
message_id	**int**	**Yes**	**17**	**Email, SMS, Push**	**Message ID in the Altcraft system**
campaign_id	**int**	**Yes**	**973**	**Email, SMS, Push**	**Mailing ID in the Altcraft system**
scampaign_id	**int**	**No**	**973**	**For all**	**Campaign ID in the Altcraft system**
segment_id	**int**	**No**	**0**	**Email, SMS, Push**	**Segment ID, if available**
resource_id	**int**	**Yes**	**3**	**Email, SMS, Push**	**Resource ID**
ip	**string**	**No**	**“10.9.0.14”**	**Email, Push**	**Subscriber IPv4 address**
country	**string**	**No**	**“.us” “.ru”**	**Email, Push**	**Country in TLD format**

Parameter	Type	Required	Example	For which channel	Description
city	string	No	“New York” “Moscow”	Email, Push	Subscriber city
postal_code	string	No	“101000”	Email, Push	Subscriber postal code
region	string	No	“Ohio” “Moskovskaya oblast”	Email, Push	Subscriber region
tz	string	No	“Europe/Vaduz”	Email, Push	Subscriber time zone
lat	float	No	55.244075	Email, Push	Geographic latitude
lon	float	No	55.244075	Email, Push	Geographic longitude
user_agent	string	No	“Mozilla/5.0 (X11; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0”	Email, Push	User-Agent header string
os	string	No	“Linux x86_64”	Email, Push	Operating system
browser	string	No	“Firefox”	Email, Push	Browser and version
device	string	No	“web”	Email, Push	Device
wick	int	No	1	Email, Push	Event source, if 1 - API
link_url	string	No	“http://www.example.com”	Email, SMS, Push	HTTP link the user was redirected to
launch_id	string	Yes	“00000017000003cd5a9bf01a”	Email, SMS, Push	Launch ID
send_message_id	string	Yes	“5a9bf01a17_3cd_11_6f_0_2.5a8c045fc1dde44cbf718f78”	Email, SMS, Push	Sent message ID
full_message_id	string	Yes	“fQIQs4FgAAEBIcARADkxODIKR8-gA”	Email, SMS, Push	Full SendMessageID
email_message_id	string	No	“mzi5c1~+9vqekqmy63mh8xhgt2qjwfkad6lmiafbyn-0te1_uzckfqyi4ry”	Email	Email-safe SendMessageID
extra_uid	string	No	“ew4Sgf41Qwjq8_2_2E_w_k__t.2SDvJv8bvBYQFtSDx”	Push	Extra UID
provider	string	No	“YandexAppMetricaDeviceIDAndroid”	Push	Provider
prev_st	int	No	1	For all	Group identifier: 1 - test, 2 - control

Messaging events

For the following event types:

- **Email: send, deliv, suppress, complain**
- **SMS: send_sms, deliv_sms**
- **Push: send_push, deliv_push**

Parameter	Type	Required	Example	For which channel	Description
message_id	int	Yes	17	Email, SMS, Push	Message ID in the Altcraft system
campaign_id	int	Yes	973	Email, SMS, Push	Mailing ID in the Altcraft system
segment_id	int	No	0	Email, SMS, Push	Segment ID, if available
resource_id	int	Yes	3	Email, SMS, Push	Resource ID in the Altcraft system
launch_id	string	Yes	“00000017000003cd5a9bf01a”	Email, SMS, Push	Launch ID
send_message_id	string	Yes	“5a9bf01a17_3cd_11_6f_0_2.5a8c045fc1dde44cbf718f78”	Email, SMS, Push	Sent message ID
full_message_id	string	Yes	“fQIQs4FgAAEBIcARADkxODIKR8-gA”	Email, SMS, Push	Full SendMessageID
email_message_id	string	No	“mzi5c1~+9vqekqmy63mh8xhgt2qjwfkad6lmiafbyn-0te1_uzckfqyi4ry”	Email	Email-safe SendMessageID
extra_uid	string	No	“ew4Sgf41Qwjq8_2_2E_w_k__t.2SDvJv8bvBYQFtSDx”	Push	Extra UID
provider	string	No	“YandexAppMetricaDeviceIDAndroid”	Push	Provider
workflow_id	int	No	1	For all	Scenario ID, if the mailing was launched from a scenario

For events sbounce, hbounce, sbounce_push, hbounce_push, the following parameters are also passed:

Parameter	Type	Required	Example	For which channel	Description
bounce_code	int	Yes	500	Email, Push	Server response code
bounce_message	string	Yes	“No such email”	Email, Push	Server response

For events undeliv, undeliv_sms, undeliv_push, the following parameters are passed:

Parameter	Type	Required	Example	For which channel	Description
undeliv_code	int	Yes	400	Email, SMS, Push	Error code
undeliv_message	string	Yes	“Invalid Caller ID”	Email, SMS, Push	Error message

Pixel events

For pixel_open events:

Parameter	Type	Required	Example	Description
ip	string	Yes	“10.9.0.14”	Subscriber IPv4 address
country	string	Yes	".ru "	Country in TLD format
city	string	Yes	“Moscow”	Subscriber city
pixel_id	int	Yes	41	Pixel ID
postal_code	string	Yes	“390000”	Subscriber postal code

Parameter	Type	Required	Example	Description
region	**string**	**Yes**	**"Ryazanskaya obl"ast**	**Subscriber region**
tz	**string**	**Yes**	**“Europe/Vaduz”**	**Subscriber time zone**
lat	**float**	**Yes**	**55.244075**	**Geographic latitude**
lon	**float**	**Yes**	**55.244075**	**Geographic longitude**
user_agent	**string**	**Yes**	**“Mozilla/5.0 (X11; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0”**	**User-Agent header string**
os	**string**	**Yes**	**“Linux x86_64”**	**Operating system**
browser	**string**	**Yes**	**“Firefox”**	**Browser and version**
device	**string**	**Yes**	**“web”**	**Device**

The full list of possible parameters is available in this article.

Popup events

For all popup_post, popup_click, popup_error, popup_bounce, popup_close, popup_open events:

Parameter	Type	Required	Example	Description
popup_id	**int**	**Yes**	**123**	**Popup ID**
referer	**string**	**Yes**	**“api.example.com”**	**Referer request header**
smid	**string**	**Yes**	**“fGmXiWU9Y4NSIpCwl4QQQQEQEAABCRA”**	**Sent message identifier**
utm_campaign	**string**	**No**	**“utm_campaign”**	**Mailing UTM tag**
utm_content	**string**	**No**	**“utm_content”**	**Content UTM tag**
utm_source	**string**	**No**	**“utm_source”**	**Source UTM tag**

Some events have their own unique parameters:

Parameter	**Type**	**Required**	**Example**	**Description**
popup_post				
post_uid	**string**	**Yes**	**"7bc142b5-7f38-4272-bc7e-7539f70b25b1"**	**Unique form submission identifier**
popup_click				
name	**string**	**Yes**	**"main_block"**	**Name of the element that was clicked**
url	**string**	**Yes**	**"example.com/page"**	**Value of the `href` attribute of the clicked element**
popup_error, popup_bounce				
error	**string**	**Yes**	**"object_not_found"**	**Error text**
error_category	**int**	**Yes**	**404**	**Error number**

Promocode events

For promocode_attach, promocode_detach, promocode_activate events:

Parameter	Type	Required	Example	Description
attached	**string**	**Yes**	**“2020-10-25T13:28:52Z”**	**Date and time the promocode was attached to the profile**
detached	**string**	**Yes**	**“2020-10-25T13:28:52Z”**	**Date and time the promocode was detached**
activated	**string**	**Yes**	**“2020-10-25T13:28:52Z”**	**Promocode activation date**
code	**string**	**Yes**	**“TEST-XOUH-KBJM-J5K9-24I1”**	**Promocode**
db_id	**int**	**Yes**	**123456**	**Profile database ID**
loyalty_id	**int**	**Yes**	**9**	**Loyalty program ID**
profile_id	**string**	**Yes**	**“000000000000000000000000”**	**Profile ID**

All dates are specified in RFC 3339 format.

Segment creation events during import

For segs_add, segs_remove events:

Parameter	Type	Required	Example	Description
campaign_id	**int**	**Yes**	**9**	**Mailing ID in the Altcraft system**
launch_id	**string**	**Yes**	**“”**	**Mailing launch ID**
message_id	**int**	**Yes**	**0**	**Message ID in the Altcraft system**
segment_id	**int**	**Yes**	**80**	**Segment ID, if available**
send_message_id	**string**	**Yes**	**“”**	**Sent message ID**
full_message_id	**string**	**Yes**	**“fQIQs4FgAAEBIcARADkxODlKR8-gA”**	**Full SendMessageID**

Events of merging multiple profiles or splitting one profile

For profile_merge, profile_split events:

Parameter	Type	Required	Example	Description

destination	**array**	**Yes**	<pre>[
			  {
			  “custom_data”: {
			  “age”: 25
			  },
			  “db_id”: 94,
			  “profile_id”: “608a8960e3224a6be26a71df”
			  }
			]</pre>
				If profile merging occurs, it contains the matching data of the profile that will result from the merge.
				
				If profile splitting occurs, it contains the matching data of the profiles that will result from the split.

Parameter	Type	Required	Example	Description

source	**array**	**Yes**	[
			{
			“custom_data”: {
			“age”: 25
			},
			“db_id”: 94,
			“profile_id”: “608a8960e3224a6be26a71df”
			}
			]
				When profiles are merged, it contains matching data about the profiles that must be merged.
				
				When a profile is split, it contains matching data about the profile that must be split.

Channel events

For events of the following channels:

- **Telegram:** `telegram_bot_send, telegram_bot_deliv, telegram_bot_click, telegram_bot_uclick, telegram_bot_undeliv, telegram_group_send, telegram_group_deliv, telegram_group_click, telegram_group_undeliv`
- **MAX:** `max_bot_send, max_bot_deliv, max_bot_click, max_bot_uclick, max_bot_undeliv, max_group_send, max_group_deliv, max_group_click, max_group_undeliv`
- **WhatsApp:** `whatsapp_deliv, whatsapp_undeliv, whatsapp_send, whatsapp_click, whatsapp_uclick, whatsapp_subscribe, whatsapp_read, whatsapp_unsub`
- **Viber:** `viber_deliv, viber_undeliv, viber_send, viber_click, viber_uclick, viber_subscribe, viber_read, viber_unsub, viber_complain`
- **Notify:** `notify_deliv, notify_undeliv, notify_send, notify_click, notify_uclick, notify_subscribe, notify_read, notify_unsub, notify_complain`

Parameter	Type	Example	Required	Description
campaign_id	**int**	**9**	**Yes**	**Mailing ID in the Altcraft system**
launch_id	**string**	**“”**	**Yes**	**Mailing launch ID**
message_id	**int**	**0**	**Yes**	**Message ID in the Altcraft system**
resource_id	**int**	**3**	**No**	**Resource ID in the Altcraft system**
segment_id	**int**	**80**	**Yes**	**Segment ID, if available**
send_message_id	**string**	**“”**	**Yes**	**Sent message ID**
full_message_id	**string**	**“fQIQs4FgAAEBIcARADkxODIKR8-gA”**	**Yes**	**Full SendMessageID**
city	**string**	**“Moscow”**	**No**	**Subscriber city**
browser	**string**	**“Firefox”**	**No**	**Browser and version**
country	**string**	**“.ru”**	**No**	**Country in TLD format**
device	**string**	**“web”**	**No**	**Device**
ip	**string**	**“10.9.0.14”**	**No**	**Subscriber IPv4 address**
lat	**float**	**55.244075**	**No**	**Geographic latitude**
link_url	**string**	**“http://www.example.com”**	**No**	**HTTP link the user was redirected to**
lon	**float**	**55.244075**	**No**	**Geographic longitude**
os	**string**	**“Linux x86_64”**	**No**	**Operating system**
postal_code	**string**	**“390000”**	**No**	**Subscriber postal code**
region	**string**	**“Ryazanskaya oblast”**	**No**	**Subscriber region**
tz	**string**	**“Europe/Vaduz”**	**No**	**Subscriber time zone**
undeliv_code	**int**	**500**	**No**	**Delivery failure error code**
undeliv_message	**string**	**“Content is required”**	**No**	**Error information**
workflow_id	**int**	**1**	**No**	**Scenario ID, if the mailing was launched from a scenario**

Mailing error events

For camp_error events:

Parameter	Type	Required	Example	Description
campaign_id	**int**	**Yes**	**9**	**Mailing ID in the Altcraft system**
error_code	**int**	**Yes**	**500**	**Error code**
error_text	**string**	**Yes**	**“LOG_INTERNAL_ERROR”**	**Error text**
launch_id	**string**	**Yes**	**“”**	**Launch ID**
message_id	**int**	**Yes**	**0**	**Message ID in the Altcraft system**
resource_id	**int**	**Yes**	**3**	**Resource ID in the Altcraft system**
segment_id	**int**	**Yes**	**80**	**Segment ID, if available**
send_message_id	**string**	**Yes**	**“”**	**Sent message ID**
full_message_id	**string**	**Yes**	**“fQIQs4FgAAEBIcARADkxODIKR8-gA”**	**Full SendMessageID**

campaign_launch structure

General campaign_launch structure:

Parameter	Type	Required	Example	Description
event_type	**string**	**Yes**	**“campaign_launch”**	**Event type**
event_date	**string**	**Yes**	**“2018-03-25T13:13:19.034315796Z”**	**Event date in RFC 3339 format**
account_id	**int**	**No**	**23**	**Platform account ID**
is_test	**bool**	**Yes**	**false**	**Test event sending flag**
is_static	**bool**	**No**	**true**	**Whether dynamic substitutions were used in the sent content**
is_historical	**bool**	**Yes**	**false**	**Whether the event is historical**

Parameter	Type	Required	Example	Description
event_data	**JSON object**	**Yes**	**{}**	**Object with mailing data**

event_data structure for campaign_launch

Parameter	Type	Required	Example	Description
campaign_id	**int**	**Yes**	**42**	**Mailing identifier**
workflow_id	**int**	**No**	**1**	**Scenario ID, if the campaign was launched from a scenario**
scampaign_id	**int**	**No**	**1**	**Campaign ID in the Altcraft system**
campaign_sub_id	**string**	**Yes**	**“x11”**	**Statistics identifier**
campaign_name	**string**	**Yes**	**“name”**	**Mailing name**
campaign_type	**string**	**Yes**	**“broadcast”**	**Mailing type**
data_type	**string**	**Yes**	**“segment”**	**Data source type**
data_id	**int**	**Yes**	**3**	**Data source identifier**
launch_id	**string**	**Yes**	**“00000017000003cd5a9bf01a”**	**Launch ID**
content	**JSON object**	**Yes**	**{}**	**Information about mailing message templates**

Campaign launch event example


```
{
  "event_type": "campaign_launch",
  "event_date": "2016-12-09T10:11:24.934079204Z",
  "is_test": false,
  "account_id": 161,
  "event_data": {
    "campaign_id": 1472,
    "campaign_type": "broadcast",
    "campaign_name": "Campaugn Launch Hook",
    "campaign_sub_id": "x11",
    "data_type": "segment",
    "data_id": 1221,
    "launch_id": "000000a1000005c0584a834c",
    "content": [
      {
        "message_name": "Hello",
        "message_id": 1412,
        "suppress_id": 271,
        "replyto_email": "",
        "replyto_name": "",
        "text": "текстовая версия письма",
        "html": "<html> html версия письма </html>",
        "from_name": "",
        "from_email": "",
        "attach_links": [],
        "subject": "Hello, {your_name}!"
      },
      {
        "message_name": "Hi",
        "message_id": 1413,
        "suppress_id": 272,
        "replyto_email": "monty@altcraft.com",
        "replyto_name": "Monty",
        "text": "текстовая версия письма",
        "html": "<html> html версия письма </html>",
        "from_name": "",
        "from_email": "",
        "attach_links": [
          {
            "name": "text_file.txt",
            "url": "http://test.altcraft.com/api/attach/link/text_file.txt?token=eyJh..."
          },
          {
            "name": "picture_file.png",
            "url": "http://test.altcraft.com/api/attach/link/picture_file.png?token=eyJh..."
          }
        ],
        "subject": "Hi, {your_name}!"
      }
    ]
  }
}
```

Subscription structure

This object contains information about the subscription to the resource through which the message was sent.

INFO: Required fields

The following fields are required for the subscription structure: `resource_id`, `channel`, `priority`, `status`, `ccsub`.

Parameter	**Type**	**Example**	**Description**
Common for all channels			
resource_id	int	12	Resource ID
channel	string	"email"	Communication channel
priority	int	0	Subscription priority
status	string	"subscribed"	Subscription status
cc_sub	JSON object	\{"custom_key": "custom_value"\}	Object with custom channel subscription fields. If there are no fields, it takes the value null
For the email channel			
email	string	"john@example.com"	Subscriber email
For the SMS channel			
phone	string	"+79991112233"	Subscriber phone number
For the push channel			
provider	string	"YandexAppMetricaDeviceIDAndroid"	Push provider
subscription_id	string	"123ab667960efec47b1afe64b56bfa386a3"	Provider subscription ID

content structure

If the “Add messages content” option is enabled in the webhook settings, the `content` field is added to the general hook message structure.

For Push and Viber, content is not transmitted. For Telegram, content fully depends on the template.

Email

```
{
    "content": {
        "amp": "<html amp4email></html>",
        "from_email": "no-reply@altcraft.com",
        "from_name": "Altcraft",
        "html": "<html></html>",
        "replyto_email": "example@example.com",
        "replyto_name": "Altcraft",
        "subject": "Weather alert",
        "text": "Text version",
        "to_email": "profile@example.com",
        "to_name": "Fname Lname",
        "webversion_email_html": "http://click.altkraft.com/0000/path.html",
        "webversion_email_text": "http://click.altkraft.com/0000/path.txt"
    },
}
```

Parameter	Type	Required	Example	Description
from_email	string	Yes	“no-reply@altcraft.com”	Sender email
from_name	string	Yes	“Altcraft”	Sender name
html	string	Yes		HTML version of the email
replyto_email	string	Yes	“example@example.com”	Reply-to email
replyto_name	string	Yes	“Altcraft”	Reply-to name
subject	string	Yes	“Weather alert”	Email subject
text	string	Yes	“Text version is important...”	Text version of the email
to_email	string	Yes	“profile@example.com”	Recipient profile email address
to_name	string	Yes	“Fname Lname”	Recipient profile name
amp	string	No		AMP version of the email
webversion_email_html	string	Yes	http://click.altkraft.com	Link to the web version of the email
webversion_email_text	string	Yes	http://click.altkraft.com	Link to the text web version of the email

SMS

```
{
    "content": {
        "fromnamesms": "abcdefghijklmnqrstuvwxyz",
        "sms": "Hello, Aaron! See the new items for you: http://click.altkraft.com/s/path",
        "webversion_sms_text": "http://click.altkraft.com/000000/path.txt"
    }
}
```

Parameter	Type	Required	Example	Description
fromnamesms	**string**	**Yes**	**“abcdefghijklmnqrstuvwxyz”**	**Caller ID**
sms	**string**	**Yes**	**“Follow the link and claim your welcome bonus!”**	**Message text**
webversion_sms_text	**string**	**Yes**	**http://click.altkraft.com**	**Link to the web version of the message**

WhatsApp

Parameter	Type	Required	Example	Description
type	**string**	**Yes**	**“template”**	**Message type**
category	**string**	**Yes**	**“marketing”**	**Message category**
name	**string**	**Yes**	**“welcome”**	**Template name**
language	**string**	**Yes**	**“en”**	**Template language**
namespace	**string**	**Yes**	**“xxx”**	**Template namespace**
wa_account_id	**string**	**Yes**	**“123456”**	**WhatsApp account ID**
components	**array**	**Yes**	**[...]**	**Message components**

MAX

Parameter	Type	Required	Example	Description
text	**string**	**Yes**	**“Hello!”**	**Message text**
buttons	**array**	**Yes**	**[...]**	**Message buttons**
disable_link_preview	**bool**	**Yes**	**true**	**Disable link preview**

Push

```
{
    "content": {
        "app_push_settings": {
            "background_color": "#00CED1",
            "badges": 0,
            "banner": "",
            "critical_push": false,
            "deliver_prior": 0,
            "expir_time": 604800,
            "led": "",
            "media": "",
            "silent_push": false,
            "sound": "",
            "subtitle": "",
            "vibro": false
        },
        "body": "See if our new items are fit to make you even more awesome!",
        "click_url": "https://click.altkraft.com/0000000",
        "custom_push_json": {},
        "icon": "",
        "result_content": {
            "content_available": true,
            "data": {
                "hub_link": {
                    "ack": "https://pxl.altkraft.com/crypto_post_events?/bFOoNcxW7lpDA",
                    "open": "https://pxl.altkraft.com/crypto_post_events?/LbFOoNcxW7lpDA"
                },
                "image": "",
                "launch_id": "2_3J_4Z2ZTRaQr7p",
                "mediaUrl": "",
                "message": "See if our new items are fit to make you even more awesome!",
                "uid": "fATFSGbgLjinkRE59ZXDOSd0BcAABARnhIxDBBAAAEbAQowRH"
            },
            "mutable_content": true,
            "notification": {
                "body": "See if our new items are fit to make you even more awesome!",
                "click_action": "https://click.altkraft.com/0000000/abcdefghijklmnqrstuvwxyz",
                "color": "",
                "icon": "",
                "sound": "default"
            },
            "title": "Just look, Alex! These are awesome!",
            "priority": "normal",
            "time_to_live": 604800,
            "to": "5dbDUxVXHV"
        },
        "title": "Just look, Alex! These are awesome!",
        "webversion_push_body": "http://click.altkraft.com/000000/path.txt"
    }
}
```


Parameter	Type	Description
body	**string**	**Push message body**
click_url	**string**	**Click-through link**
custom_push_json	**JSON object**	**Custom JSON of the message**
icon	**string**	**Path to the push message icon**
title	**string**	**Message title**
result_content	**JSON object**	**Stores the structure of data sent to the push service**
webversion_push_body	**string**	**Link to the web version of the push message**
app_push_settings	**JSON object**	**Advanced settings for mobile push notifications**
web_push_settings	**JSON object**	**Advanced settings for browser push notifications**

app_push_settings structure

Parameter	Type	Example	Description
background_color	**string**	**#00CED1**	**Push notification background color**
badges	**int**	**0**	**Event indicator**
banner	**string**	**http://example.com/path.img"**	**Path to the banner image**
critical_push	**bool**	**false**	**Enables critical mode in the push notification**
deliver_prior	**int**	**0,1**	**Delivery priority**

Parameter	Type	Example	Description
expir_time	**int**	**604800**	**Push notification TTL in seconds**
led	**string**	**#00CED2**	**LED indicator color**
media	**string**	**http://example.com/path**	**Path to a local file in the application**
silent_push	**bool**	**false**	**Enables or disables silent mode**
sound	**string**	**/sound/sound.mp3**	**Path to a local file in the application**
subtitle	**string**	**“Just for you!”**	**Message subtitle (for iOS)**
vibro	**bool**	**true**	**Vibration when the push notification is received**
soundless	**bool**	**true**	**Indicates whether the push message will be accompanied by sound (used by providers: android firebase, android huawei, android rustore)**
actions	**array**	`[ { “label”: “Open”, “action”: “https://example.com” } ]`	**Buttons in push notifications**

web_push_settings structure

Parameter	Type	Example	Description
actions	**JSON object**	`[]`	**The array contains push buttons: click-through link (`action`) + button text (`label`)**
button_label	**string**	**“Sure”**	**Action button text (the value is passed only for the Safari provider)**
large_image	**string**	**“https://.altkraft.com/284834790/1/664c6da207767c769a/image.png”**	**Path to the banner image**
expir_time	**string**	**604800**	**Push notification TTL in seconds**
large_image	**string**	**“https://click.example.com”**	**Path to the local banner image**

**Meta, which owns Instagram, Facebook, and WhatsApp, has been recognized as an extremist organization and banned in the Russian Federation.*

Viber is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.

JSON batch request (HTTP POST action hook)

Batch requests allow you to combine many HTTP requests into one by specifying each request as a separate object in the body. The server returns one HTTP response that contains the responses to each submitted request.

When we choose this request type, we send the received events in JSON format in the body of the POST request to an external API. When sent, a JSON array with events will be generated.

Request settings

The standard request is sent without authorization. However, if you want to send data to an external service that uses the OAuth 2.0 framework as its authorization method, change the authorization type to OAuth2. In this case, you will need to select the connection that links Altcraft Platform and the service.

You have to specify:

- **URL of the service that receives the data**
- **Maximum batch size: limit the number of events in JSON array. POST request will be sent as soon as there are enough events.**

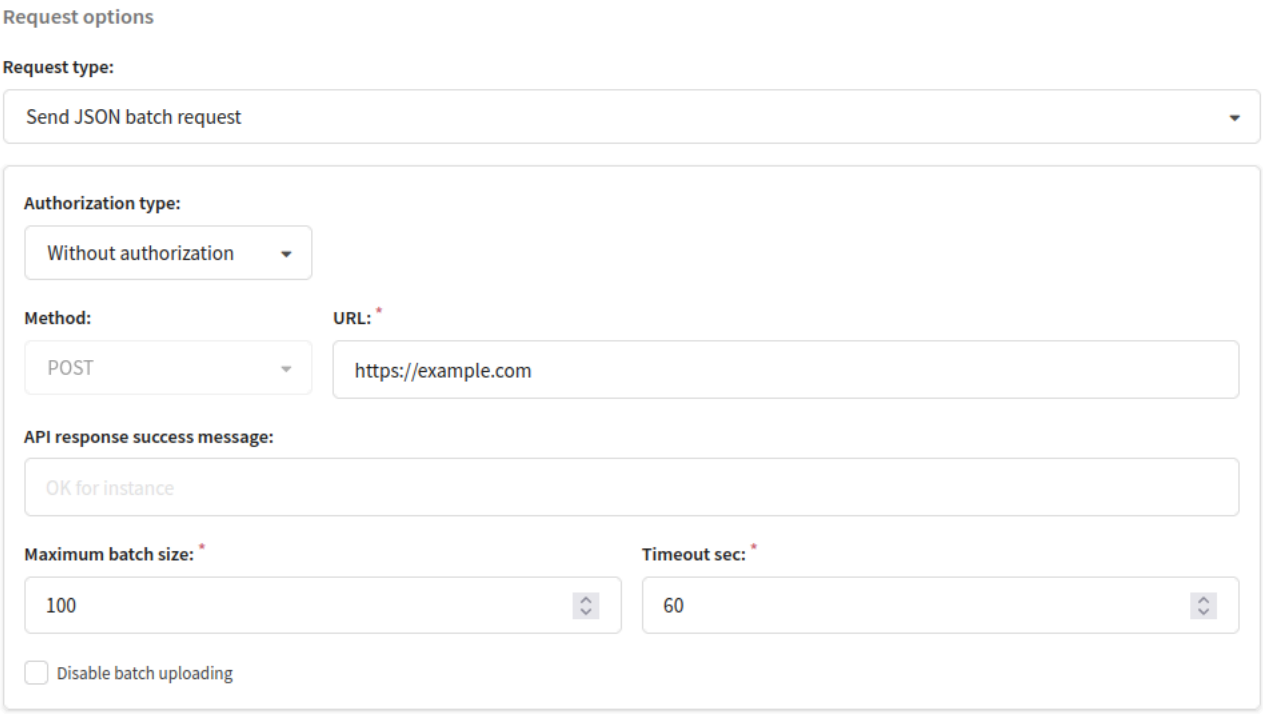
```
[{"event": "send", ... }, { }, ... ]
```

- **Timeout sec: the period of time during which events are accumulated in a batch to be sent to the server. If during this time the maximum batch size is not reached, the POST request is still sent. In this case, the JSON array contains the number of events that have accumulated by the time the timeout expires.**
- **API response success message: the response from the server must go into this Regex, only then the request to the server will be considered successful. For instance:**

`.*OK.*`

CAUTION

If you leave field API response success message empty, any response the status code of which is 200 is considered successful.

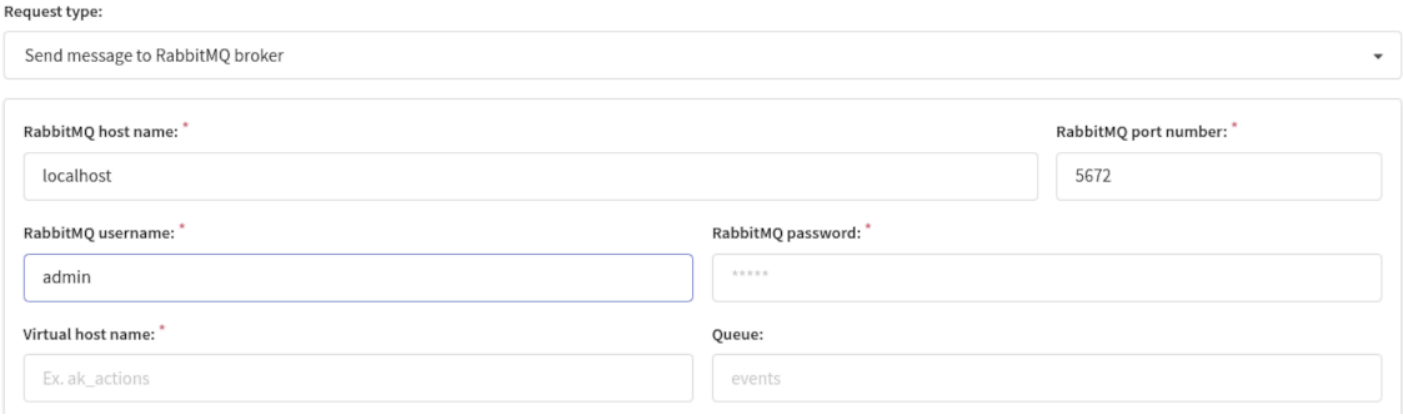


Email delivered message example


```
[
  {
    "event_type": "deliv",
    "list_id": 50,
    "account_id": 161,
    "_xxh": "5ada5e6ea3c6da5f",
    "_md5": "a5656dc1e3f603063a1918d582a10f58",
    "event_data": {
      "message_id": 5,
      "segment_id": 0,
      "campaign_id": 130
    },
    "event_date": "2016-11-15T09:12:28.514388956Z",
    "custom_data": {
      "email": "eddie@test-ukr.net",
      "_fname": "Eddie"
    },
    "is_test": false
  },
  {
    "event_type": "deliv",
    "list_id": 50,
    "account_id": 161,
    "_xxh": "bb42e541c7267fa6",
    "_md5": "593b250db256a764da8d2da4be694b31",
    "event_data": {
      "message_id": 5,
      "segment_id": 0,
      "campaign_id": 130
    },
    "event_date": "2016-11-15T09:12:28.569124547Z",
    "custom_data": {
      "email": "simon@test-gmail.com",
      "_fname": "Simon"
    },
    "is_test": false
  }
]
```

Message to RabbitMQ broker

Each single event will be sent immediately to the message broker queue. Both a local RMQ, which is part of the platform, and a broker located on an external host can be used.



In the settings, specify the broker's parameters:

- **host name**
- **port number**
- **username**
- **password for authorization**
- **virtual host name**
- **name of the queue into which event messages will be sent.**

CAUTION

RabbitMQ user must have access to the selected virtual host. Create a virtual host and give the RabbitMQ user the right to use it right in RabbitMQ.

You are free to choose whether to set up RabbitMQ queue name: if it is not set - a queue will be named after event type.

Email link click message example.


```
{
  "event_type": "click",
  "event_date": "2018-03-04T13:13:19.034315796Z",
  "list_id": 111,
  "_xxh": "ad1850e0d1d9481e",
  "_md5": "145bb30f7ddf1184cb560bc4b7705646",
  "is_test": false,
  "event_data": {
    "browser": "Firefox",
    "campaign_id": 973,
    "city": "",
    "country": "",
    "device": "web",
    "ip": "10.9.0.14",
    "lat": 0,
    "launch_id": "00000017000003cd5a9bf01a",
    "link_url": "http://www.ford.com",
    "lon": 0,
    "message_id": 17,
    "os": "Linux x86_64",
    "postal_code": "",
    "region": "",
    "segment_id": 0,
    "send_message_id": "5a9bf01a17_3cd_11_6f_0_2.5a8c045fc1dde44cbf718f78",
    "tz": "",
    "user_agent": "Mozilla/5.0 (X11; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0"
  },
  "custom_data": {
    "email": "alice@example.org"
  },
  "account_id": 23
}
```

Message to RabbitMQ exchanger

Exchange is a point of exchange, in other words, another player between the message sender and the queue. This exchange point distributes messages to one or more queues.

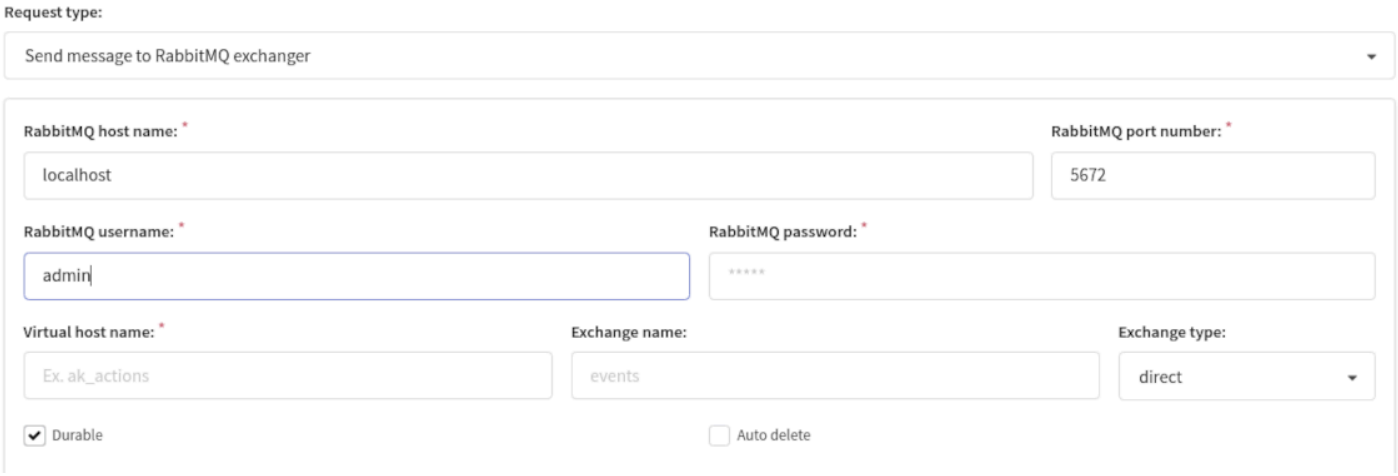
To configure sending a message to exchange RabbitMQ, specify host name, port name, username, passwordand virtual host name.

Next select an exchange type:

- **Direct type allows sending messages to only one queue subscribed to the corresponding routing key. The key is generated automatically using the pattern:** `<db_id>.<event_type>`**.**
- **Topic type allows sending messages to multiple queues that are subscribed to a routing key pattern.**

The durable option is enabled by default. This means that the exchange point will be permanently stored on disk and will be able to survive a server restart. If you disabled this field, the exchange will become temporary and will be deleted if the server is restarted.

Also pay attention to the Auto delete field. If you enable this field, then exchange will be deleted as soon as all associated queues are deleted.



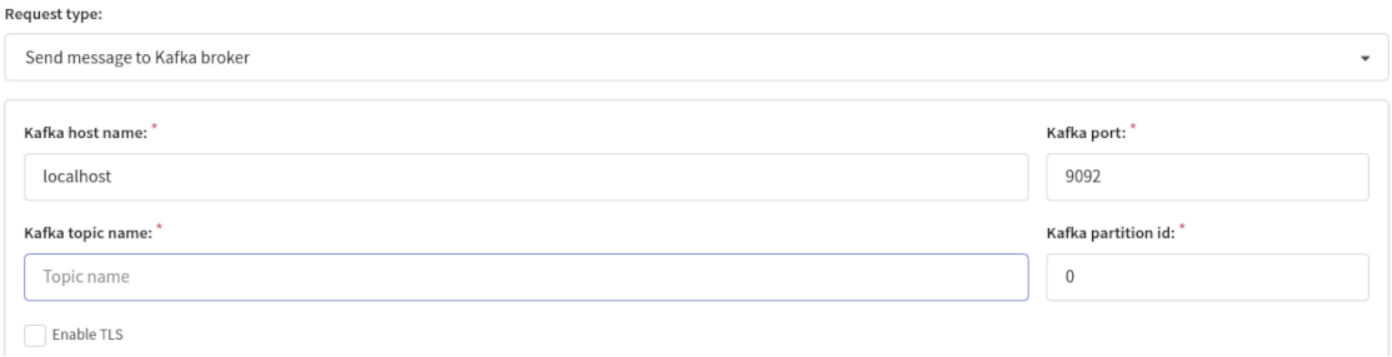
Message to Kafka broker

To configure sending events to Apache Kafka, specify the next parameters:

- **kafka host name,**
- **kafka port,**
- **kafka topic name,**
- **kafka partition id.**

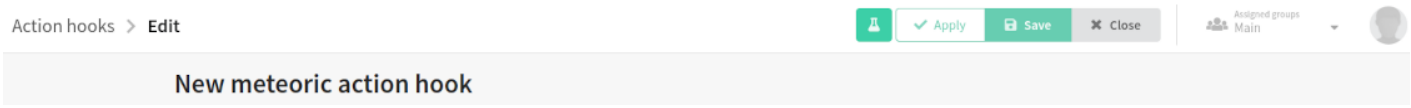
Activate TLS if you need. When using a self-signed certificate, if it is not included in the trusted ones, specify the CA certificate chain in PEM format. This also takes into account the global option `ALLOW_INSECURE_REQUESTS` in the main.json configuration file, if it is set.

The validity of certificates is not checked. If authorization with a client certificate is used, then you must specify both the client's private key and the certificate, also in PEM format.

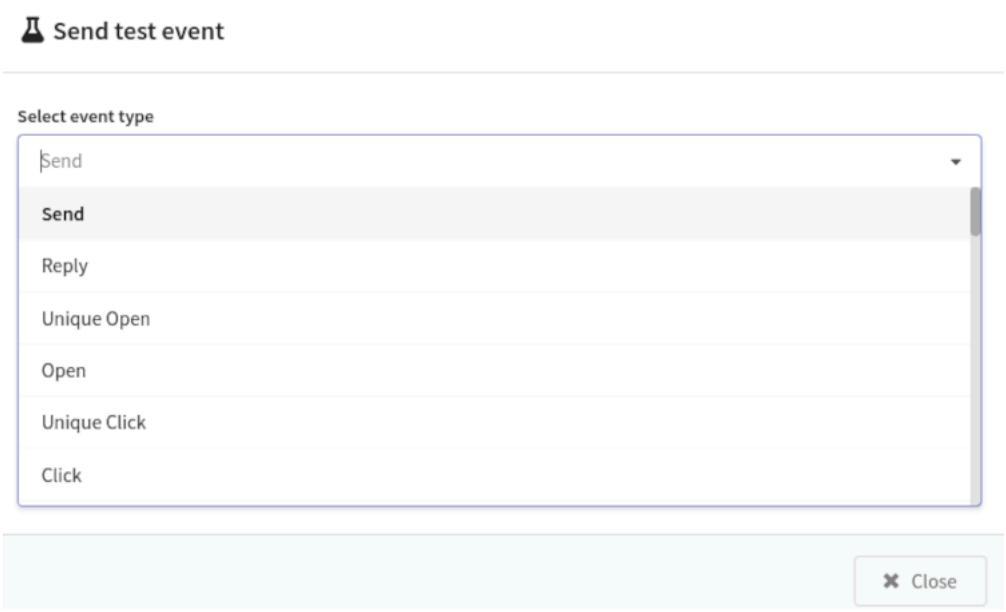


Test event

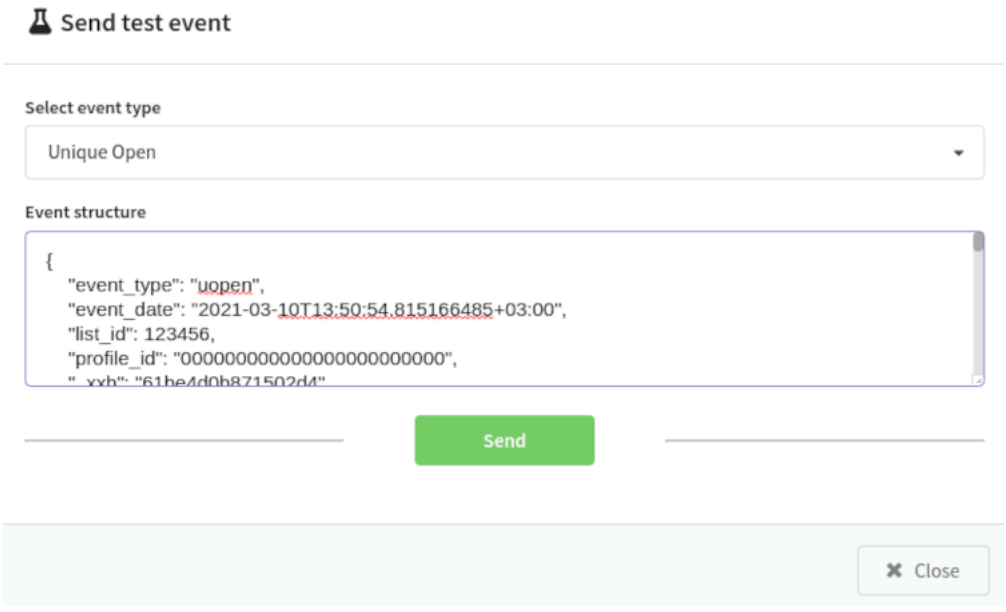
When you are working with action hooks you can test any event using Altcraft Platform interface. Click *flask icon* in the upper panel.



Select event type in new window:



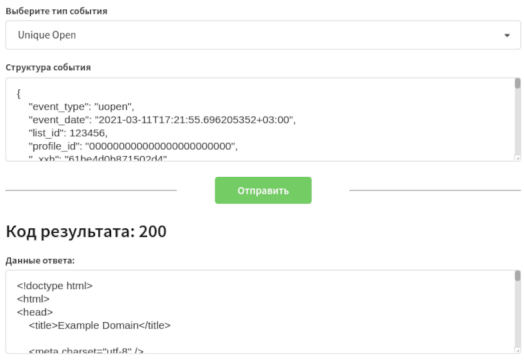
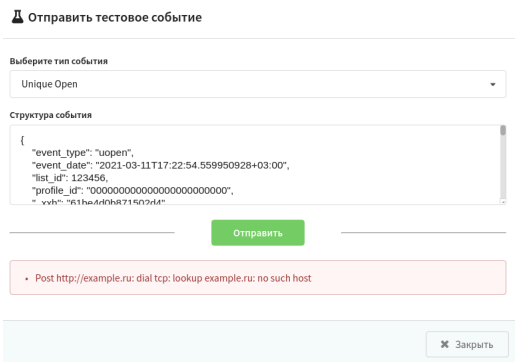
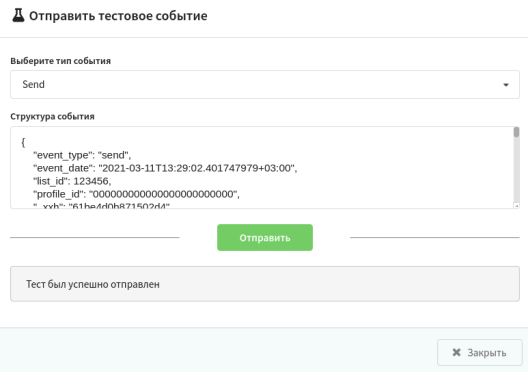
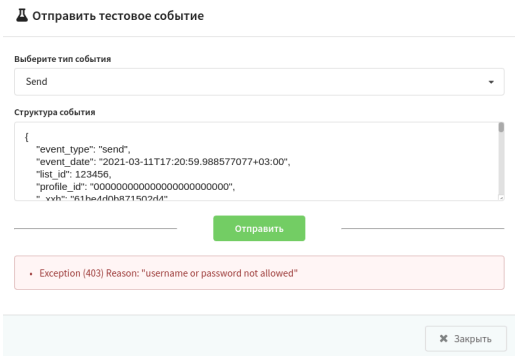
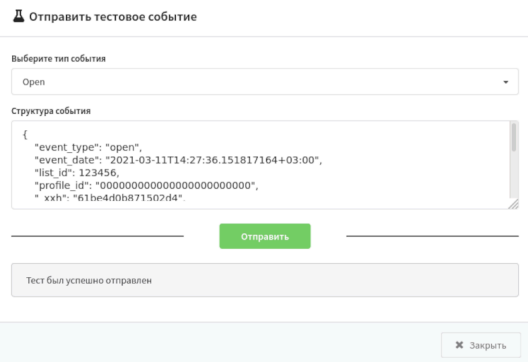
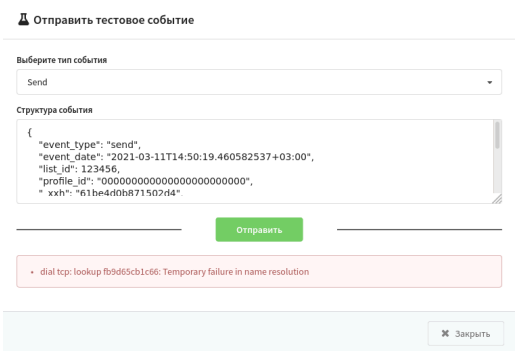
En example of the structure of selected event will be generated automatically. This structure can be corrected.



Click Send to test the event.

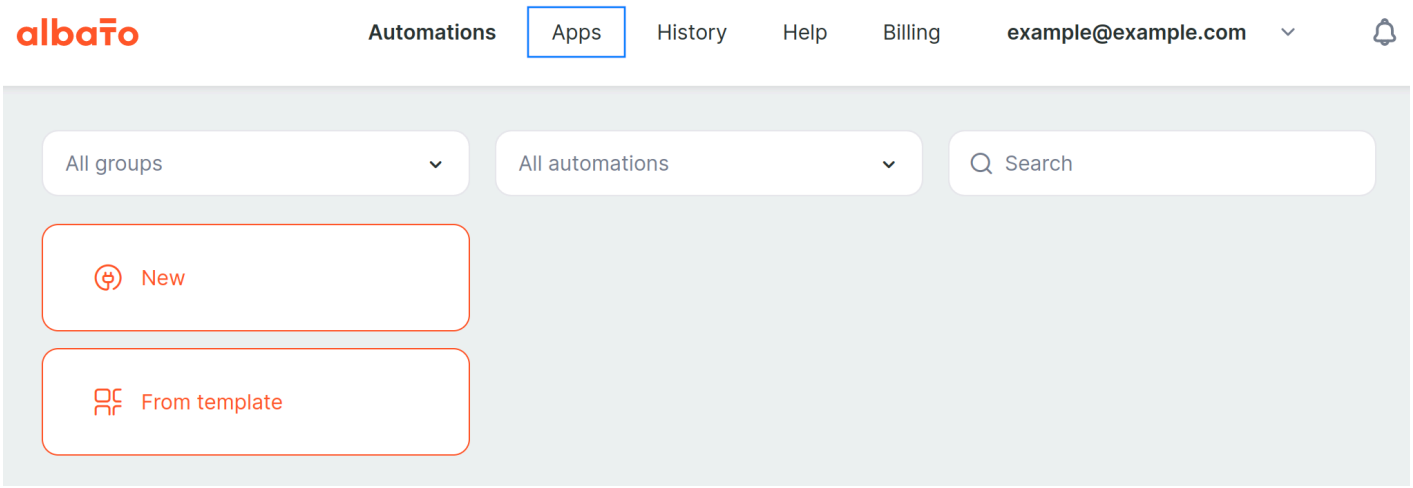
- **If the message was successfully sent via an HTTP request, you will get Result code: 200 and Response data.**
- **If the message was successfully sent to the queue or exchanger RabbitMQ or to Kafka broker, then you will see the text: "The test was sent successfully."**

In other case there will be red field, where you can know the error code and its reason.

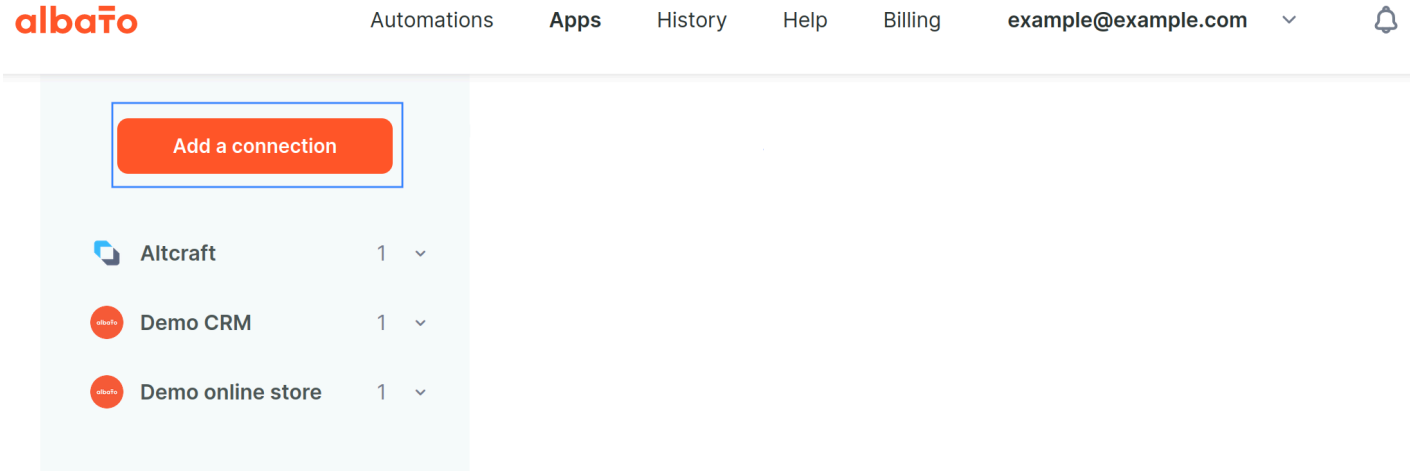
Request type	The message is successfully delivered	Error sending message
JSON batch request		
Message to RabbitMQ broker / exchanger		
Message to Kafka broker		

Connecting Altcraft to Albato

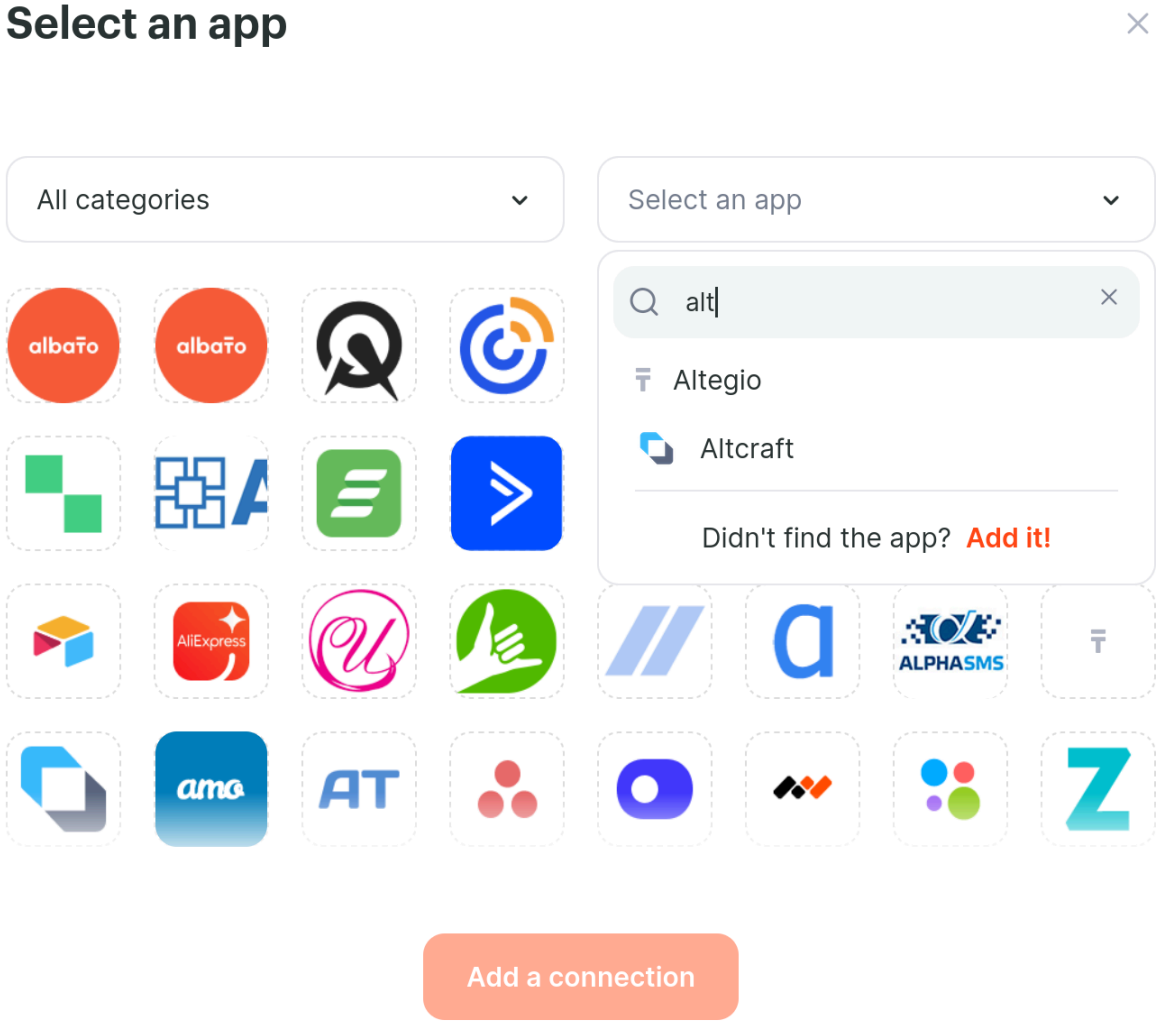
Before you can set up integration with the desired service, you need to connect Altcraft to Albato. Start by logging into your Albato account. Then select “Apps” from the top menu.



In the “Apps” section in the menu on the left, click “Add a connection”.



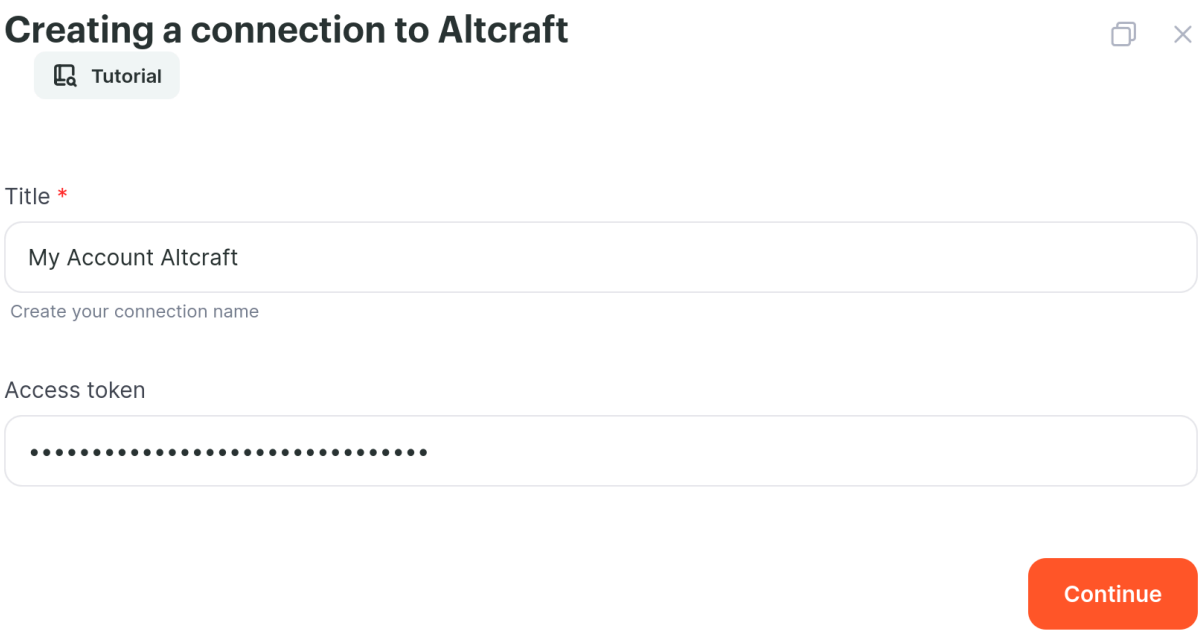
Now you need to select a service. In the “Select an app” field, specify the name of the platform — Altcraft. After you click the app, you must confirm the selected service by clicking on the “Add a connection” button.



After adding the connection, in the next window you can specify the connection title (or leave the default name) and Access token (required field).

TIP

Obtaining an Access Token in Altcraft Platform is described in the documentation.



After adding the Access token, click “Continue”. The following window will open, displaying information about the newly created connection.

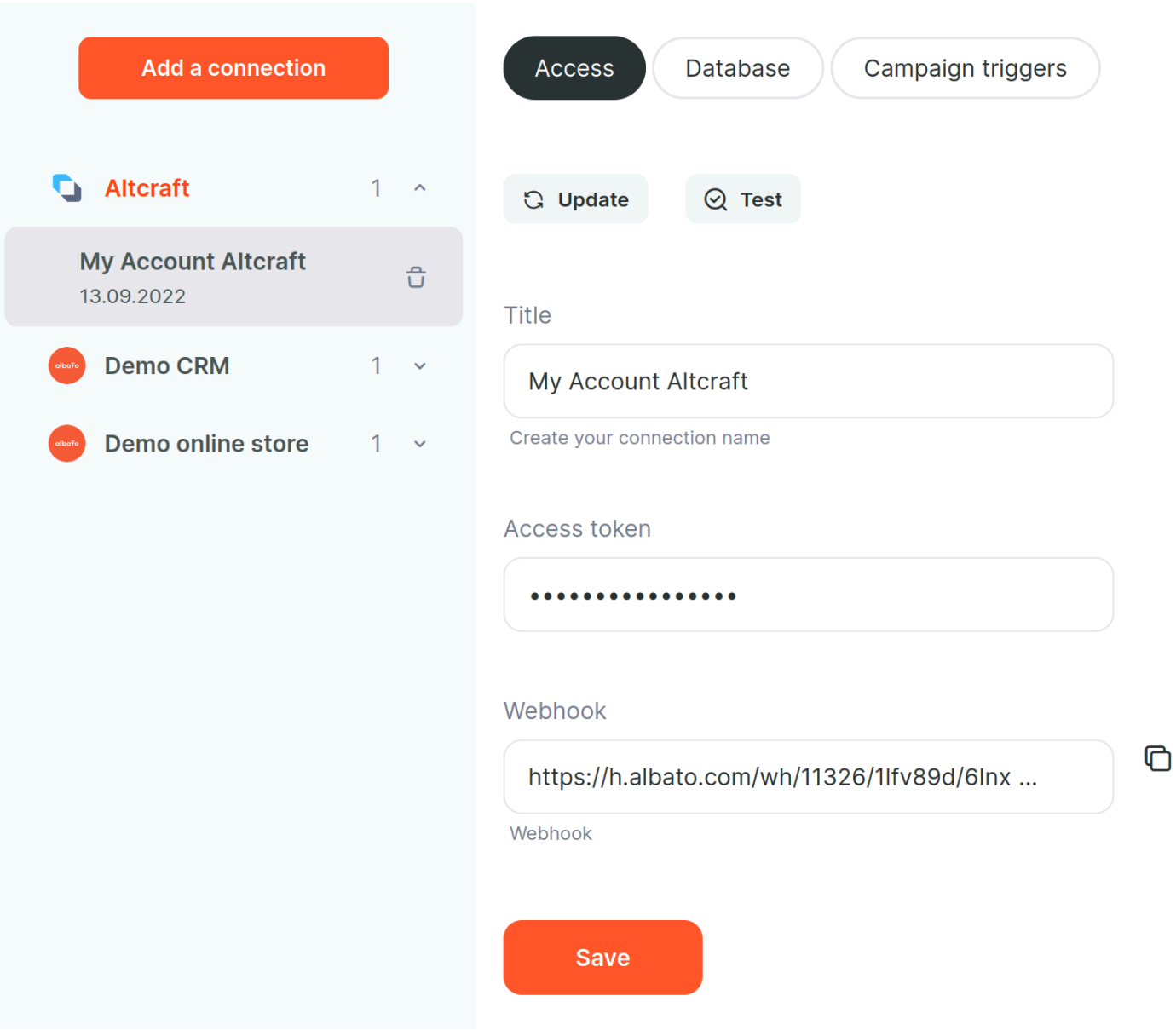
If the connection is successful, in the “Apps” section, in the left menu, Altcraft will be displayed in the list of all connections.

CAUTION

Albato can connect multiple Altcraft users with limited access. To do this, you need to generate a new Access Token, assign the necessary rights and create a new connection as described above.

It is also necessary to specify the names for each connection, so that later it will be convenient to view and test the connections.

To test the connection, you must select a connection from the list (if there are several). Next, you will see a window with connection information.



In this window, you can view the list of available databases (there may be restrictions depending on the specified Access Token role), as well as the list of available trigger mailings. You can also test the connection.

Launching the welcome scenario using Albato

Albato allows you to configure data transfer from an external service to the Altcraft platform. Using a special API request (for more information see the developer guide section), which comes from Albato, the platform decides what to do with the client's data.

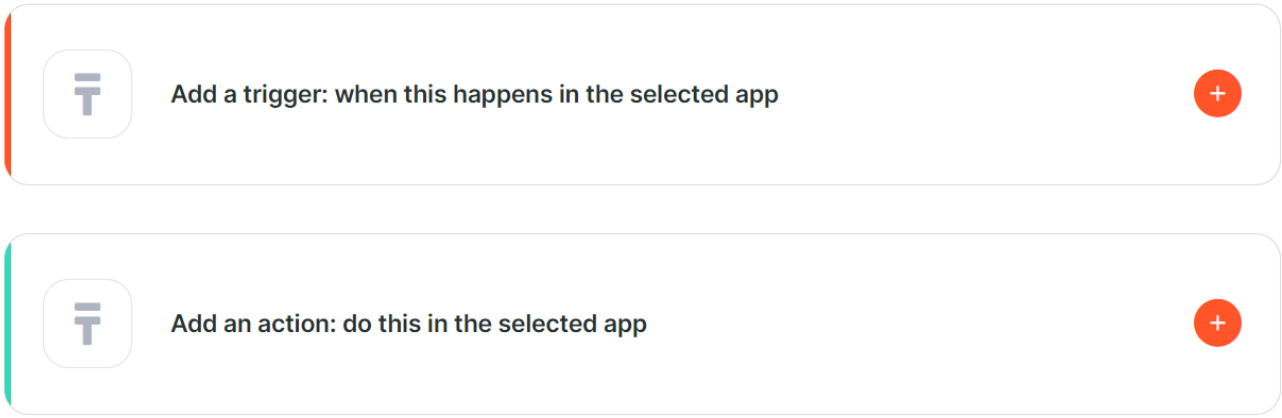
The platform can:

- **update the data in the client's profile by importing**
- **add a profile to a segment**
- **send the client to the scenario**

In this article, we will know how to use Albato to add a client profile to a private scenario.

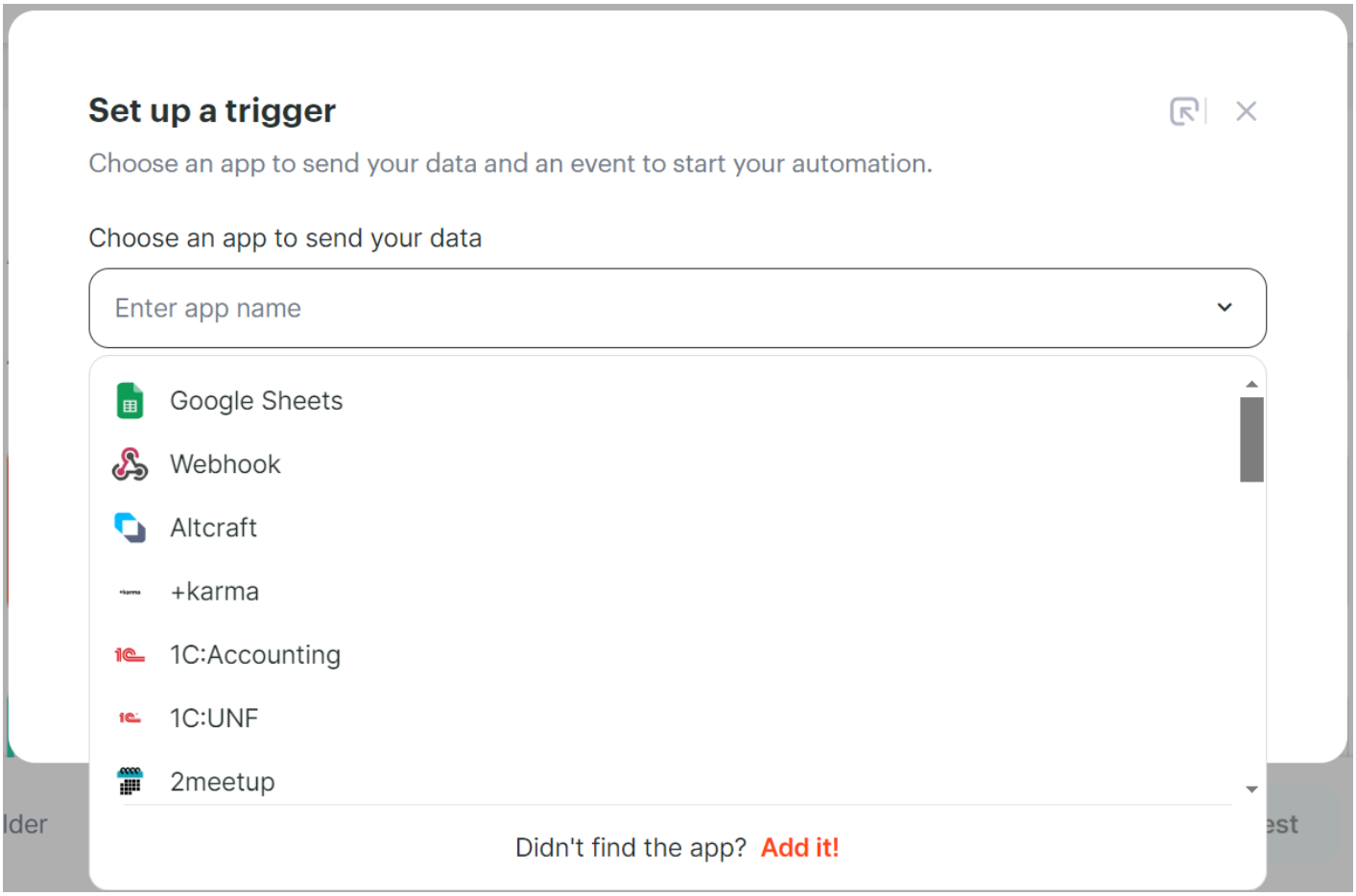
Create a new automation in Albato

A automations in Albato consists of two components: a trigger that start the transfer of data from an external service to Albato, and an action that occurs after the trigger is started.

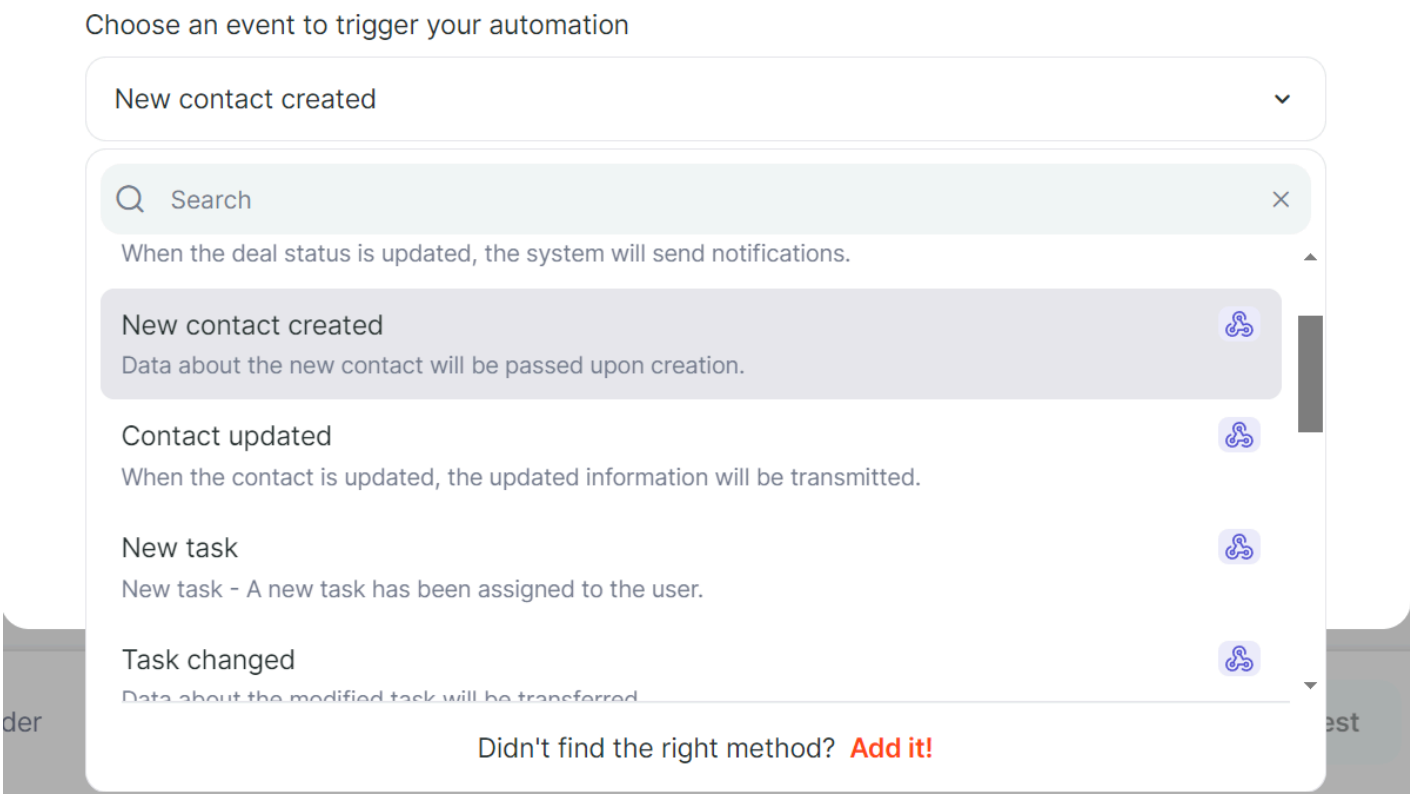


Add a trigger

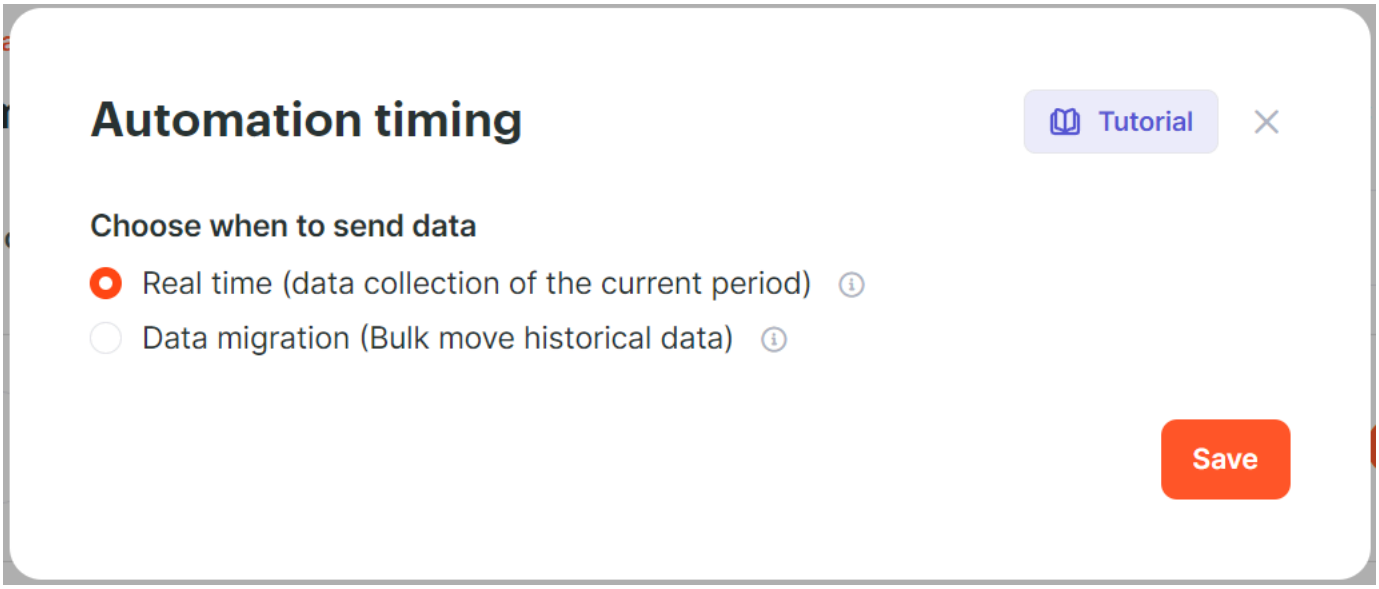
Select the service from which Albato will receive data:



Select the event that will trigger the data transfer. In our case, the action will be “New contact created”.

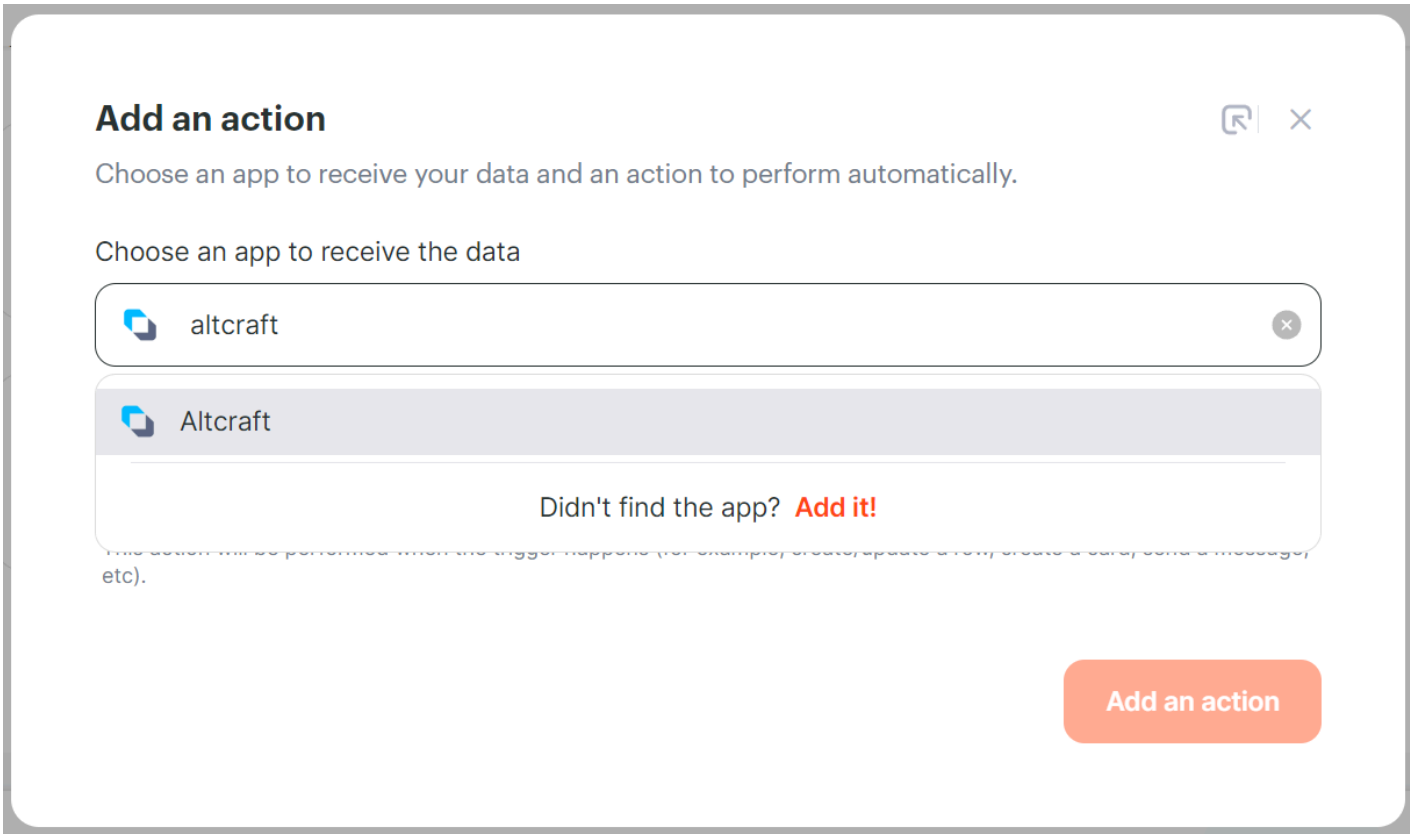


Select the mode of operation of the automations: data can be sent to Altcraft in real time or specify for which period to collect it.

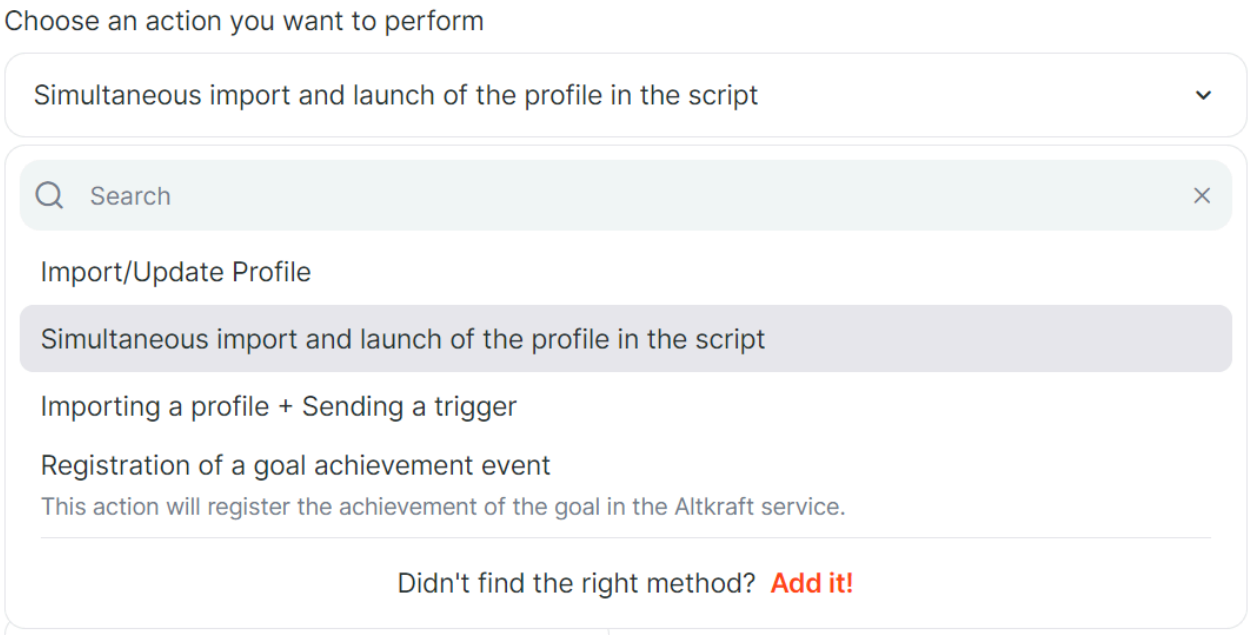


Add an action

Select the Altcraft platform as the server that will receive data from service.



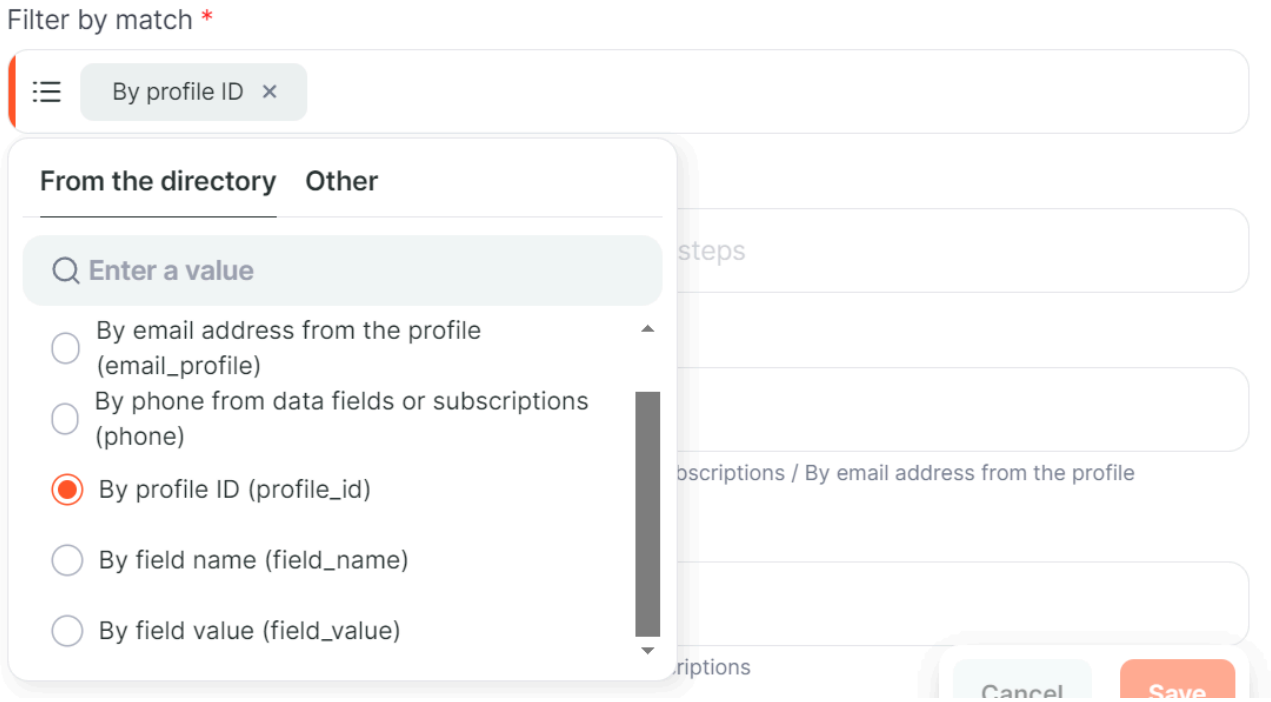
To import or update client data and transfer the profile to the scenario, select Simultaneous import and launch of the profile in the script.



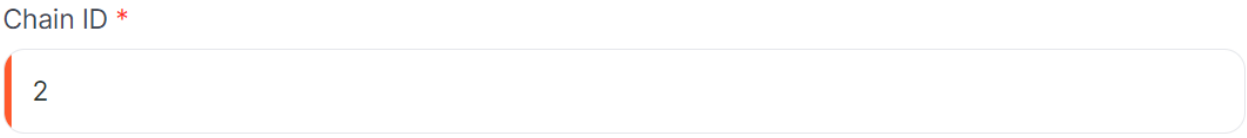
Connect an Altcraft account and specify the database for interaction. Click Add an action.

Configure the fields

Specify the filter by which the profile will be searched in the database.



Specify the ID of the scenario to which the profile will be sent.



Depending on the selected filter for matching, specify the field by which the search will be performed.

Optionally, specify additional fields to match. After setting up, you can test or run the automations.

Working with the scenario on the platform

After creating a automations, customer data will be transferred via Albato to the platform when a trigger event occurs. The client's profile will be included in the previously prepared welcome scenario.

You can read more about how events are registered on the platform to return data to Albato in the next article.

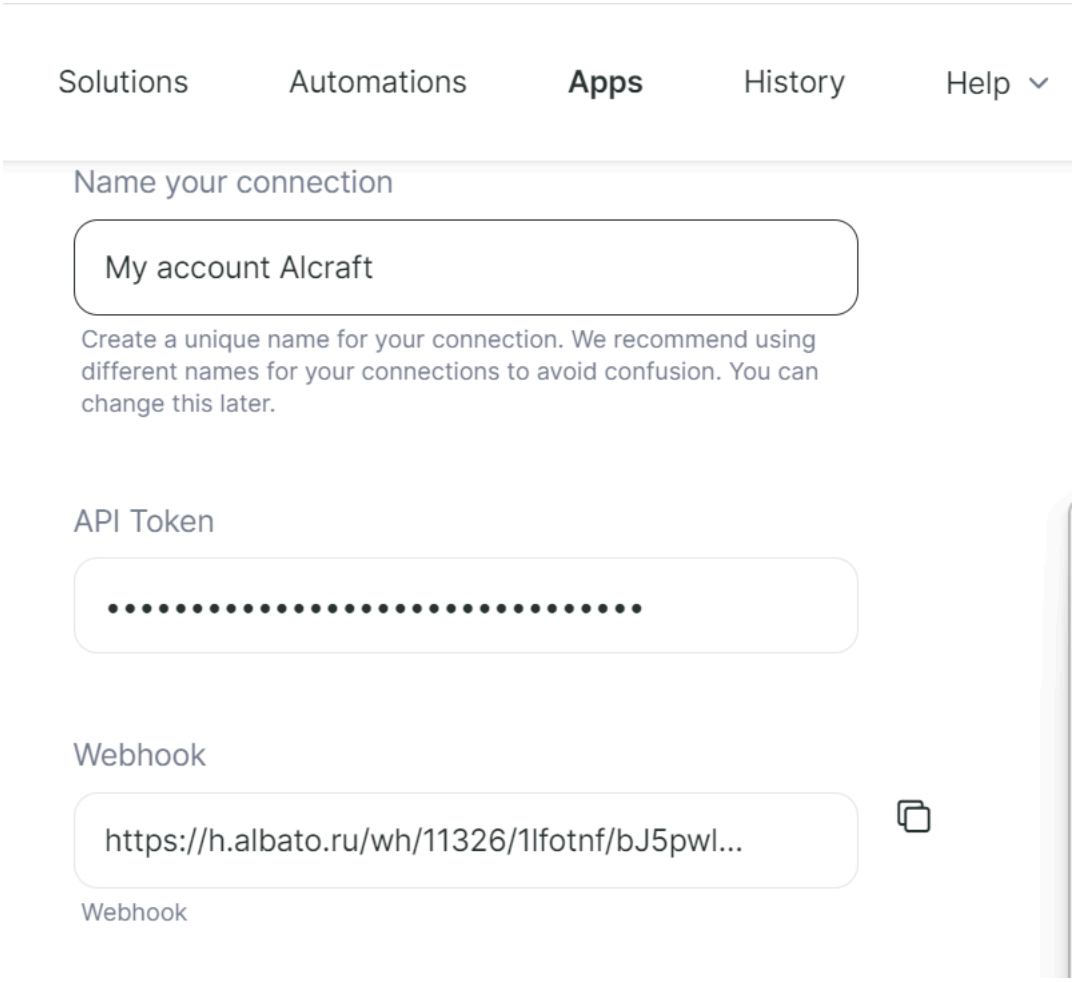
Transmitting event data

An event is an action related to the delivery of messages or the import of subscriber profiles, as well as customer actions tracked by Altcraft platform pixels on your website or on the app.

Event data can be transferred to various services using action hooks, including Albato.

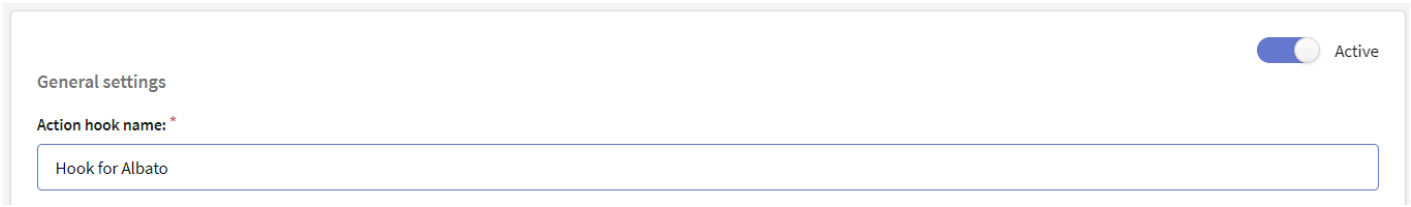
Setting up action hook on the platform

The main steps for configuring action hook are described in this article. On the URL field, specify the address, the way is Albato —> Apps —> Altcraft —> My Altcraft account



List of available events:

- **Mailing launched**
- **An SMS message has been sent for delivery to the subscriber.**
- **Not a unique click on a link in an Email message**
- **The SMS message has been delivered**
- **Click on the subscription confirmation link in the double opt-in message**
- **SMS message not delivered**
- **The subscriber followed the unsubscribe link in the email and was unsubscribed**
- **Push message sent for delivery to subscriber**
- **Not a unique opening of a Push message**
- **Push message delivered**
- **Not a unique click on a link in an SMS message.**
- **Not a unique click on a link in a Push message**
- **The promo code is activated**
- **Not a unique opening of an Email message**
- **Email message sent for delivery to subscriber**
- **A promo code is assigned to the subscriber**
- **The sender delivered the Email message to the subscriber**
- **AK-Pixel is open on the website**
- **The sender could not deliver the Email message to the subscriber**
- **You can activate event capture using the switch at the top of the screen:**

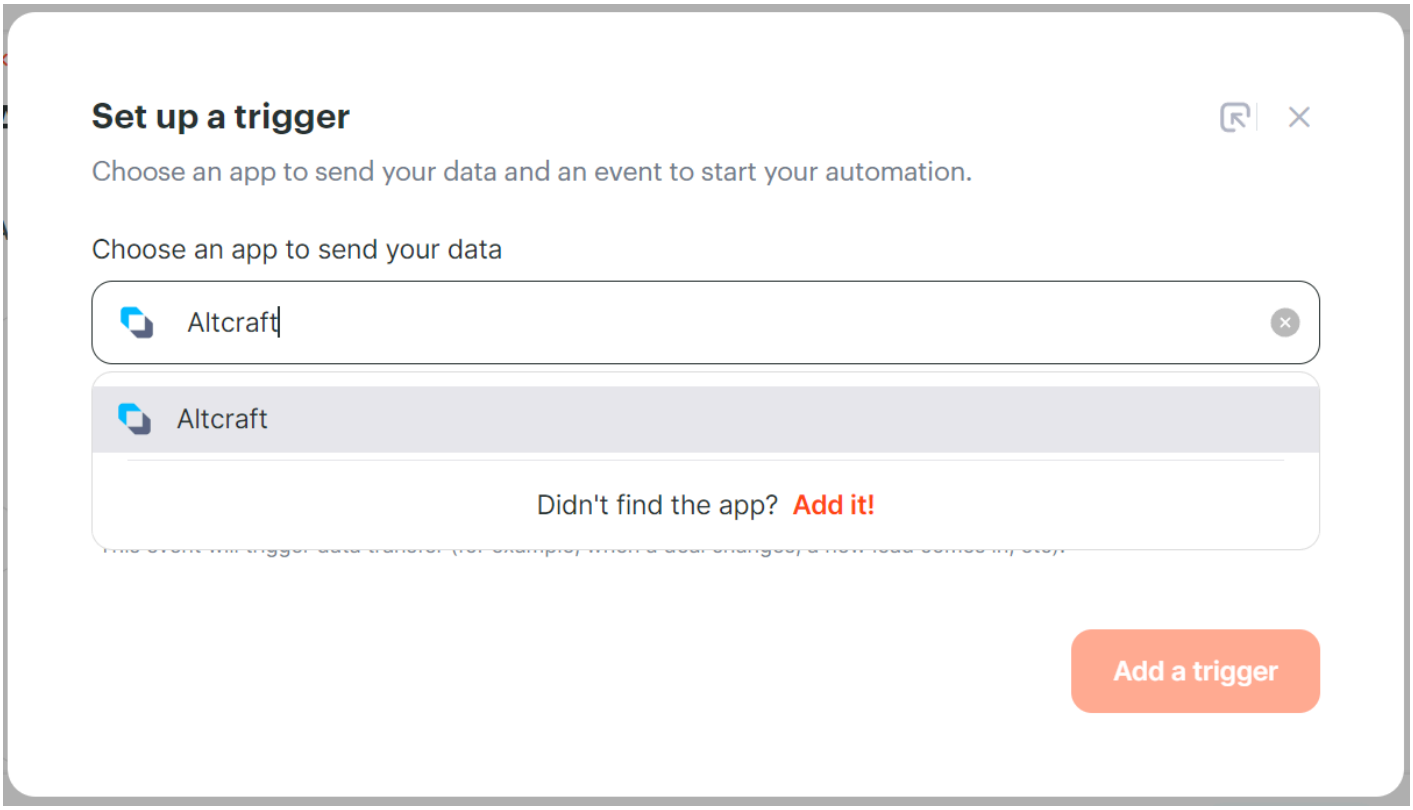


Setting up a automation in Albato

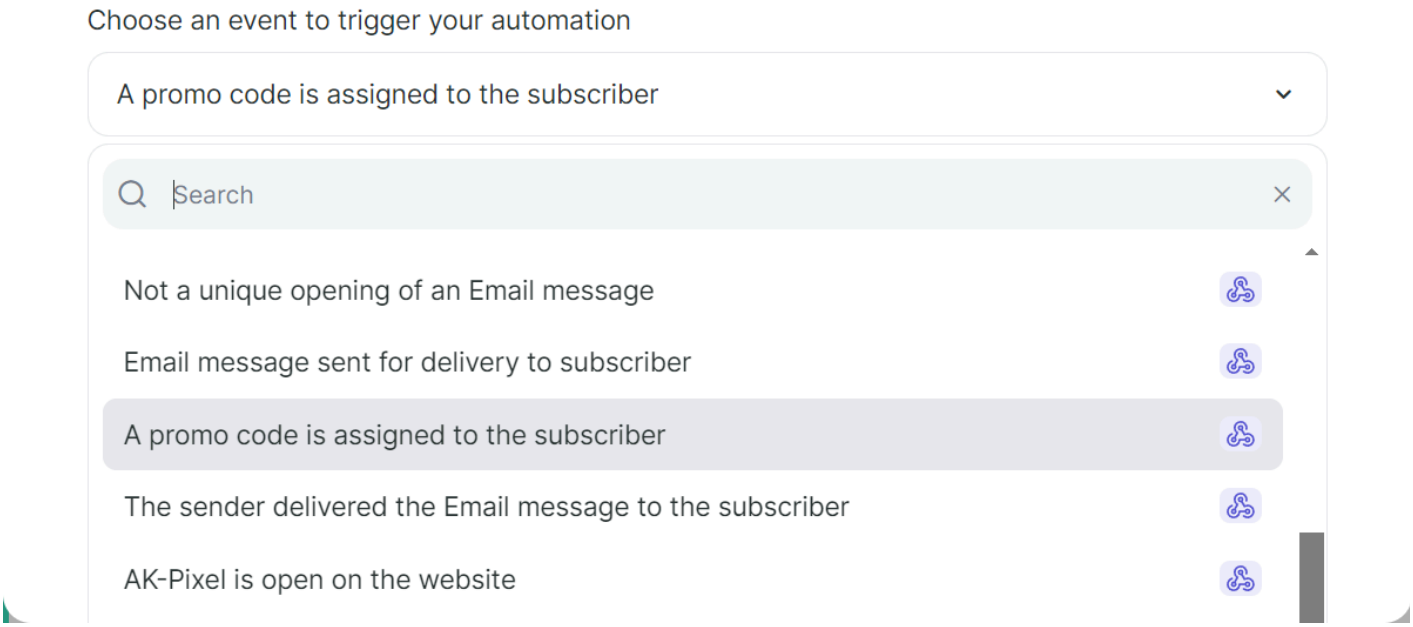
To set up a automations to transfer data from Altcraft to an external service, click Create new automations

Set up a trigger

Select Altcraft as the service that will transfer data to Albato



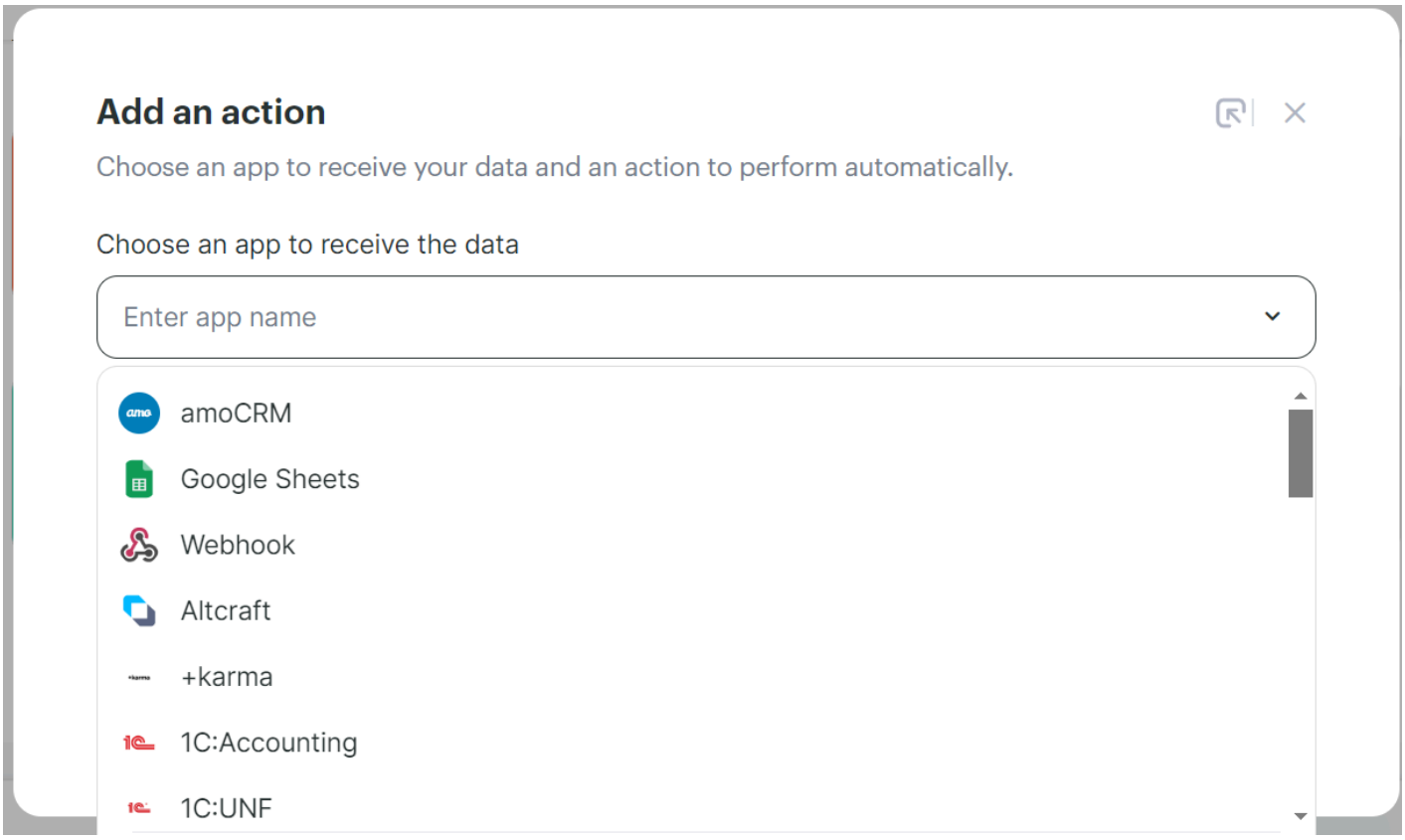
Select the event that will trigger the transfer of data to Albato:



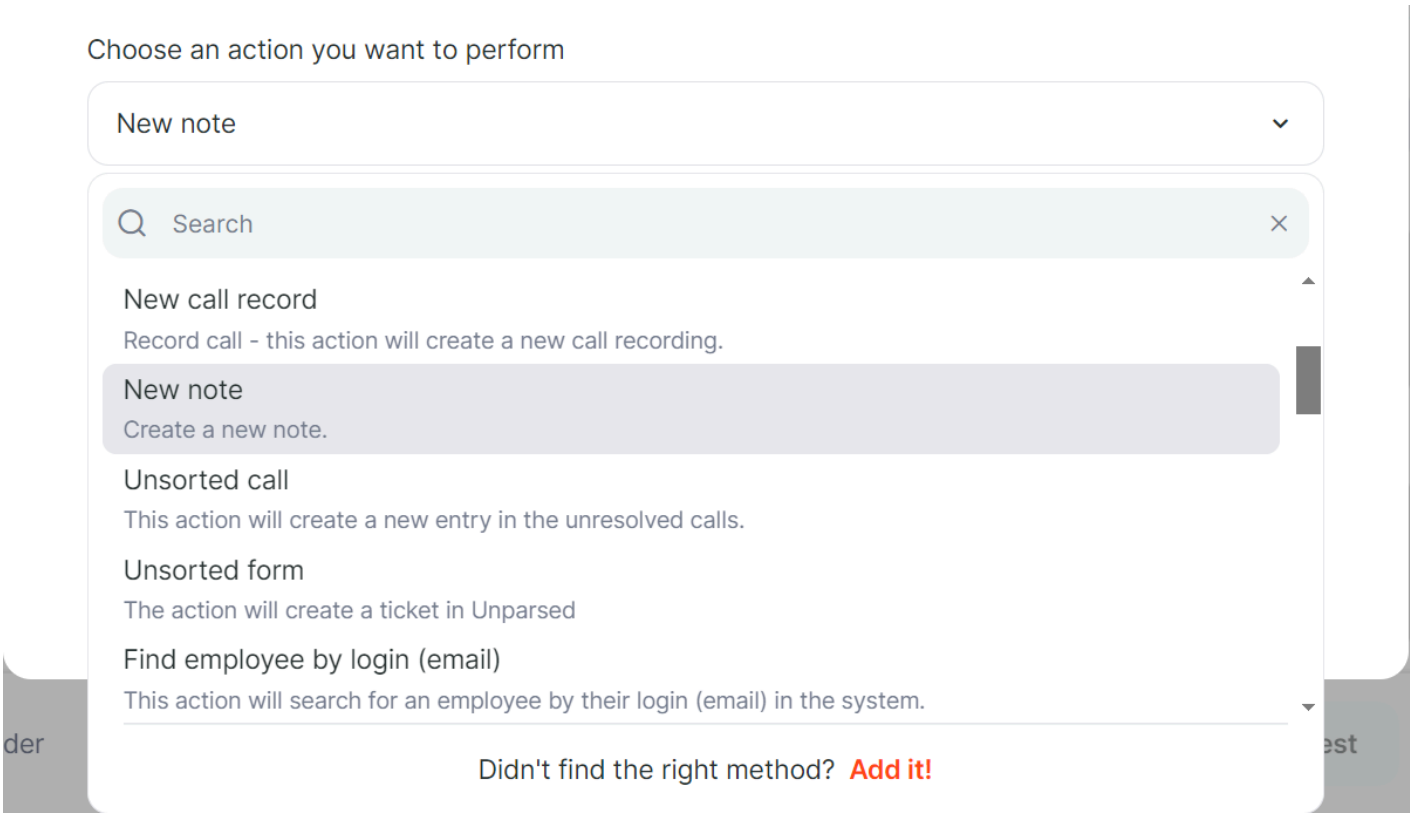
Connect your account.

Add the action that will occur after the trigger is started

Select the service where the data will be sent:



Assign an event to be performed:



Connect your account to Albato.

Click save and then configure the fields: add a “Contact”, the text “add a promo”. You can activate the automations after adding necessary information.

Setting app a trigger mailing

Using Albato, you can create a automation that will launch a trigger mailing on the platform.

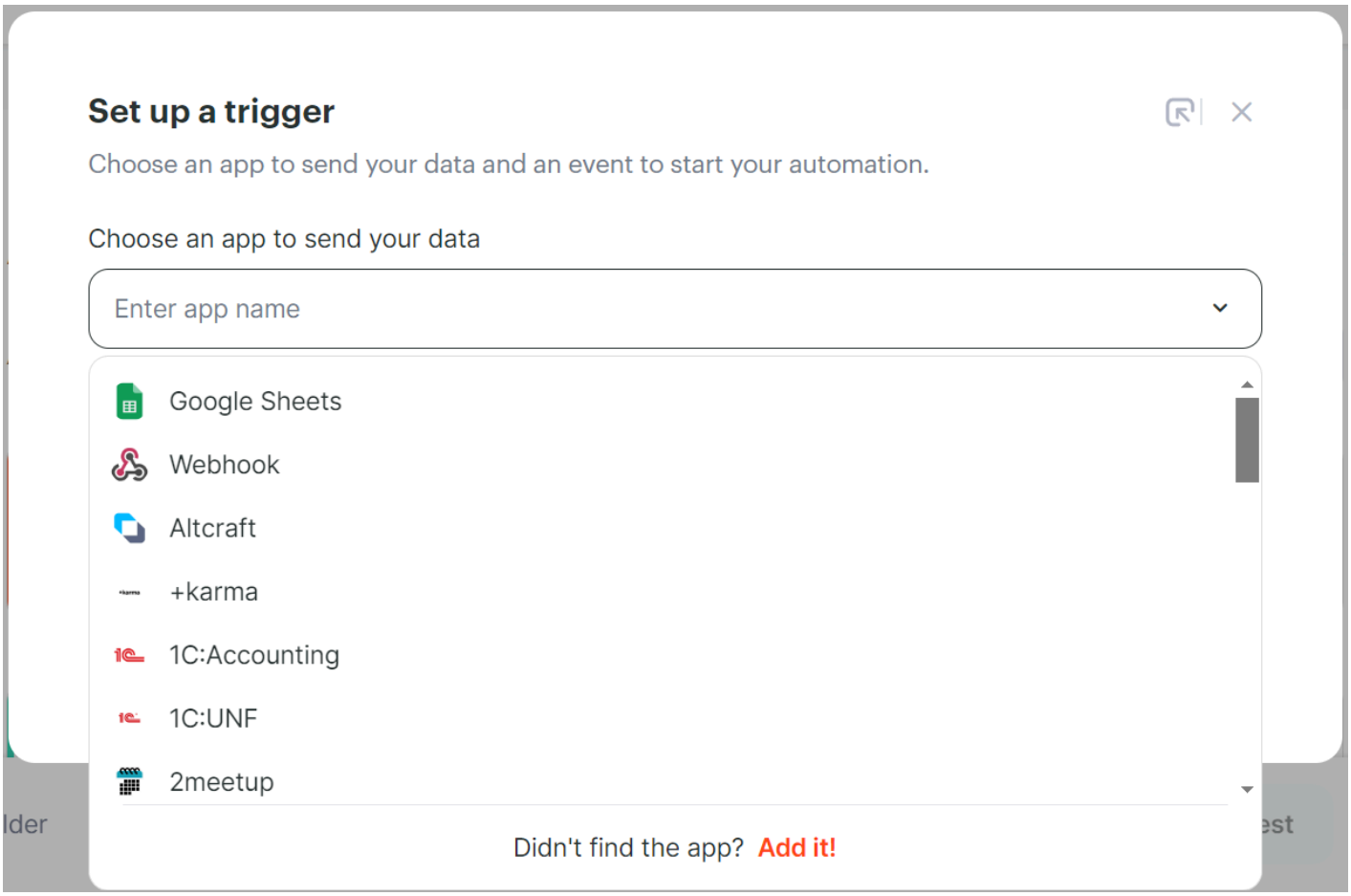
Let’s launche this type of mailing using the example of sending an invoice to a client.

Setting up automation in Albato

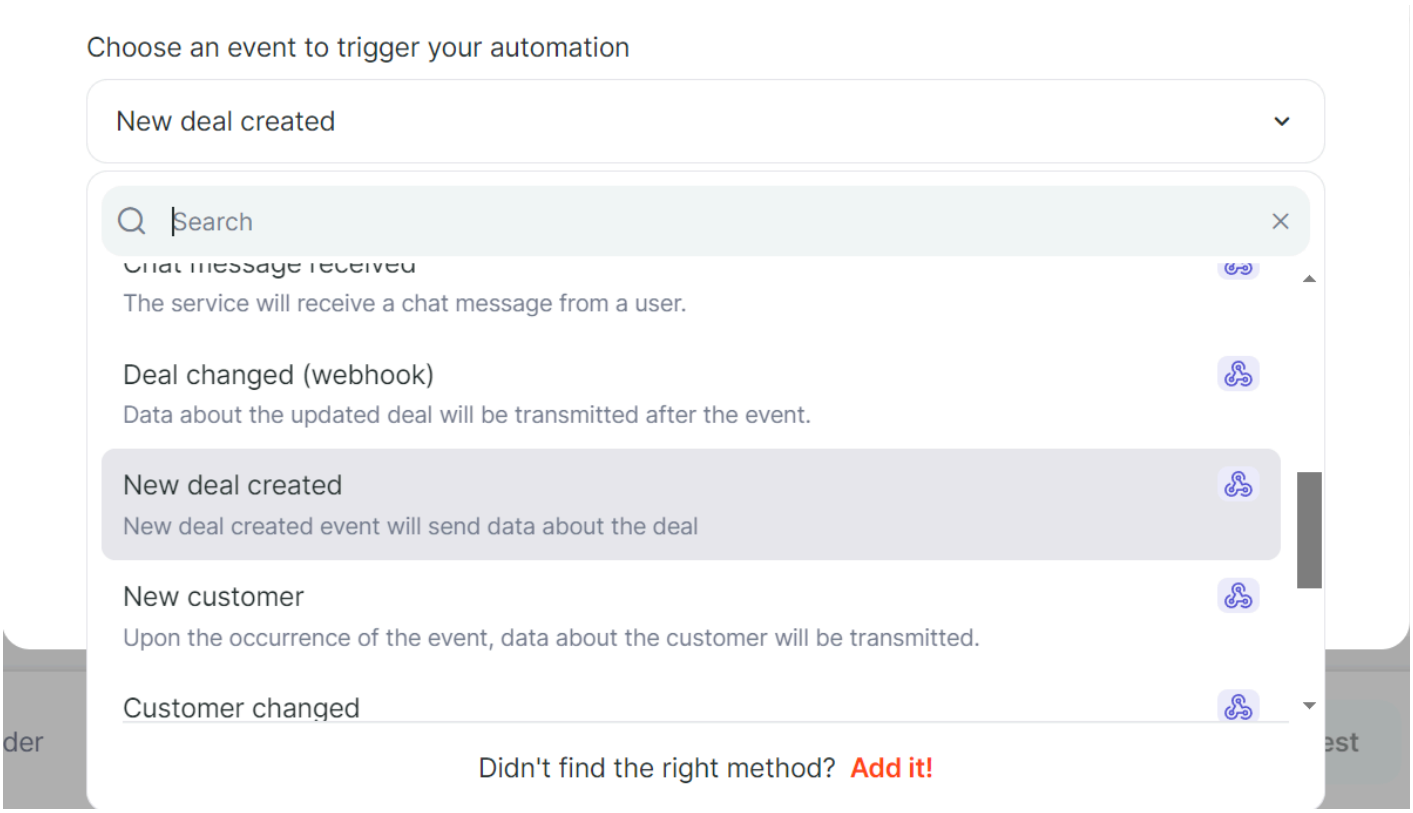
Set up a automation using this way External Service -> Albato -> Altcraft.

Add a trigger

Select the external service from which the data will be received:

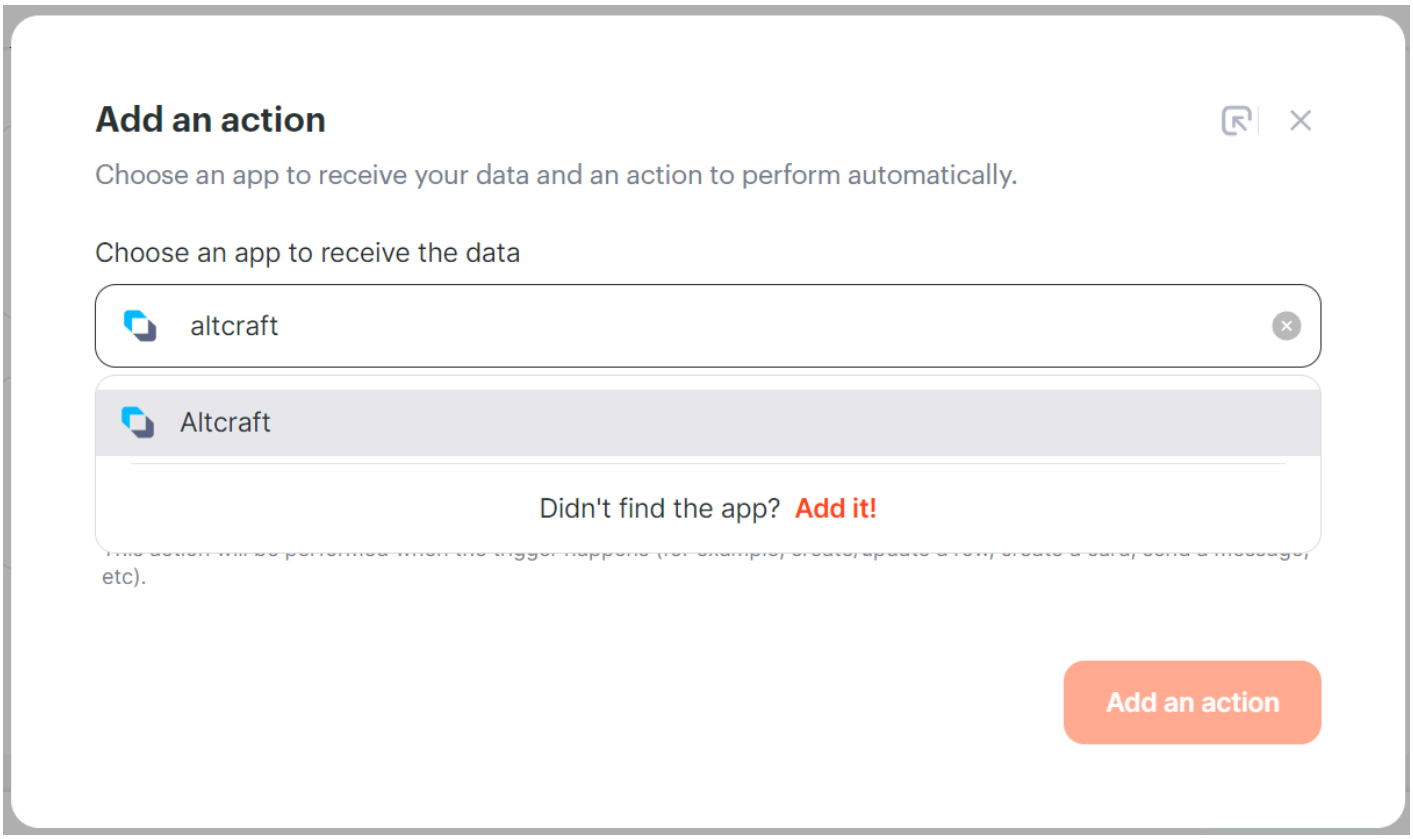


Select the event that will trigger the data transfer:

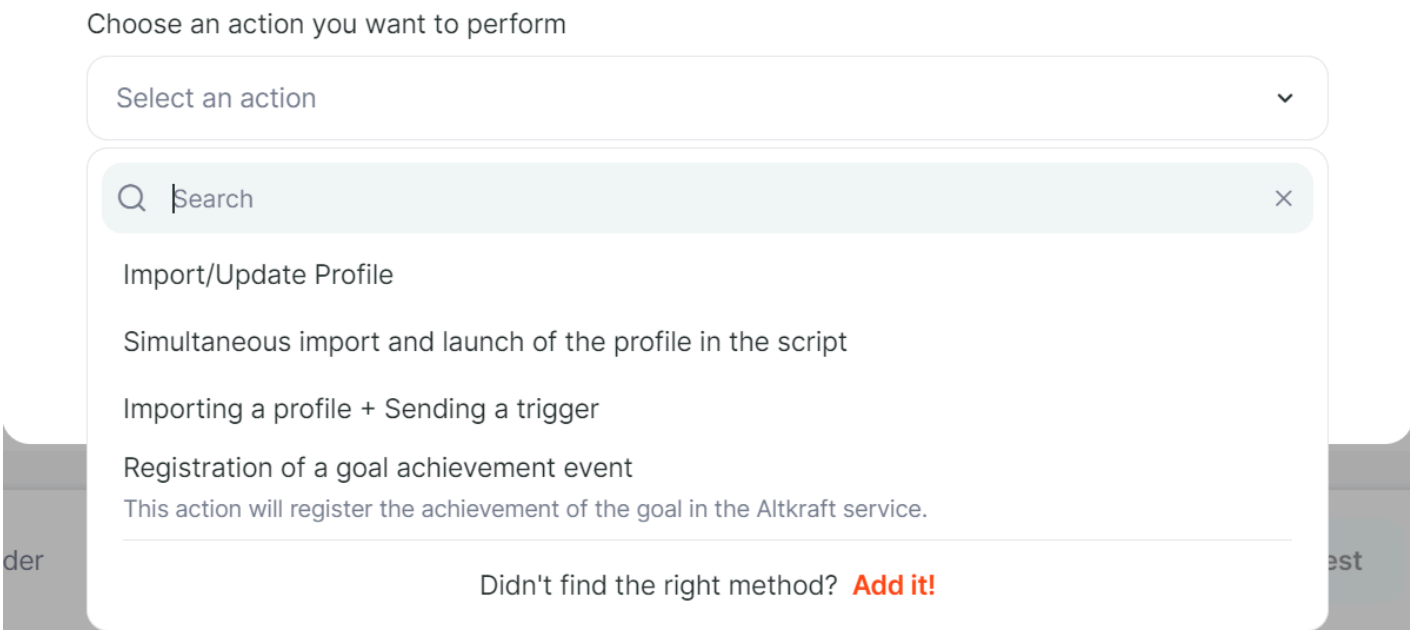


Add an action

Specify Altcraft as a service for receiving data:

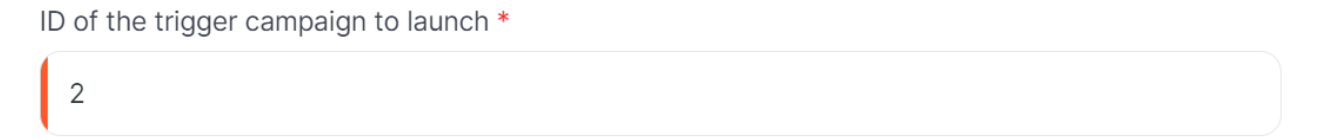


Assign an action that will take place after the event occurs. In our case, we need Importing a profile + Sending a trigger.

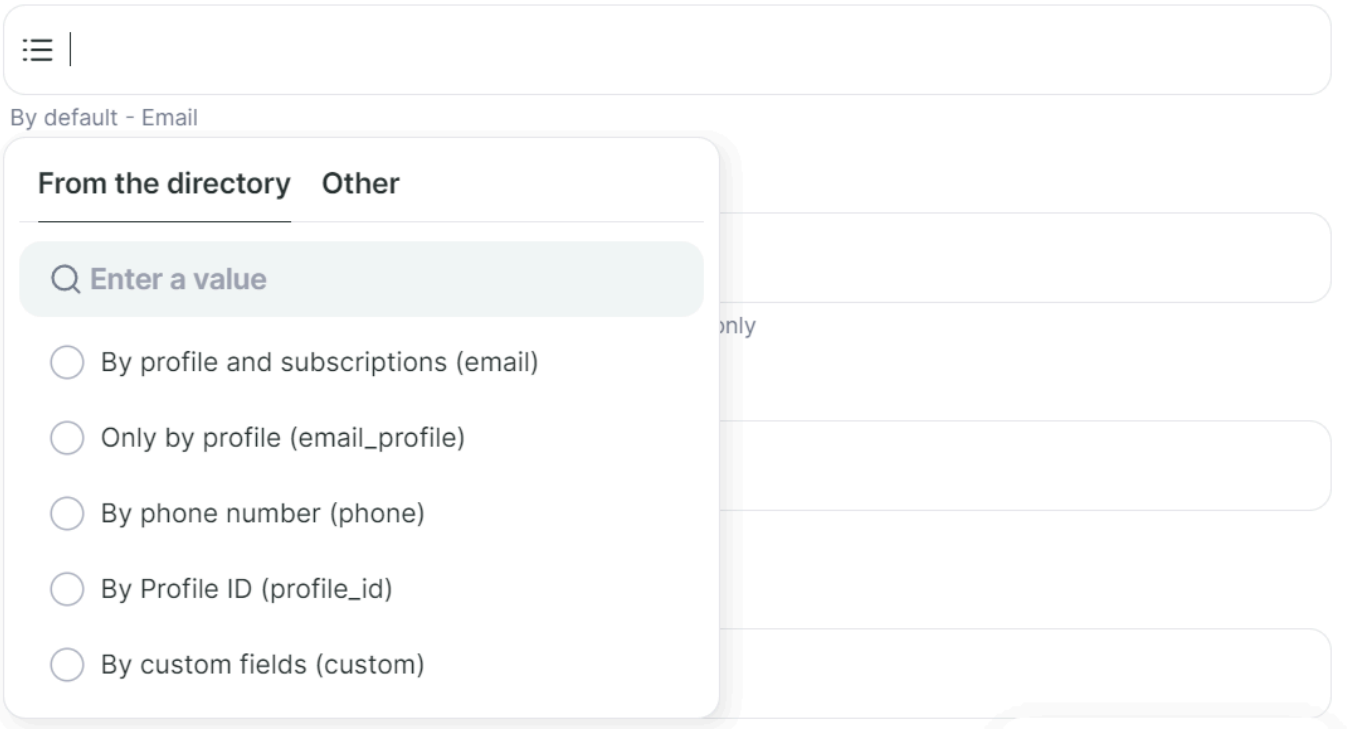


Connect your account and select the profile database.

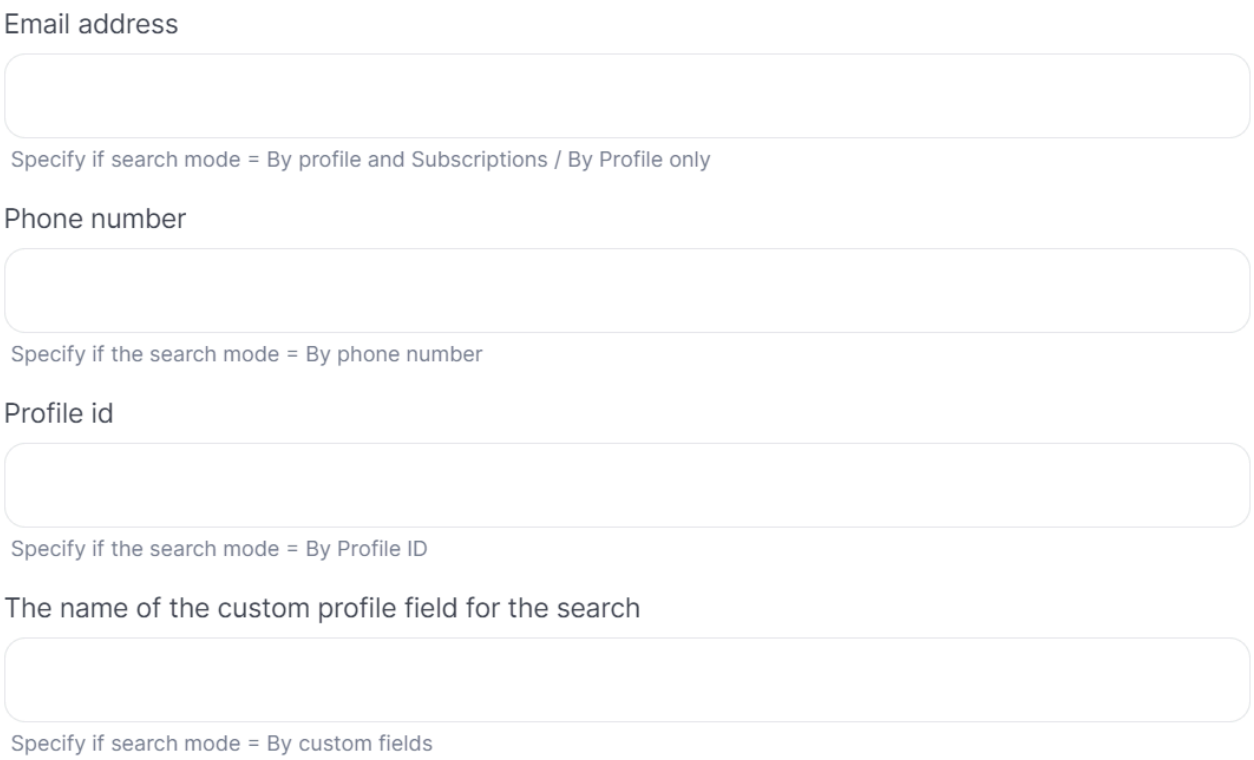
Next, specify the ID of the trigger mailing that you would like to launch:



Select the parameter by which the subscriber will be located in the database:



Depending on the selected parameter, specify the search field:

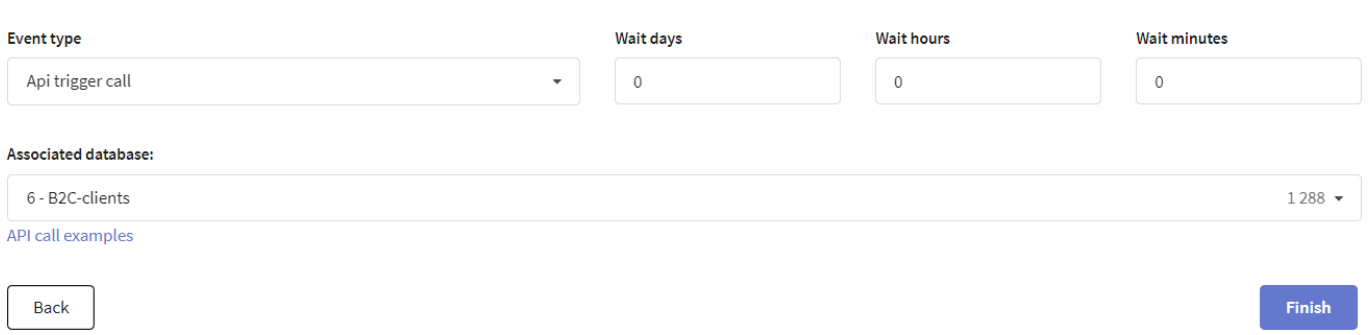


Optionally, specify the fields for matching.

Setting up a trigger mailing on the platform

First, you need to set up a trigger mailing.

In the Trigger section, specify Event type: API trigger call (accepts API calls from external services). We specify the database, the waiting time is optional.



Next we have to save and launch the mailing.

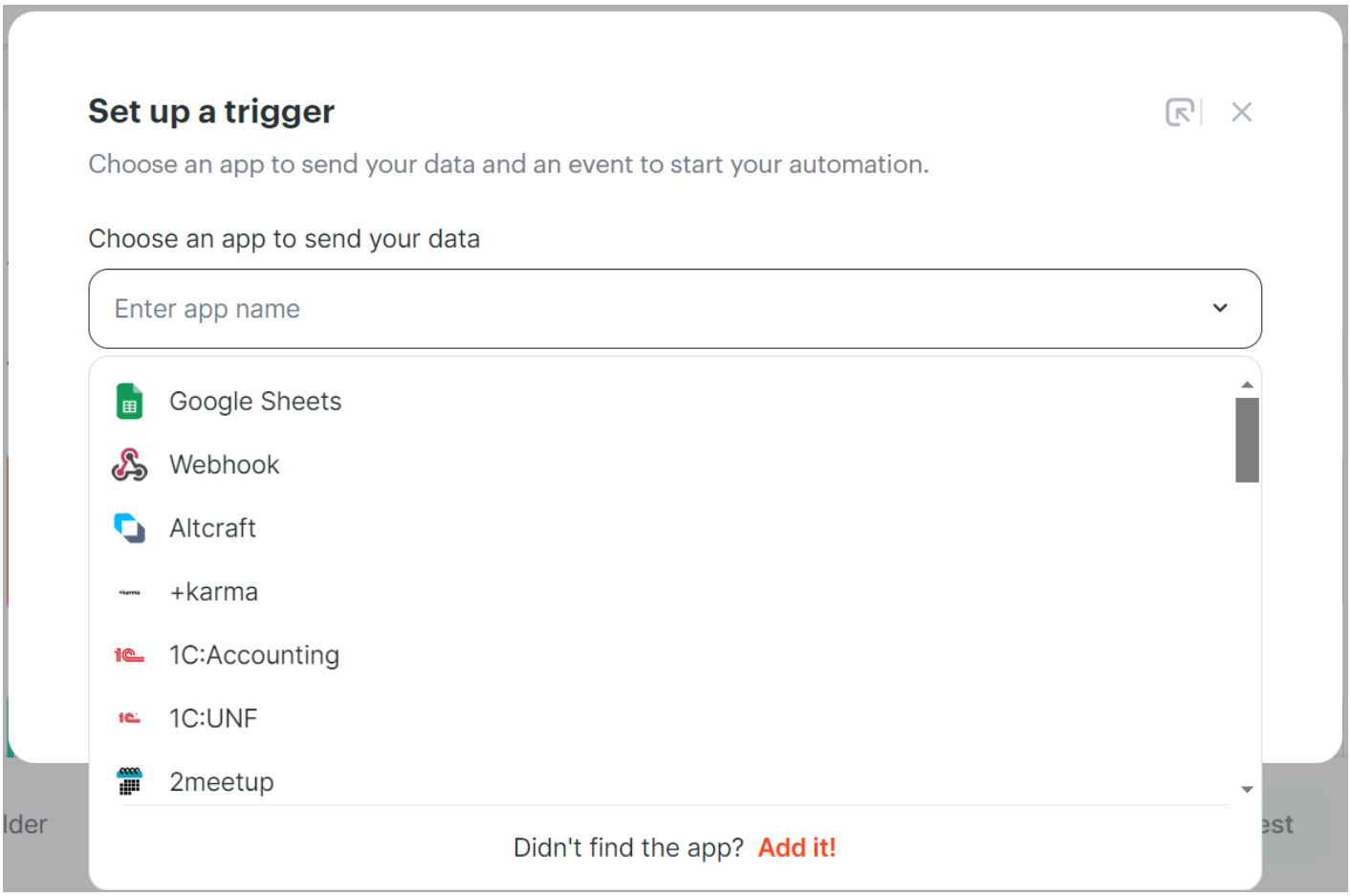
Event registration

Albato allows you to transfer customer data to subscriber profile on the platform.

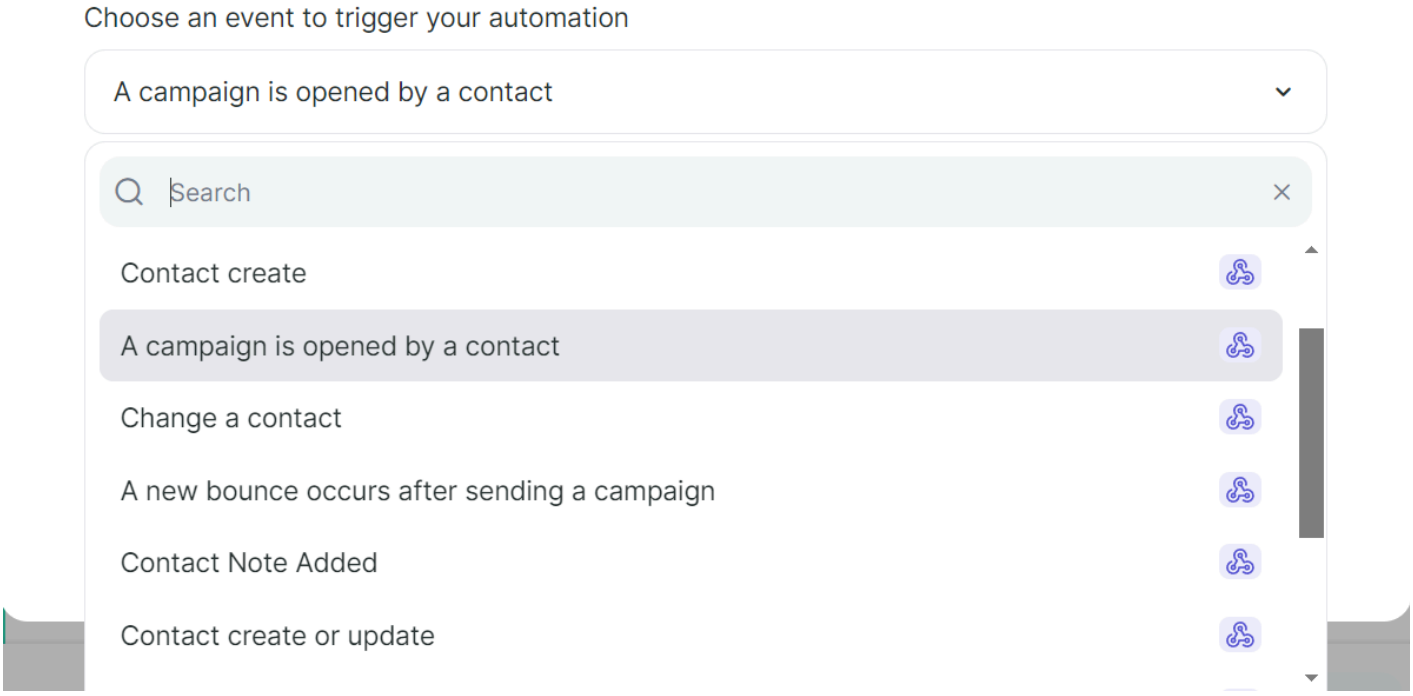
To do this, you need to set up a way External service —> Albato —> Altcraft

Set up the trigger

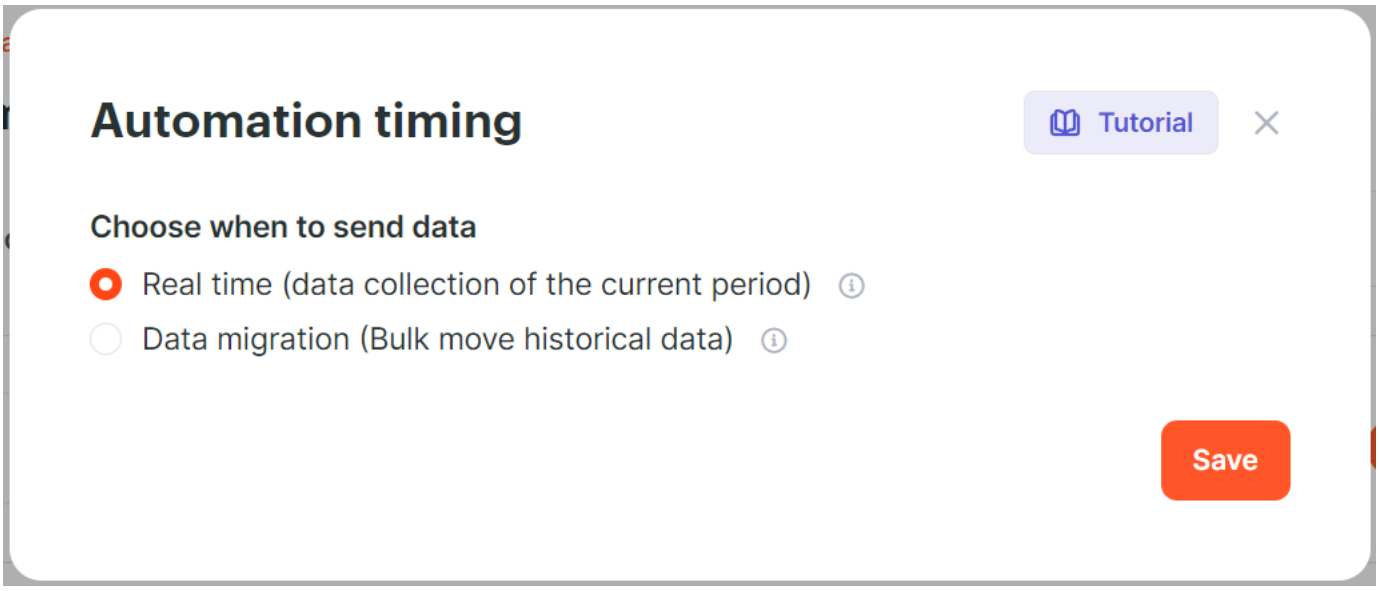
Specify the external service that will transfer data to the platform:



Select the event that will trigger the data transfer:

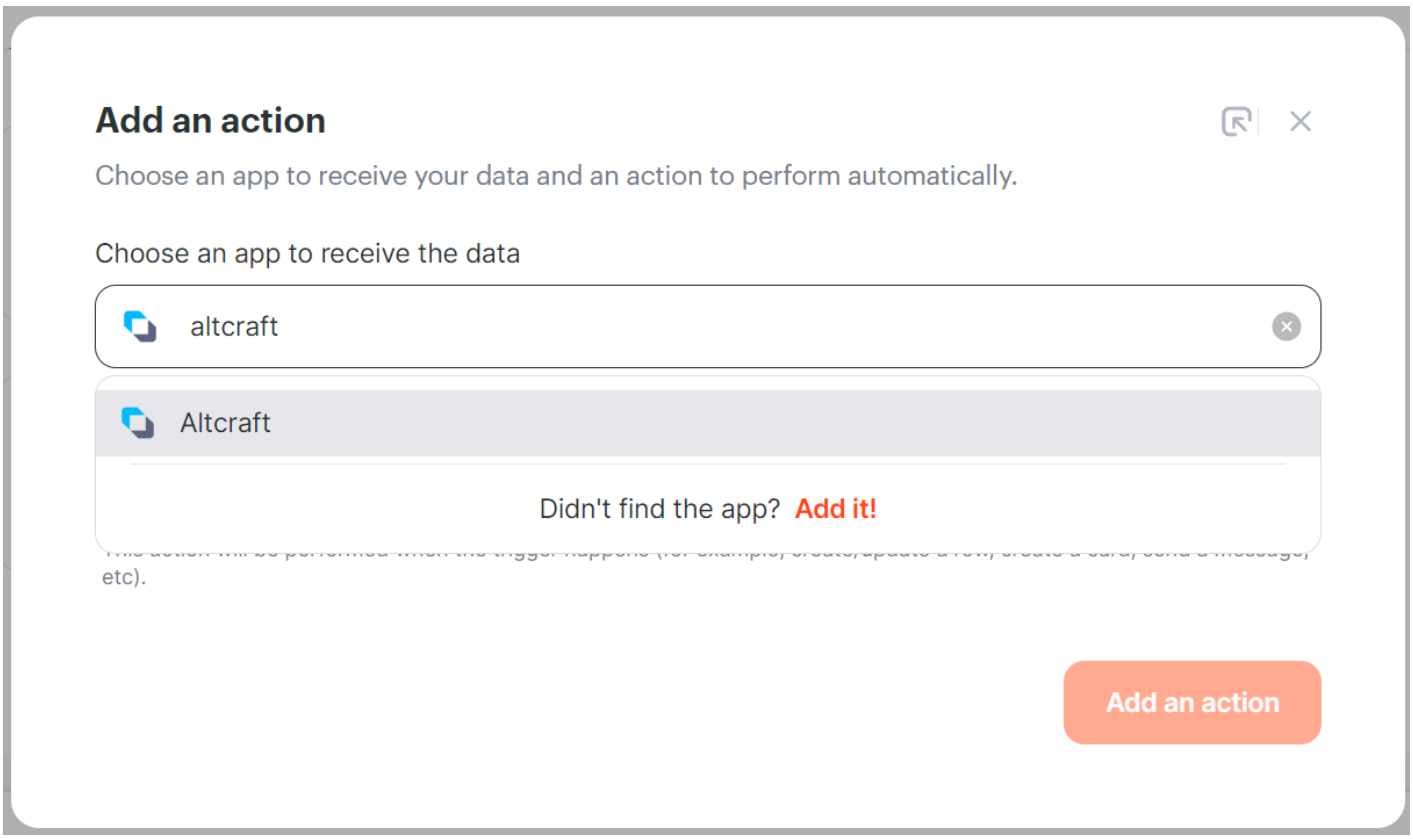


Select the mode of operation of the bundle: data can be sent to Altcraft in real time or specify for which period to collect it.

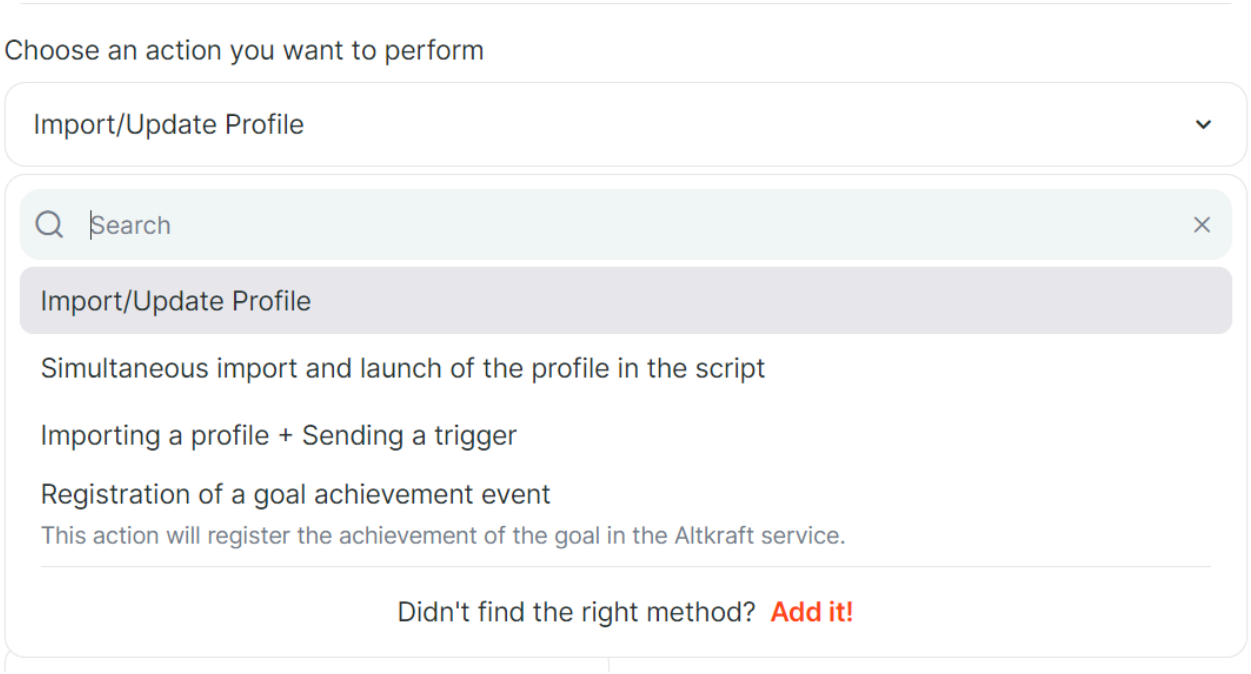


Set up the action

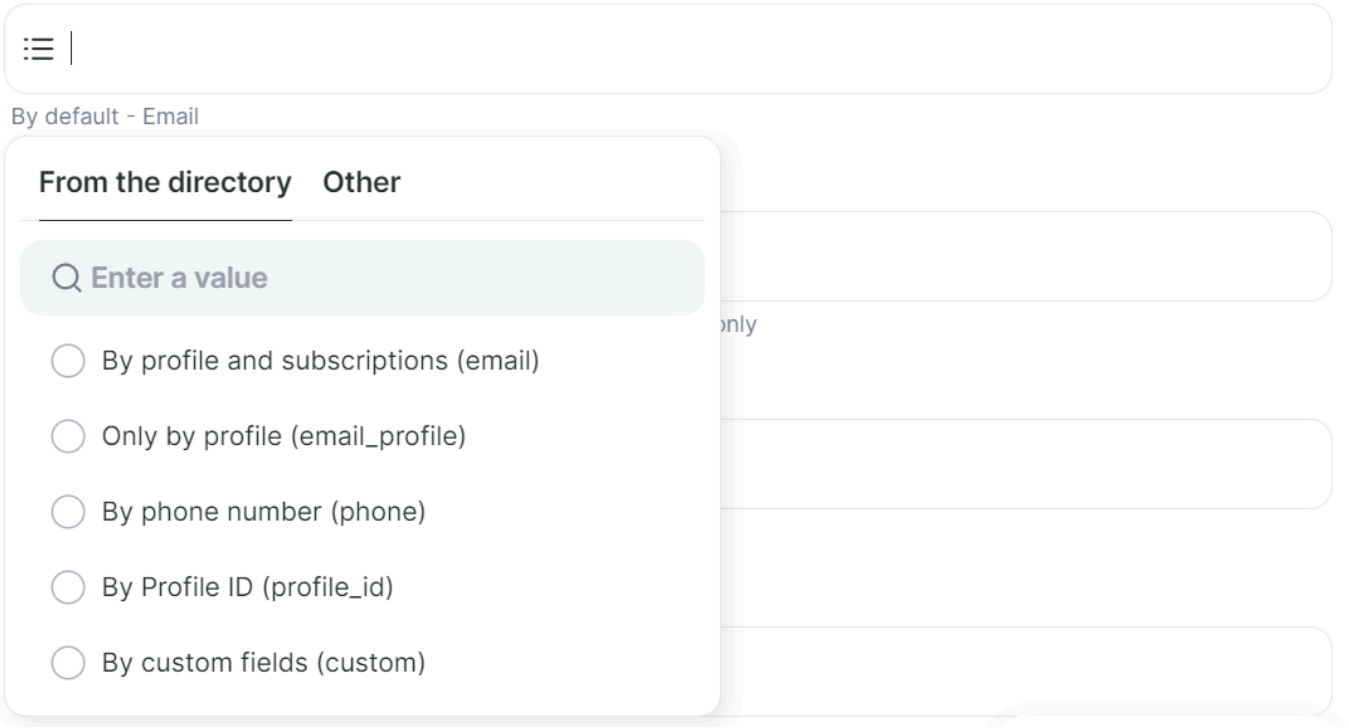
Specify Altcraft as a service for receiving data:



Assign an action that will take place after the event occurs. In order for the data to be imported into the subscriber's profile, select Import/Update profile:



Adjust the filter accordingly:



Optionally, set up additional fields.

After saving the bundle, when the trigger is started, the data will be transferred to the client's profile card.

Data analytics

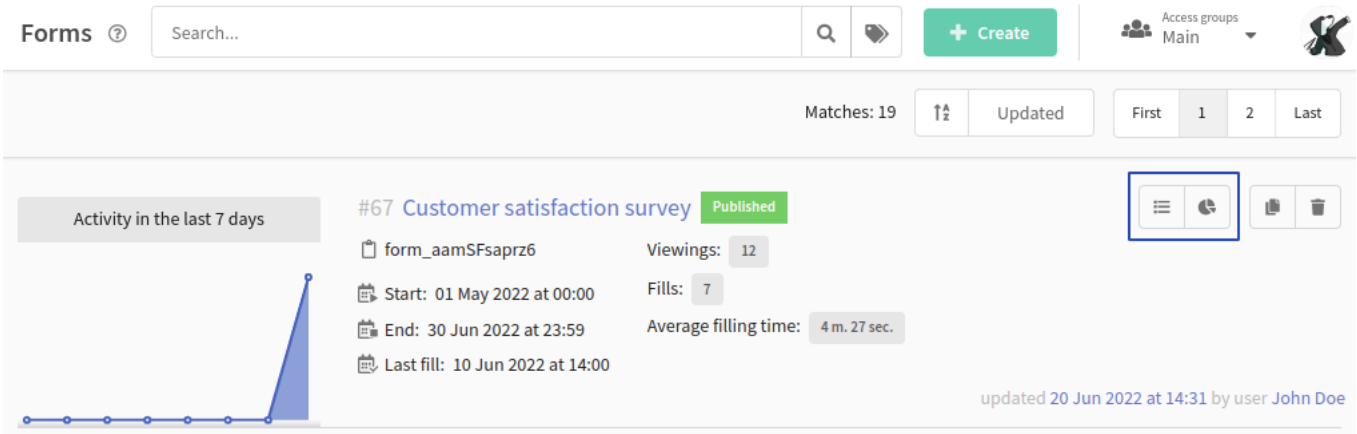
Description

You can analyze the results obtained after filling out the form in the “Collected data” section or in the “Insights” section.

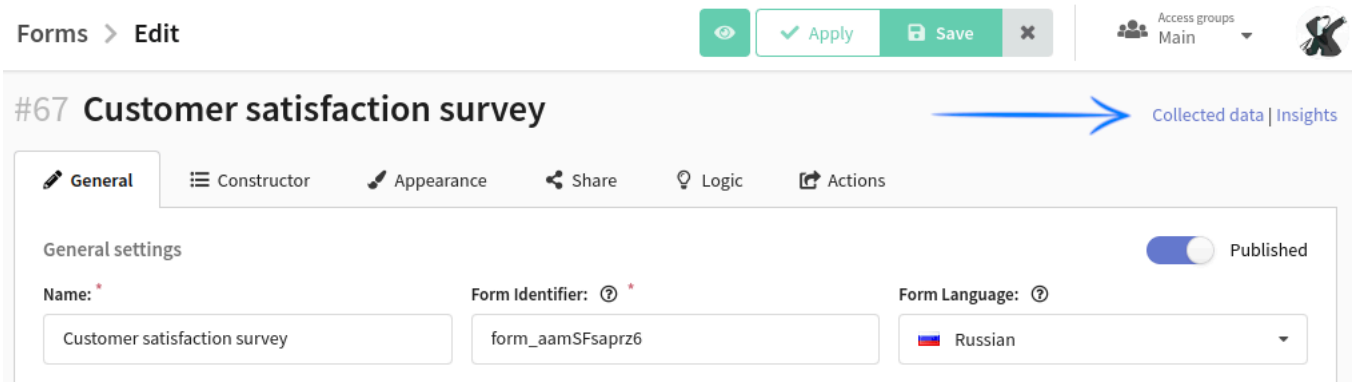
Collected data shows the responses of a specific user.

Insights provides a general summary of the form (views, completions) and statistics on a specific question.

To go to one of the sections, find the required form in the general list and click the corresponding icon on the toolbar:

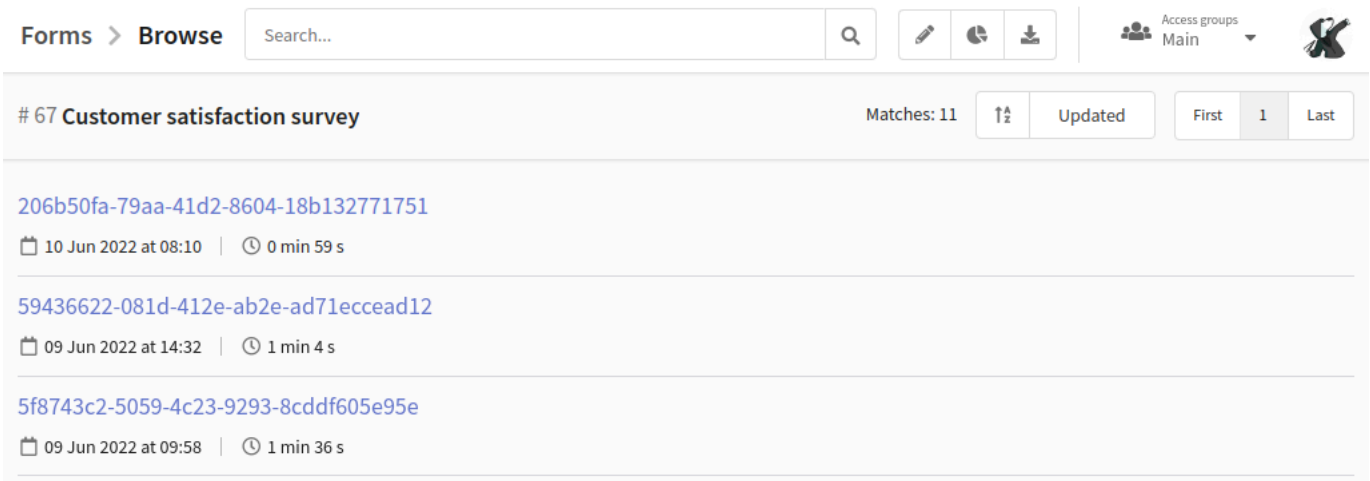


You can also go to analytics from the form editing window:



Collected data

Open the form and go to the Collected data section. Here you will see a list of IDs of all completed forms. Click on an ID to get detailed information.



INFO

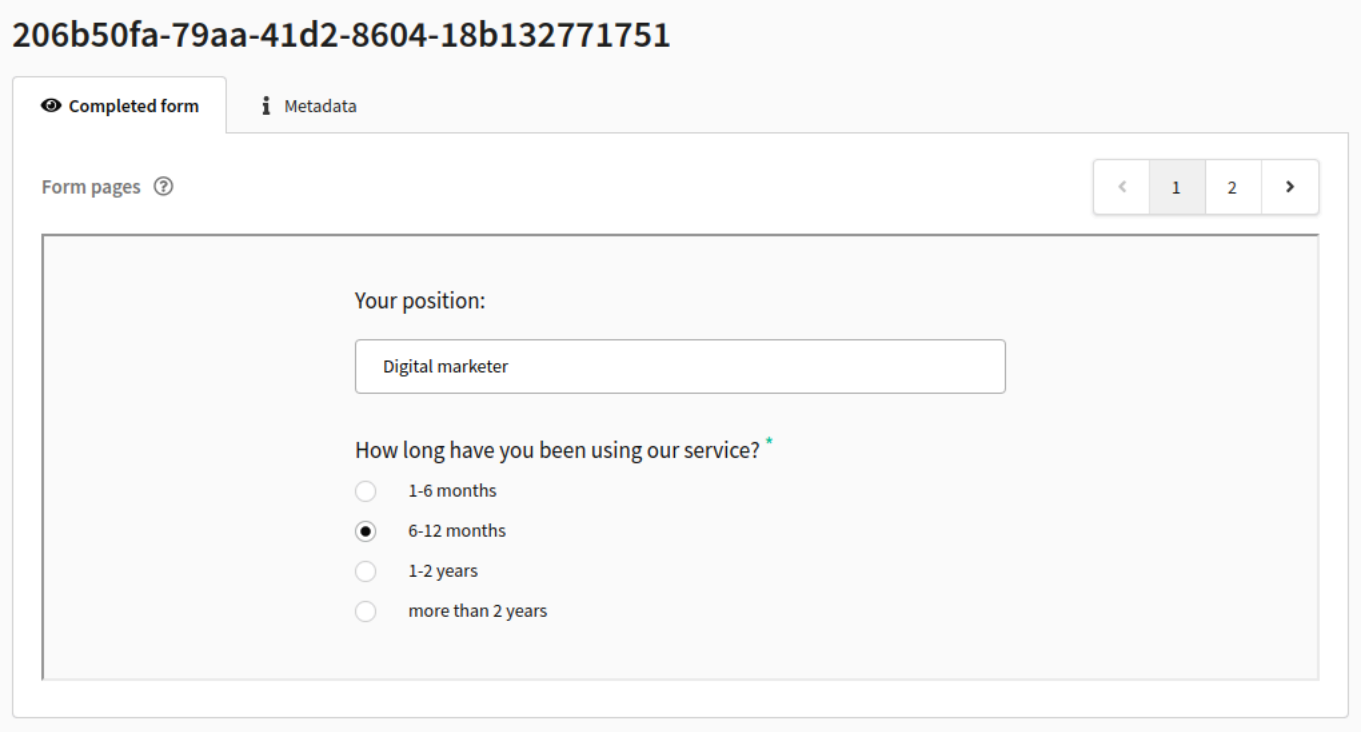
The following buttons are located in the upper right corner:

— go to the form editing window.

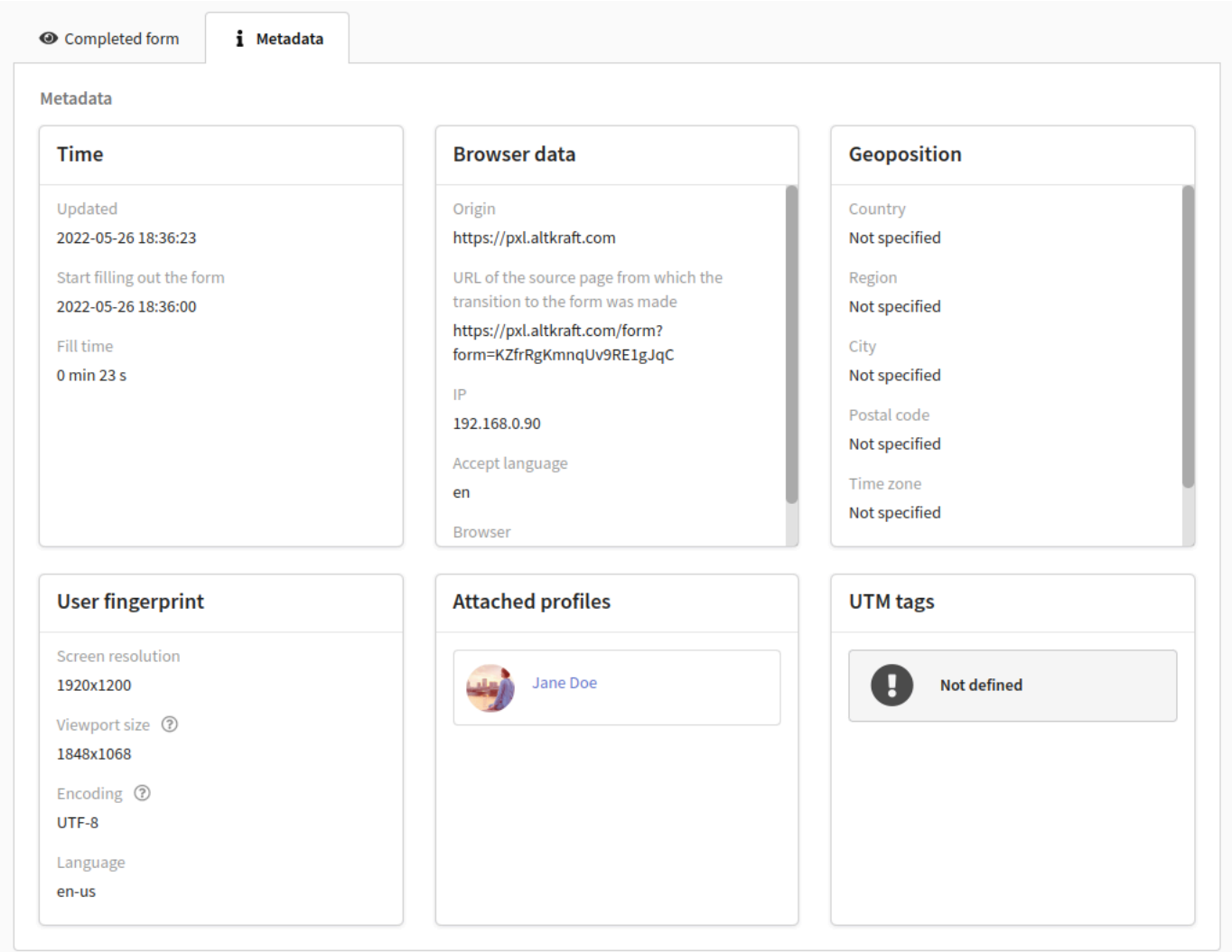
— open form insights.

— export data on form completions for a certain period.

On the Completed form tab, you can find out exactly how the user filled out each field:



The Metadata tab displays detailed information about filling out the form: time, browser data, geolocation, user fingerprint, and UTM tags.



Form insights

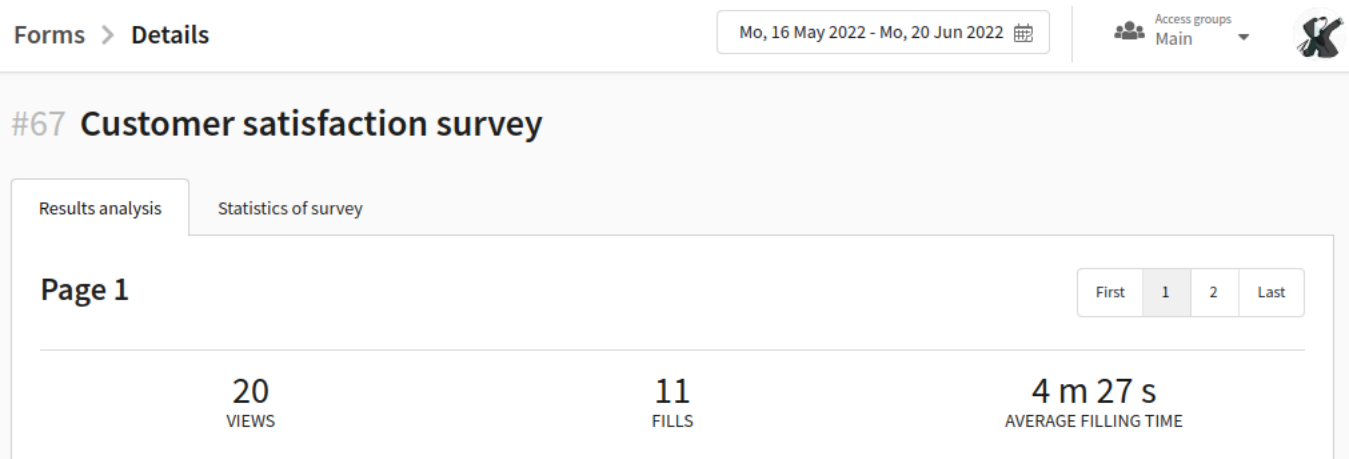
Open the form and go to the "Insight section. On the top panel, open the calendar and select the period for which you want to receive data.

Analytics provides the following information:

- **Views (how many times the form page was loaded).**
- **Fills (how many times the form was filled out and submitted).**
- **Average filling time .**

INFO

These metrics are not unique. If the same user has viewed or completed the form multiple times, all views and submissions will be counted.



There are two tabs available in Form insights:

- **Results analysis — statistics on a specific field in the form.**

- **Statistics of survey — a general summary of the form.**

Results analysis

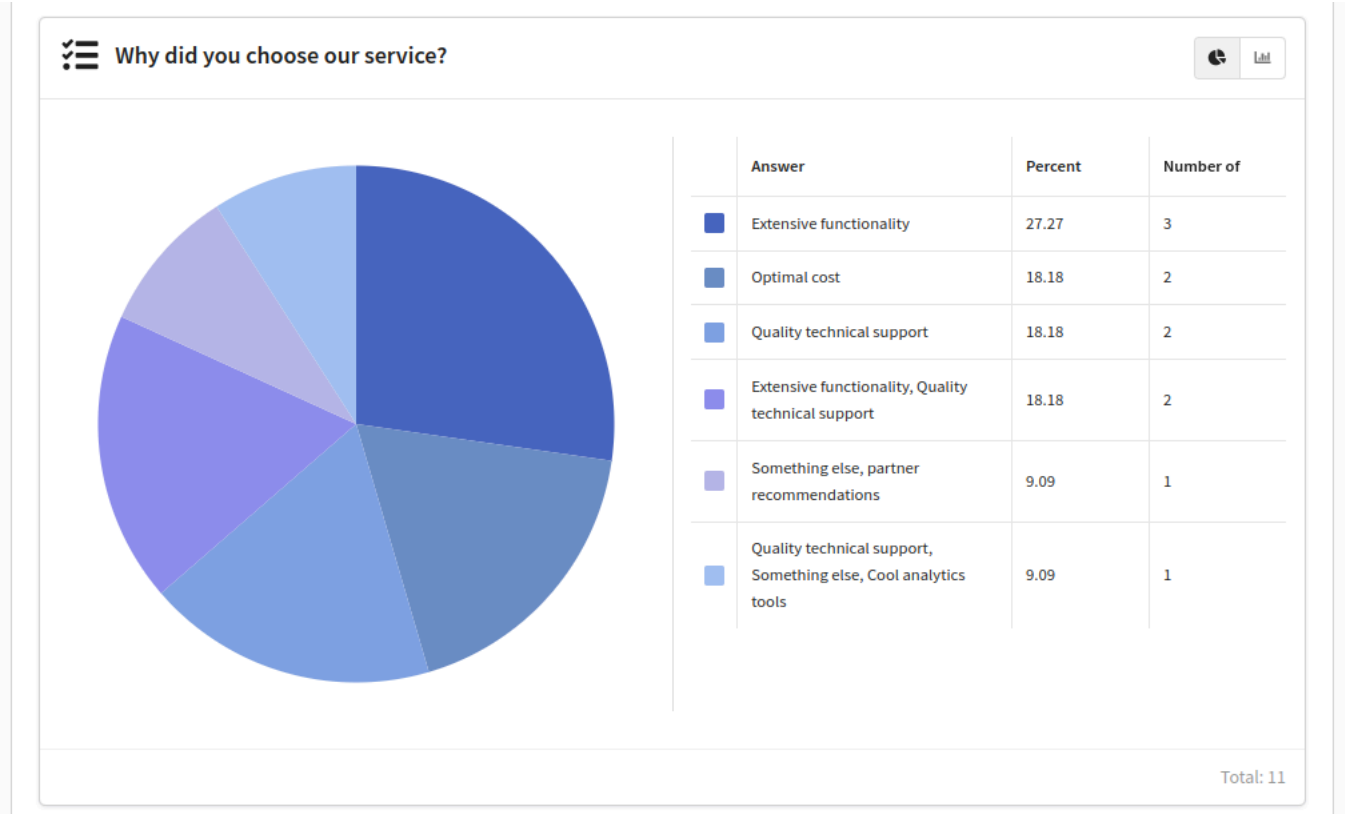
This tab displays statistics on the completion of a particular field in the form. To switch between pages of the form, use the buttons in the upper right corner.

Data for each field is collected in a separate block. Here you can see the question and answers given by users. At the bottom of the block, it shows how many times the field was filled in total.

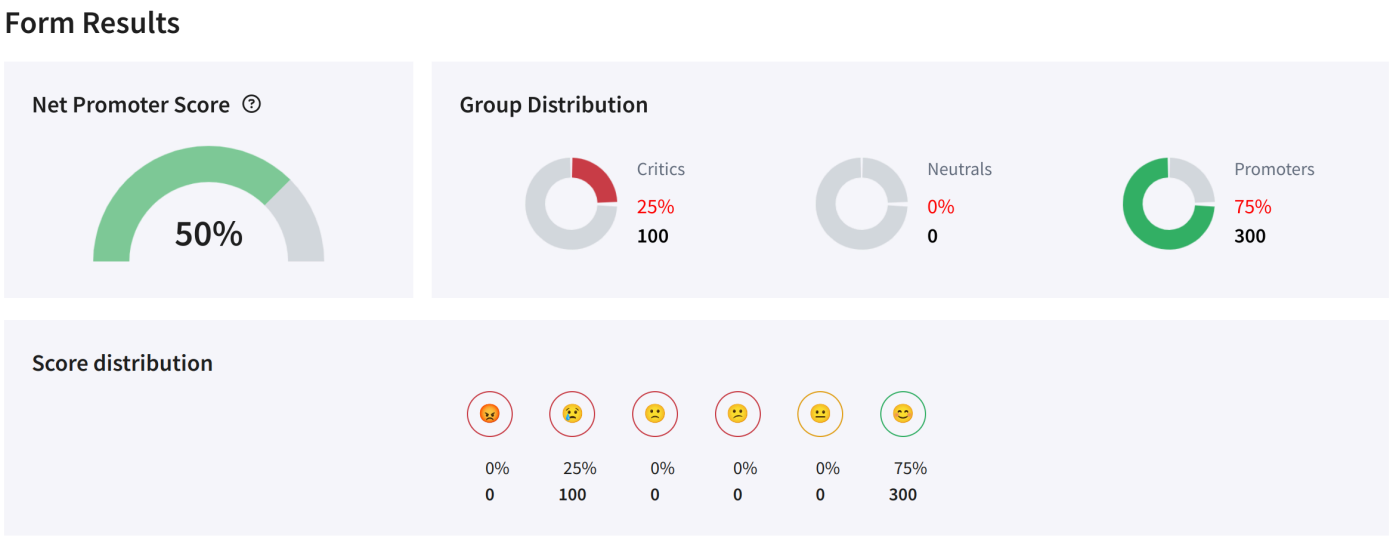
Most of the data are presented in percentage and quantitative terms. Additionally, it is possible to visualize them in the form of a bar -

or pychart

.

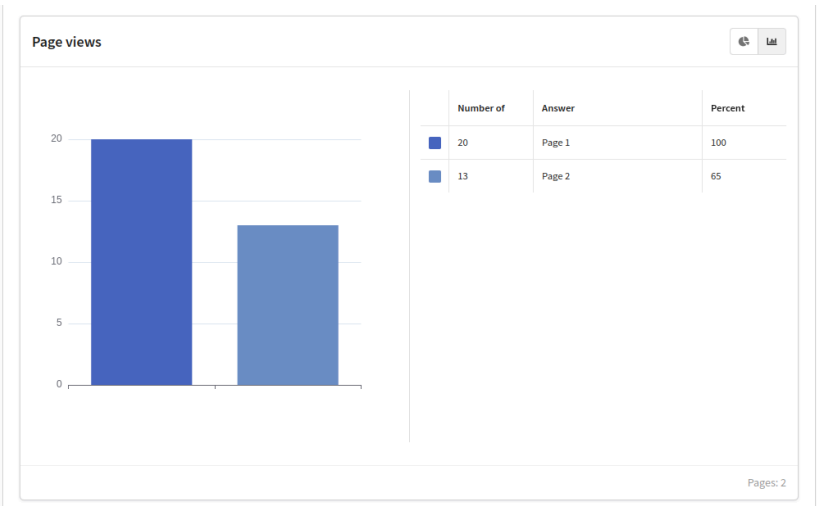
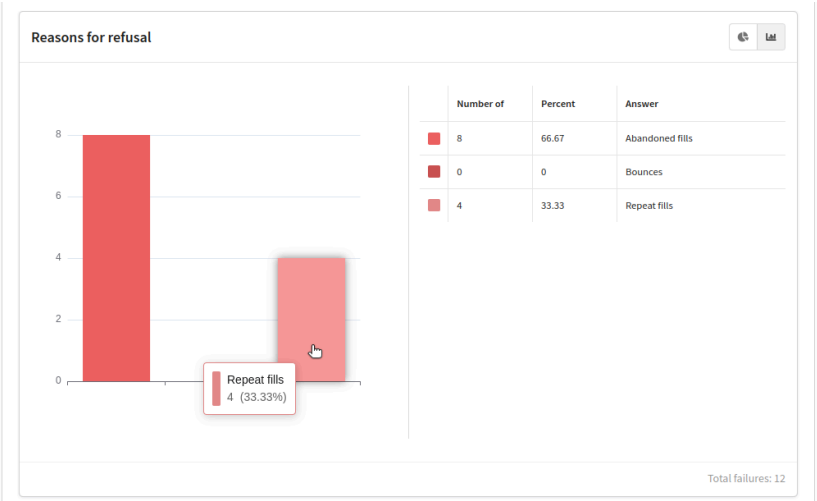


For forms containing NPS survey, the Net Promoter Score and the distribution of responses by group will be provided:



Statistics of survey

On this tab you can find the following information:

Page views **Views will show how many times each page has been loaded. With this metric, you can find out on which page users most often abandon the form, and further optimize it.**	
Reasons for refusal **1. Abandoned fills — the user partially or completely fills out the form, but does not submit it.** **2. Bounces — there were technical problems when filling out the form (for example, the server stopped).** **3. Repeat fills — the same user fills out the form several times. Although such fillings appear in the rejection reasons, they are also included in the overall summary.**	

Analytical slices of the report

Below you can group respondents by additional parameters:

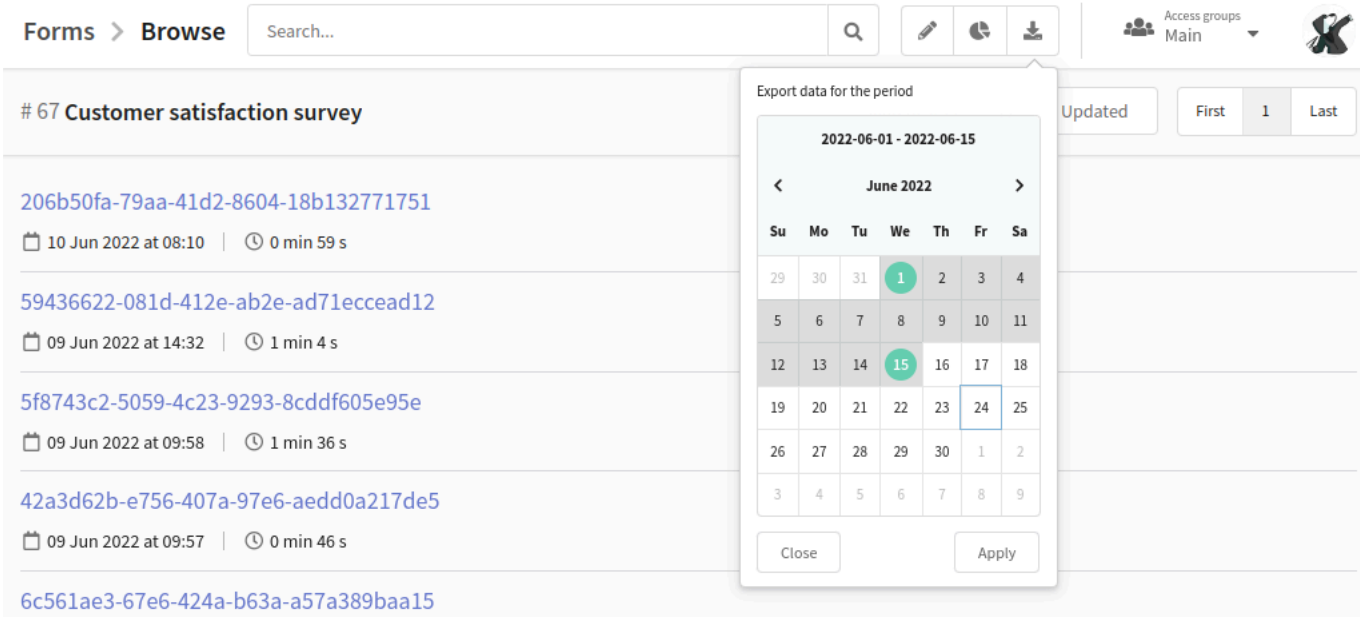
- **referrer**
- **browser**
- **operation system**
- **language**
- **time zone**
- **UTMs**

Form export

Choose a form. Go to the “Collected data” section. There is an export button

at the top of the page. Click on it.

In the calendar, specify for which period to export data. Click Apply. Go to Tasks and download the file.


Forms > Browse
Search...
Access groups
Main
67 Customer satisfaction survey
Updated
First
1
Last
206b50fa-79aa-41d2-8604-18b132771751
10 Jun 2022 at 08:10
0 min 59 s
59436622-081d-412e-ab2e-ad71eccead12
09 Jun 2022 at 14:32
1 min 4 s
5f8743c2-5059-4c23-9293-8cddf605e95e
09 Jun 2022 at 09:58
1 min 36 s
42a3d62b-e756-407a-97e6-aedd0a217de5
09 Jun 2022 at 09:57
0 min 46 s
6c561ae3-67e6-424a-b63a-a57a389baa15
Export data for the period
2022-06-01 - 2022-06-15
June 2022
Su Mo Tu We Th Fr Sa
29 30 31 1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 1 2
3 4 5 6 7 8 9
Close
Apply

Binding data channel and forms

Description

In most cases, the form is intended for website visitors. We create forms to invite new visitors to subscribe to our content newsletter. How does it work? The client goes to the site, sees the form and, if he has both strength and desire, fills it out. The next step is all the data the user left is sent to one of the databases on the platform and we get a new digital profile. As you may have noticed, the people who fill out the form usually don't subscribe to any of our resources yet.

But what if we'd like to get feedback from the client who is already subscribed to one of the resources? Or what if we'd like to collect additional information about the profile in order to decide whether it is worth subscribing to a new resource? In this case, we send the form to the person whose digital profile is already in our database. Will the Altcraft platform be able to recognize a familiar profile?

The answer is yes. If the profile that decided to fill out the form is already in one of your databases, the platform will identify it the moment the person opens the form.

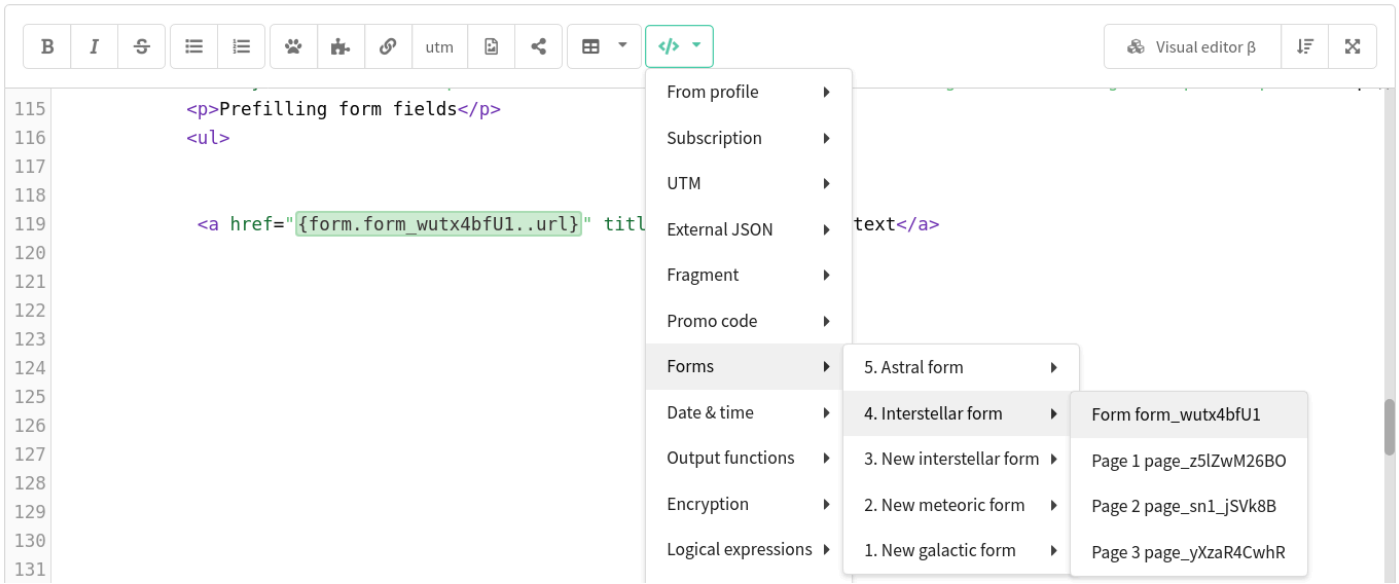
What are the advantages? You can set up automatic prefilling of form fields with data from the profile. For example, in a form, you have fields where you ask the customer to enter his email and phone number, but you already know all this information. Why not save our client's time and just send him a link to the template, where both his email and his phone number will be filled out?

The only thing is, all this is possible only if you use one of the communication channels to send the form. If you just add HTML code of the form in the website structure, you can't prefill the form. Setting up automatic identification of a profile when filling out a form occurs at the time of creating a template for the mailing — in the HTML version of the letter.

How to add a form to a template

First of all, let's add a link to our template.

Now in the HTML version of the letter, click `</>` and select "Forms" from the drop-down list. You'll see all existing forms on the right. Choose one of them.



Then, move the form variable you've just added to the value of the href attribute.

```
<a href="{form.form_eyyVPnQTE.url}" name="link1">Link to the form "form_eyyVPnQTE"</a>
```

```
<p>Prefilling form fields</p>
<ul>


 <a href="{form.form_wutx4bfU1..url}" title="Link text">Link to the form</a>
```

TIP

By the way, if there are several pages in the form, you can add a link to the page you need.

```
<a href="{form.form_eyyVPnQTE.page_eq55Lrs_J.url}" name="link2">Link to the first page of the form "form_eyyVPnQTE"</a>
```

```
<p>Prefilling form fields</p>
<ul>


 <a href="{form.form_wutx4bfU1.page_z5lZwM26BO.url}" title="Link text">Link to the first page of the form</a>
```

How to set up auto-complete fields

First, add a link to the form. We have described how to do this above.

After that, you need to pass the parameters and their values. Parameters are the fields of our form. For example, “name”, “email”. But we cannot pass such general names — it is necessary to find the identifier of each field.

Where can you find it? Go back to your form. Each element you add has its own ID.

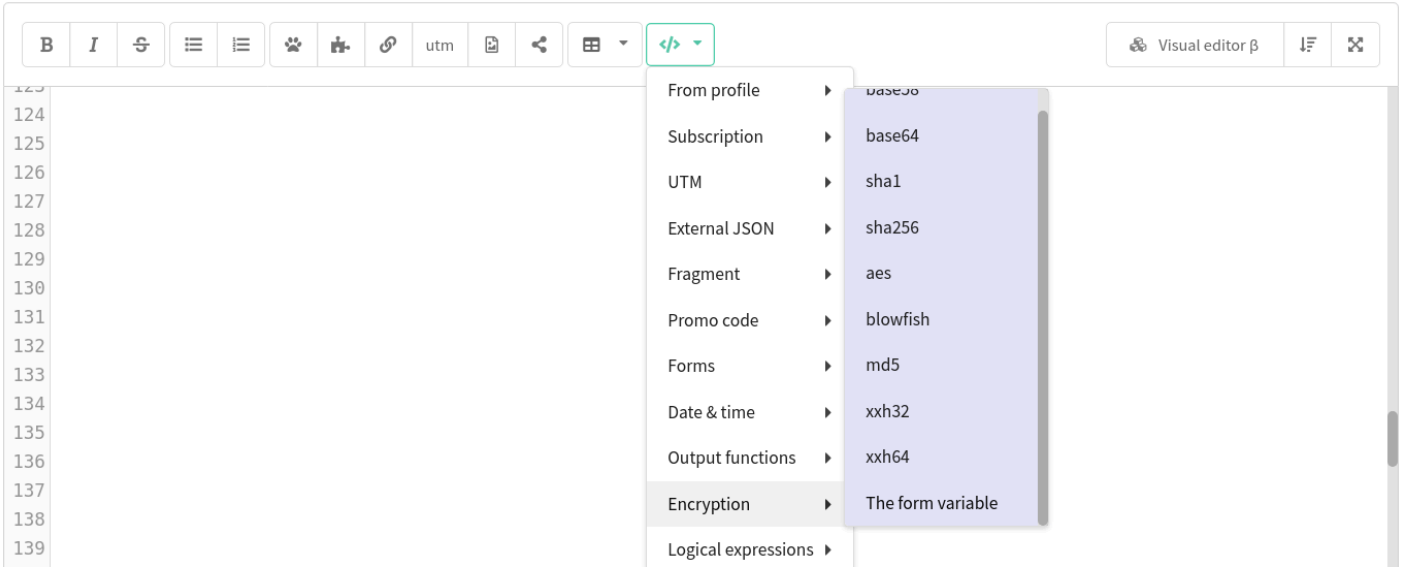
Copy it. Go back to the template and paste the parameter, separated by the ampersand &.

```
<a href="{form.form_eyyVPnQTE.url}&field_54vkfiZJC" name="link3">Link to the form with automatic prefilling of fields</a>
```

Now it"s time to figure out the values of the parameters. The simplest option is to fill them on your own. In other words, enter a text or a number that will be displayed the same for each customer.

```
<a href="{form.form_eyyVPnQTE.url}&field_54vkfiZJC={formcrypt("Свое значение")}" name="link3">Link to the form with auto-filled fields</a>
```

However, if you want to personalize your content, use variables. To do this, in the top panel, click "`</>`", select “Encryption” → “The form variable”.



The variable `{formcrypt(lead.email)}` will be added to the template.

`lead.email` is the email of the subscriber, who is already in the Altcraft profile database. You can change the variable. Read the article about available variables.

The variable must be added to the parameter after `=`.

```
field_54vkfiZJC={formcrypt(lead.email)}

field_fxljWrmV5={formcrypt("Your text")}
```

You can add as many parameters as you'd like to by listing them separated by `&`.

```
<a href="{form.form_eyyVPnQTE.url}&field_54vkfiZJC={formcrypt(lead.email)}&field_fxljWrmV5={formcrypt("Custom value")}" name="link3">Link to the form with automatic prefilling of fields</a>
```

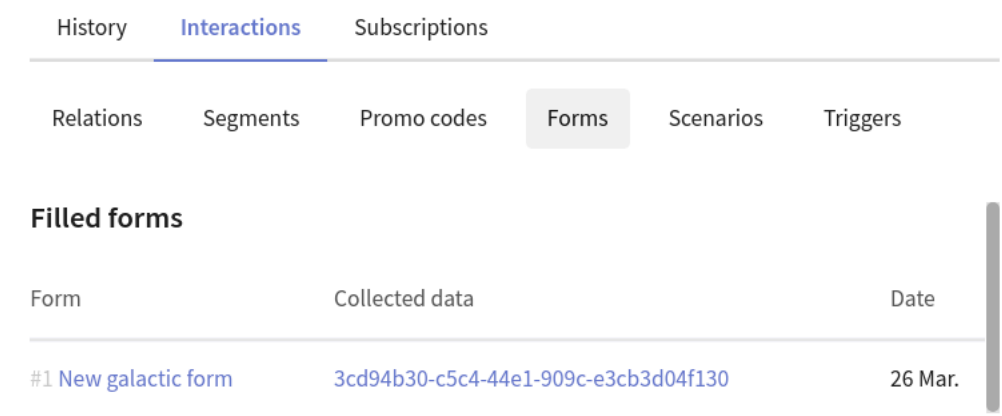
Matching by ID

When you create a form, you can use an additional matching mode in the Actions section: By message ID. To use this type of matching, you need to pass the smid parameter with the send_message_id value in the link.

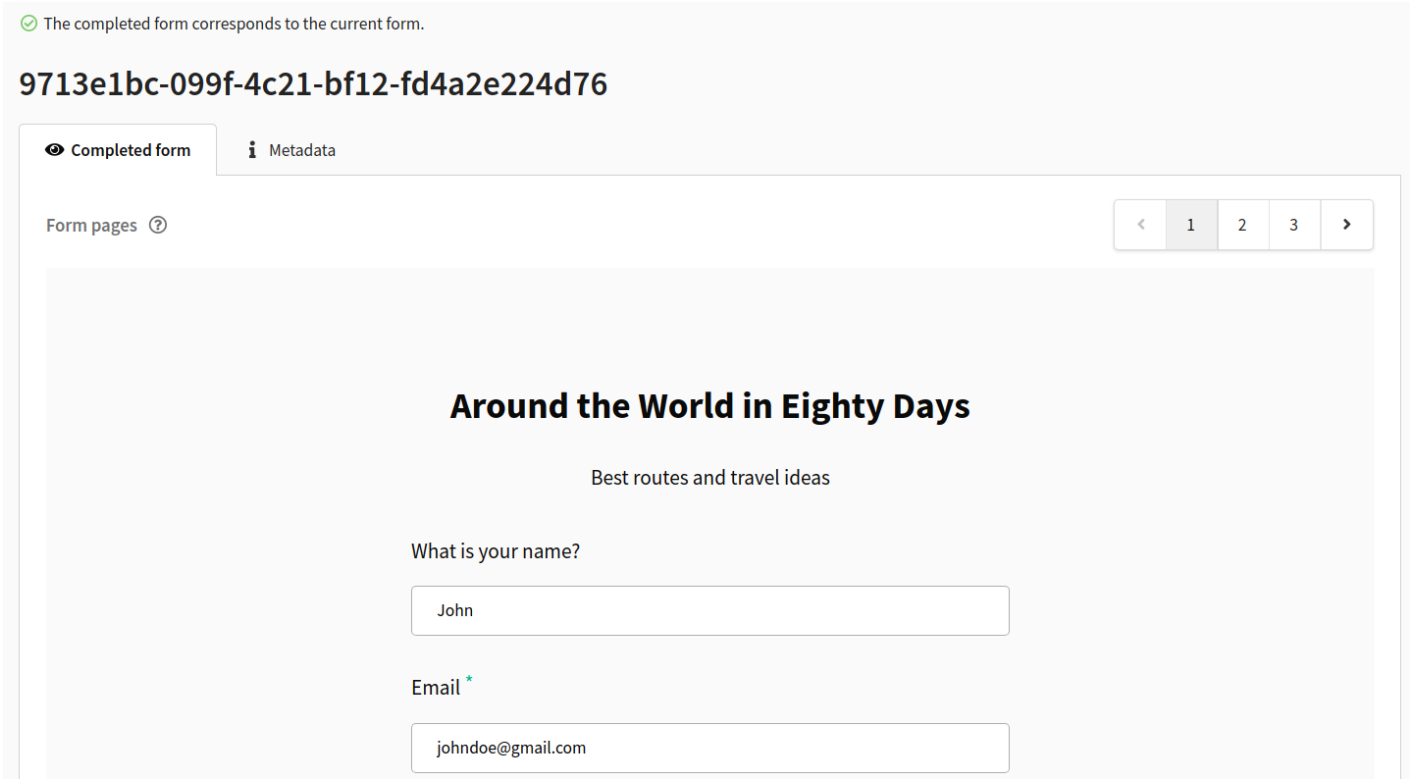
```
<a href="https://example.com/?smid={send_message_id}&field_54vkfiZJC={formcrypt(lead.email)}&field_5S0nwR_AD={formcrypt(lead.phone)}" name="link4">The link passes Message ID to the form and fills in the fields</a>
```

Collected data

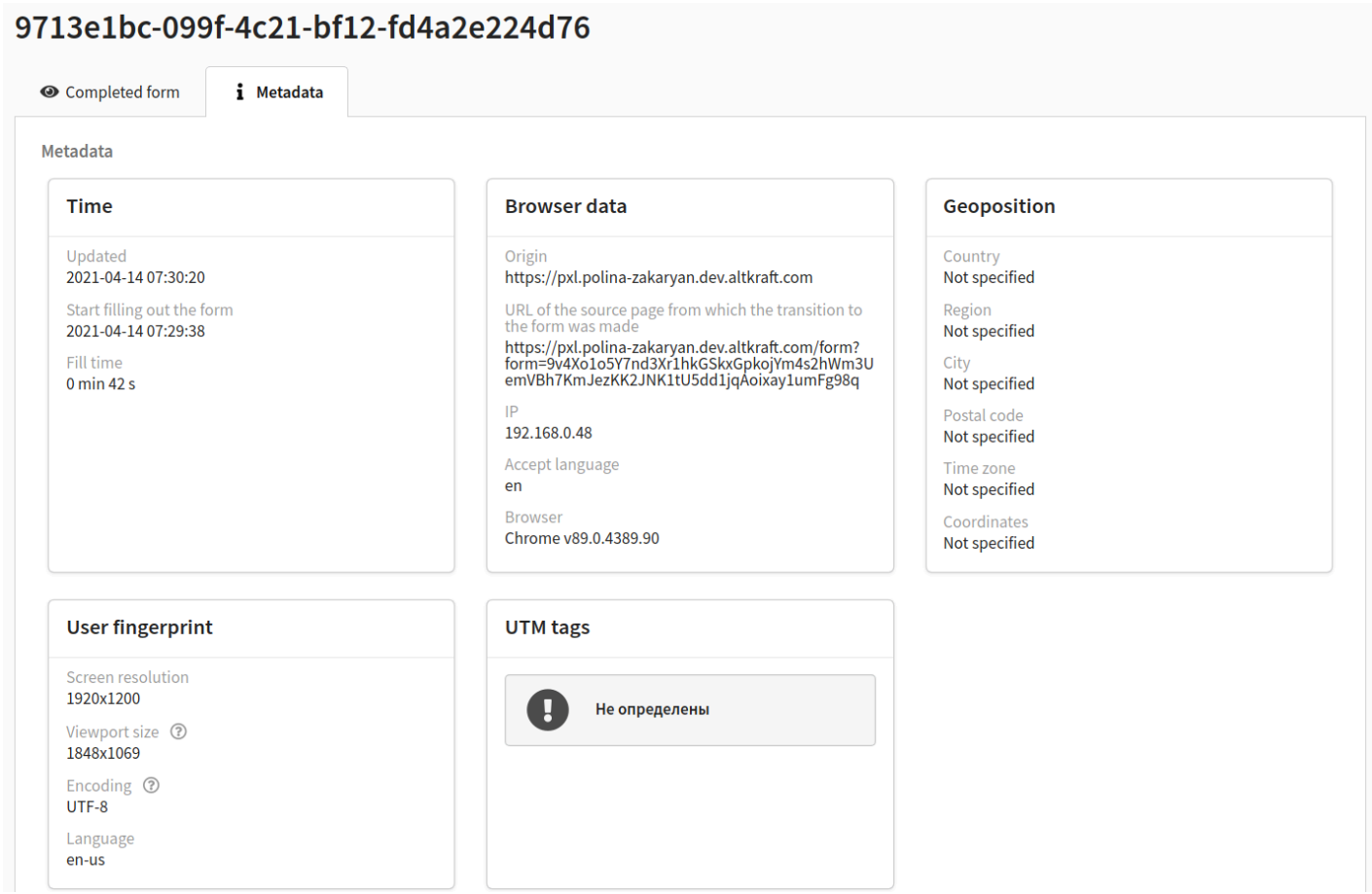
In the profile history on the "Interactions tab, you can find information about the subscriber's interactions with forms. For example, when and what forms he filled out.



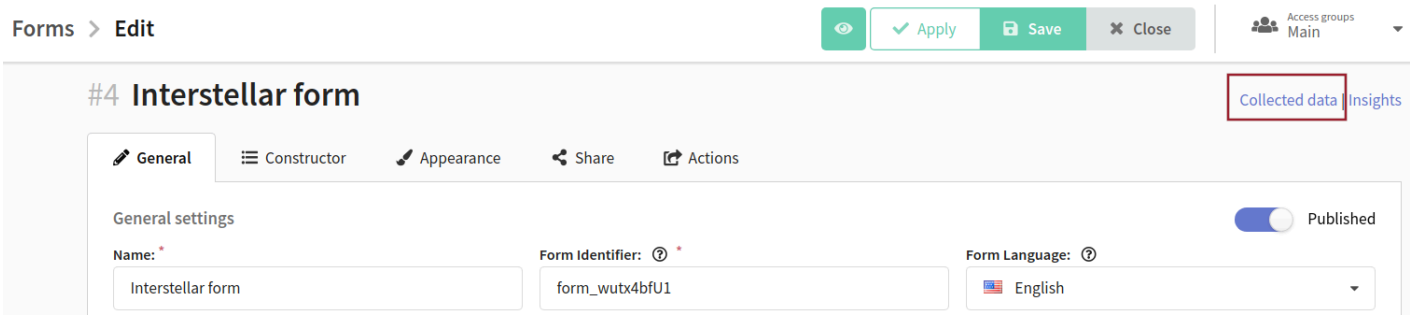
If you click on the identifier in the Collected data" column, you will be taken to a new page, where you can get more detailed information about the interaction of the subscriber with a specific form. For example, see how exactly he filled in each field.



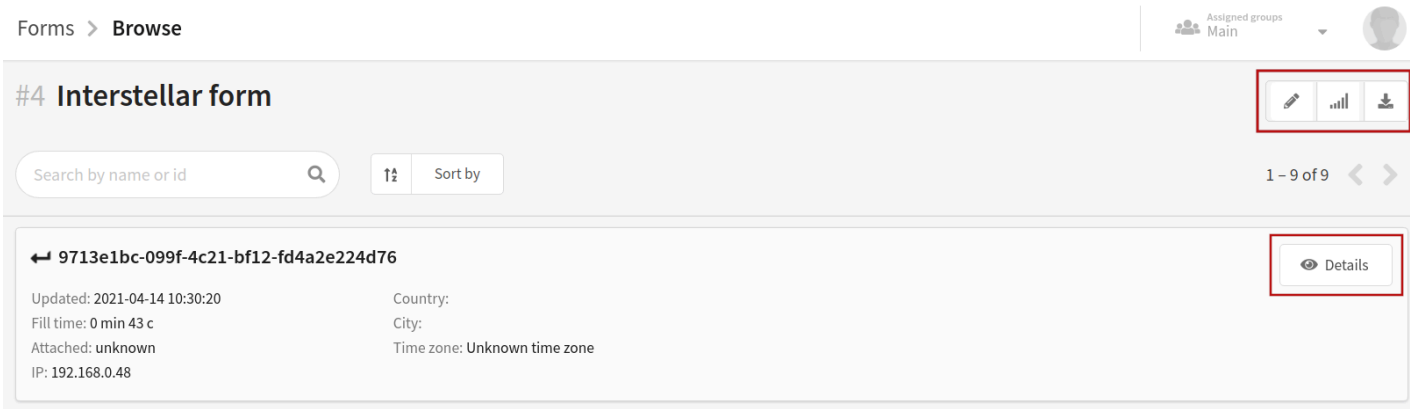
Go to the next tab - “Metadata”. Here you will find detailed information: from the time of filling and the browser that the subscriber was using at that moment to his geolocation and UTM tags.



By the way, you can go to the same panel with data for a specific subscriber directly from the form. Click Collected Data in the upper left corner.



Here you will see a list of the IDs of all users who filled out the form. Click “Details” to get detailed information about each of them.



NPS testing

The Net Promoter Score (or NPS for short) is a metric that shows customer loyalty to a company or product. In other words, the NPS index reflects the willingness of customers to recommend your product or services. This index is one of the main indices for measuring customer loyalty. NPS allows you to analyze the strengths and weaknesses of your product and adjust your marketing strategy.

Usually NPS testing is a form with a question How ready are you to recommend us on a scale from 1 to 10? and answer options. Depending on the answer, several types of respondents are distinguished:

- **Promoters - loyal customers who are ready to leave positive feedback and recommend the product**
- **Neutrals - neutral customers who use your product now, but will stop if they find a better offer.**
- **Critics (or detractors) - customers who have had a negative experience with you and are now ready to leave negative feedback about your company.**

NPS values of 30% and above are considered acceptable. A value between 0% and 30% indicates that there are some problems with the product. Negative index values indicate serious problems that require immediate intervention.

NPS calculation

In mathematical terms, NPS is the difference between the percentages of promoters and critics. The expression can be visualized as follows:

$$NPS = \frac{P-D}{T} \times 100\%$$

Where P are promoters, D are critics, T is total number of respondents.

The platform can use either average or aggregated NPS index when analyzing the resulting data. Average NPS is the arithmetic mean of all NPS indices within the form. Aggregated NPS is the overall NPS index derived from the responses of all respondents in each question. These can be calculated using the formulas:

$$NPS_{avg} = \frac{NPS_1 + NPS_2 + NPS_3 + \ldots + NPS_n}{n}$$

$$NPS_{agr} = \frac{\Sigma(P_i - D_i)}{\Sigma T_i} \times 100\%$$

Let's look at an example of calculating the NPS index. Let's say we ran 3 tests with the following results:

- **Test #1 - 6 critics, 9 neutrals and 27 promoters for a total of 42 respondents**
- **Test #2 - 10 critics, 15 neutrals and 25 promoters for a total of 50 respondents.**
- **Test #3 - 21 critics, 35 neutrals and 14 promoters for a total of 70 respondents.**

The NPS calculation for these tests will look like this:

$$NPS_1 = \frac{🙂 - 😡}{42} \times 100\% = 50\%$$

$$NPS_2 = \frac{🙂 - 😡}{50} \times 100\% = 30\%$$

$$NPS_3 = \frac{🙂 - 😡}{70} \times 100\% = -10\%$$

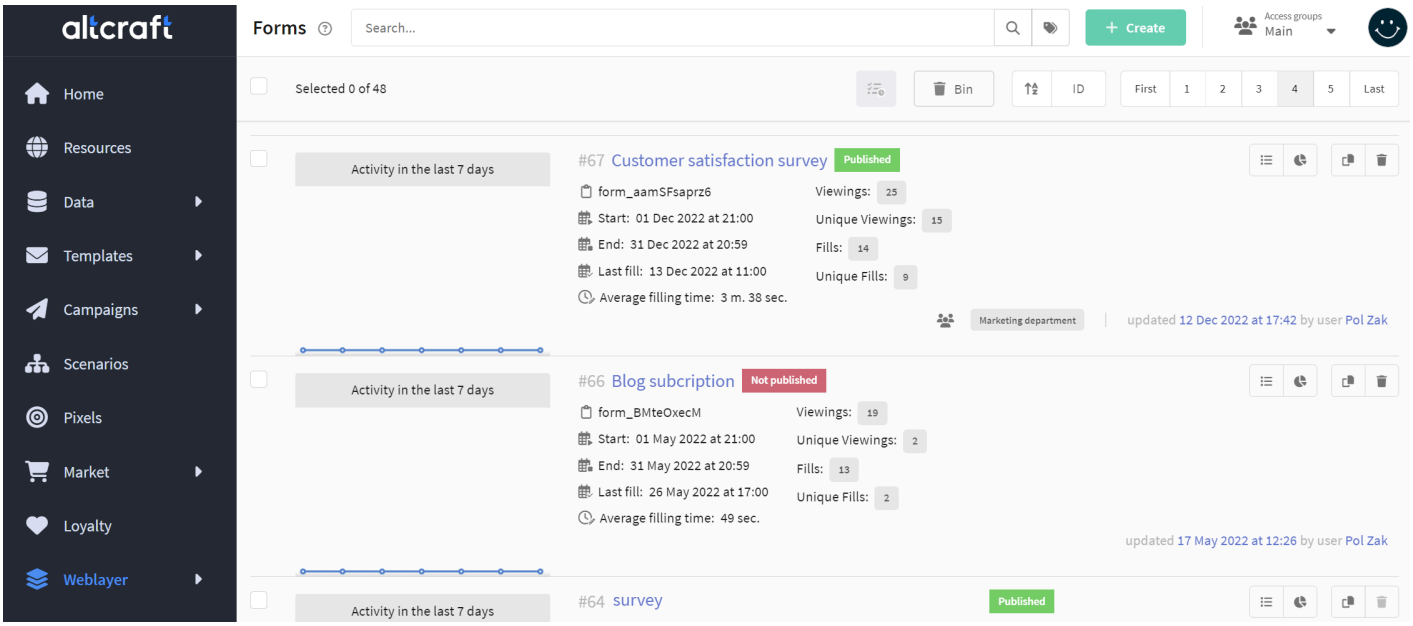
From this data, you can find the average and aggregated NPS:

$$NPS_{avg} = \frac{50\% + 30\% - 10\%}{3} = 23.3\%$$

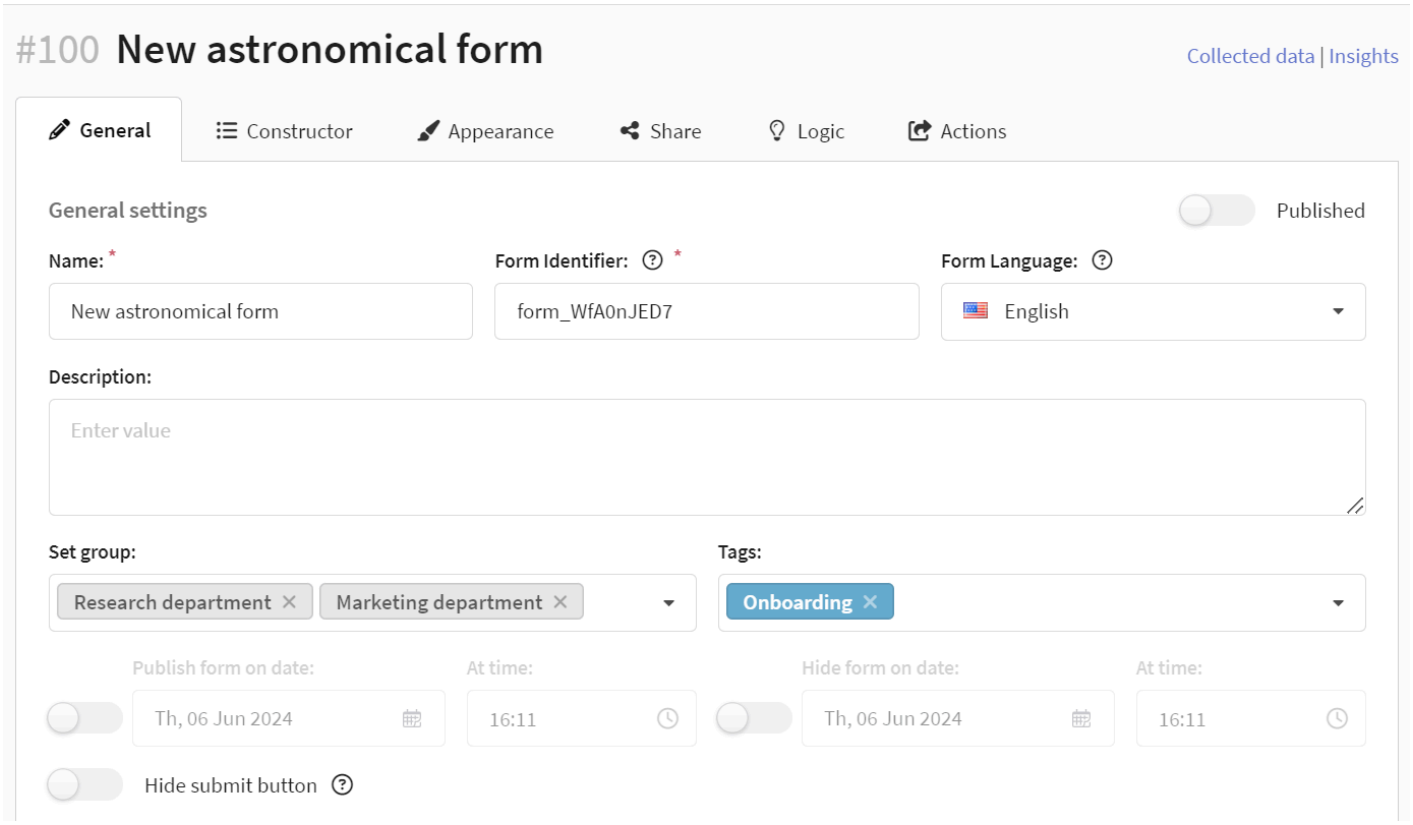
$$NPS_{agr} = \frac{(🙂_{27} + 🙂_{25} + 🙂_{14}) - (😡_{6} + 😡_{10} + 😡_{21})}{42 + 50 + 70} \times 100\% = 17.9\%$$

Creating NPS testing in the platform

NPS testing is implemented in the Altcraft platform using webforms. To create one, go to “Weblayer” – “Forms” in the main menu and click on Create:



On the first tab “General” the main form settings are available: name, description, identifier, language. To make the form available by URL or as an iframe element, you need to activate the Published switch in the upper right corner. Please note that until the form is unpublished, it cannot be deleted:



Below you can customize settings such as publishing schedule, final form page and set a refill restriction:

Publish form on date: | At time: | Hide form on date: | At time:

Mo, 11 Nov 2024 | 18:31 | Mo, 11 Nov 2024 | 18:31

Hide "Next" and "Submit" buttons

Final page

Redirect to the specified link bypassing the final page

Title:

Thanks!

Template:

Your form has been submitted successfully.

Refill restriction

Restrict form refill

Restrict within days

1

Redirect to the specified link if the form has been previously filled out

Title:

Restriction!

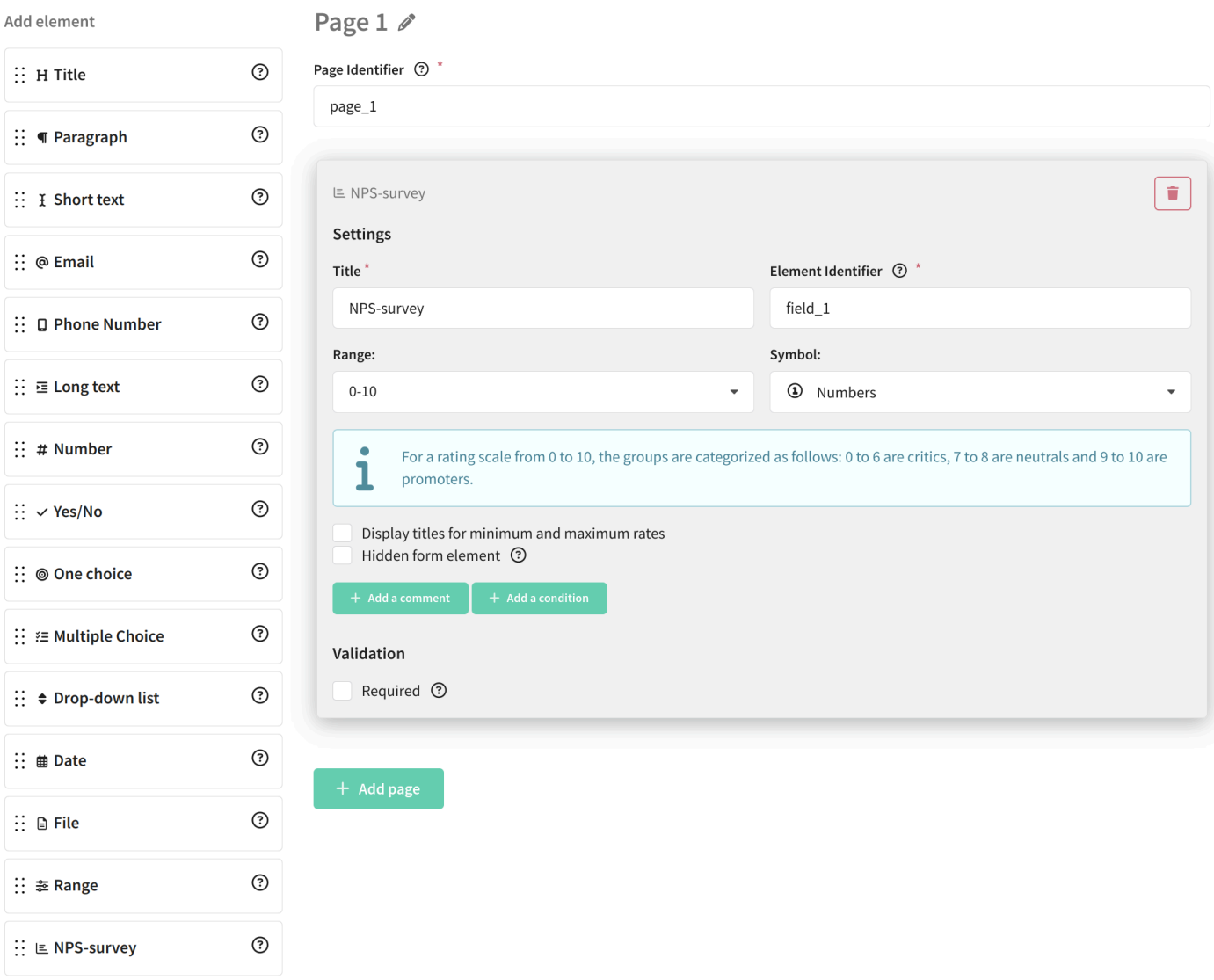
Template:

You have already completed this form.

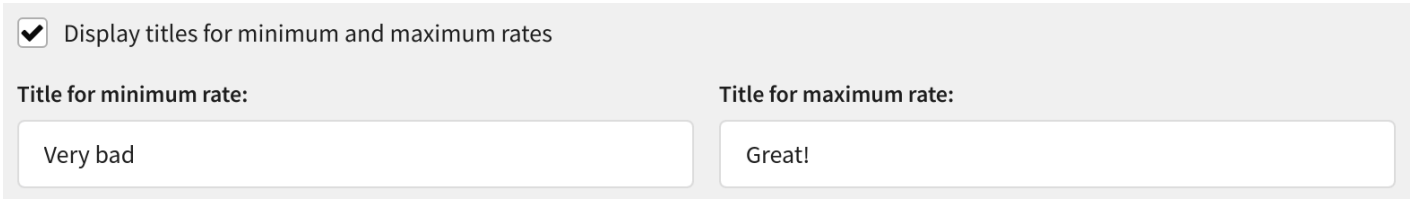
Insert additional code

- Additional code in <head /> tag
- Additional code at the beginning of the <body> tag
- Additional code at the end of <body> tag

To add an NPS survey to the form, go to the Constructor tab and add the NPS survey element:



Here you can change the title, scale and symbols for the scale. You can also make the item mandatory and add titles for the minimum and maximum score:



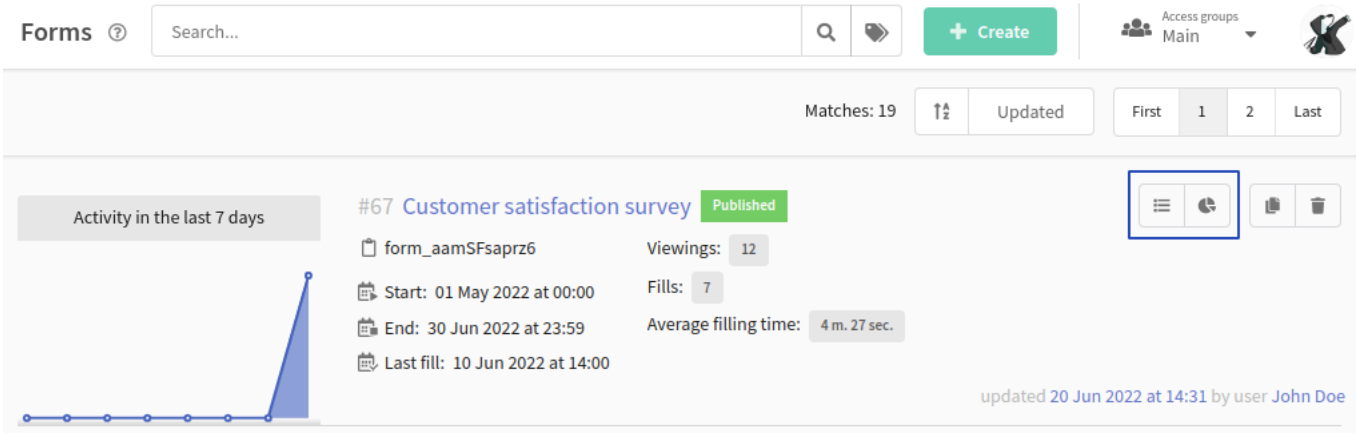
You can add other elements to the form if you wish. Once you have made your changes, save the form and toggle the Published option. The NPS survey is ready.

TIP

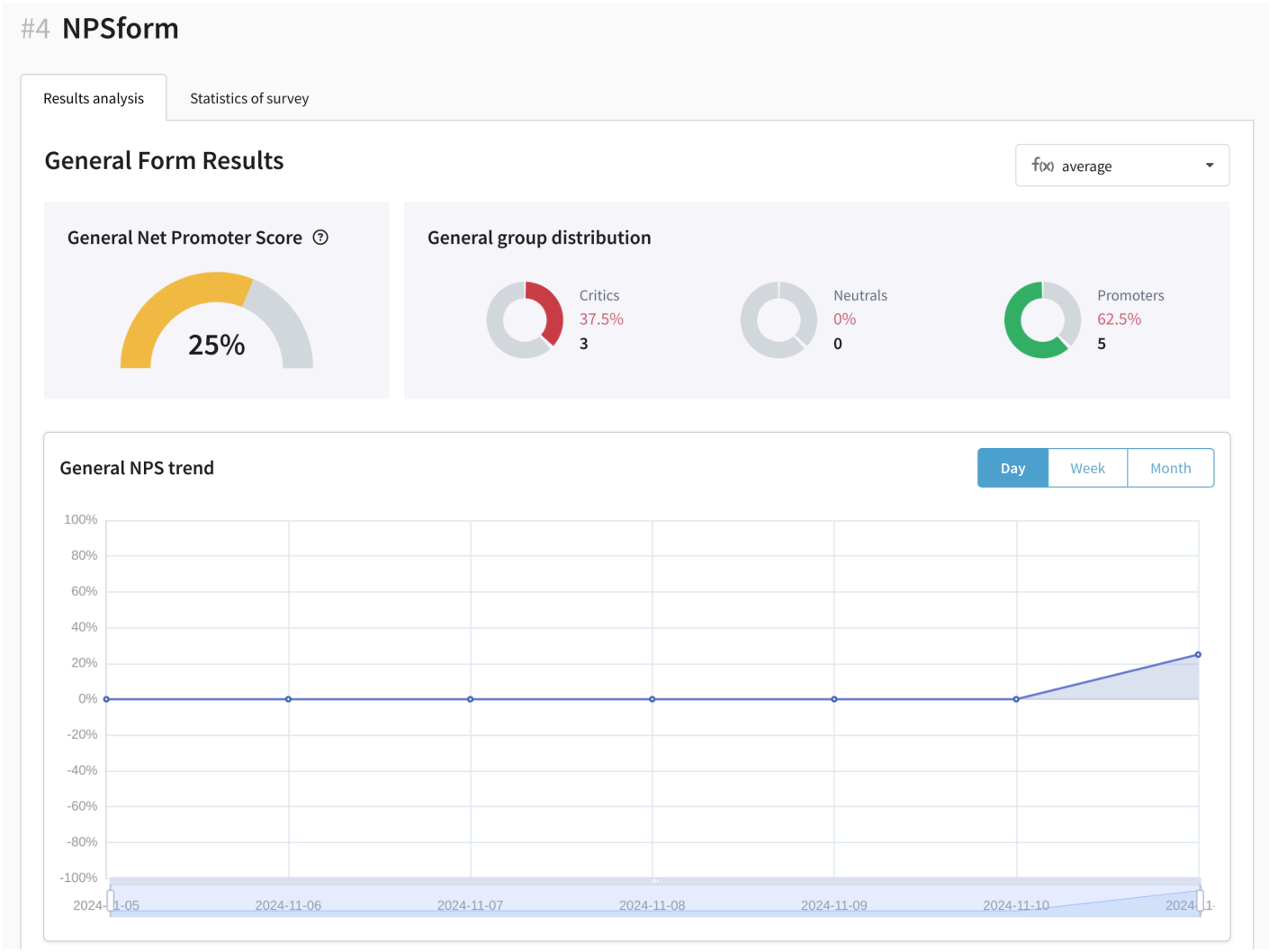
You can read more about creating and customizing a form on this page.

Testing results

You don't need to manually collect information from forms and calculate the NPS index, you can view the test results and their interpretation in statistics of form. To do this, open the Weblayer - Forms section, find the form you need and click on the Insights button in the quick actions menu:



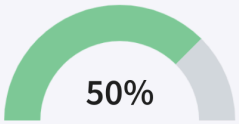
The screen will display the overall form scores - Net Promoter Score, group distribution and NPS dynamics:



You can also see information on every element on each page:

Form Results

Net Promoter Score



Group Distribution

Critics
25%
100

Neutrals
0%
0

Promoters
75%
300

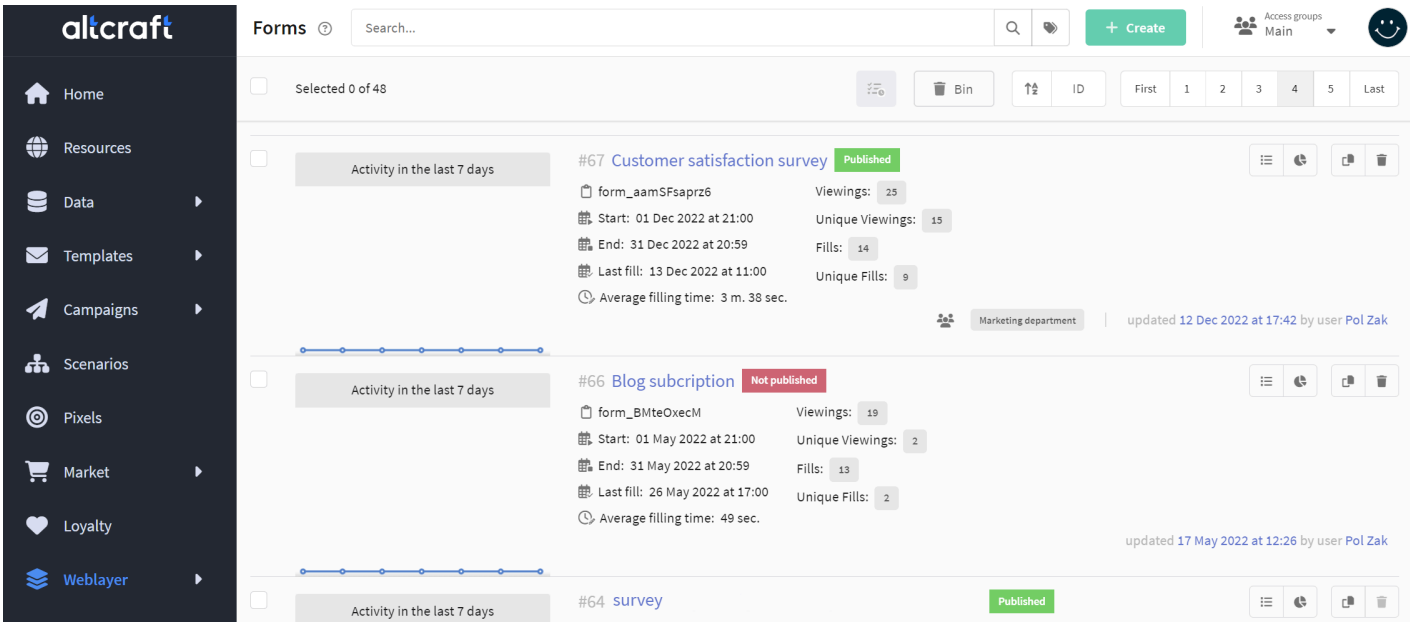
Score distribution

😡	😢	😕	😕	😐	😊
0%	25%	0%	0%	0%	75%
0	100	0	0	0	300

Create a form

Forms allow you to collect information from users: contact details, feedback, requests, event registrations, and much more. Once created, a form can be embedded on a website, sent via email, or used in newsletters.

To create a form, go to Forms in the main menu and click the Create button:

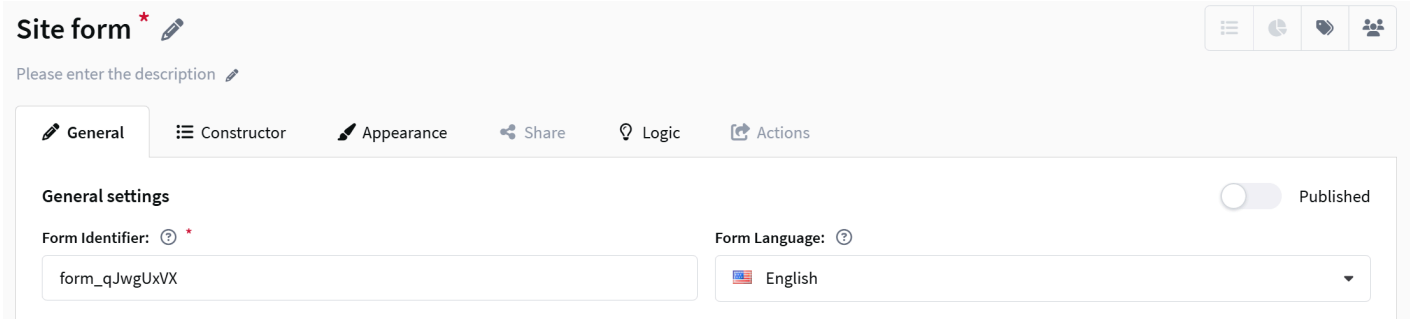


General settings

The General tab contains the main form settings: name, description, publication options, and additional features. Here you set the basic parameters of the form before moving on to its content and design.

Key features:

- **Configure the name, description, and form identifier**
- **Manage publication and scheduling**
- **Configure the final page after form submission**
- **Restrict repeat submissions**
- **Add custom code for customization**

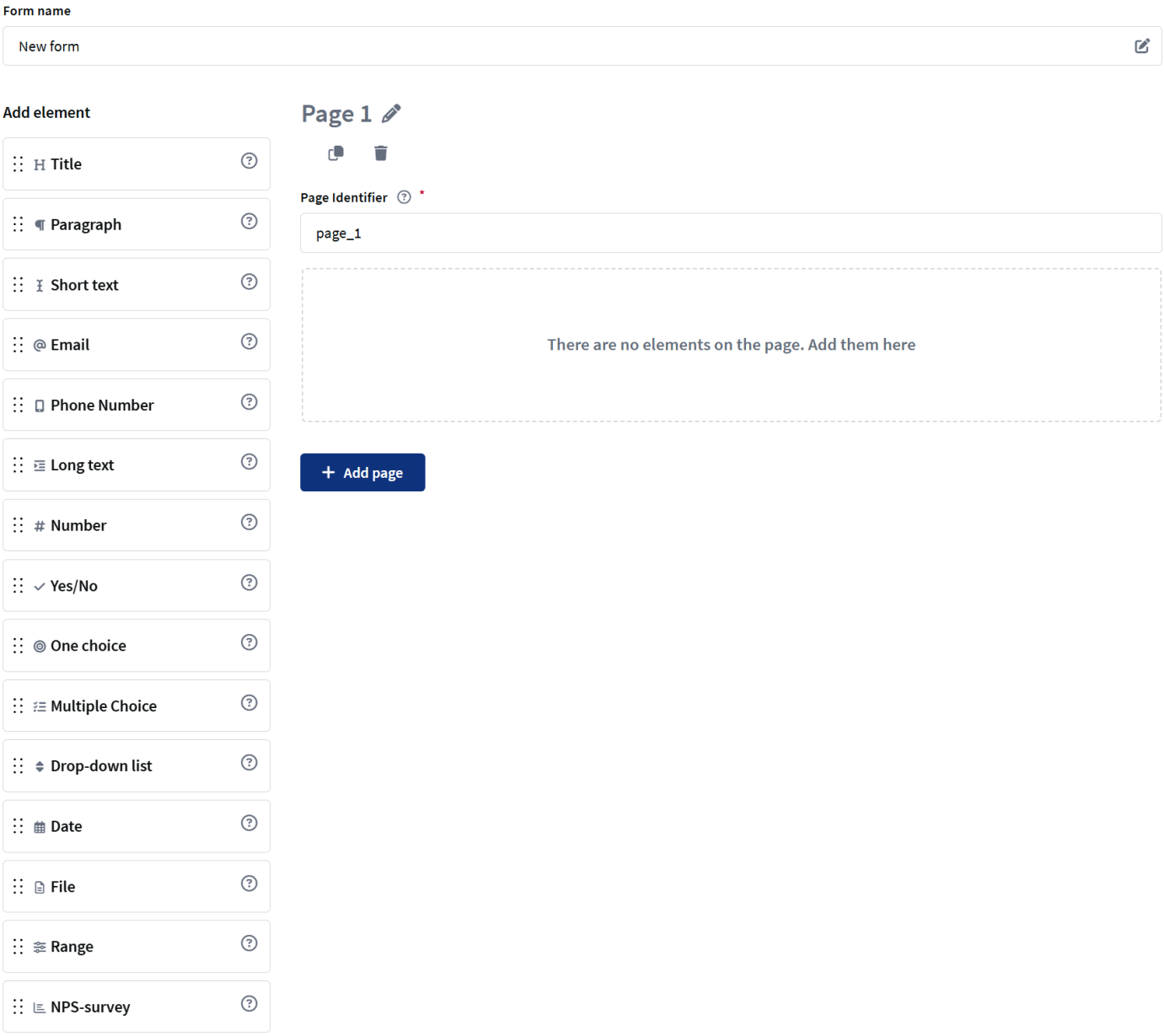


TIP

You can learn more about general settings in this article.

Form constructor

The Constructor tab is used to build the form structure — here you add fields, configure questions and answers, and manage form pages. This is the main tool for creating the form content.



To add a selected field, simply drag it from the left panel. The following field types are available:

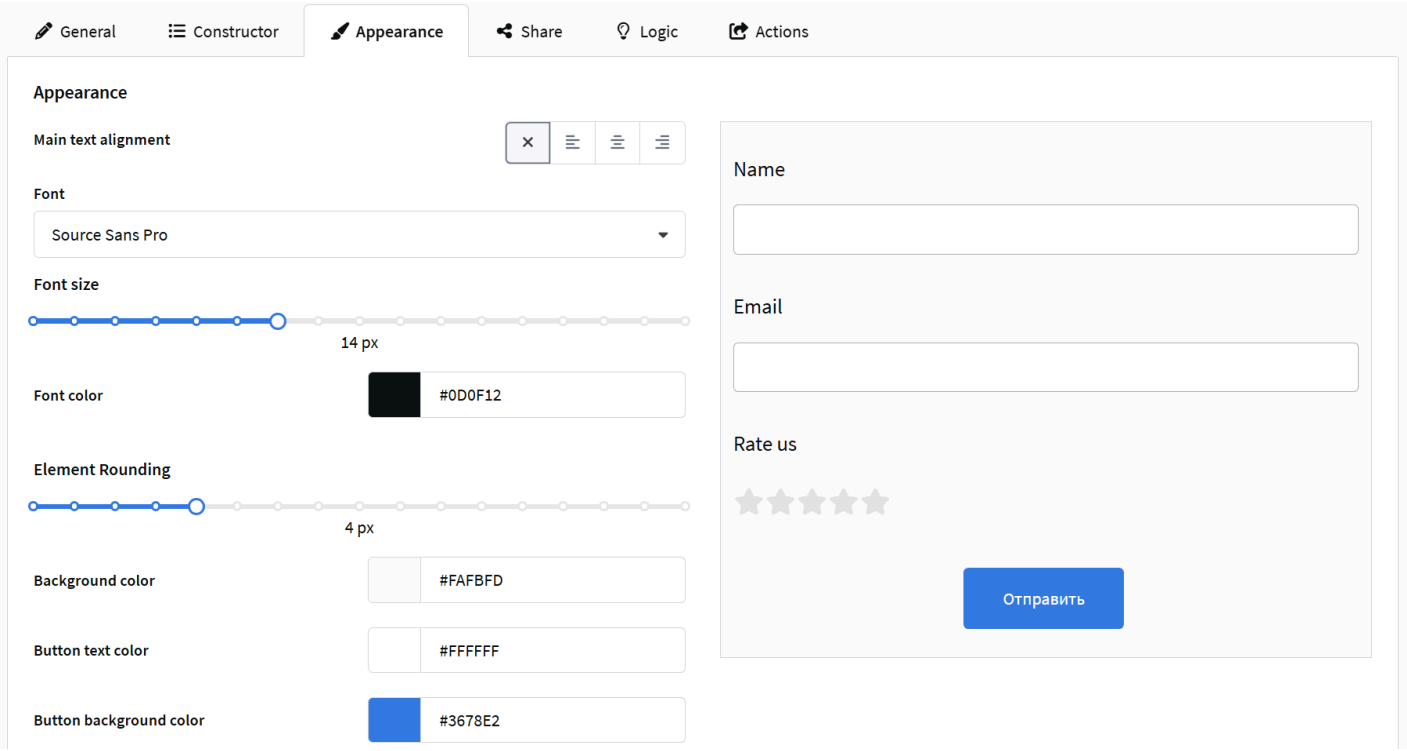
Field Type	Description
Title	**A text header row**
Paragraph	**A text block for information**
Short text	**A field for a short answer**
Email	**An email input field with validation**
Phone number	**A phone number input field**
Long text	**A multi-line text field**
Number	**A field for numeric values**
Yes/No	**A toggle for consent**
One choice	**Radio buttons for selecting a single option**
Multiple choice	**Checkboxes for selecting multiple options**
Drop-down list	**A dropdown list of options**
Date	**A date picker field**
File	**File upload (up to 20 MB)**
Range	**A rating scale (stars, numbers)**
NPS Survey	**A field for NPS testing**

TIP

You can learn more about the form constructor in this article.

Appearance

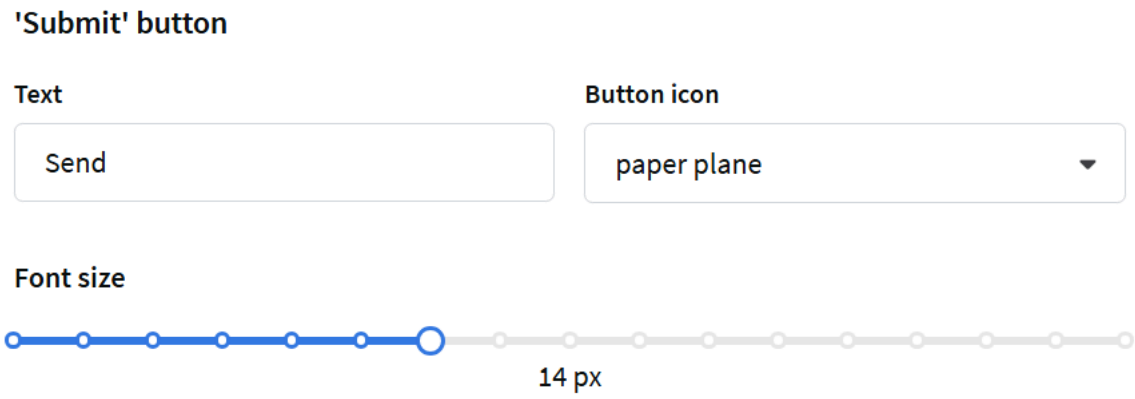
The Appearance tab allows you to customize the form's visual design — colors, fonts, element layout, and button styles. Here you can adapt the form design to match your website or brand style.



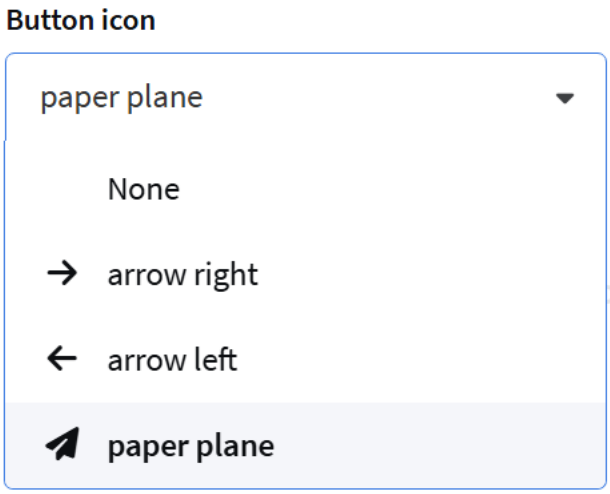
Main settings:

- **Fonts — choose the font style and size**
- **Color scheme — configure text, background, and element colors**
- **Alignment — set the form element layout**

Button styling:



For each button, configure the text, font size, and icon:



TIP

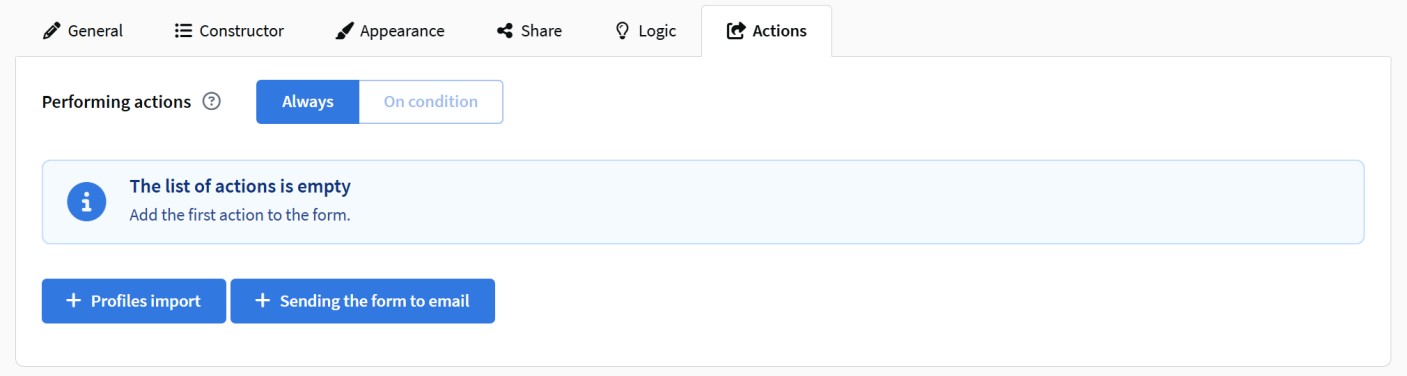
You can learn more about form appearance in this article.

After saving the form

After configuring all the parameters, you need to save the form. Only then will additional tabs for configuring actions and publication become available.

Actions

The Actions tab configures automated processes after form submission — importing data into the database and sending notifications. Here you define what happens to the information after the form is submitted:



- **Profiles import — upload data to the database**
- **Send email notifications — email alerts about form submissions**

TIP

Learn more about the Actions tab.

Share

The Share tab provides access to codes and links for publishing the form on your website. Here you get ready-made options for embedding the form:

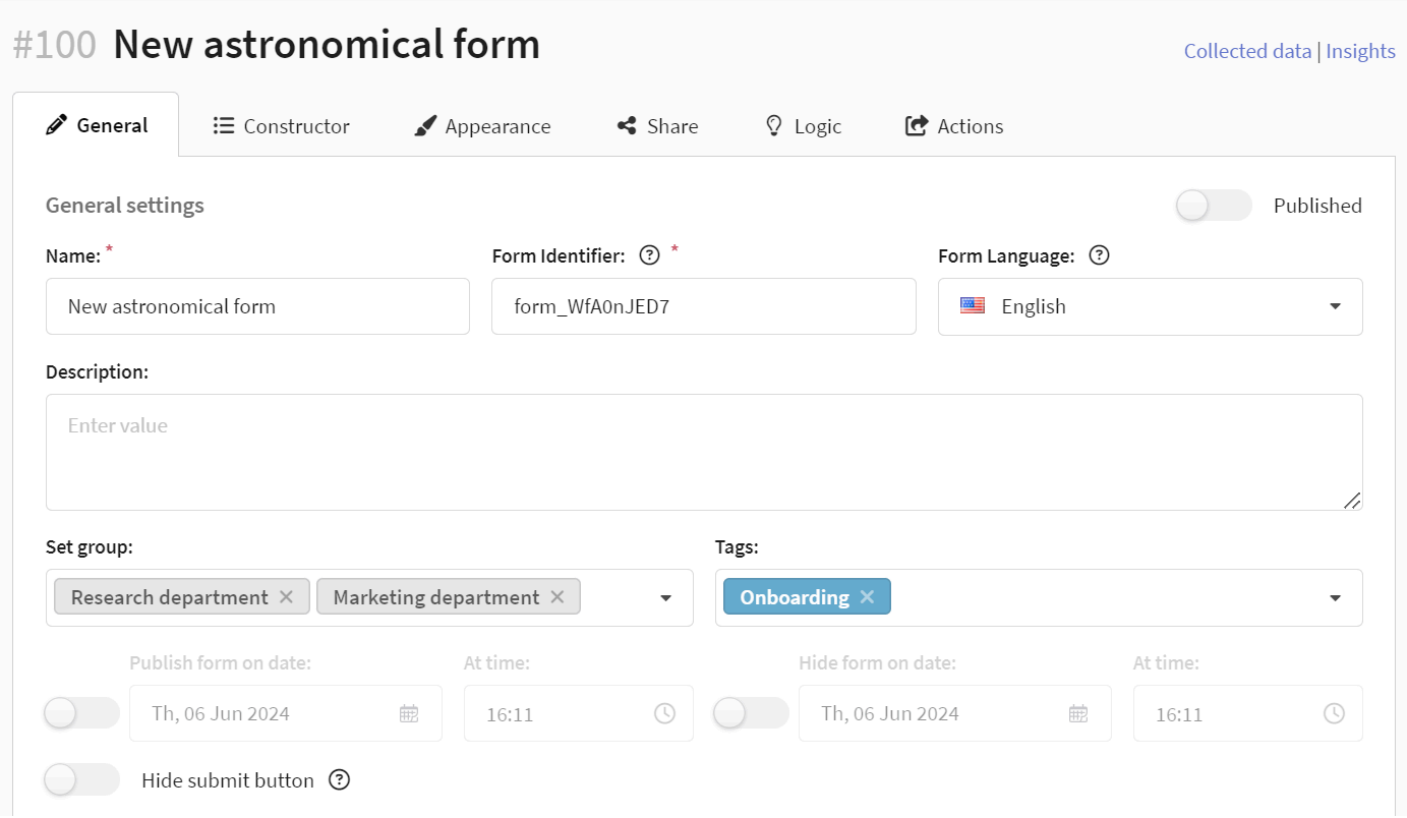
- **URL — a direct link to the form**
- **HTML code — code for embedding on the website**
- **Embedding parameters — as an iframe or popup window**

TIP

Detailed information can be found in this article.

General settings

On the General tab, you can configure the basic form parameters: name, description, publication settings, final page, and additional options.



Basic form settings

- **Name — form name for identification in the system**
- **Description — additional information about the form**
- **Identifier — unique form ID**
- **Language — form interface language**
- **Access group — access rights restrictions for the form**
- **Tags — labels for convenient form organization**

Form publication

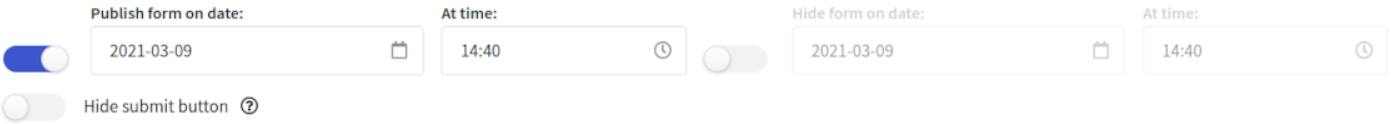
To make the form available via URL or as an iframe element, activate the Published switch in the upper right corner.

CAUTION

While the form is published, it cannot be deleted.

Publication scheduling

Set the start and end date and time for publishing the form for temporary access:



The form will be available for filling out on the specified date and time and will be automatically unpublished after the period ends. The Published switch should remain inactive — the system will update its status automatically.

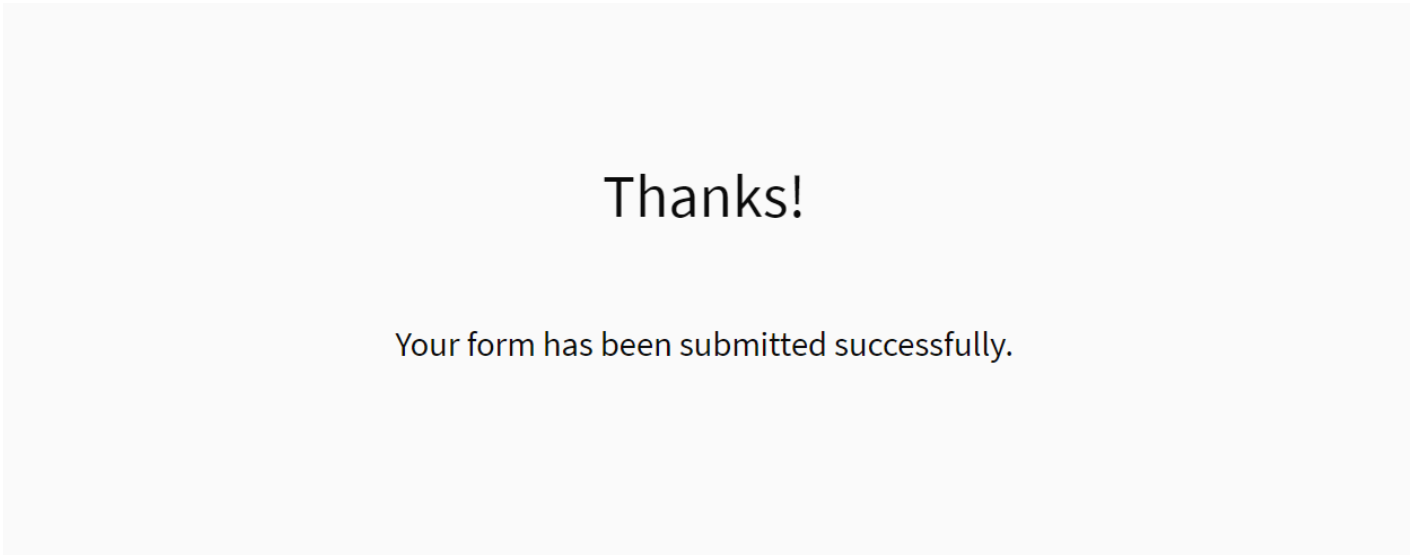
INFO

The system uses a cache to store forms. Changes in form publication may take up to 3 minutes to take effect.

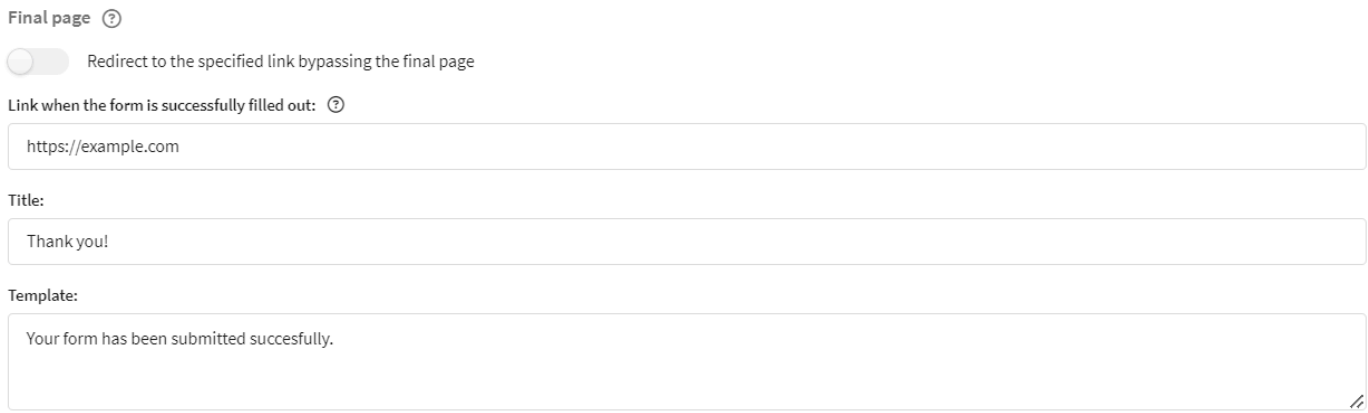
When the form is not published, users trying to access it will see the message Form is unavailable.

Final page

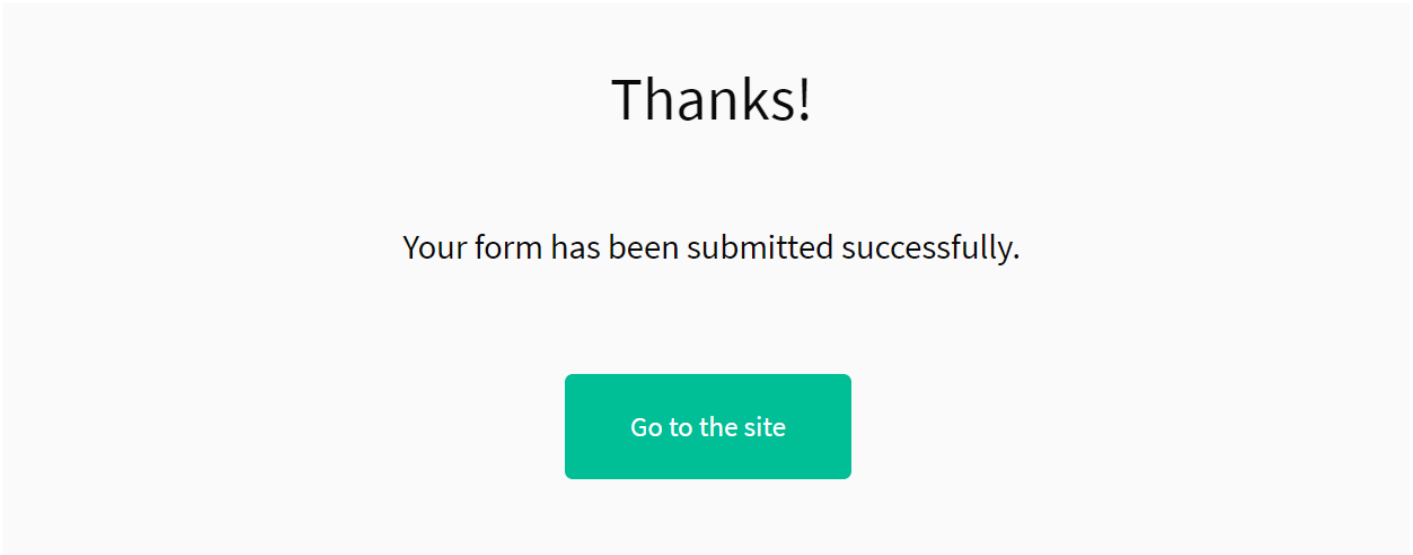
Configure the page that users will see after submitting the form:



Change the Title and Message on the final page:



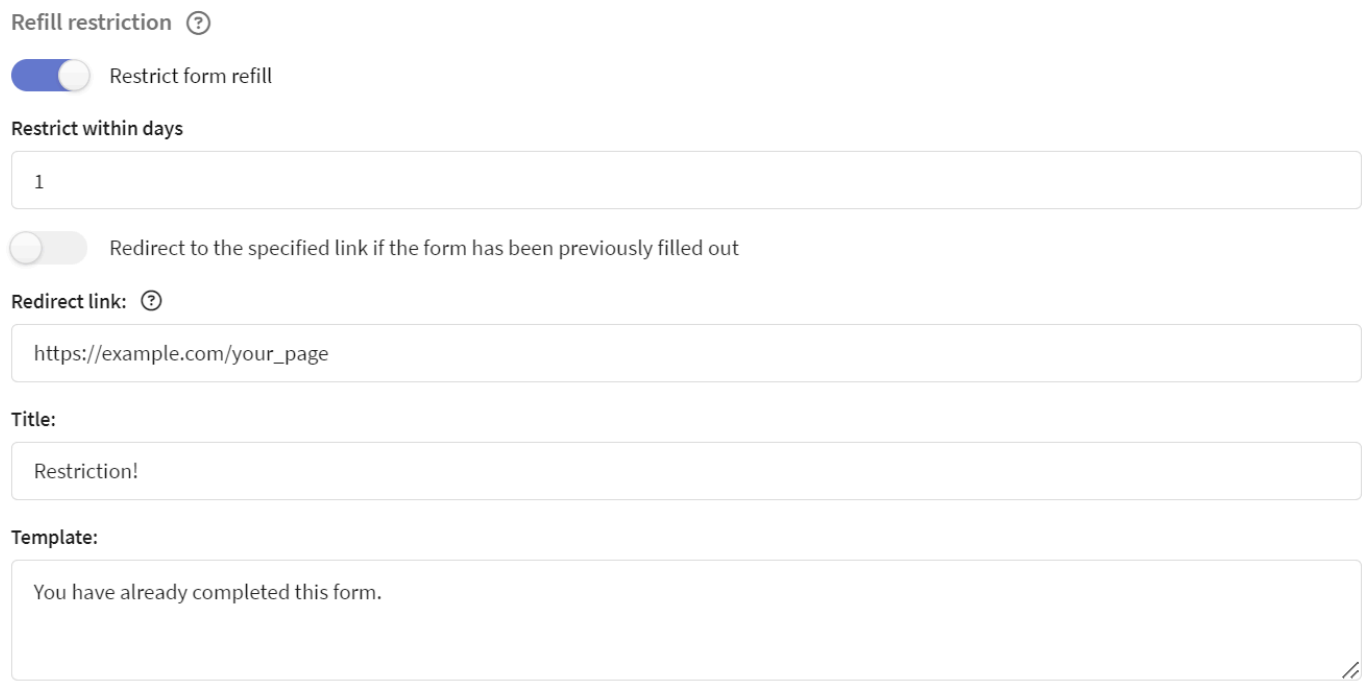
The link when the form is successfully filled out is displayed as a Go to the site button. The button text can be changed in the Appearance tab:



Activate the Redirect to the specified link bypassing the final message switch to redirect the user immediately after form submission.

Refill restriction

Protect the form from multiple submissions:



Activate the switch and set the number of days to restrict refilling.

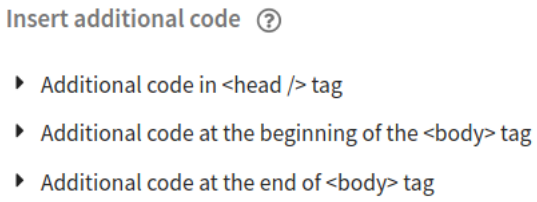
The system records form submission data in cookie files. When attempting to reopen the form, the user will see the message: Restriction! You have already completed this form. The message text can be changed in the Title and Message fields.

Additional options:

- **Specify a redirect link when attempting to resubmit**
- **Activate Redirect to the specified link if the form has been previously filled out for automatic redirection**

Form customization

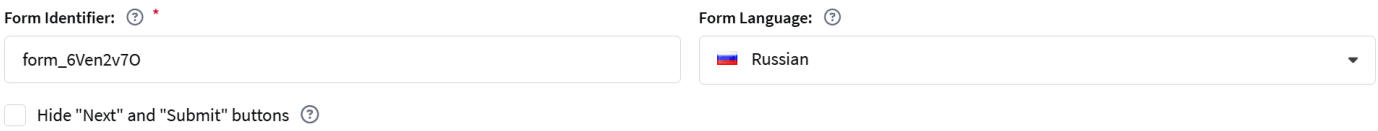
Add custom code for advanced customization:



- **In the `<head>` tag — for connecting styles and scripts**
- **After the opening `<body>` tag — for additional elements**
- **Before the closing `</body>` tag — for analytics scripts**

Additional options

Hide submit button — activate this option to prevent showing the “Submit” and “Next” buttons to the user until they have filled in all required fields on the page:

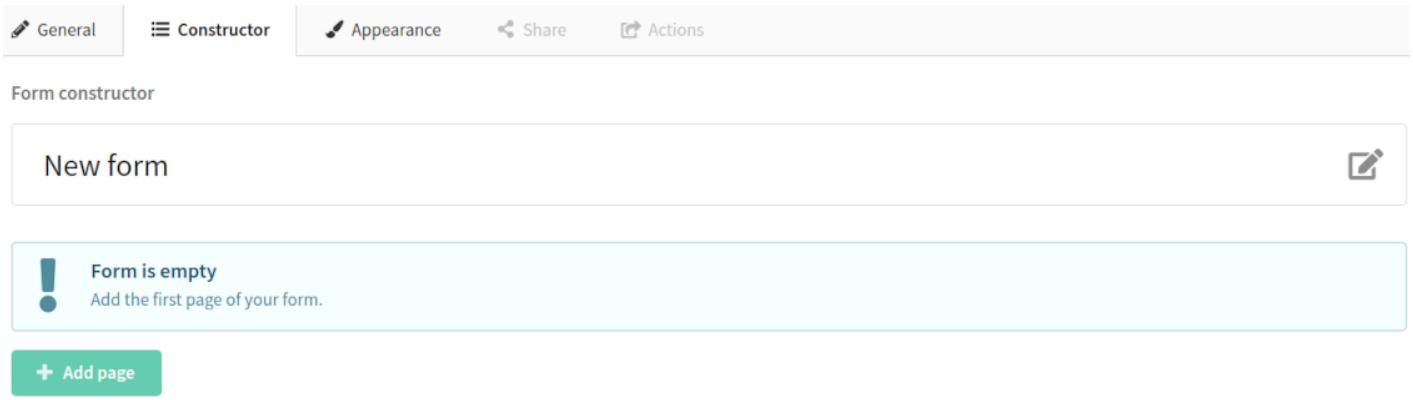


TIP

In the on-premise version of the platform, administrators can customize the appearance and behavior of forms (error page, inactive form page, etc.) by modifying template files. More details in admin documentation.

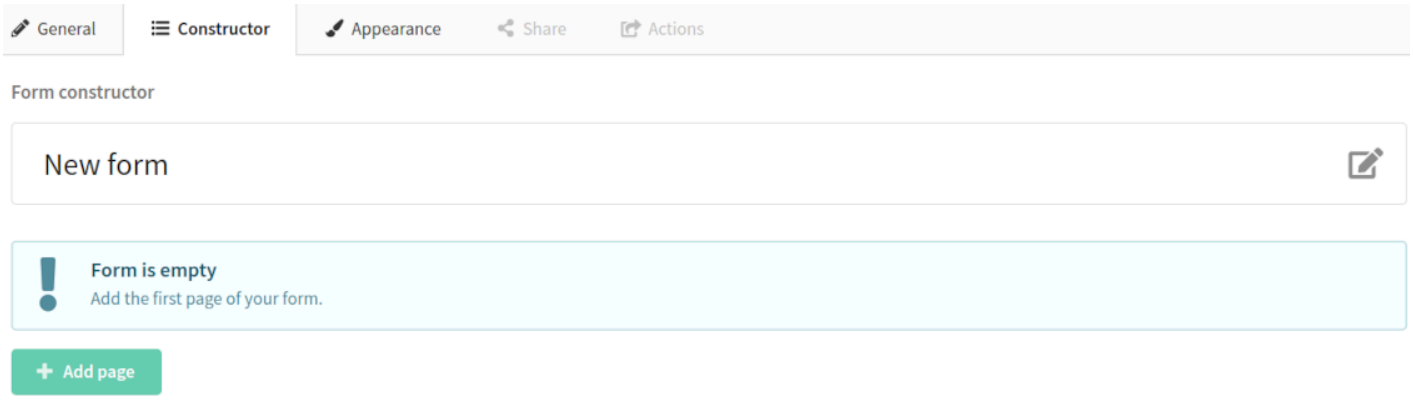
Form constructor

The second tab Constructor is used to configure the form structure and its fields. Here you can set up form pages, add and configure fields — define their type, required status, visibility, and display logic:



- **The right side of the screen displays the form pages and their elements.**
- **The left column shows the element types that can be added to the form.**

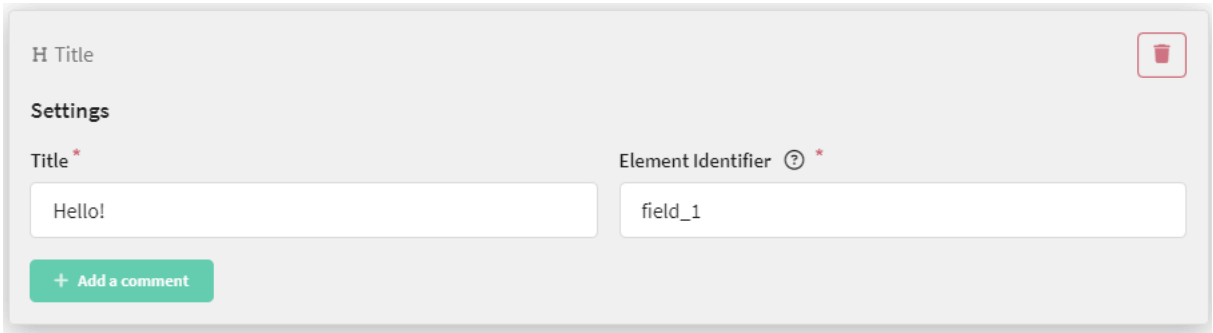
Working in the constructor starts with creating a page — click Add page:



Elements and their description

Title

Title is a text element where you can specify the main topic of your form.



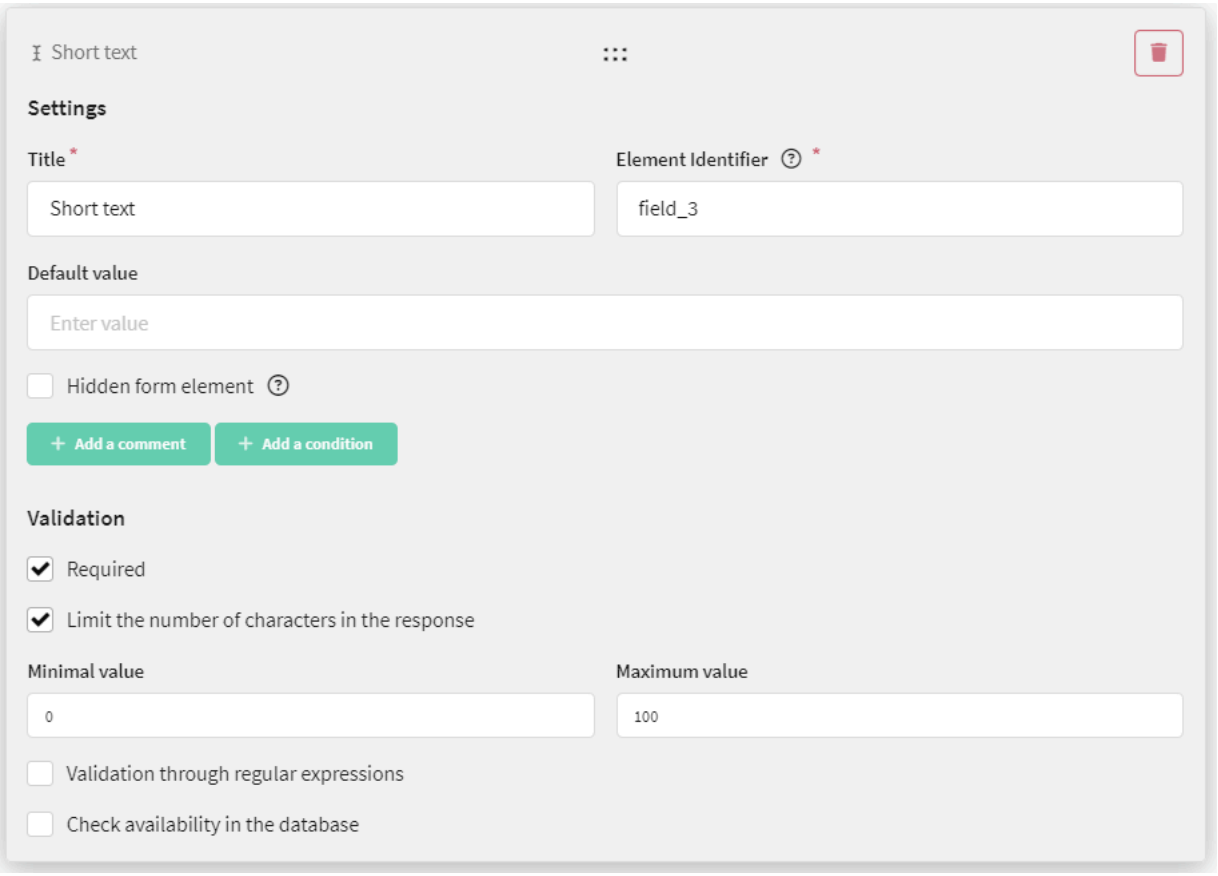
Paragraph

Paragraph is a text element. It can contain information that should be displayed in the form.

Short text

Short text is a text field where you can ask a question with a short answer. For example, the client's first or last name.

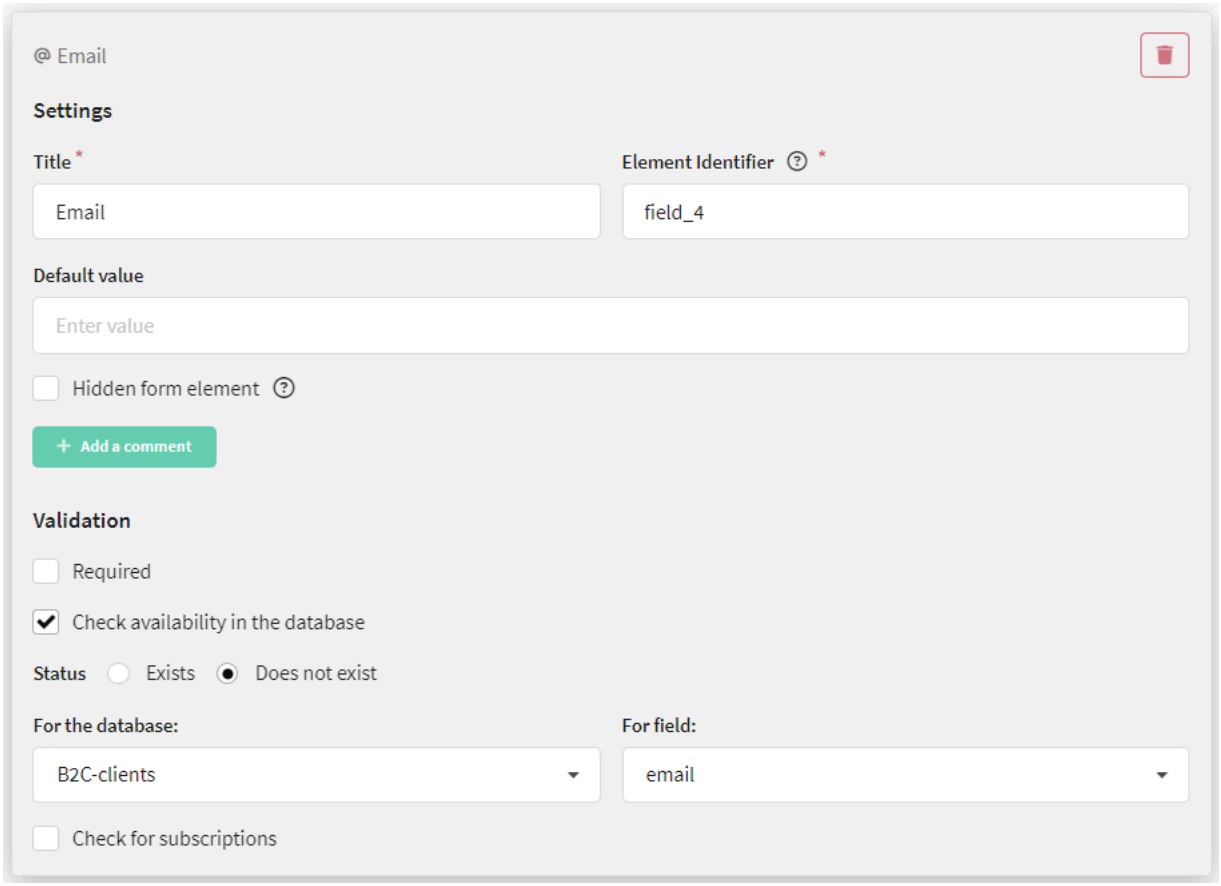
For this field, you can set a title, a default value, and configure element visibility and data validation.



Email

Email is a field where you need to specify an email address.

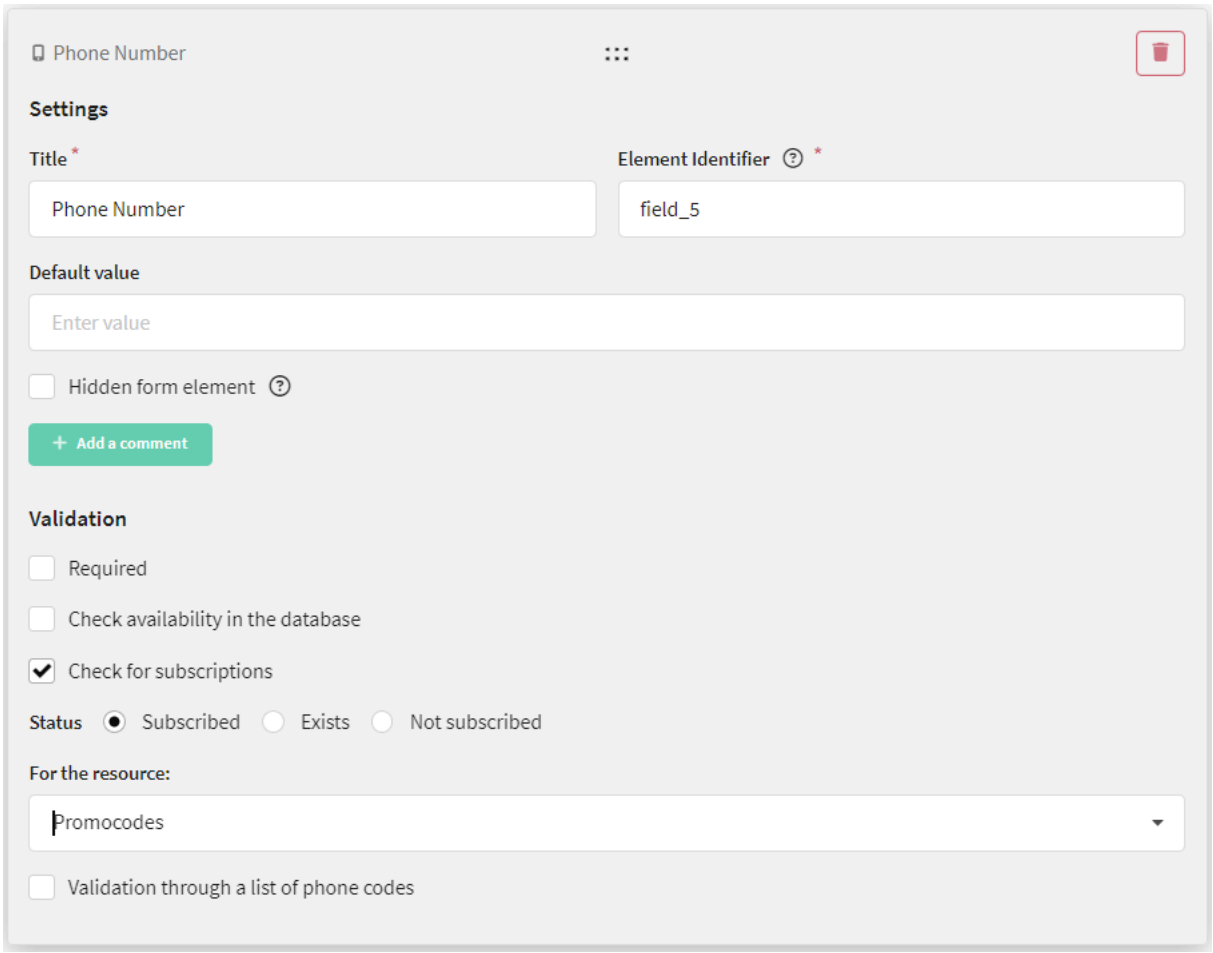
For this field, you can set a title, a default value, and configure element visibility and data validation.



Phone number

Phone number is a field where you can specify a phone number.

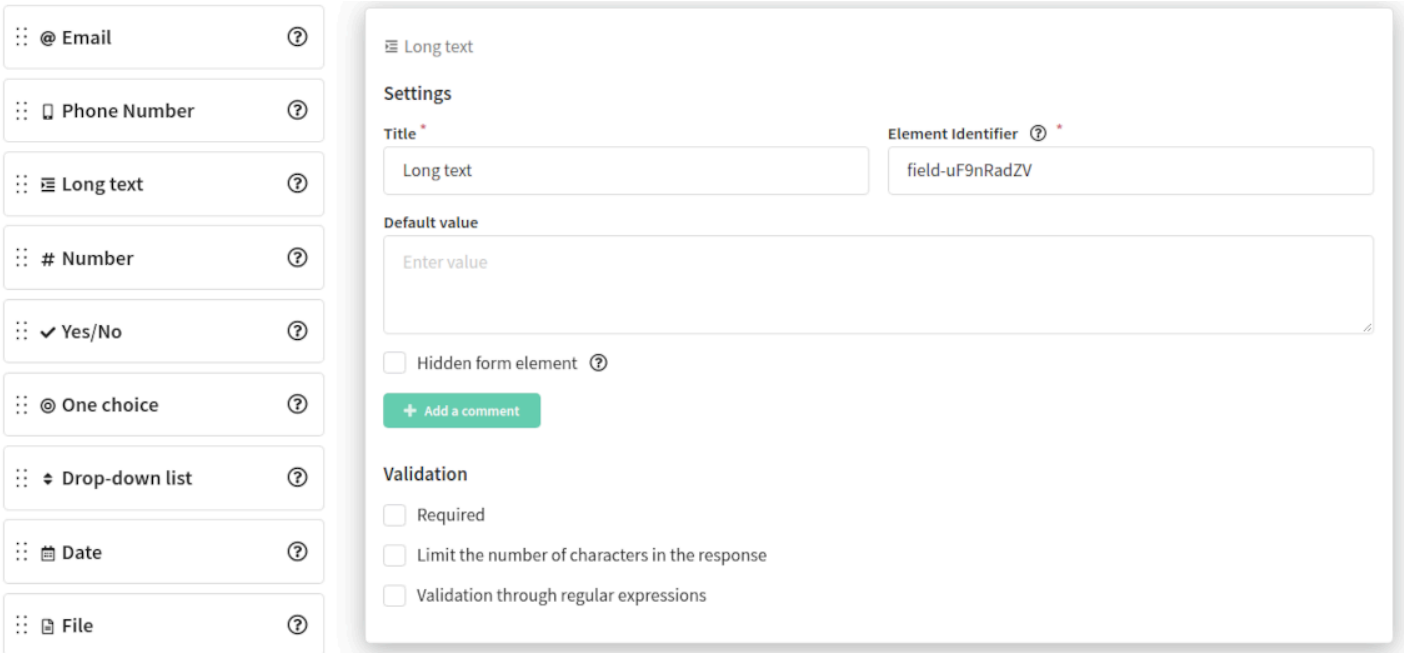
For this field, you can set a title, a default value, and configure element visibility and data validation.



Long text

Long text is a text field where you can ask a question with a long answer, for example, feedback about a product or event.

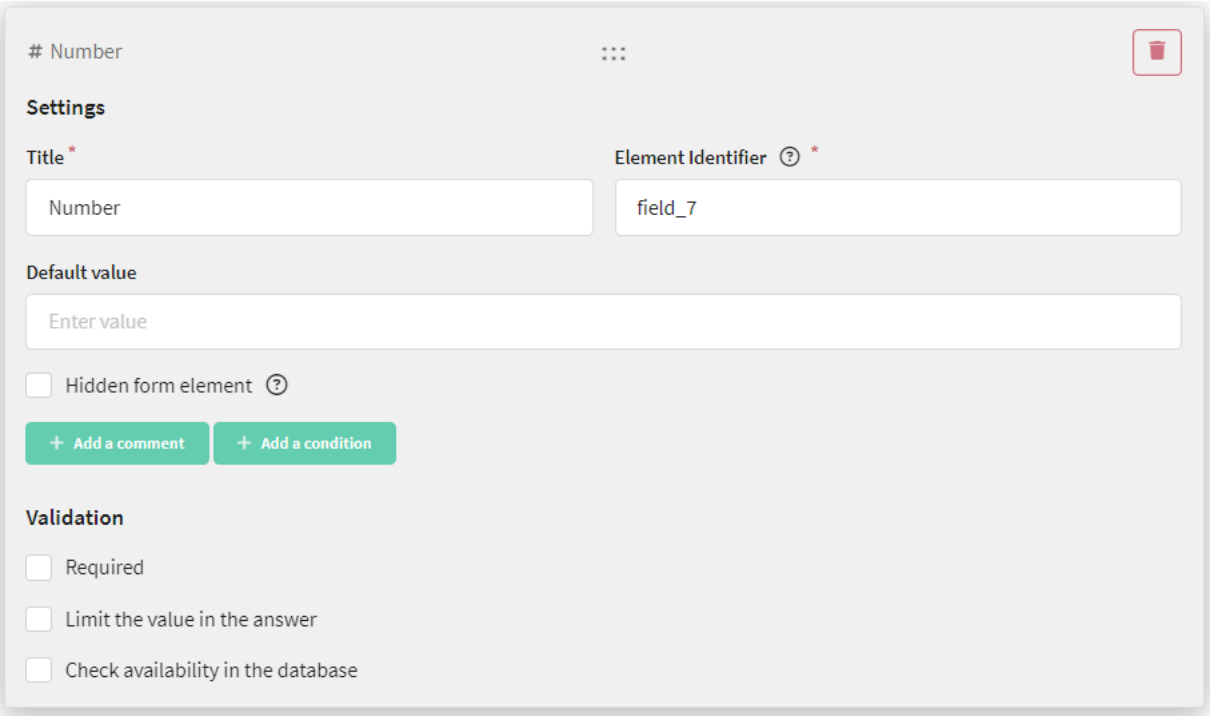
For this field, you can set a title, a default value, and configure element visibility and data validation.



Number

Number is a field where you can ask a question with an answer in the form of an integer or fractional number, for example, age.

For this field, you can set a title, a default value, and configure element visibility and data validation.

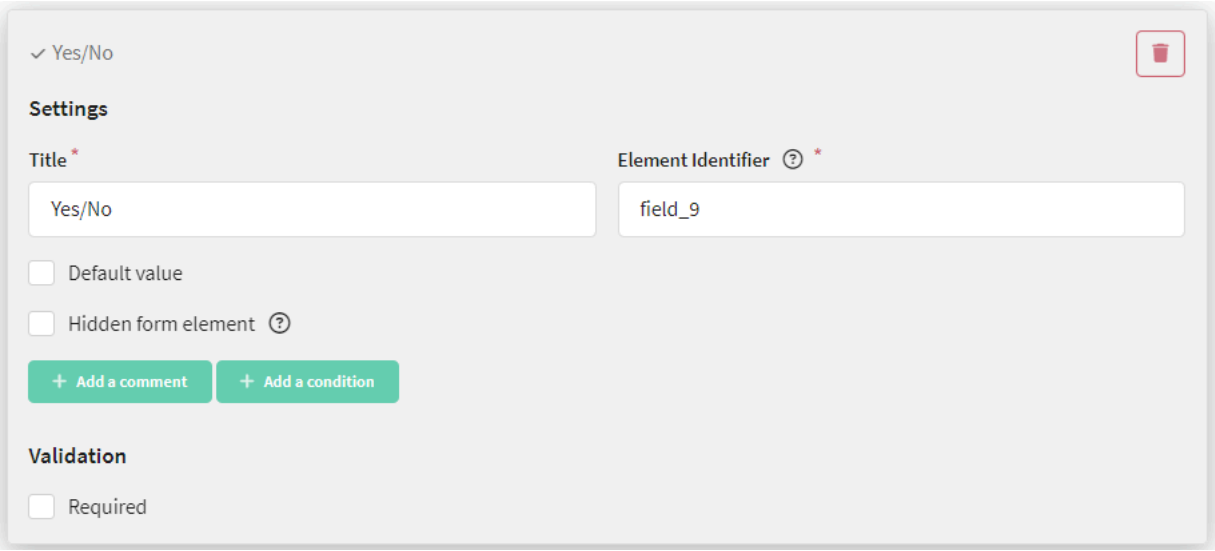


Yes/No

Yes/No is a field where you can ask a question to obtain consent, for example, to receive email newsletters.

For this field, you can set a title, a default value, and configure element visibility.

If you check the Default value field, the answer “yes” will be selected by default in the form.



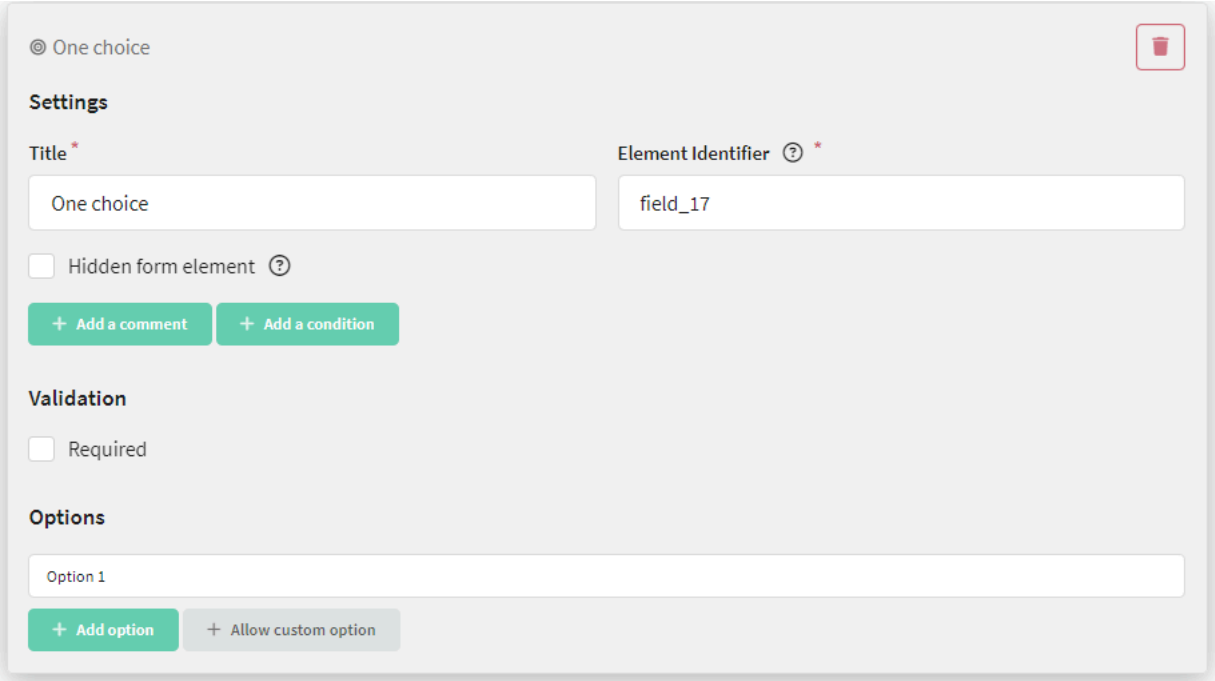
One choice

One choice is a field where you can offer to choose one of several answer options. For example, product color — red or blue.

For this field, you can set a title, answer options, and configure element visibility and data validation.

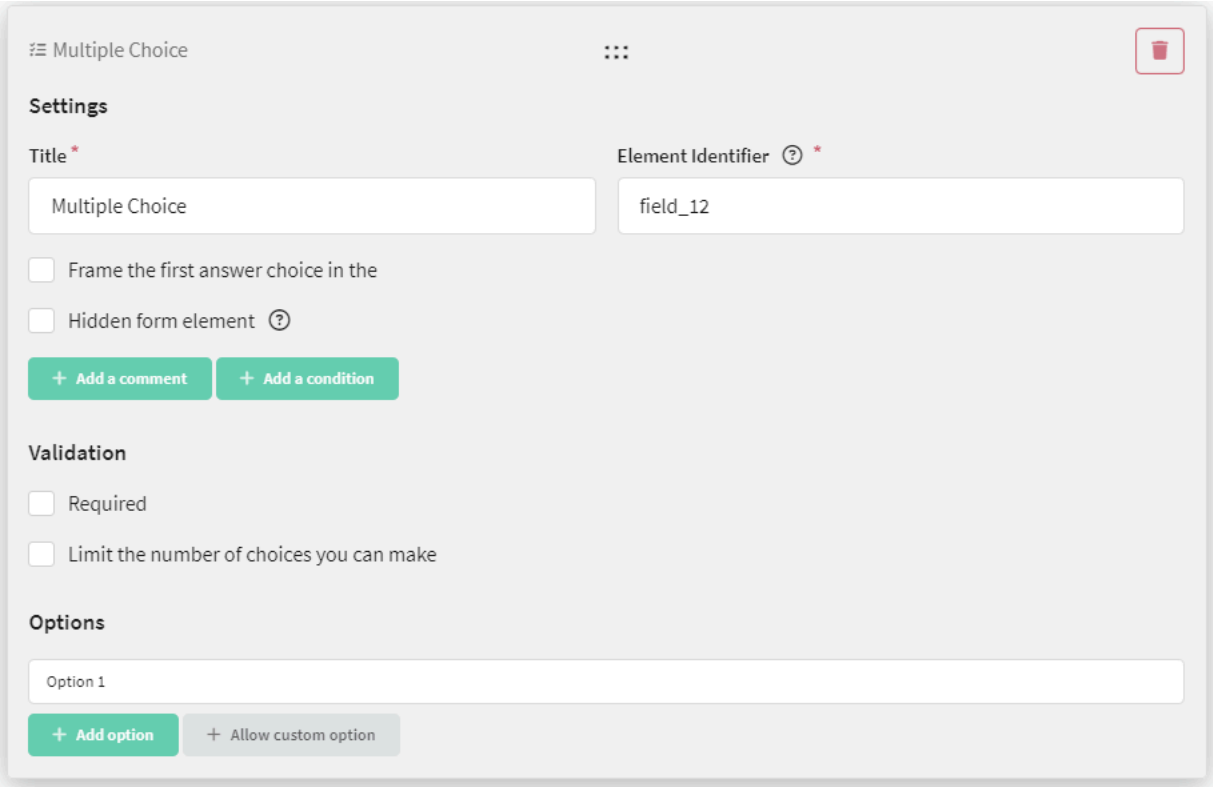
Click the Add option button and enter a value. The number of options is unlimited.

You can allow the user to enter their own answer by clicking the Allow custom option button.

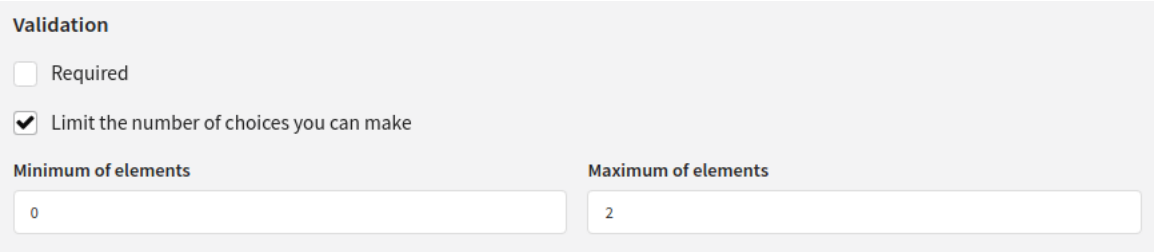


Multiple choice

This element allows you to select several answer options from the proposed ones. Use the Add option button to add possible answers.



The element can be made required, the number of options to select can be limited, and you can also frame the first answer choice in the field:



Drop-down list

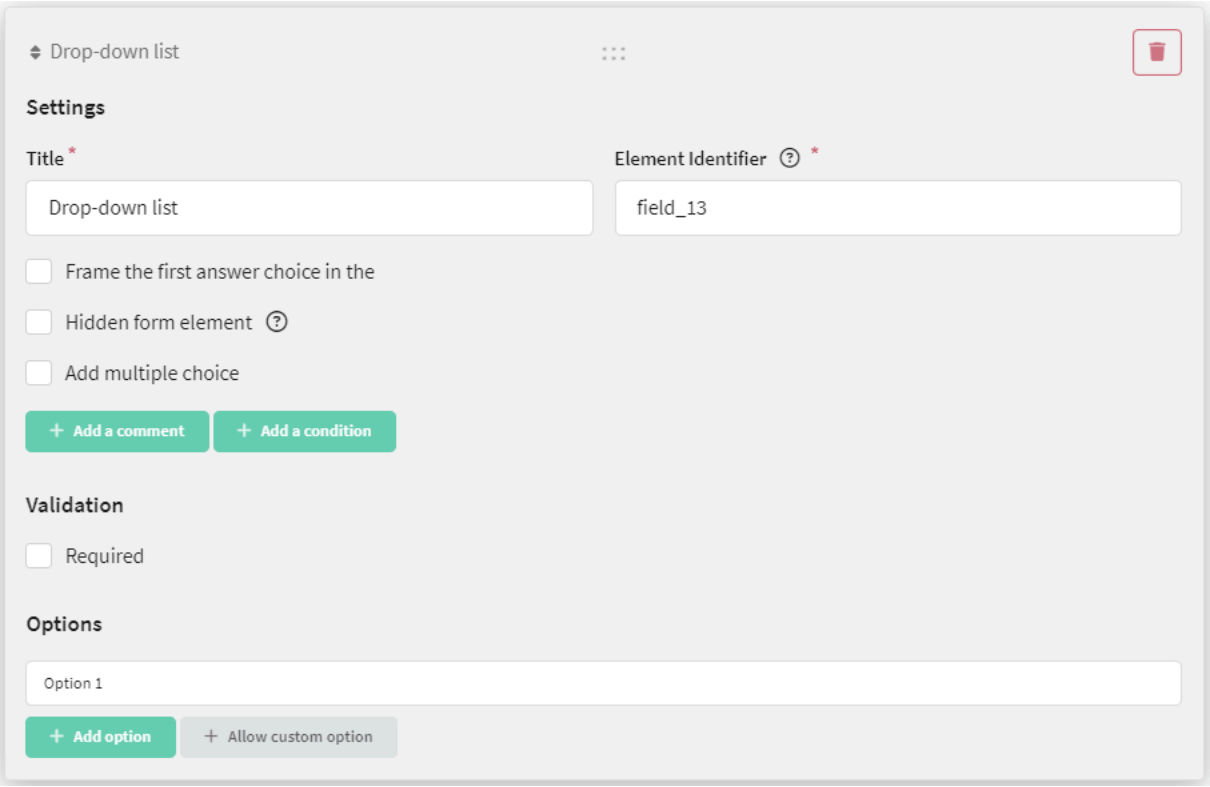
Drop-down list is a field where the user needs to select one option from a dropdown list.

For this field, you need to set a title and answer options. Click the Add option button and enter a value. The number of options is unlimited.

You can allow the user to enter their own answer by clicking the Allow custom option button. When this option is selected, an additional field will open for text input.

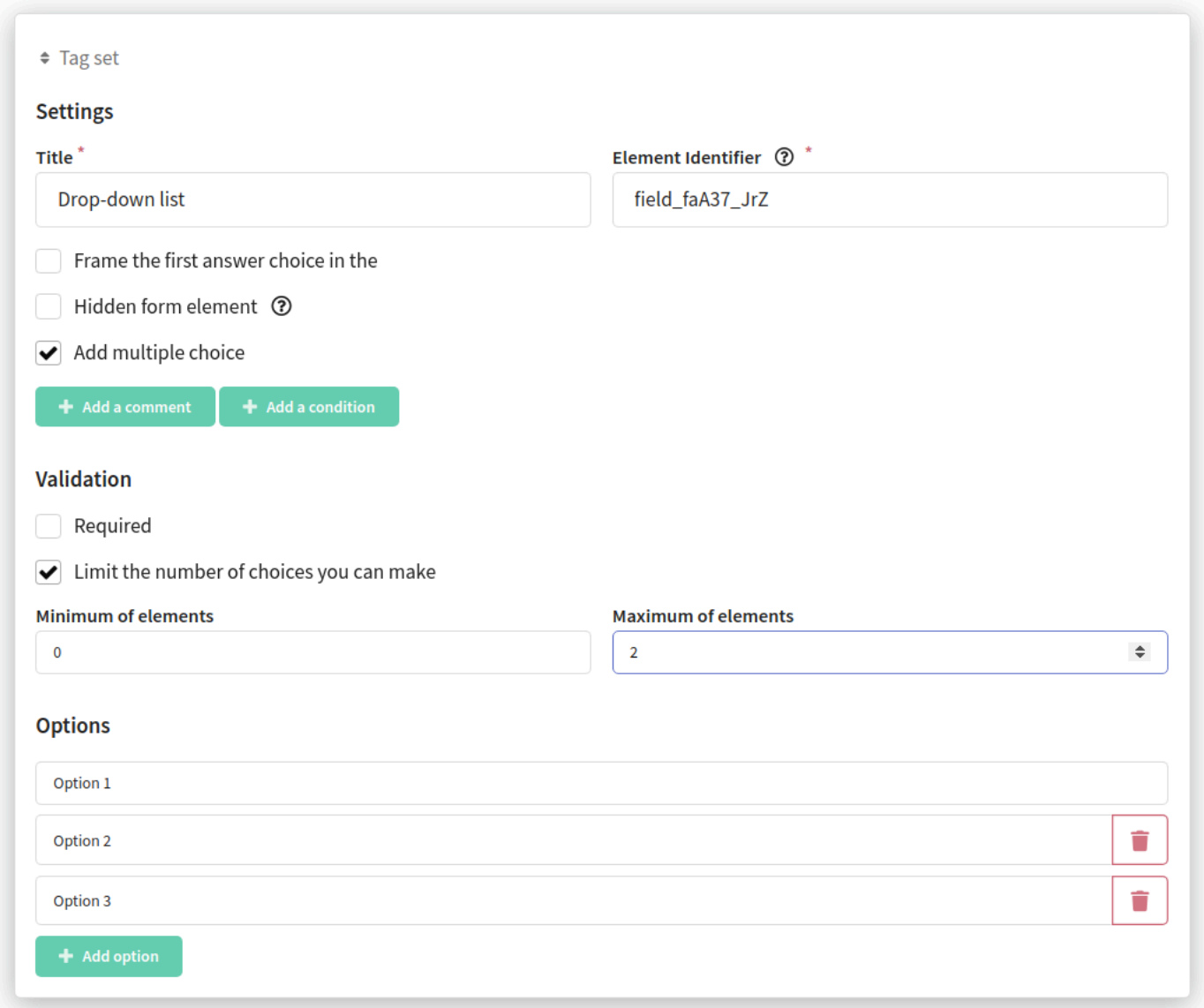
In addition, you can frame the first answer choice in the field by clicking the corresponding button.

You can also configure element visibility and data validation.



If you want to allow the user to select multiple answer options, click Add multiple choice.

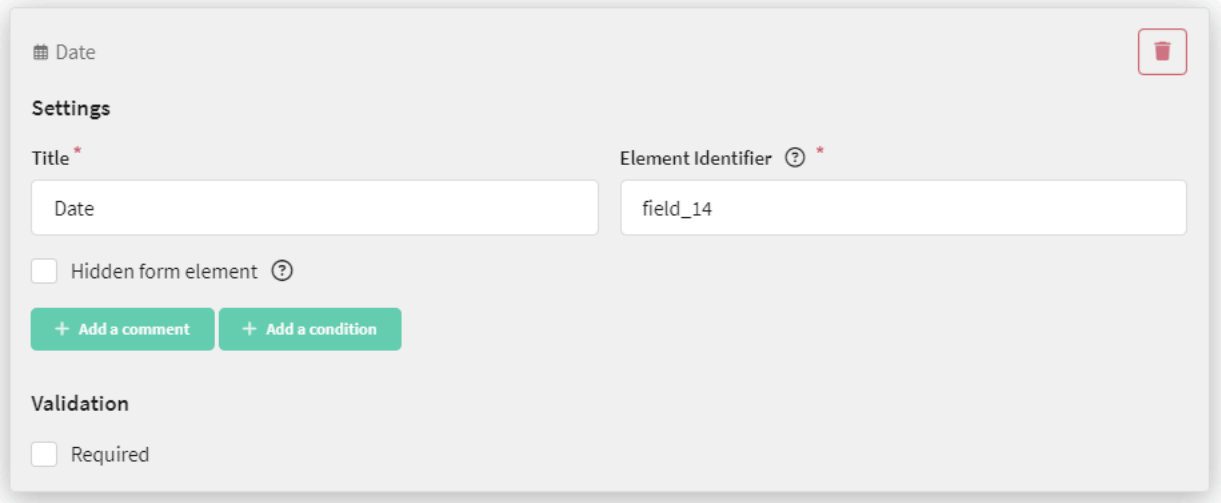
Note that when multiple choices are enabled, a new validation type appears — Limit the number of choices you can make.



Date

Date is a field where you are asked to specify a date, for example, the client's birthday.

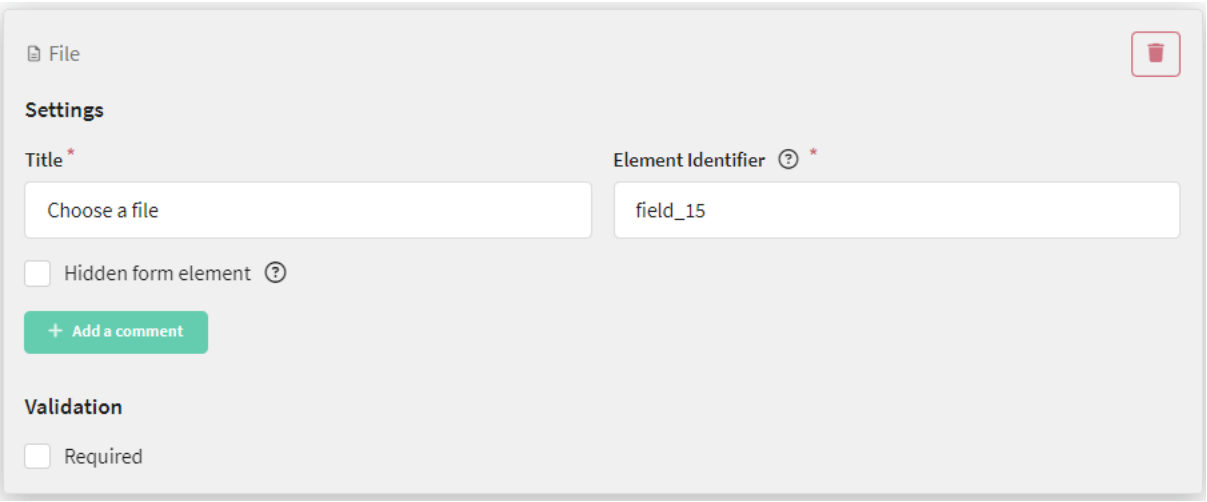
For this field, you can set a title, and configure element visibility and data validation.



File

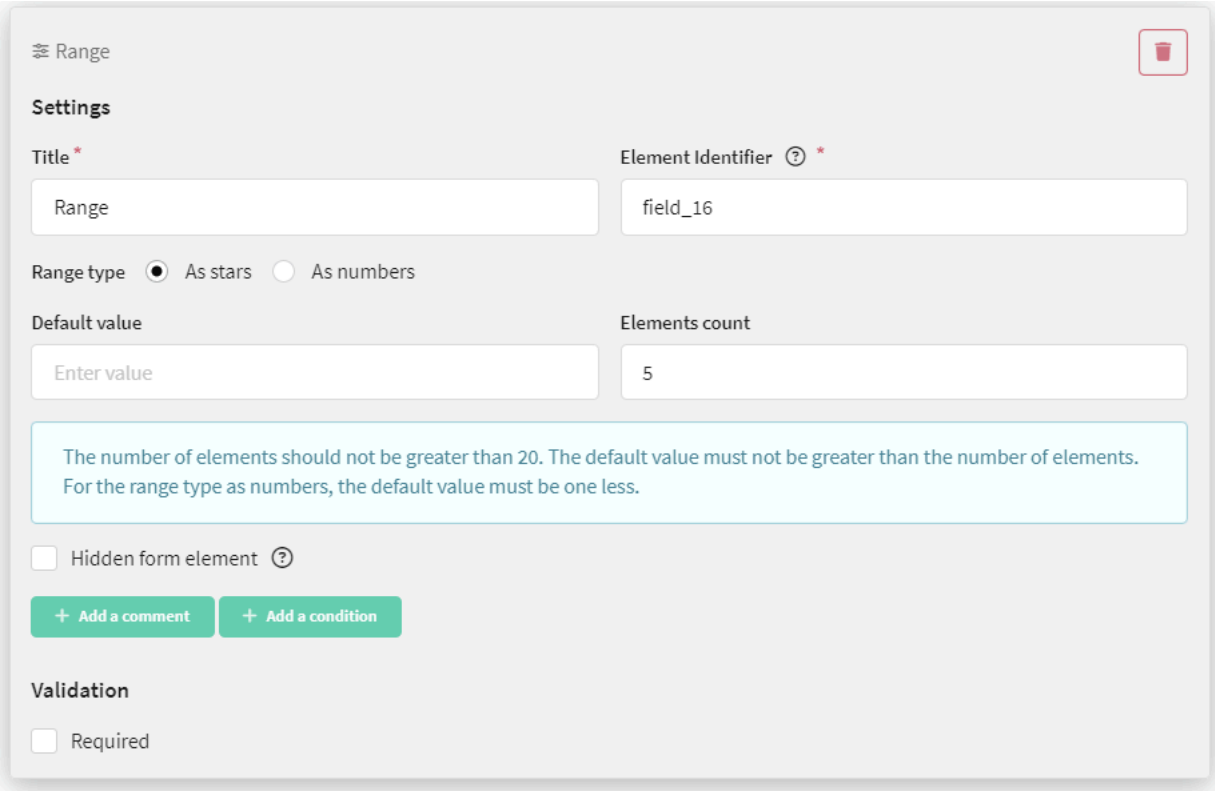
File is a field where files can be attached to the form, for example, a photo. The maximum file size is 20 MB.

For this field, you can set a title, and configure element visibility and data validation.



Range

Range is a rating selection scale. For this field, you can set a title, and configure element visibility and data validation.

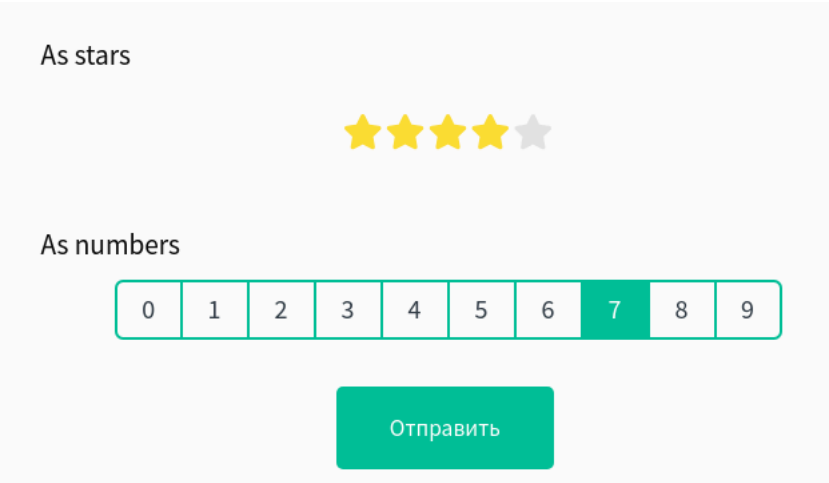


Additionally, you can choose the range type: as stars or as numbers.

You can also specify the number of elements (for example, 5 stars or 10 numbers).

TIP

If you choose numbers as the range type, note that the count starts from 0.

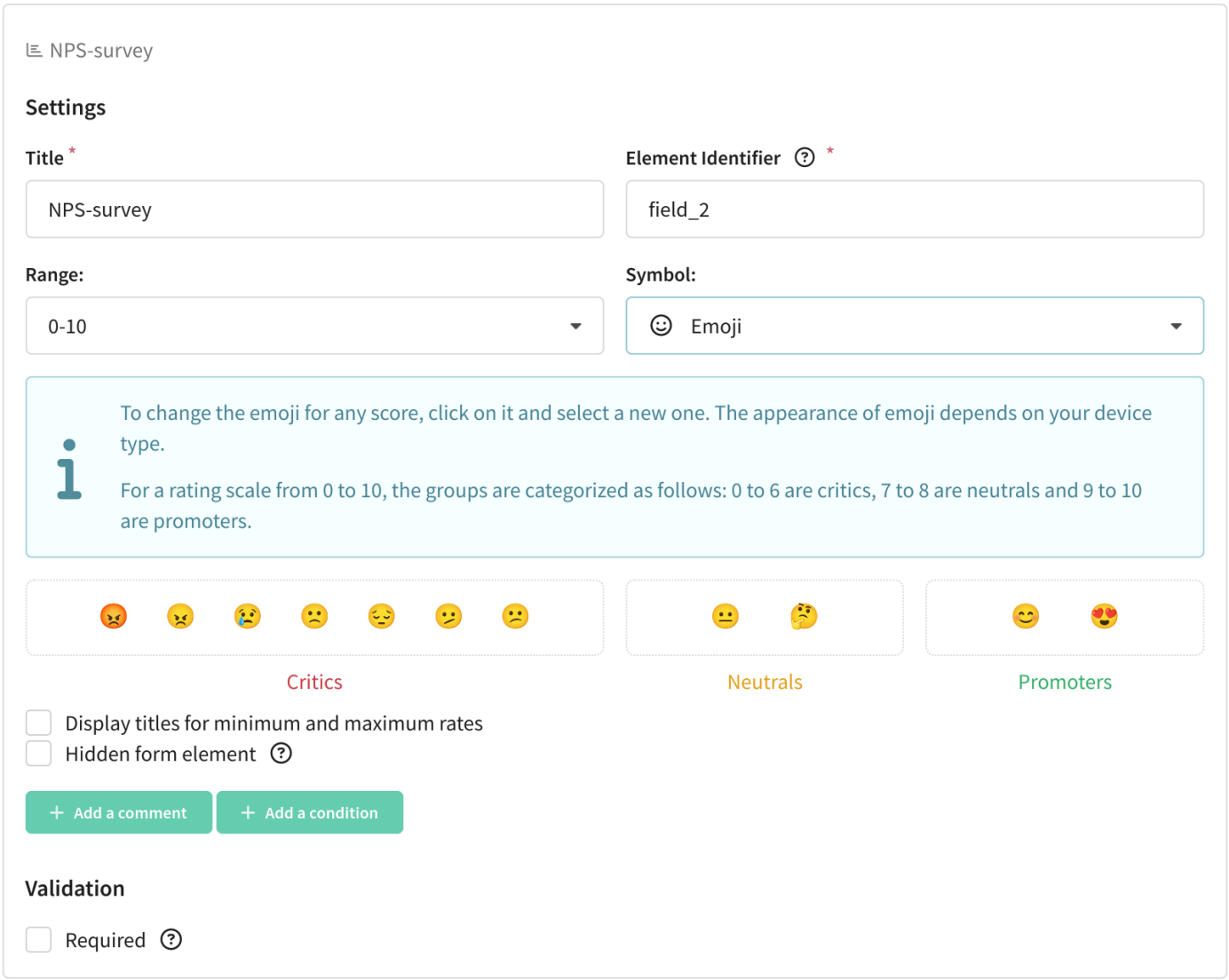


NPS Survey

NPS Survey is a test to obtain the Net Promoter Score. For this field, you can set a title, configure element visibility and data validation, as well as set labels for the maximum and minimum scores.

TIP

You can learn more about what NPS testing is and how to create it in this article.

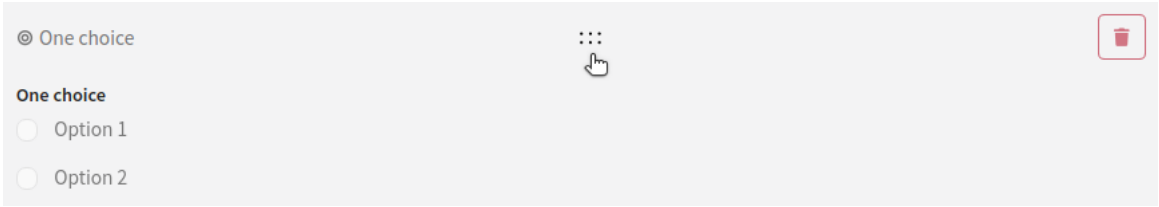


You can select the symbol type: as stars, hearts, emoji, or as numbers.

For the scale, you can also specify the size — 0 to 5, 0 to 10, 1 to 5, and 1 to 10.

TIP

You can drag and drop form elements, easily arranging them in the order you want. When you hover over an element, a cluster of dots appears at the top. Hover over them, hold down the left mouse button, and drag the element wherever you wish.



Form field validation

Most fields have one or more validation types:

Validation	Form field
Required	**Short text, email, phone number, long text, number, one choice, drop-down list, date, file, range**
Limit the number of characters in the response	**Short text, long text**
Validation through regular expressions	**Short text, long text**
Check the availability in the database	**Short text, email, phone number, number**
Check for subscriptions	**Email, phone number**
Validation through a list of telephone codes **(list the telephone codes allowed in the form using the + symbol)**	**Phone number**
Limit the value in the answer	**Number**

Validation	Form field
Limit the number of choices you can make **(specify the minimum and maximum number of options the user can select)**	**Drop-down list**

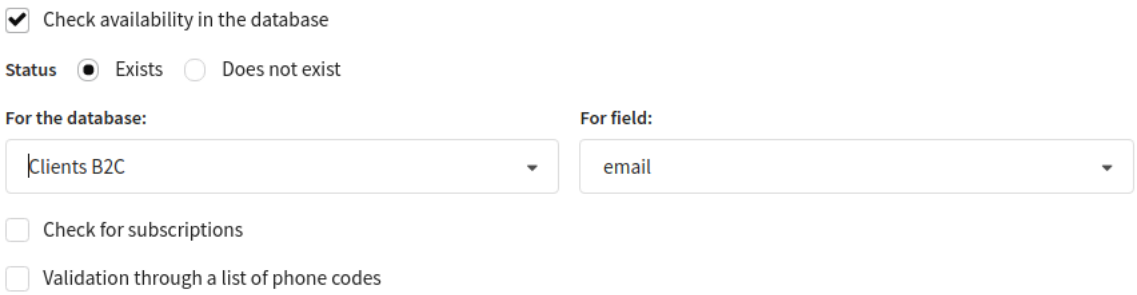
The validation types Check the availability in the database and Check for subscriptions allow you to restrict form submissions for certain users. For example, you can allow only registered users to submit, or prevent the same user from resubmitting a form.

Check the availability in the database

At the moment of form submission, the platform will check if the profile of the person who filled out the form exists in the database. If a profile with the specified data does not exist, the user will not be able to submit the form.

When configuring the form, you need to select:

- **the status you want to check (exists or does not exist),**
- **the database in which to look for the profile,**
- **the field in the database to search for an existing profile.**

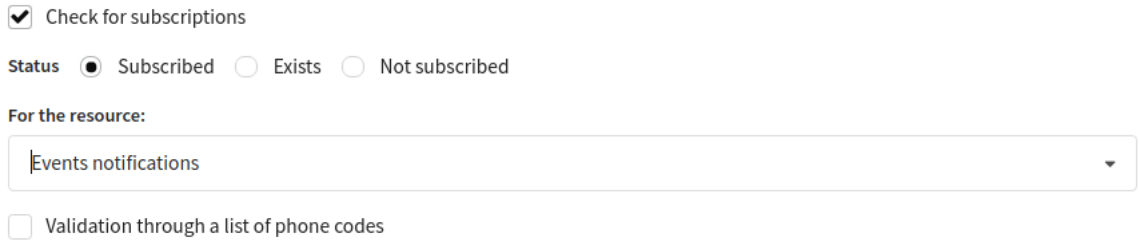


Check for subscriptions

At the moment of form submission, the platform will check if the profile of the person who filled out the form is subscribed to a specific resource. If a profile with the specified subscription does not exist, the user will not be able to submit the form.

When configuring the form, you need to select:

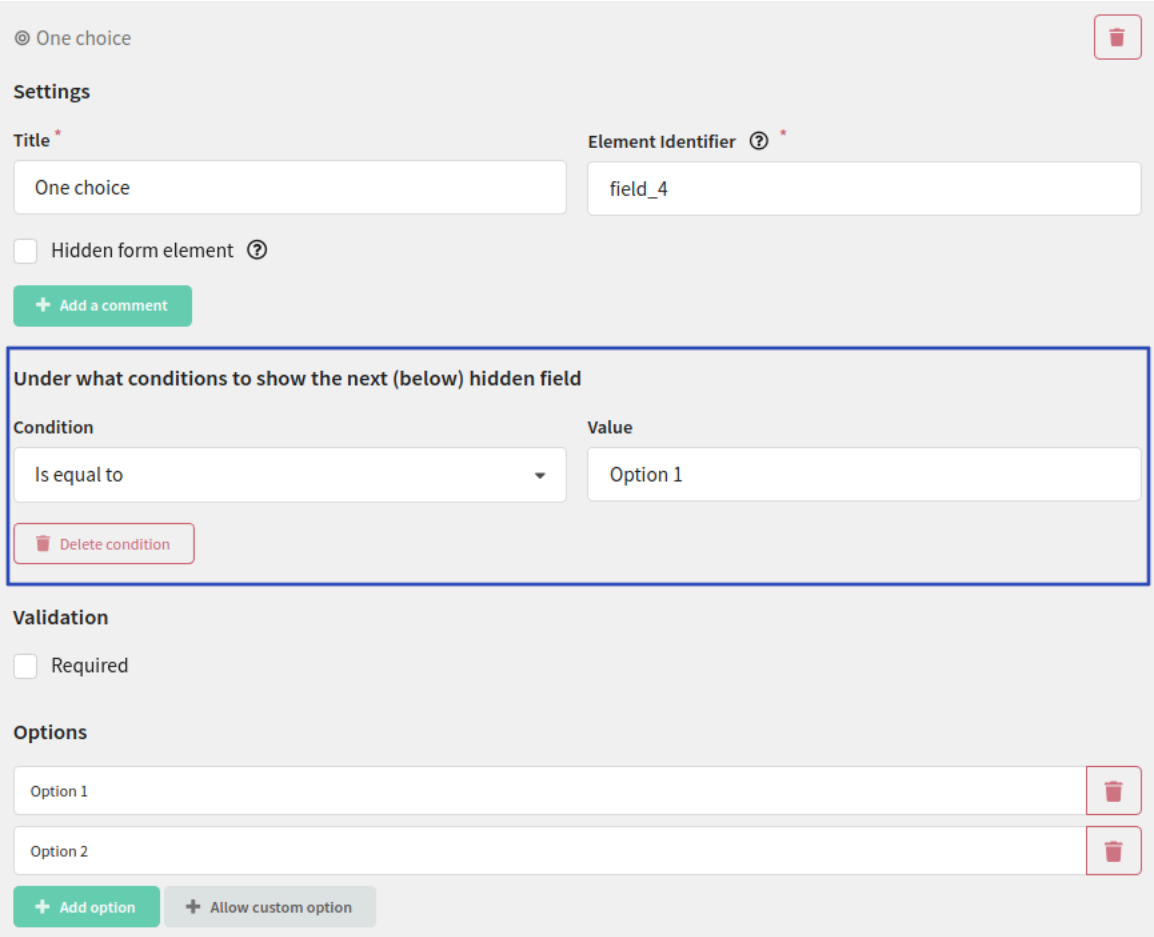
- **the status you want to check (subscribed, exists, or not subscribed),**
- **the resource to check subscription for.**



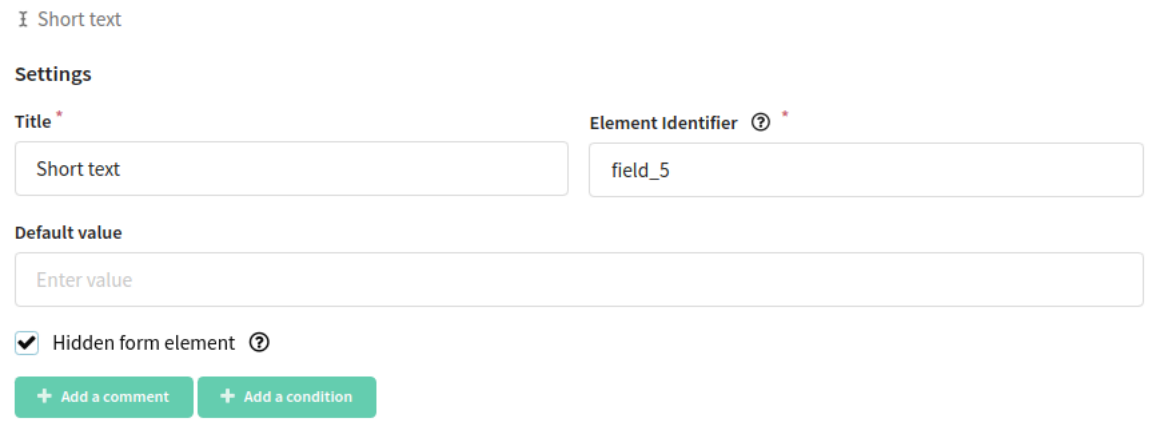
Form element display condition

You can show a form field to the user only if they gave a specific answer to the previous question. To do this, when configuring the element, click the Add condition button — additional fields will open (Under what conditions show the next hidden field).

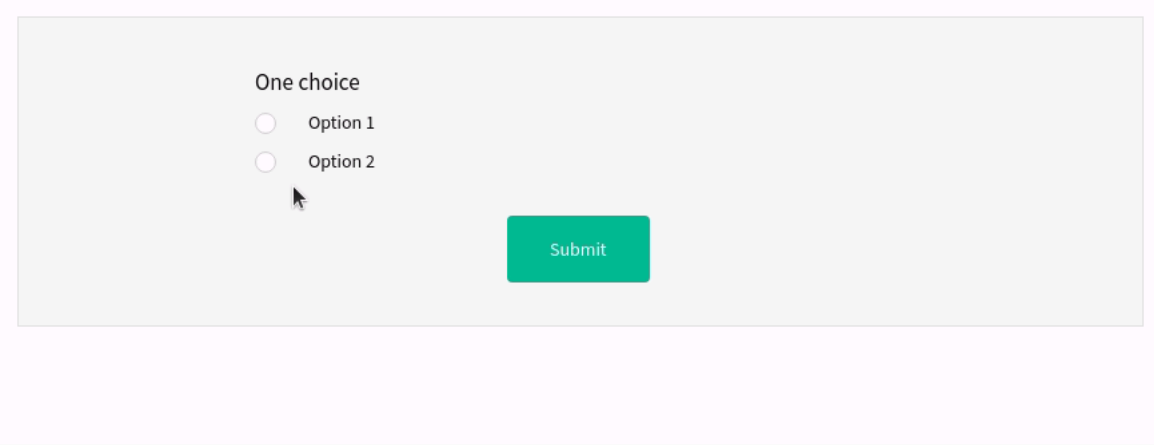
Specify the condition and the value the user should select.



Next, add the element you want to show if the user selects a certain value in the question above. Activate the Hidden form element option:



Now if the user selected a specific answer option, the hidden element will be shown to them. Otherwise, they will skip to the next question.

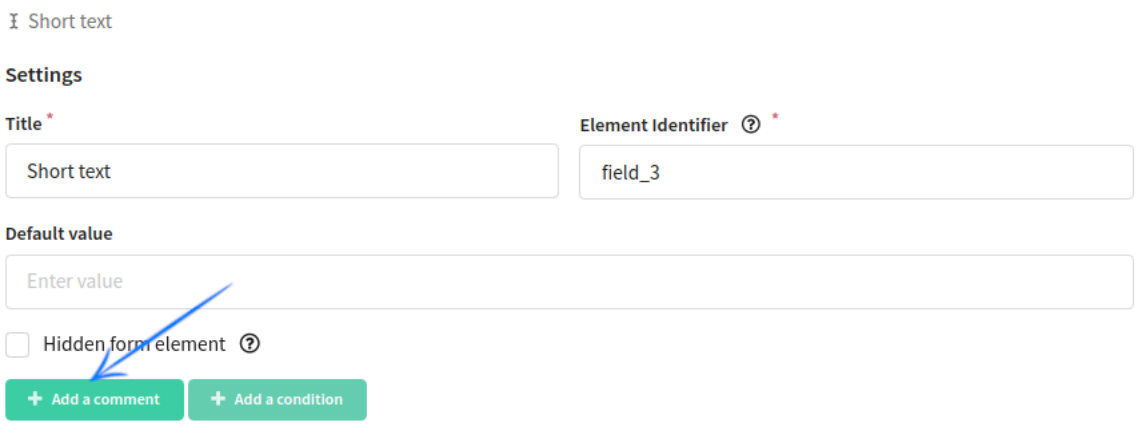


TIP

To add more complex conditions, such as transitioning between form pages depending on answers, configure the Logic block.

Comment

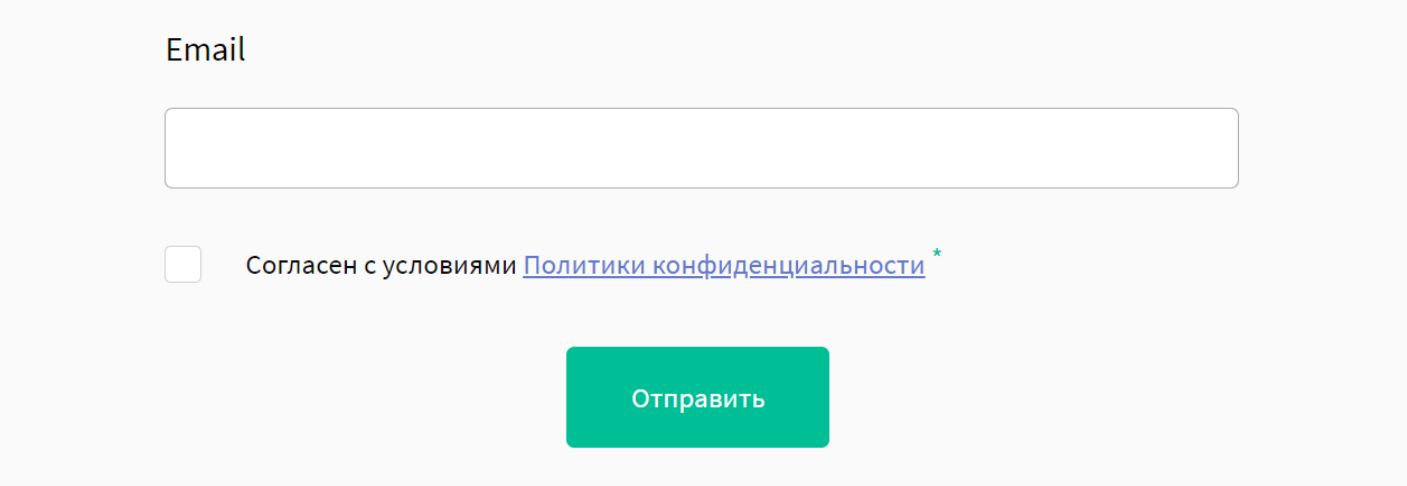
For each element, you can add a comment that will be displayed below the form field.



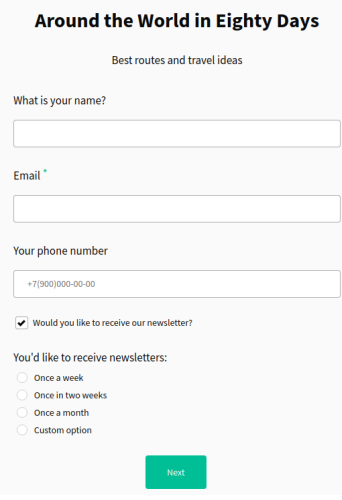
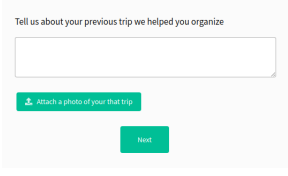
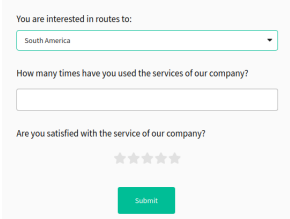
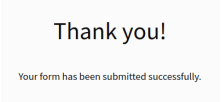
Links in form fields

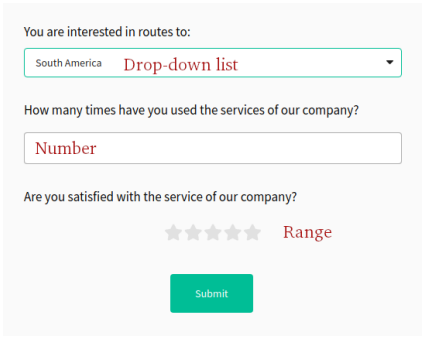
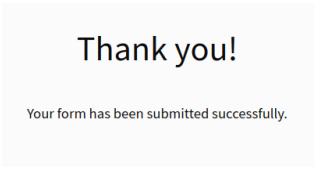
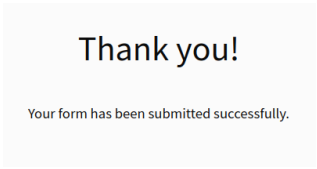
You can add a link to any form element using HTML:

```
I agree to the <a target="_blank" href="#">Privacy Policy</a> terms
```



Form example

Page 1	Page 2	Page 3	Final page
			

Page 1	Page 2	Page 3	Final page
			

**Page 1 **

Page 2

Page 3

Final page

Appearance

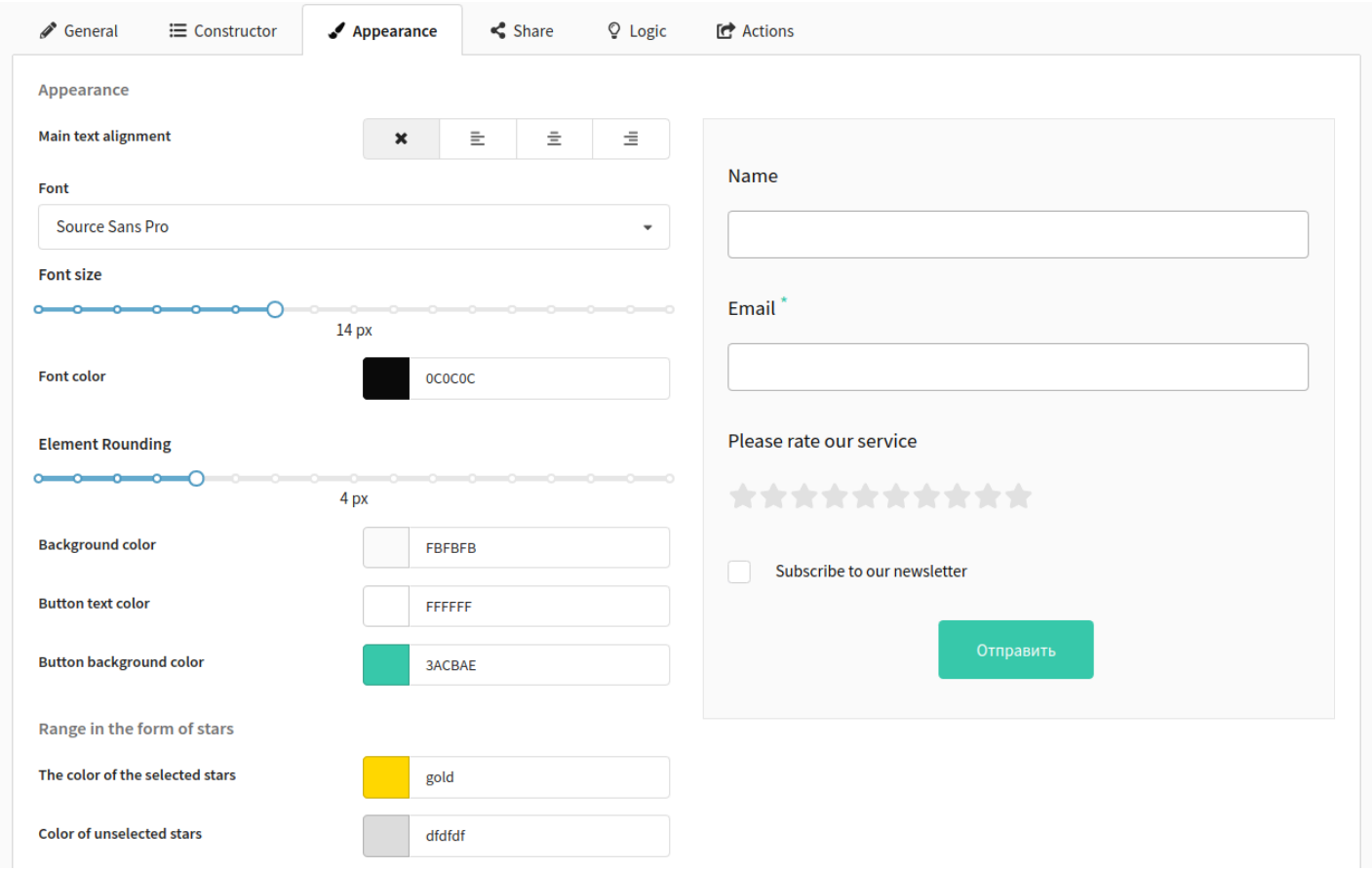
On the Appearance tab, you can customize the form's look and feel: fonts, colors, element alignment, and button styling. Here you can adapt the form design to match your website or brand style.

Main appearance settings

Configure the visual appearance of the form:

- **Fonts — font family and size**
- **Color scheme — text, background, and button colors**
- **Alignment — element layout (left/right alignment, center)**

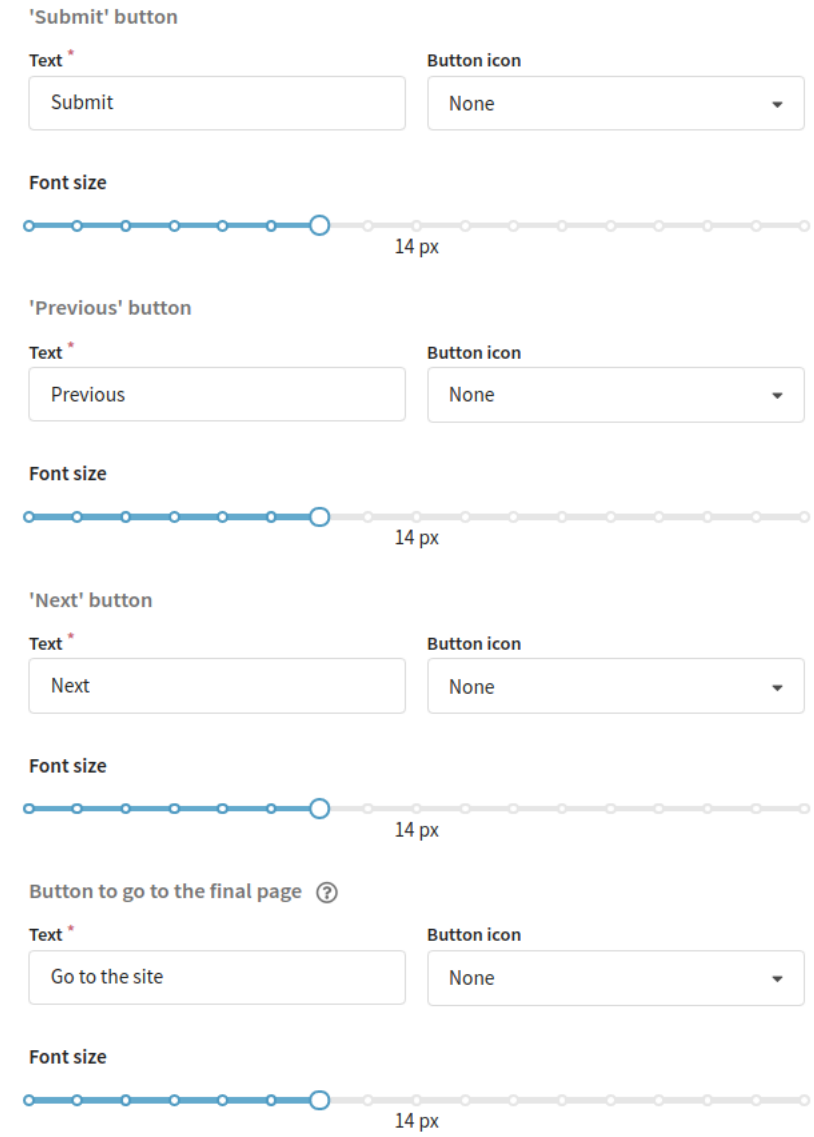
The left side displays a real-time preview of the form.



Button styling

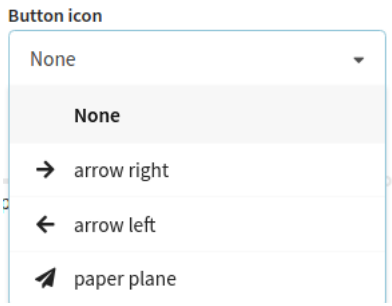
The following buttons are available for styling:

- **Submit — to submit the form**
- **Previous — to go to the previous form page**
- **Forward — to go to the next form page**
- **Button to go to the site — on the final page**



For each button, you can specify:

- **the text displayed on the button**
- **font size (in pixels)**
- **an icon**



TIP: Button to go to the final page

If you leave the button text field empty or do not specify a link to go to after filling out the form on the General tab in the final page settings, the button will not be displayed.

Saving the form

After configuring the field structure and appearance, you need to save the form. Only after this will the actions settings upon form activation and the ability to publish it become available.

Actions and form publication

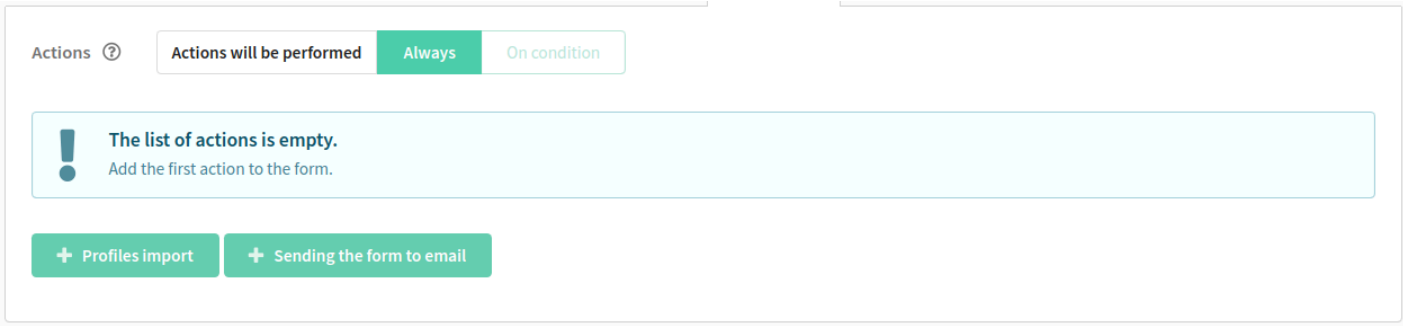
On the Actions tab, you can configure automatic processes that run after successful form submission. Here you can set up user data import into the database and send notifications about form completions.

TIP

Before you start working with actions, save the form.

On the Actions tab, specify the actions to be performed upon successful form submission:

- **Profiles import allows you to upload the data of the user who filled out the form into the database;**
- **Sending the form to email enables sending notifications about form completions to specified email addresses.**



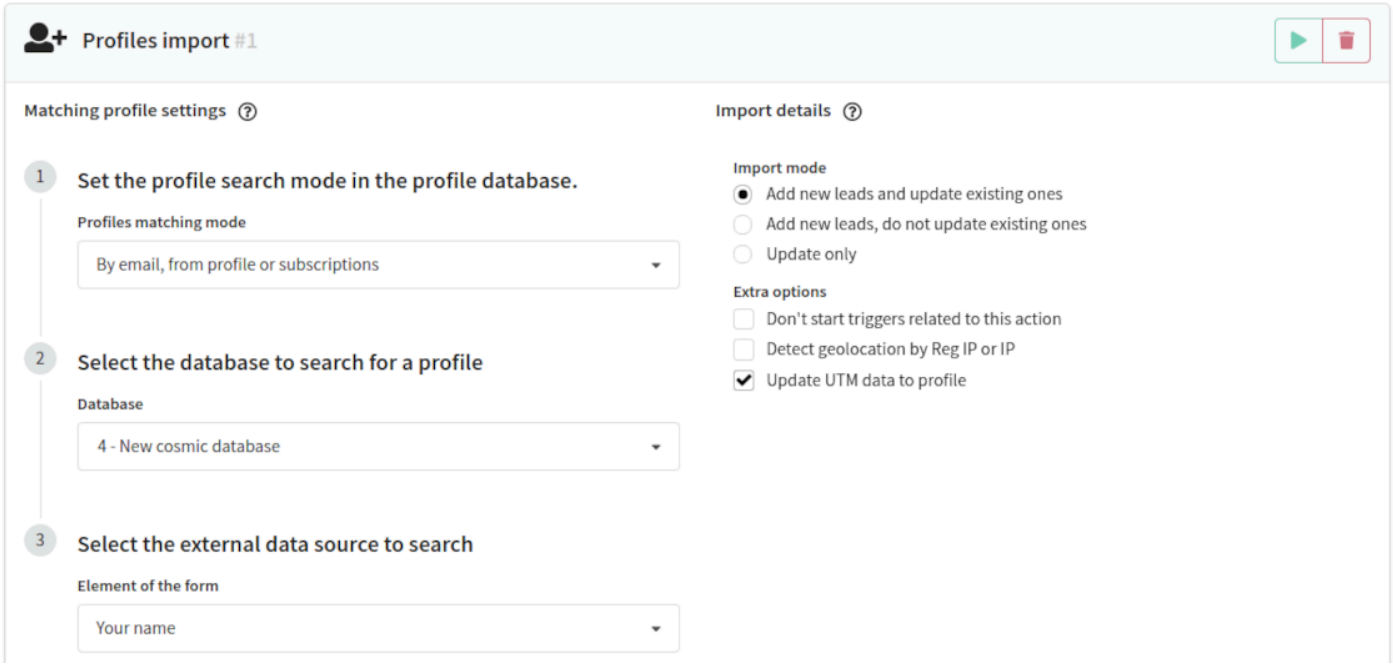
Profiles import

Setting up profile search

Specify the profile search mode in the database:

- **by email address from profile or subscriptions — by the profile field `Email` or by email addresses from subscriptions**
- **by email address from profile — only by the profile field `Email`**
- **by Message ID — by the message template identifier**
- **by additional field — by a custom database field**

Next, select the database where the profile will be imported, and the form element from which to take data for search:



In the right part of the window, you can specify the import method:

- **Add new leads and update existing ones;**
- **Add new leads only, do not update existing ones;**
- **Update only, do not add new ones.**

You can also add additional rules:

- **Don't start triggers related to this action: disable trigger notifications for the new profile after importing into the database;**

- **Detect geolocation by Reg IP or IP: enable automatic detection of the client's geolocation based on the data in the Reg IP (registration IP address) or IP (last known IP address) fields;**
- **Update UTM data to profile: if the form contains UTM tags, the information about these tags is automatically updated in the profile structure after import.**

Column mapping

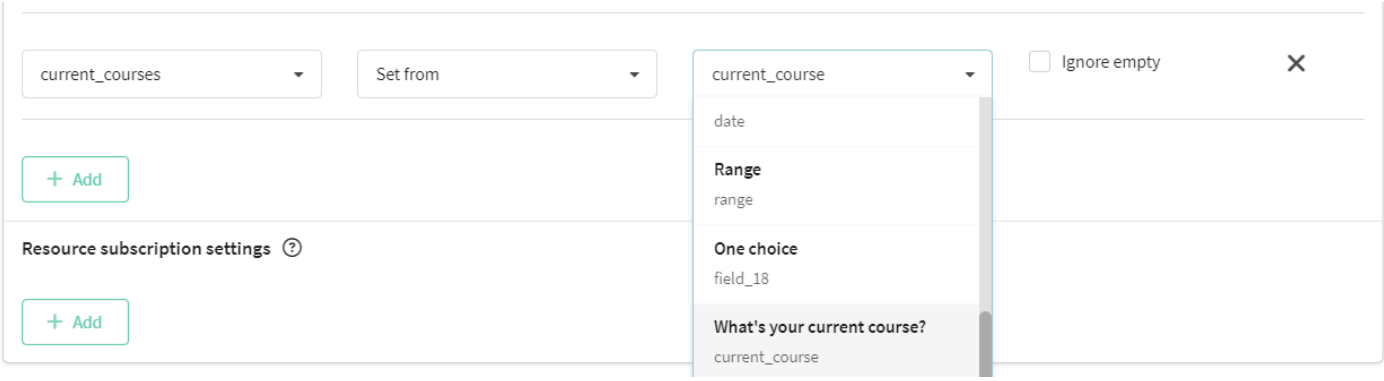
You can set the matching between the form elements and the fields in the profile card.

In the Field in the database drop-down menu, fields from the database are available: system and custom. They are necessary for matching with the main fields of the form. For example, the Email field in the database corresponds to the field of the form where you want to specify the Email.

In the Action menu, you can choose three options: set from, set custom, or remove field.

Source column contains fields that exist in the form. You can name them yourself for easier matching. We recommend naming them the same as the fields, system and custom, are called in the database.

Value handling contains the Ignore empty action. If this field in the form is not filled in, then its value does not overwrite the value in the database that is set by default.

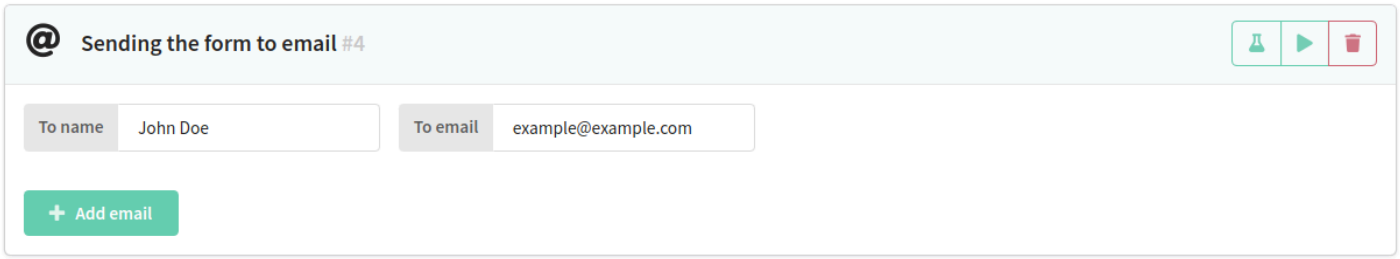


To avoid errors when transferring responses from the form to the profile card, you need to correctly specify the field type when creating the database and match them with the form fields.

Form element	Field's type in a database
Short text	**Line**
Email	**Line**
Phone number	**Phones list / Line**
Long text	**Line**
Number	**Integer**
Yes/No	**Boolean**
One choice	**Line**
Multiple choice	**Tags**
Drop-down list	**Tags**
Date	**Date (RFC 3339 (ISO 8601))**
Range	**Integer**

Resource subscription settings

When importing profiles from forms, you can immediately add a subscription to one of your resources for the client. Select the resource to subscribe to, the channel and subscription status, as well as the field from the form to be used for subscription (for example, email).

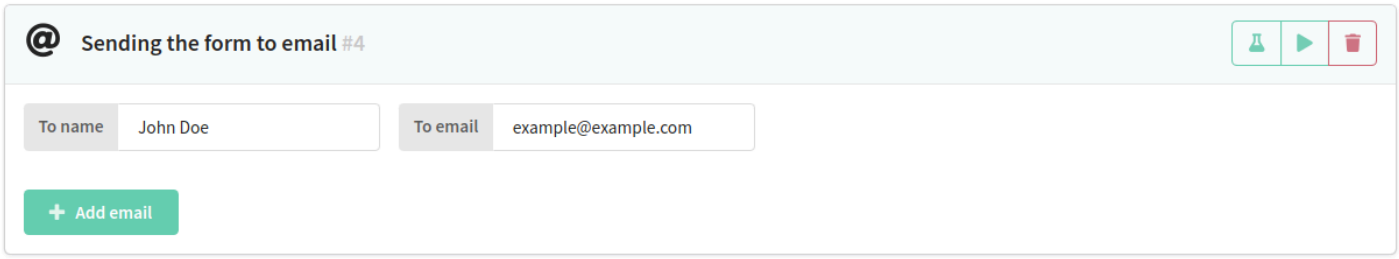


TIP

You can subscribe client profiles to multiple resources at once. To add another subscription, click the + Add button.

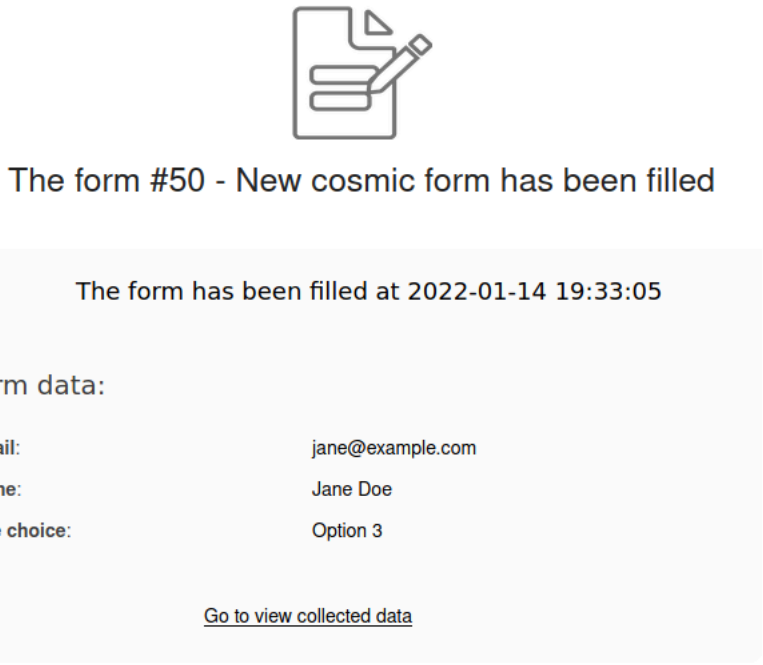
Sending the form to email

Add this action if you want to receive notifications about form completions. Enter your name and email address. You can add multiple recipients.



Each time a user fills out the form, an email will be sent to the specified address with the following data:

- **form completion notification**
- **form name and identifier**
- **link to the form (*Go to view collected data*)**



— test action (send a test message to the specified email address to make sure that the entered data is correct)

— start action (a message will be sent to the mailbox with information about how many times the form was filled out, and an archive containing data on fillings)

— remove action

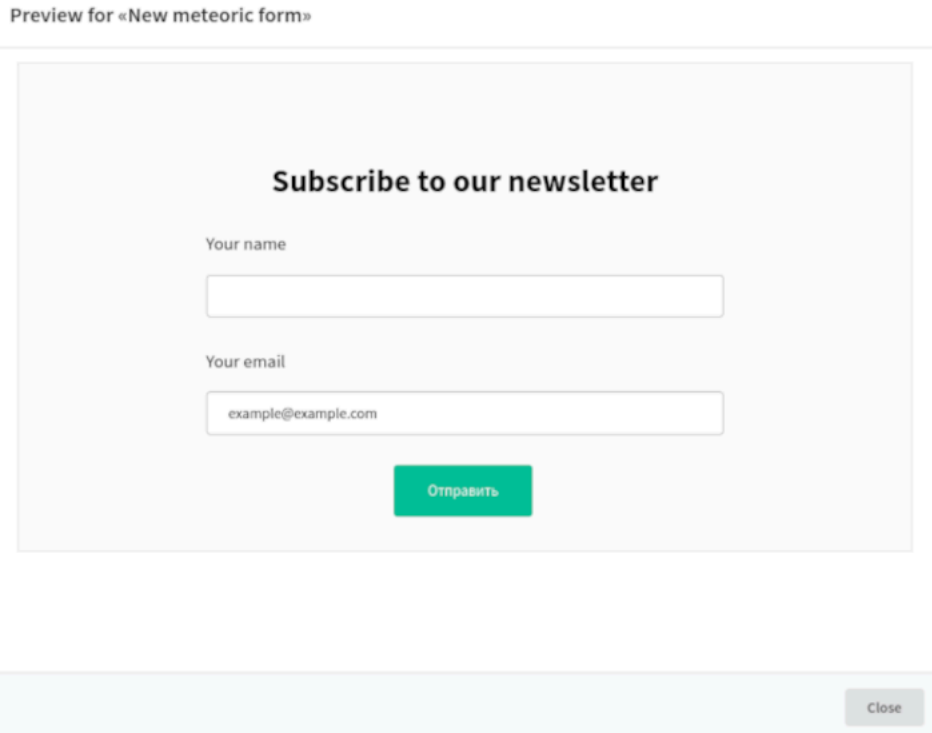
TIP

The Test action and Start action buttons work only if the form is published.

Form preview

At any time while creating and editing the form, a preview is available. To see how the published form will look, click the preview button in the upper right corner of the page

, and the form will open in the way the user will see it.



Share

When the form is ready, you can publish it. To do this, activate the Published switch on the General tab.

Once the form is published and saved, it can be used in two ways:

- **URL: The platform generates a URL that leads to a separate page with the form. You can send this link to subscribers through any communication channel by adding the form variable to the message template. Learn more.**
- **HTML: The platform generates HTML code that needs to be added to your website page. The form will be embedded as an iframe element.**

The URL and HTML code for the form are available on the Share tab:

TIP

After filling out and submitting the form, the user can be redirected to the system's final page or to any other page specified in the form settings. Please note that when embedding the form code on the website, this page will not open inside an iframe element on the current page, but the user will be redirected to a new page.

Conditional logic in forms and surveys

Description

Conditional paging logic allows you to customize the logic of transitions between form pages. Depending on the answers given by the user, you can change the order of displaying pages, as well as set the conditions for displaying a particular page.

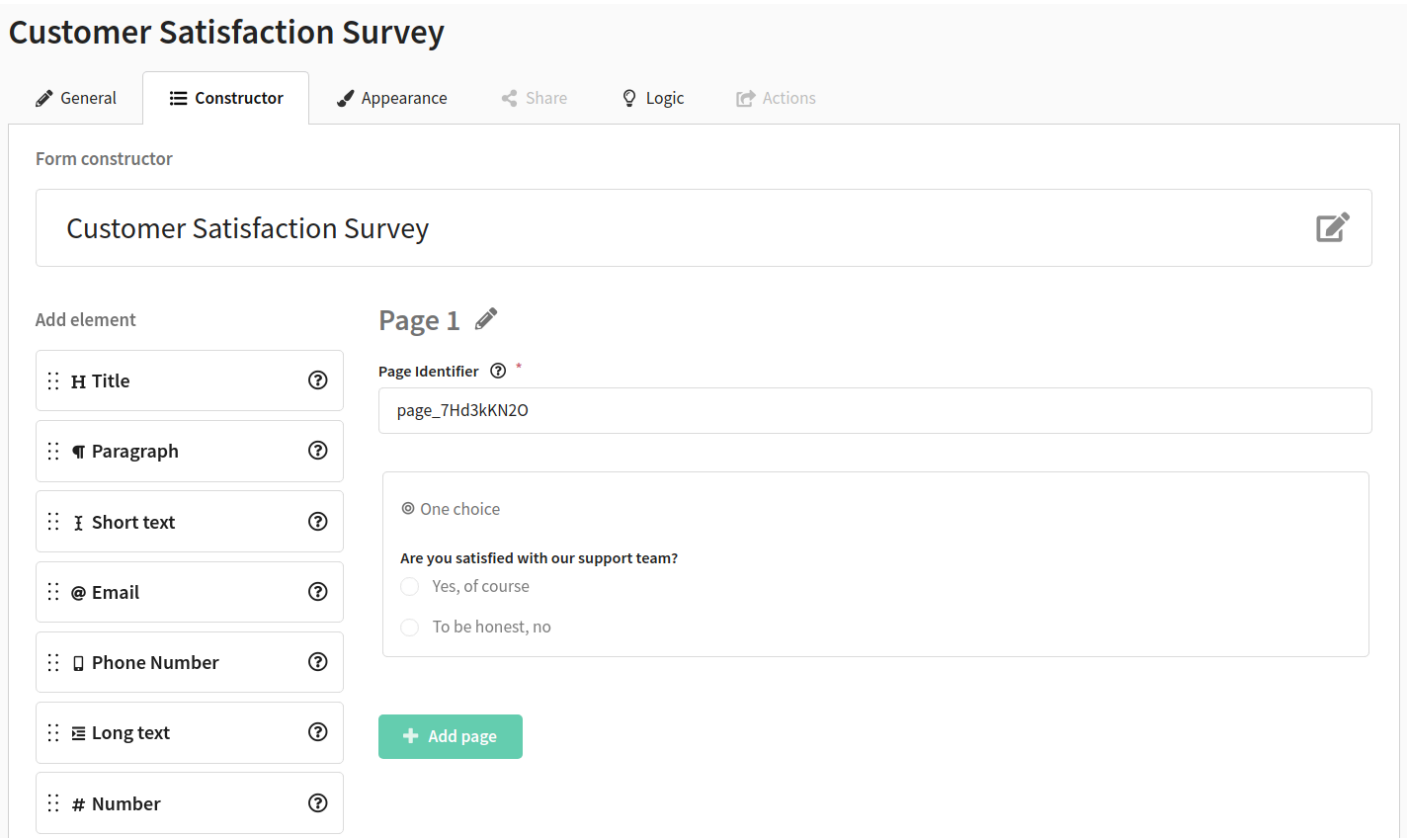
How to set up form logic

Activate logic

Create a form and fill in the basic settings. After that, go to the Logic tab and activate the option by clicking on the toggler in the upper right corner.

Create form structure

Go back to the Constructor tab. Create form pages and add the required elements:

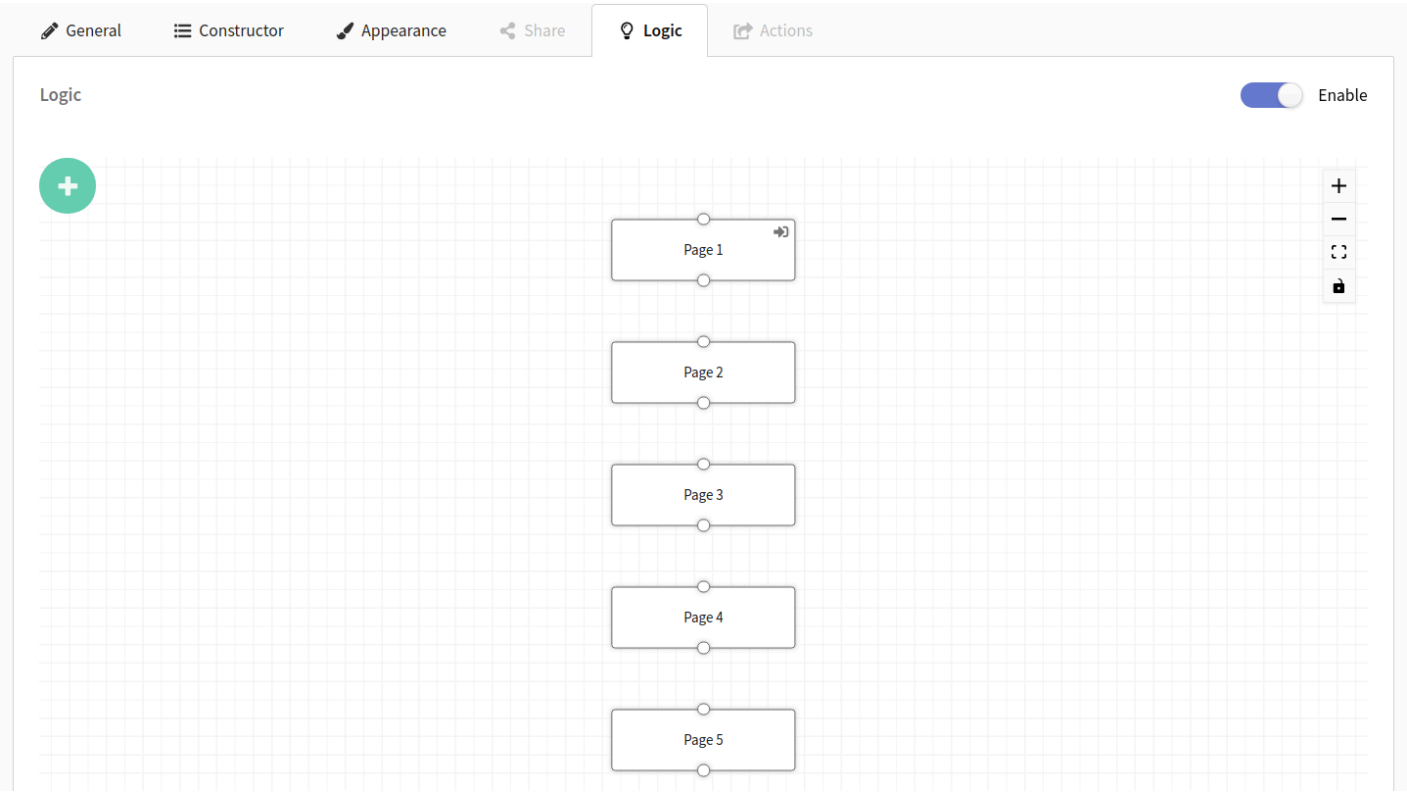


INFO

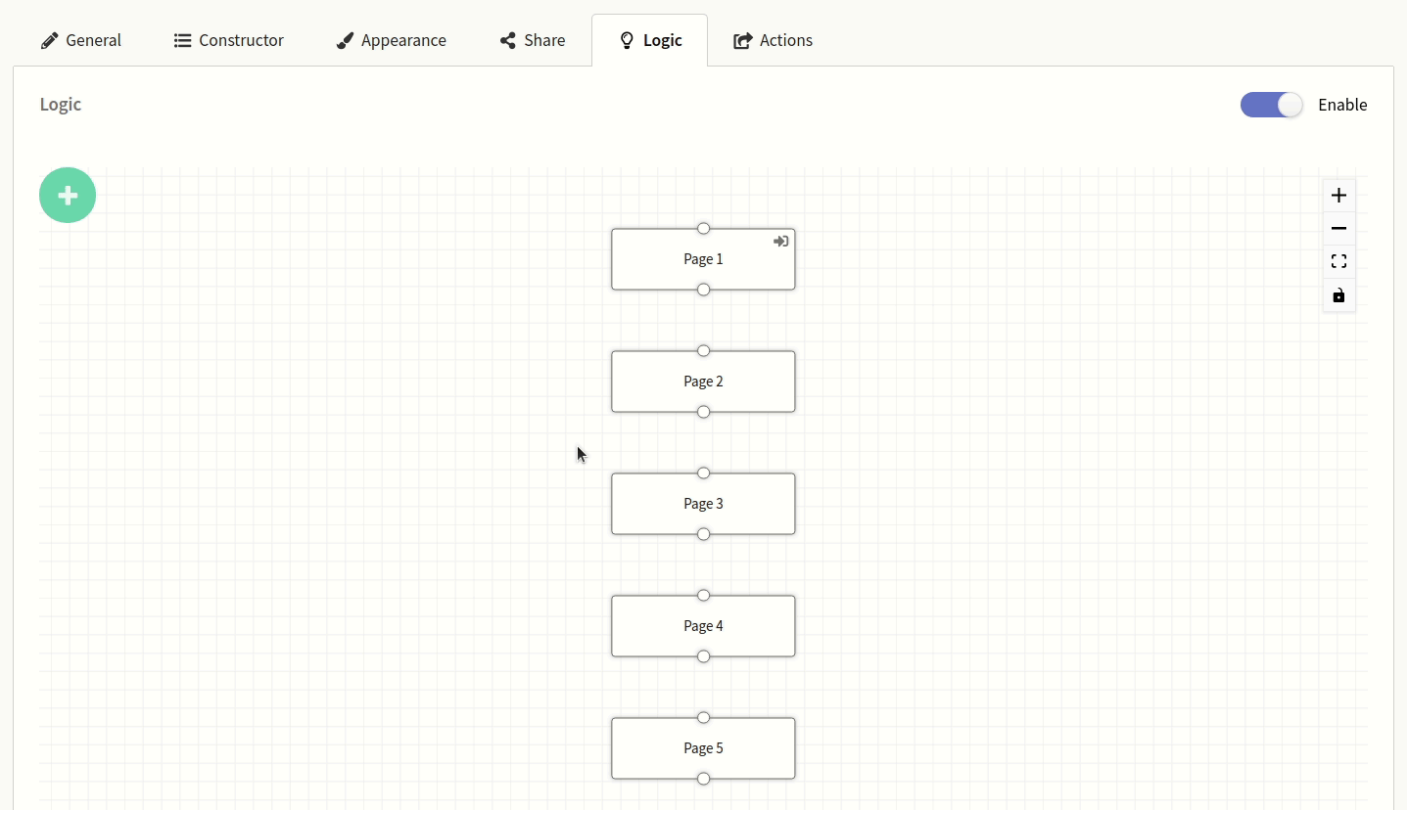
Please note that you are configuring the logic for transitions between pages. In order for the user to go to the desired page after a certain answer to a question, it is necessary to create separate pages for each question, after which the transition will occur.

Working with logic

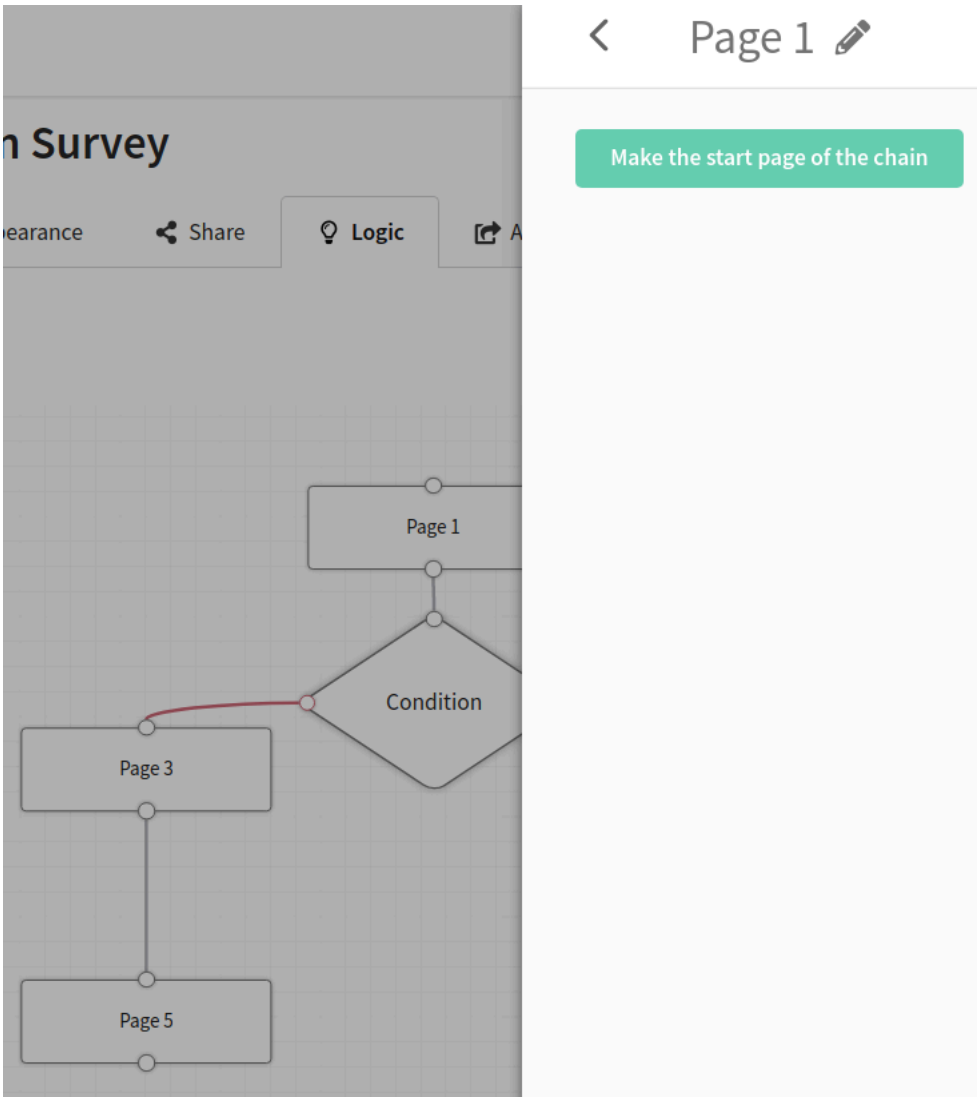
Go to the Logic tab. All created pages will be displayed in a diagram:



To change the order of pages, drag the blocks with your mouse:



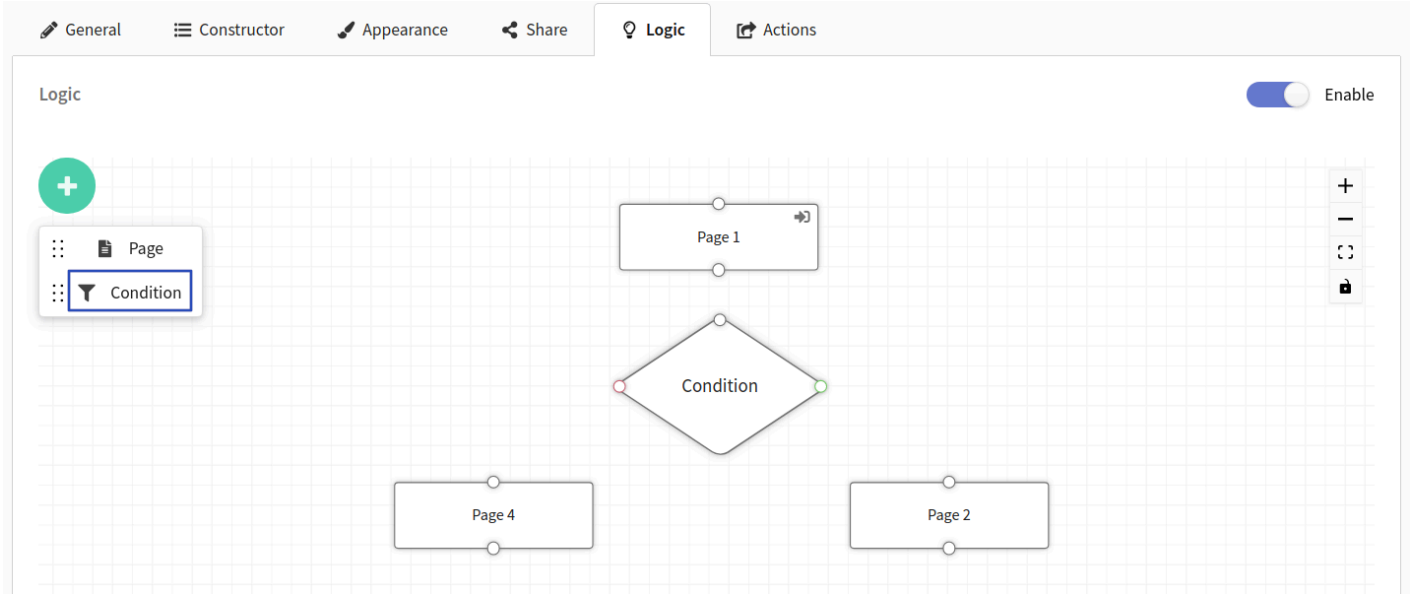
Select the start page by clicking on the page block and then clicking Make the start page of the chain. The start page can be any page in the form.



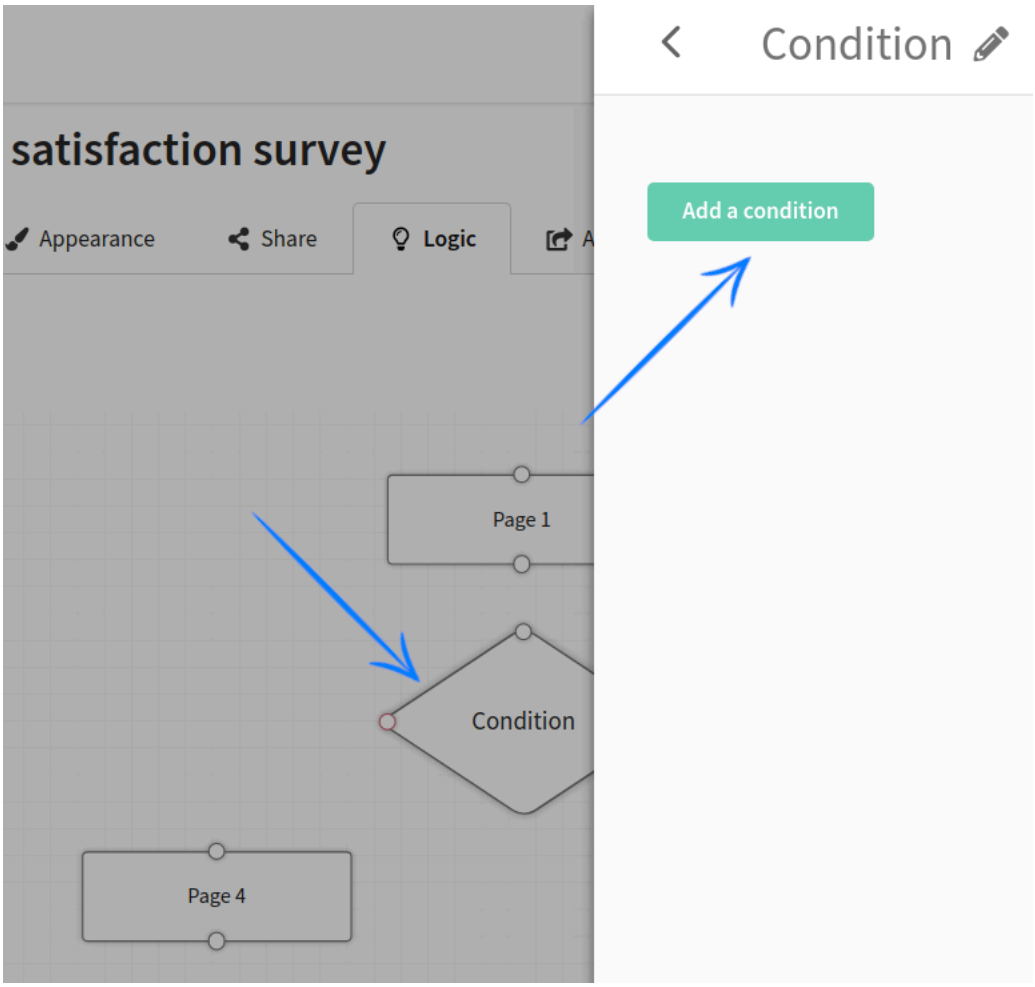
Adding conditions

To add a condition block, click the button

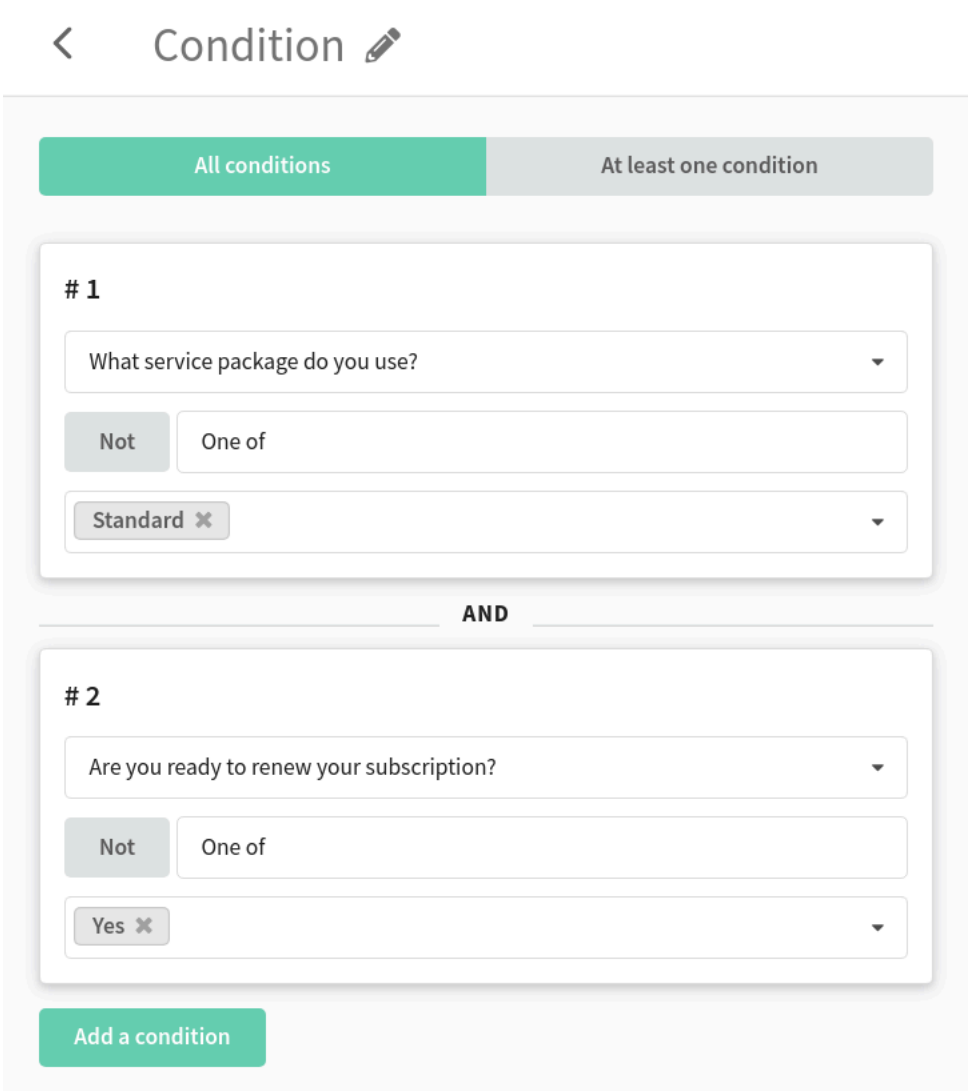
and select Condition:



Click on the condition block (diamond) and click Add a condition:



For one condition block, you can add several conditions at once. It is possible to check the user's response for meeting all conditions (All conditions) or for meeting at least one of them (At least one condition).



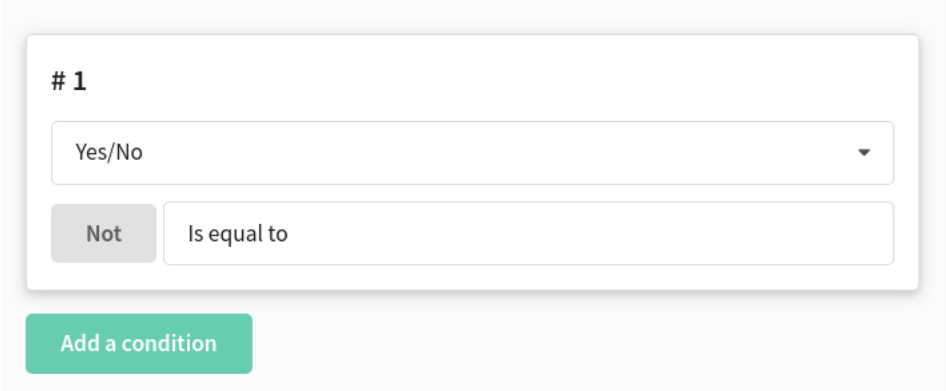
TIP

To remove the entire condition block from the board, use the trash can icon in the upper right corner.

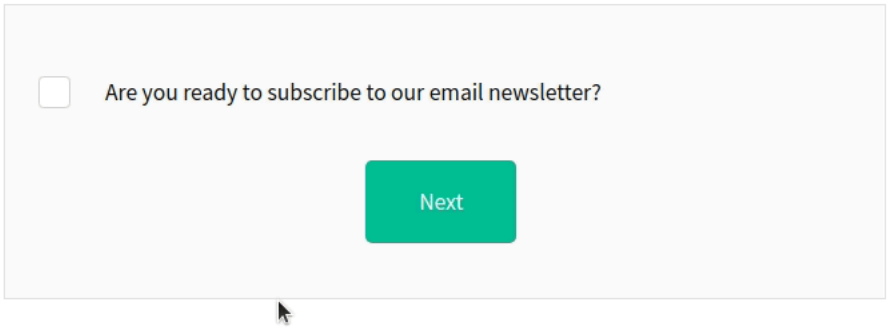
Condition types

Yes/No

Yes/No is the simplest condition. If the user gave a positive answer, they will go to one page; if negative — to another.

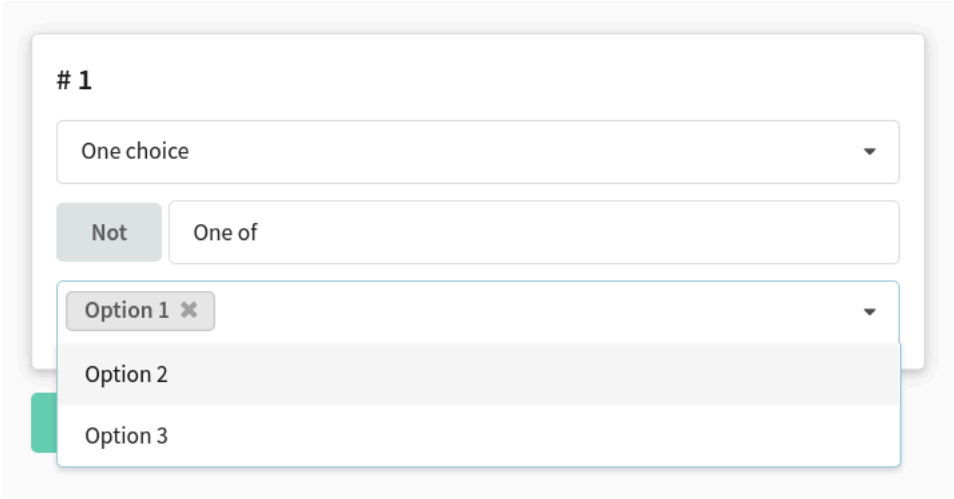


Example

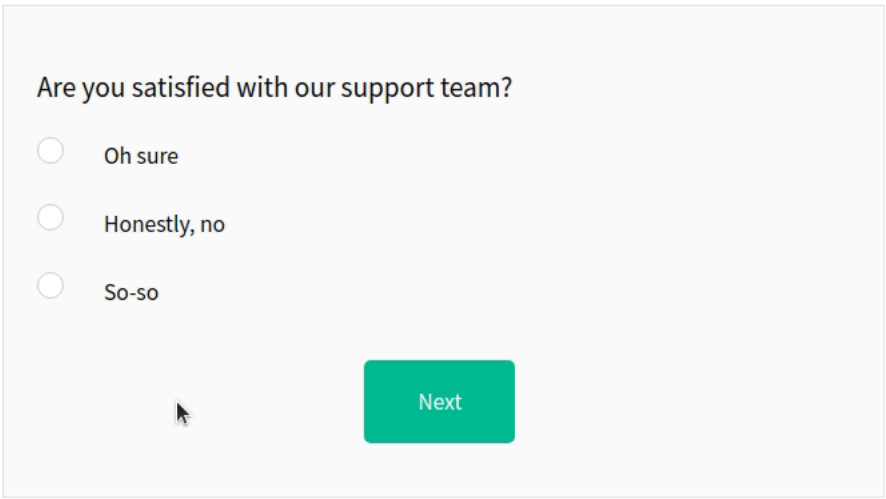


One choice

This element allows you to redirect the user to a specific page depending on the answer option they selected.



Example

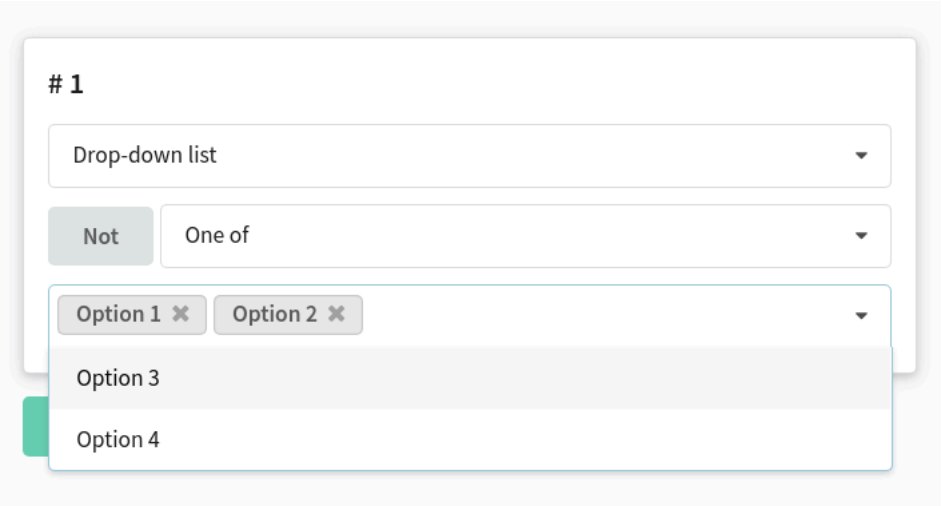


Drop-down list

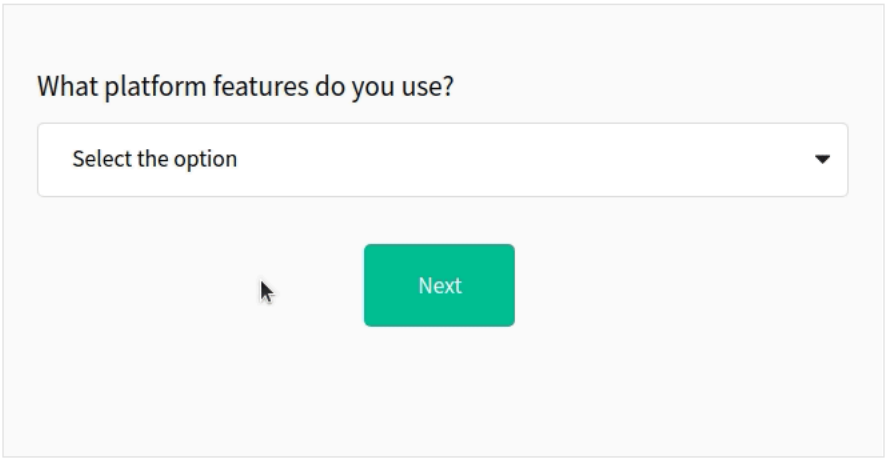
The drop-down list allows the user to select one or more answer options from the proposed ones. You can customize the logic so that when certain options are selected, the user is taken to one page of the form, and when others are selected, they are taken to another page.

For a drop-down list with multiple choices, the following conditions are available:

- **One of — at least one of the answer options selected by the user matches the specified value**
- **All — all answer options selected by the user match the specified value**



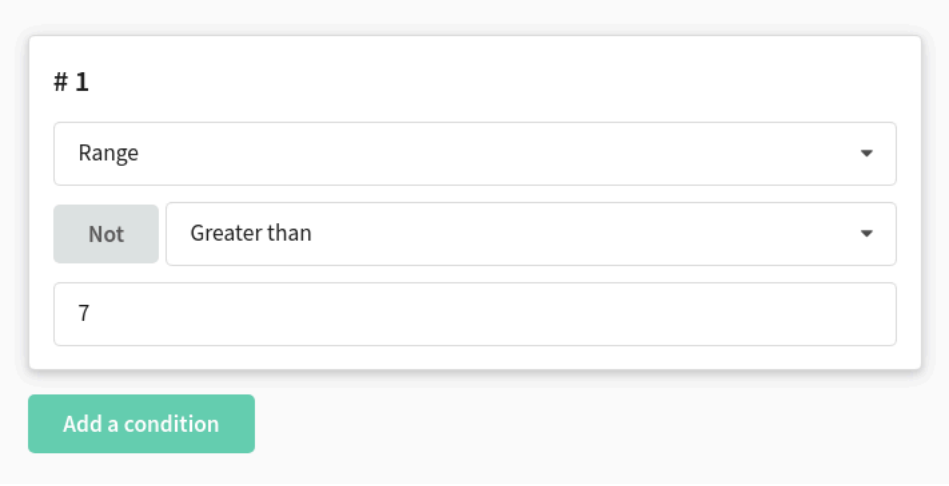
Example



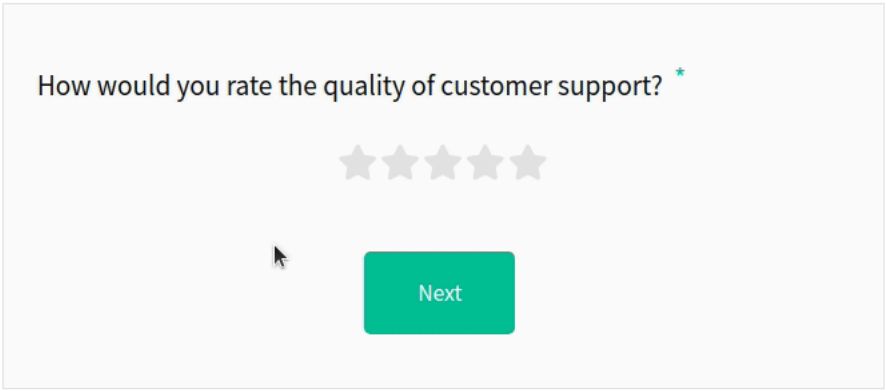
Range

The user will go to a specific page depending on which range value they check. You need to select:

- **the value with which the user's response will be compared**
- **the comparison operator (equal, greater than, less than)**



Example

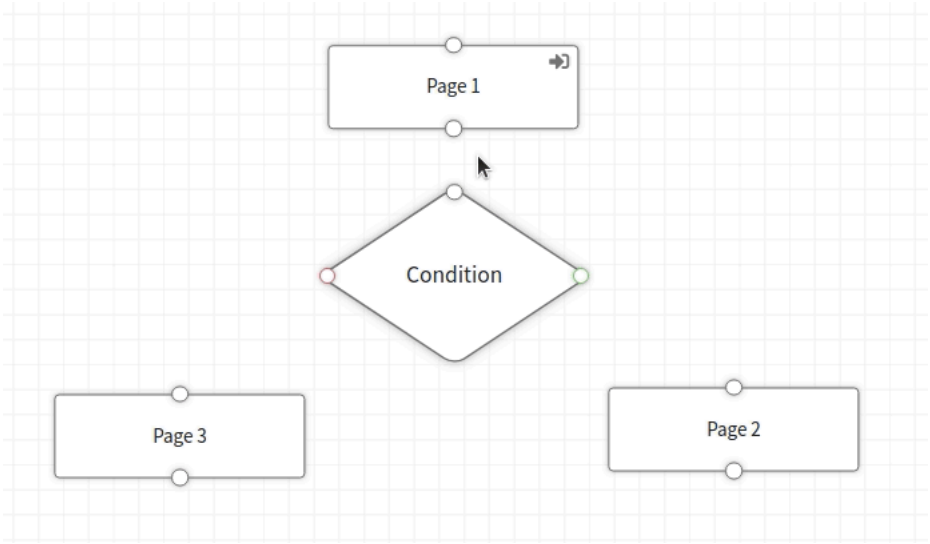


Linking blocks

Link the form pages in the required sequence. Click and hold the icon

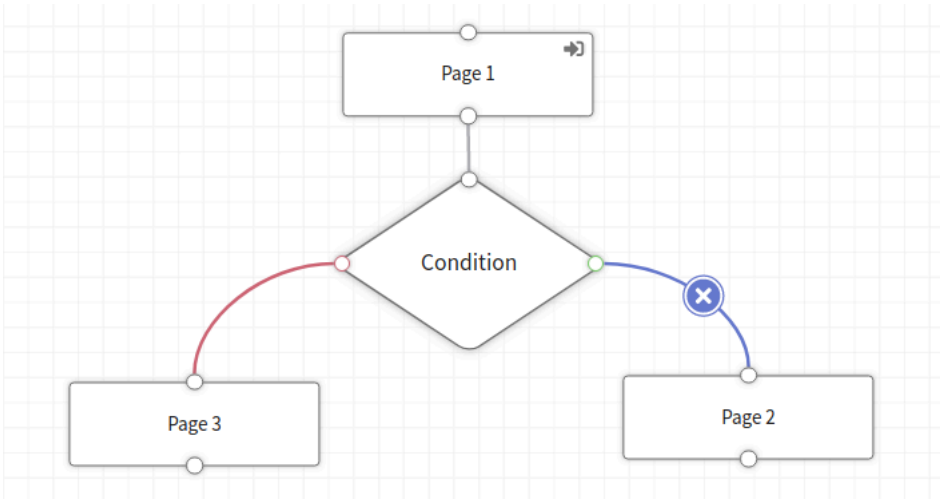
on the block border and drag the line to the page the user should be taken to.

- **Gray line — transition between pages without conditions**
- **Green line — transition if the user's answer matches the condition**
- **Red line — if the condition is not met**



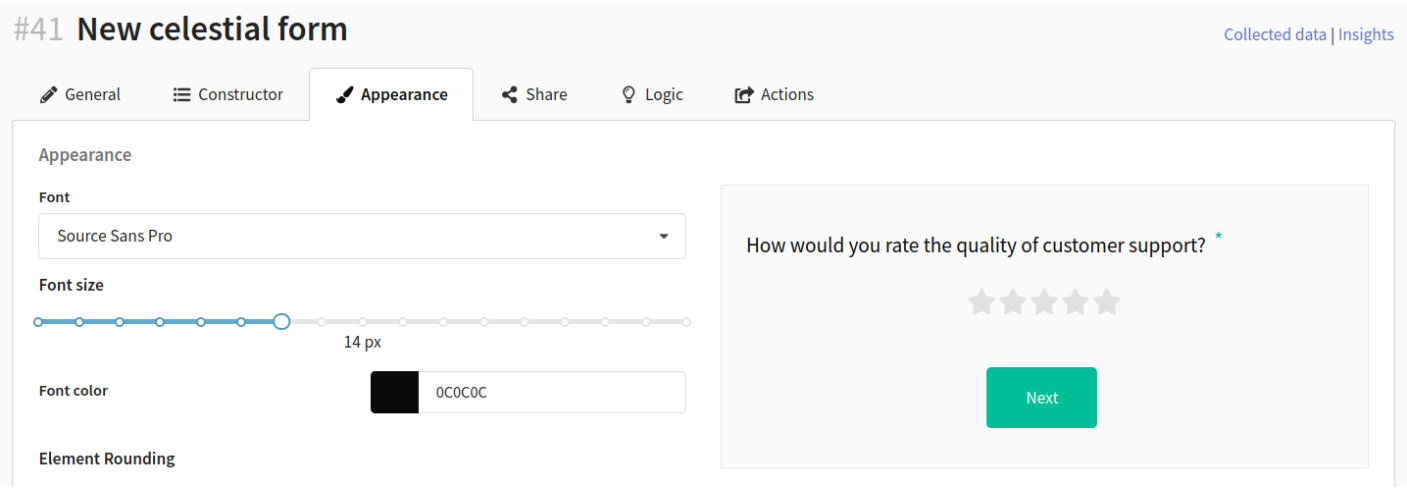
To remove a link, click on the line and then on the icon

:



Preview and publication

You can check the form logic on the "Appearance" tab:



After checking, save the form. After that, you can publish it and use it on the website, or send the client a link to the form page in an email.

Pixels

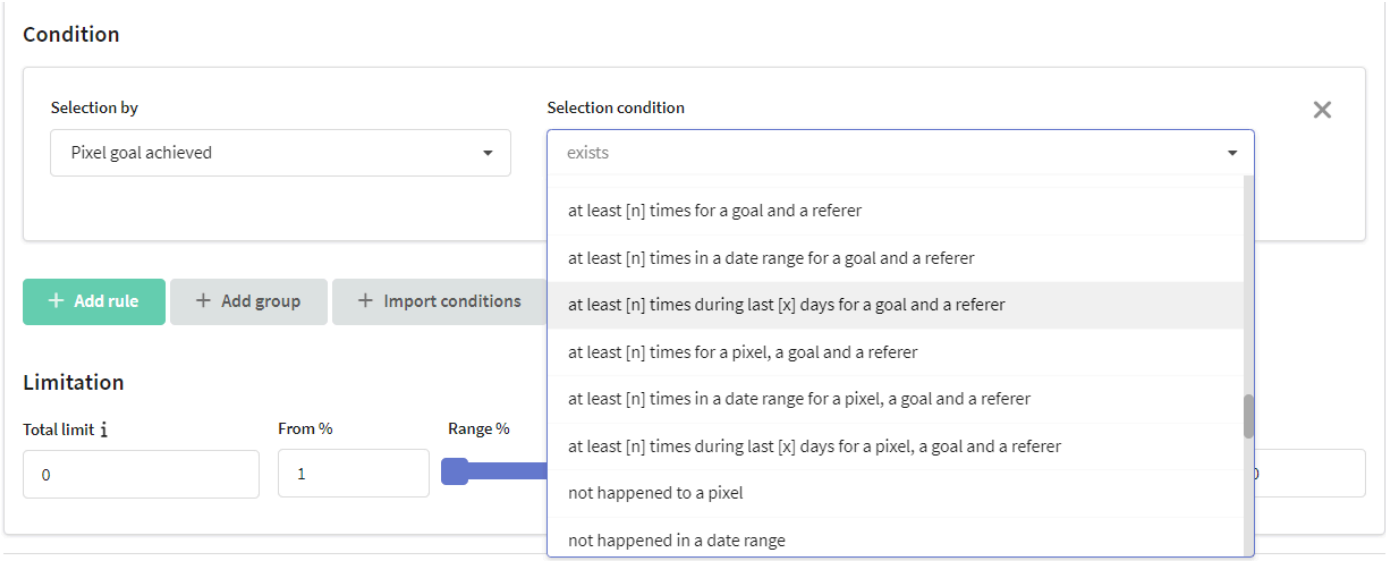
Tracking Pixels are a tool for collecting data about user actions on a website. A pixel is a small piece of code, usually a JS script or a transparent image, invisible to the visitor. It can be linked to a page load or any interactive element — a button, form, or pop-up window.

When a user triggers the pixel through their actions, a request with event information is sent to the platform's tracking server. A pixel can be used to track:

- **the geography of impressions and clicks (country, city, region) determined by IP;**
- **the browser and device platform;**
- **subscriber activity on the site before and after subscription;**
- **the URL of the page from which the user navigated;**
- **subscriber interests and preferences.**

This data is collected from request headers. Additionally, a pixel can transmit business information via goals and their values. A goal is any target action on the site (for example, a product purchase), and a value is a numeric quantity associated with that goal (for example, the purchase amount). Goals and values are defined manually according to your needs.

The collected data is stored in the user's profile in the database if the platform can identify the user. Later, this information can be used for segmentation, scenario building, and triggering automated mailings. For example, when creating a profile segment based on events, you can use pixel goals, website addresses, time intervals, and the number of pixel activations:



Pixel technologies also take a great part in keeping your databases up to date by refreshing profile location and device information.

Pixel goal achievement can be monitored via Goals report.

More details about using pixels you can see in this article.

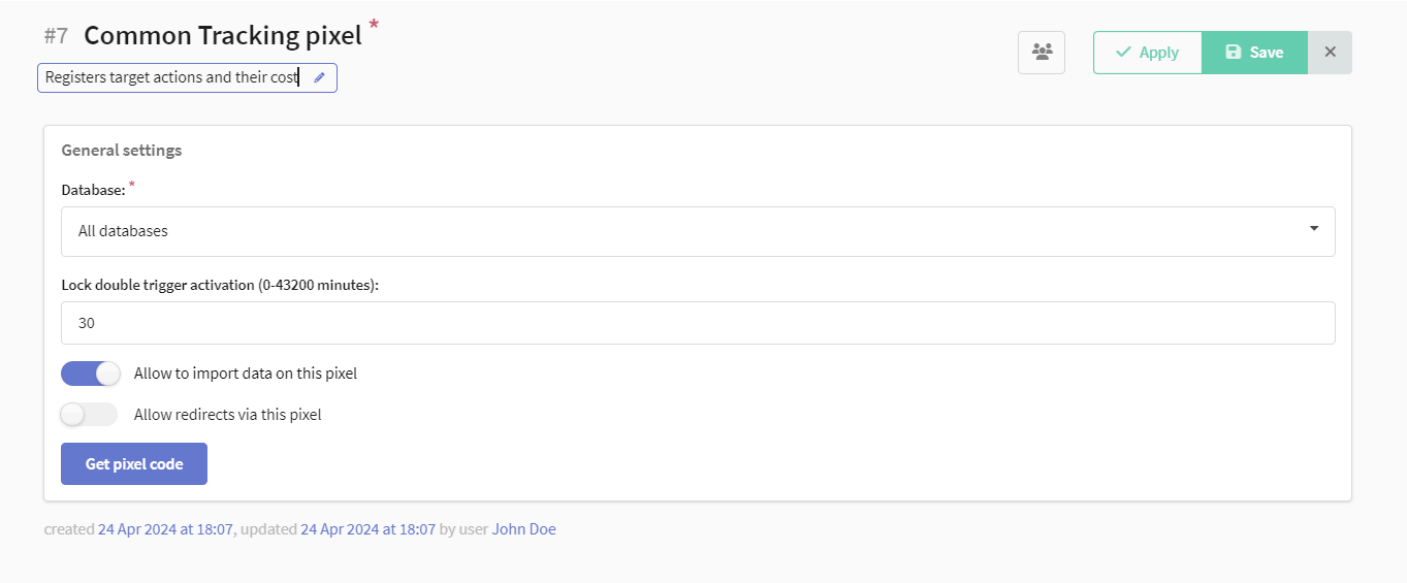
Enabling pixels

	When using Altcraft Marketing On-premise on your hardware, Pixels must be first enabled in platform administrative panel by ticking "Enable pixel" in account settings. NOTE: To use pixels, you need to configure HTTPS protocol for both your website and the platform. **After activation you'll see the new section "Pixels" in the platform interface.**

Creating pixels

Let's create a new pixel script. First of all, you have to give it a name, write a description, set access groups and a database to process.

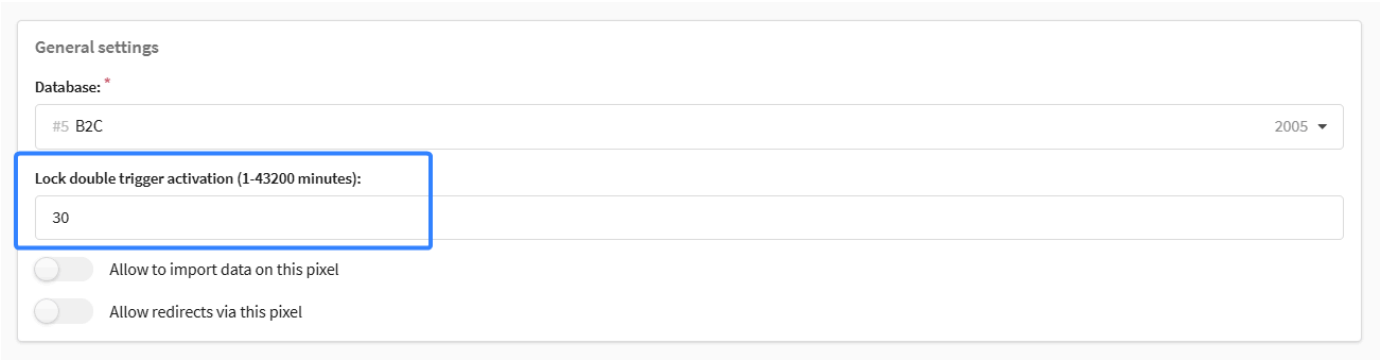
If you'd like to prevent unwanted pixel double activation, limit the time before another activation in the Lock double trigger activation field.



The trigger will be blocked for each specific goal rather than for the entire pixel. This means you can use the same pixel indefinitely, but with different goals for tracking.

TIP

If the trigger does not activate in mailings or scenarios, although the trigger conditions are met, it is recommended to check pixel trigger interval.



INFO

Allow to import data on this pixel — use this field to enable customer profile import at the moment your customer activates a tracking code.

Pixels can update your existing profiles or create the new ones, including for anonymous website visits:

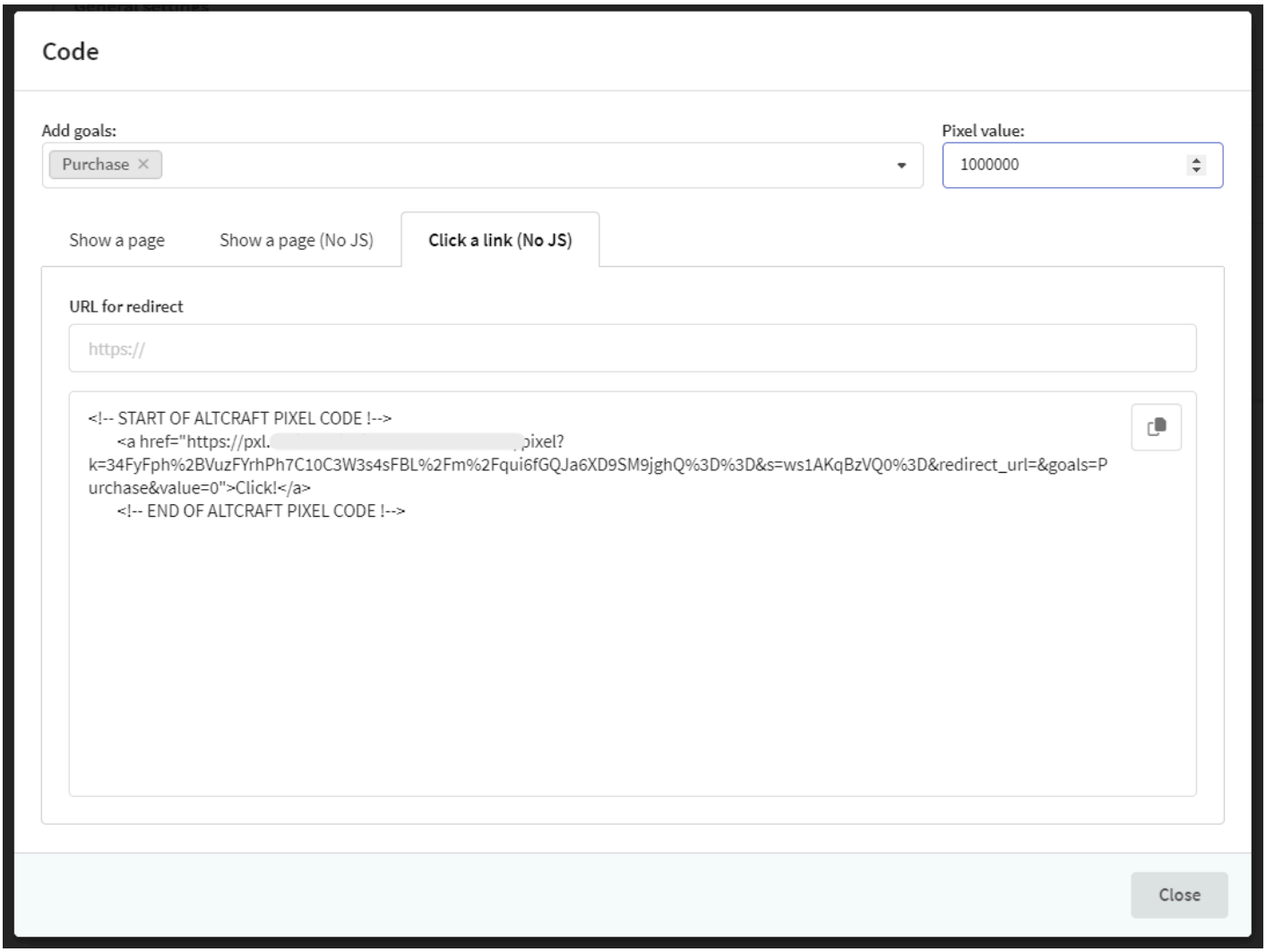
- **HTML pixels (image and URL redirect) use your customers' session data (cookie) for matching profiles. If a profile with the same cookie (e.g. pened mail from the same device) is not found - a new anonymous profile will be imported.**
- **JavaScript tracking codes can use both cookie data and customer profiles information. If a profile is found, its data will be updated; if the profile isn't found, a new profile will be created with the data you can specify.**

JS pixel import is described further on this page.

After you save a new pixel, you'll be able to open Get pixel code interface. The first thing is to decide upon tracking technology you need:

- **Show a page — a JavaScript code that can be customized to detect user actions like scrolling a page within a specific time period.**
- **Show a page (No JS)— this is a somewhat classic pixel technology, representing a transparent image wrapped in a tracking HTTP request. It can be used for both your website and messages — no JavaScript required.**
- **Click a link (No JS) — this technology is a hyperlink wrapped in a redirect tracking request. It can be used in social networks and any external resources.**

For all pixel types you can set goals and goal values:

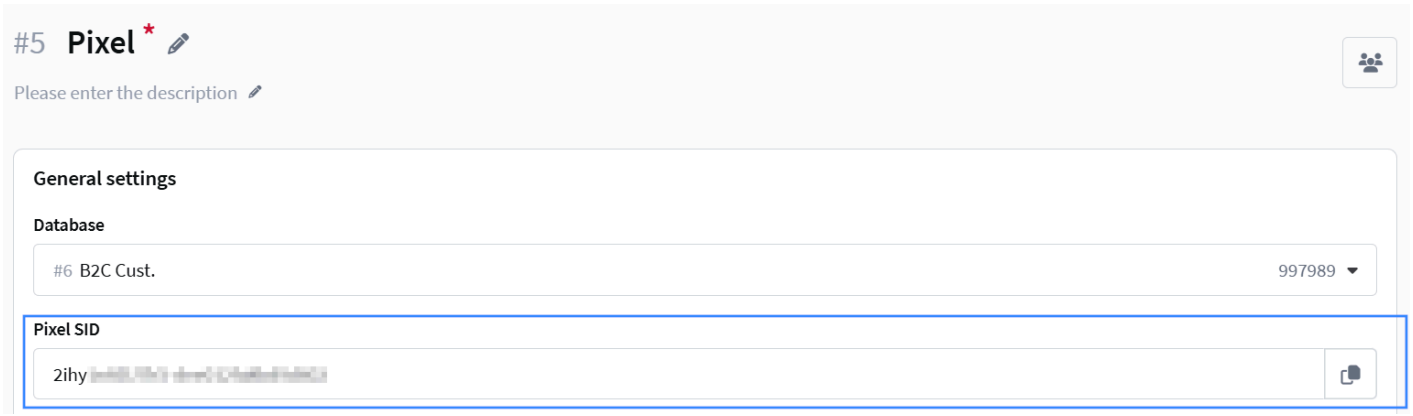


When a pixel code is created, give it to your website developers to implement it into your web site or application code — and it will start tracking your customers' actions.

For each script, you can select several goals and a common value for them (cost).

When placed on the page, the code will register the specified goals, no additional action is required. Each code created in a pixel will associate goals with that particular pixel. This logic can be used to compare conversions across different landing pages, websites, and data collection forms.

Pixel SID is a service string identifier of the pixel. The SID is generated automatically after the pixel is saved and is displayed in the pixel editor. It is used by the platform to register events and is not required for standard website tracking.



The SID may be required when configuring mobile events via mSDK.

Web page tracking

Configuring the code

The script is placed on those pages where you want to track your customers' actions. When a user visits the page, pixel will record goals you specify in pixel settings (Add goals). You can record one value for these initial goals.

INFO

Symbols available for use in goal names:

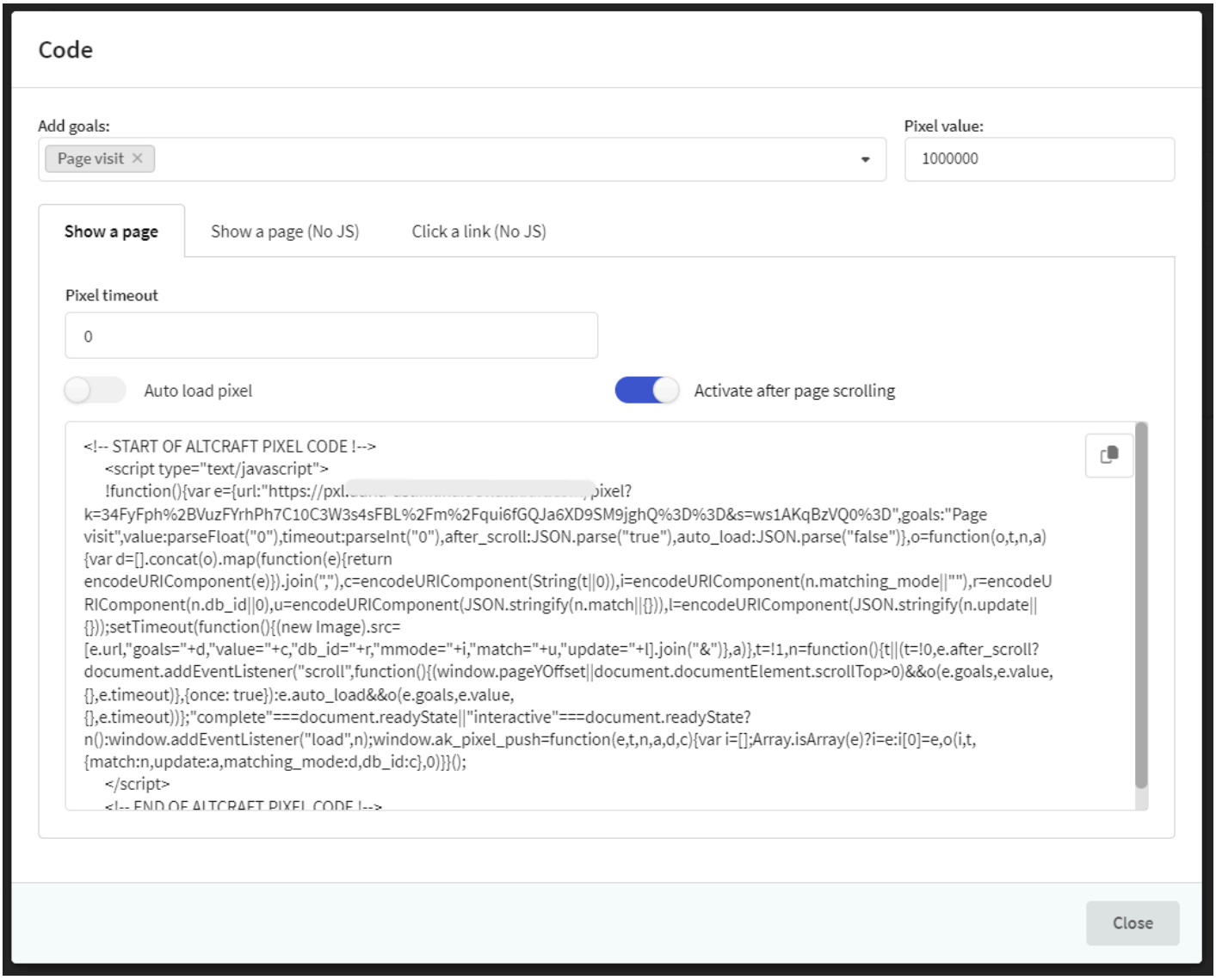
- **letters of the Latin and Russian alphabet (upper and lower case)**
- **spaces and tabs**
- **currency symbols**
- `! # $ % & * + - . , : ; '`
- `= ? @ ^ _ ( ) [ ] {} / \ |`

The maximum length is 128 characters.

Below you can set to activate pixel at once when the page is loaded, or to wait for page scrolling to exclude random website visitors from the query.

TIP

After you configure a new pixel, give its code from the Show a page tab to your website developers along with the information from this page.



Recording goals via API methods

On the web page with the script implemented you can track any of your customers actions based on browser events. For example, these can be a mouse focus on a specific banner — or a popup activation — or leaving a page. On these events you can record customer's target actions and their values. For commercial actions you can register both profit and fees, if any.

For the page elements and browser events use the following method - `ak_pixel_push()` **:**

These arguments are used to record target actions:

- **tags — goal name or names. Can contain a string value or an array of strings.**
- **value — value or price for the goals. Can contain positive and negative floating point values.**

Description	Method
Simple call	`ak_pixel_push()`
Goal registration	`ak_pixel_push("click")`
Goal and value registration	`ak_pixel_push("click", 100)`
Several goals with a negative value. One value will be recorded for all the goals	`ak_pixel_push(["click", "cars"], -100.40)`

Customer profiles import and update (JS)

If customer profiles import is enabled in pixel settings, you can add arguments for searching an existing profile to update and for adding profile data: ak_pixel_push(tags, value, match, update, matching_mode):

- **match — this one is used for profile search. It is an object with one or two properties: “search field”: “field value” и “db_id”: “profile database ID” — in case a database is not specified for the pixel.**

> TIP
>
> `db_id` **can be ignored when a profile database is specified in pixel settings.**

- **update — can contain any customer profile data including subscriptions.**
- **matching_mode - can contain any type of matching.**

> TIP
>
> `matching_mode` **can be ignored when searching by custom fields.**

Example of goal registration with profile import

Below are examples of goal registration with different type of matching:

email

```
ak_pixel_push(
  "click",
  100,
  {
    db_id: 1,
    email: "user@mail.ru"
  },
    {
        "_fname": "John",
        "_lname": "Doe",
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "john.doe@example.com"
            }
        ]
    }
);
```

email_profile

```
ak_pixel_push(
  "click",
  100,
  {
    db_id: 1,
    email: "user@mail.ru"
  },
    {
        "_fname": "John",
        "_lname": "Doe",
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "john.doe@example.com"
            }
        ]
    },
  "email_profile"
);
```

custom

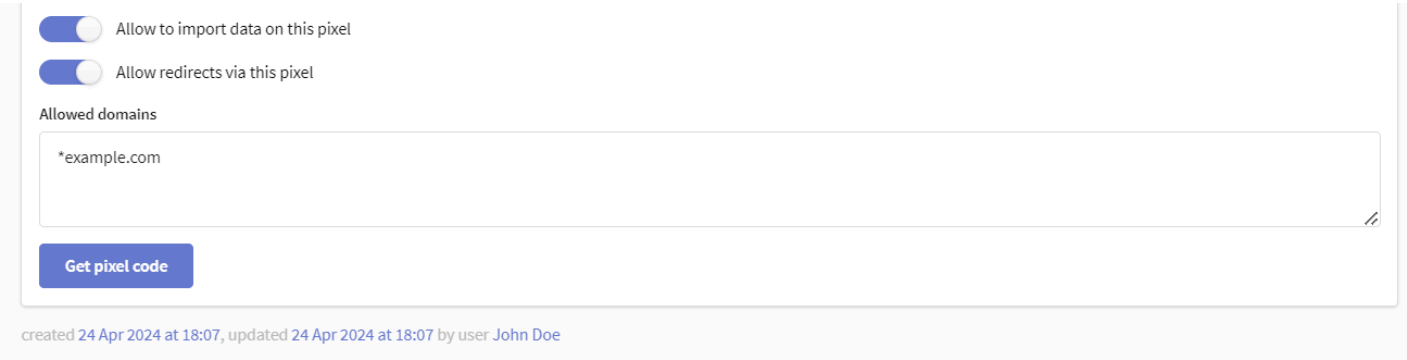
```
ak_pixel_push(
    "click",
    100,
    {
      db_id: 1,
      field_name: "client_id",
      field_value: 1,
    },
    {
      _fname: "John",
      _lname: "Doe",
      subscriptions: [
        {
          resource_id: 1,
          channel: "email",
          email: "john.doe@example.com",
        },
      ],
    },
  "custom"
);
```

phone

```
ak_pixel_push(
    "click",
    100,
    {
      db_id: 1,
      phone: "+79000000000"
    },
      {
          "_fname": "John",
          "_lname": "Doe",
          "subscriptions": [
              {
                  "resource_id": 1,
                  "channel": "email",
                  "email": "john.doe@example.com"
              }
          ]
      },
    "phone"
);
```

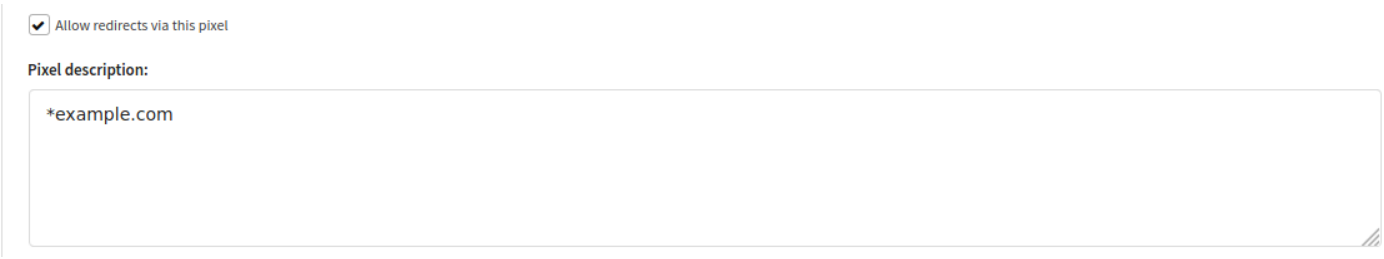
Redirects via pixel settings

If you allow redirects via the pixel, the user will be redirected to the URL you specified at the moment he hits your pixel.



Redirect links can be used for fraud by sending phishing message with redirect_url. To avoid this, pixel in Altcraft MP does not support redirect by default. To engage this option, you need to activate this flag:

When flag is activated, you can enter domains list allowed for redirect, in the field below



After these settings, redirect in messages with pixel will be available on for enlisted domains. Redirect for other domains will be restricted.

Setting up Altcraft pixel goal registration using Google Tag Manager (GTM)

For the convenience of placing the Altcraft pixel on the site pages, you can use the Google Tag Manager containers (if it's installed on the site).

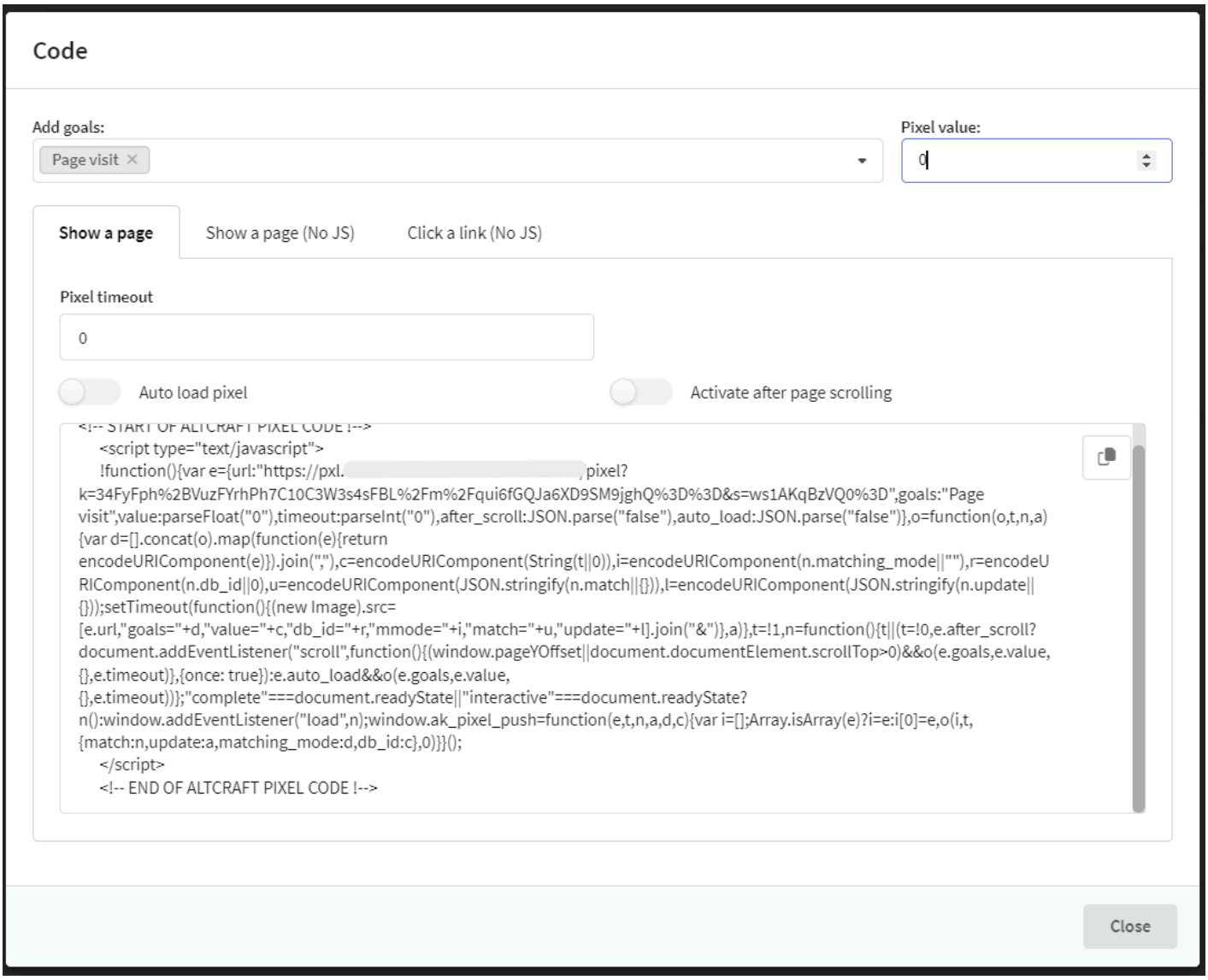
For this:

1. Create a new pixel and set basic settings: pixel name, description, access groups and associated databases.

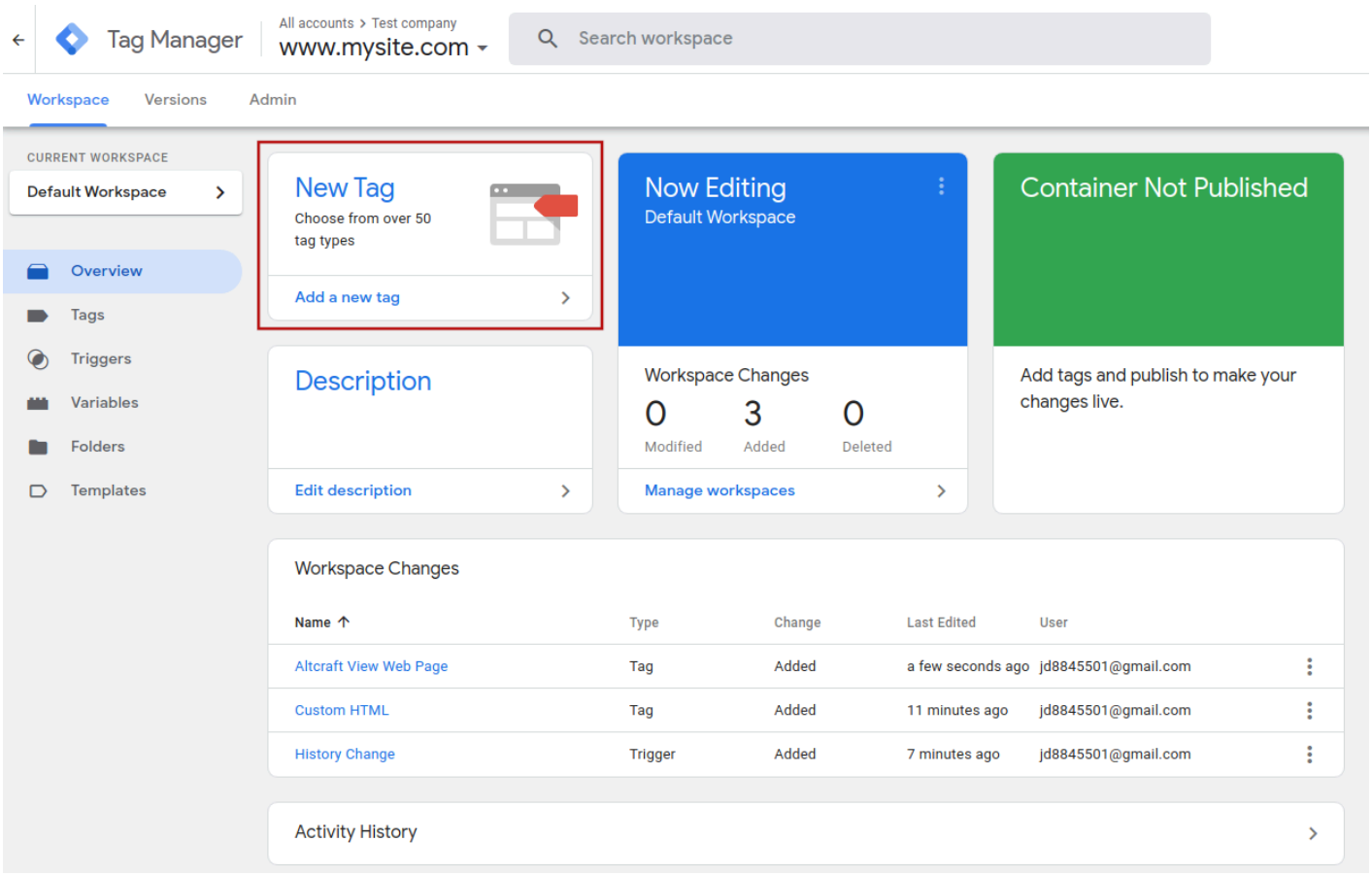
2. After saving the pixel, the tracking code configuration interface will be available. Copy the pixel code from the "Show a page" tab.

CAUTION

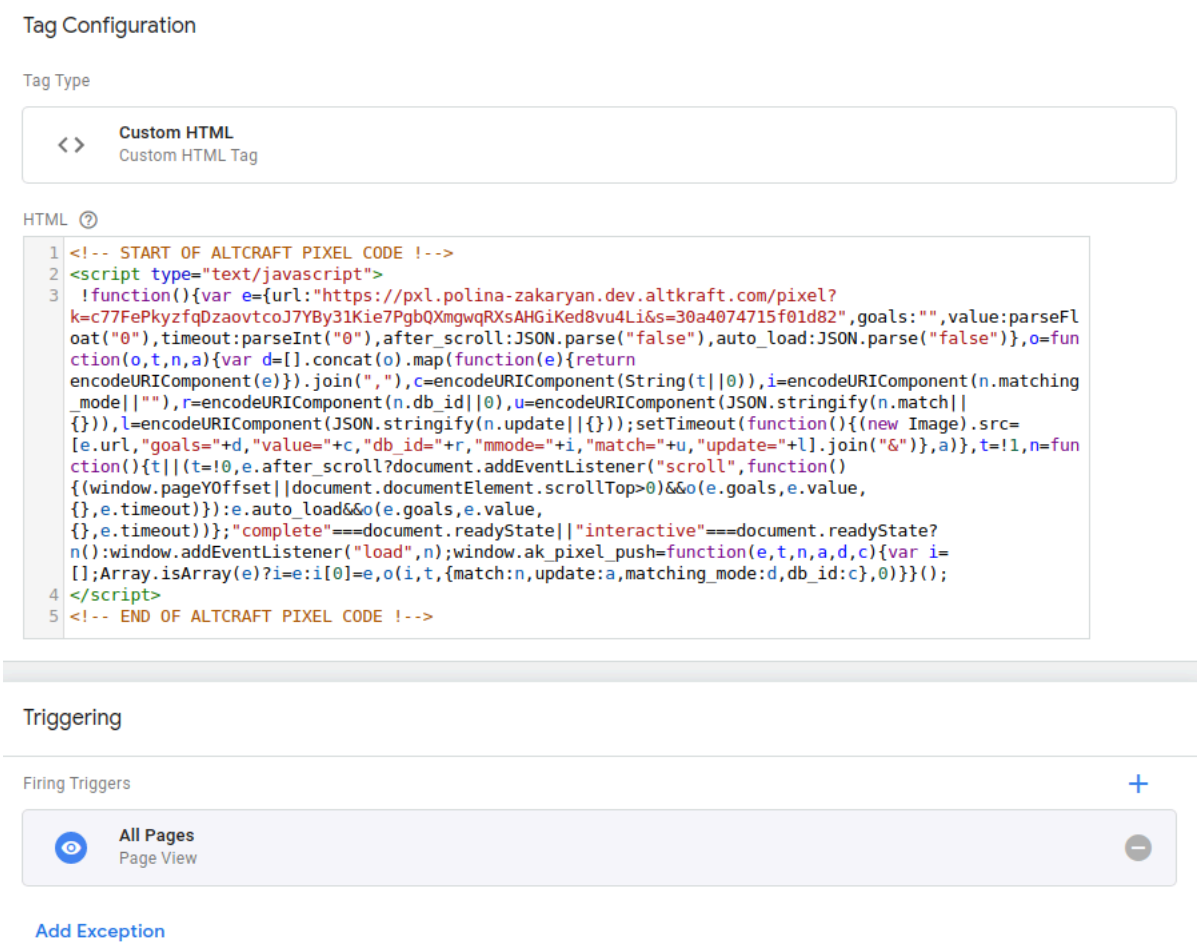
Before copying the pixel code, uncheck the "Auto load pixel" box. Only this way you can use the tracking code as a library.



3. Your next step is to go to the GTM workspace and create a new tag. Choose Custom HTML as a tag type.



4. Add the Altcraft pixel code to the HTML field in the Tab Configuration window. Choose All pages in the Triggering tab. Save it.



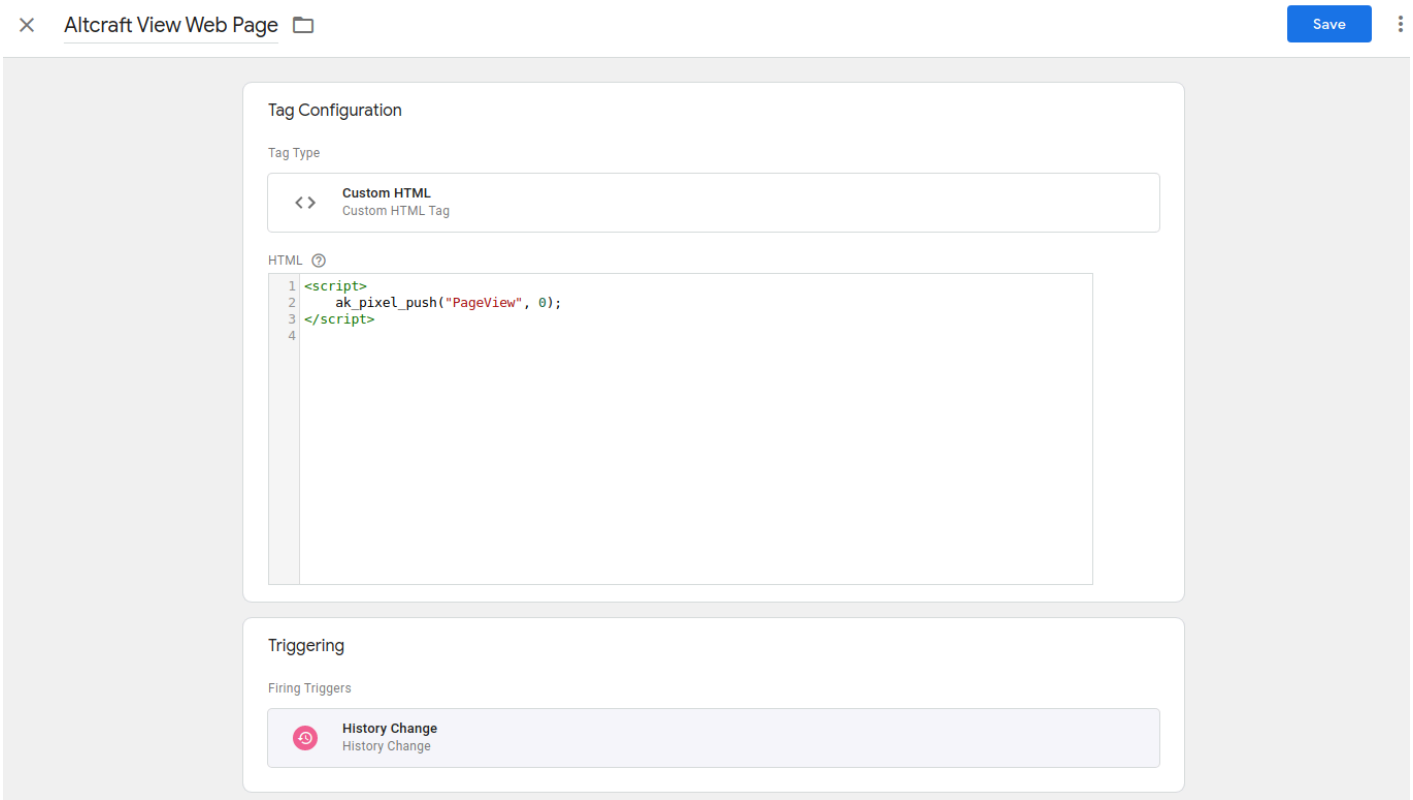
When you submit the changes, the container with Altcraft pixel code will be loaded on all pages of your site.

5. Now, to transfer goals and, if necessary, their values to Altcraft, add the required number of Custom HTML Tags configured for the Triggers you need.

```
<script>
        ak_pixel_push("PageView", 0);
</script>
```

CAUTION

Arguments are added by analogy with [Recording goals via API methods](#).



Goal customer actions and scoring

Description

Altcraft pixels allow you to track customer actions on third-party network resources, such as:

- **Websites and landing pages, including affiliate ones.**
- **Forums, social networks, blog posts and similar services.**
- **Mobile and desktop applications.**
- **CRM and integration services of the company.**

Each registered client action contains two parameters:

- **Goal — the name of the goal action;**
- **Value — the numeric value related to the goal action.**

At the same time, the Pixel itself groups goals by type or traffic source, which can be used in various mechanics described in this article.

INFO

Symbols available for use in goal names:

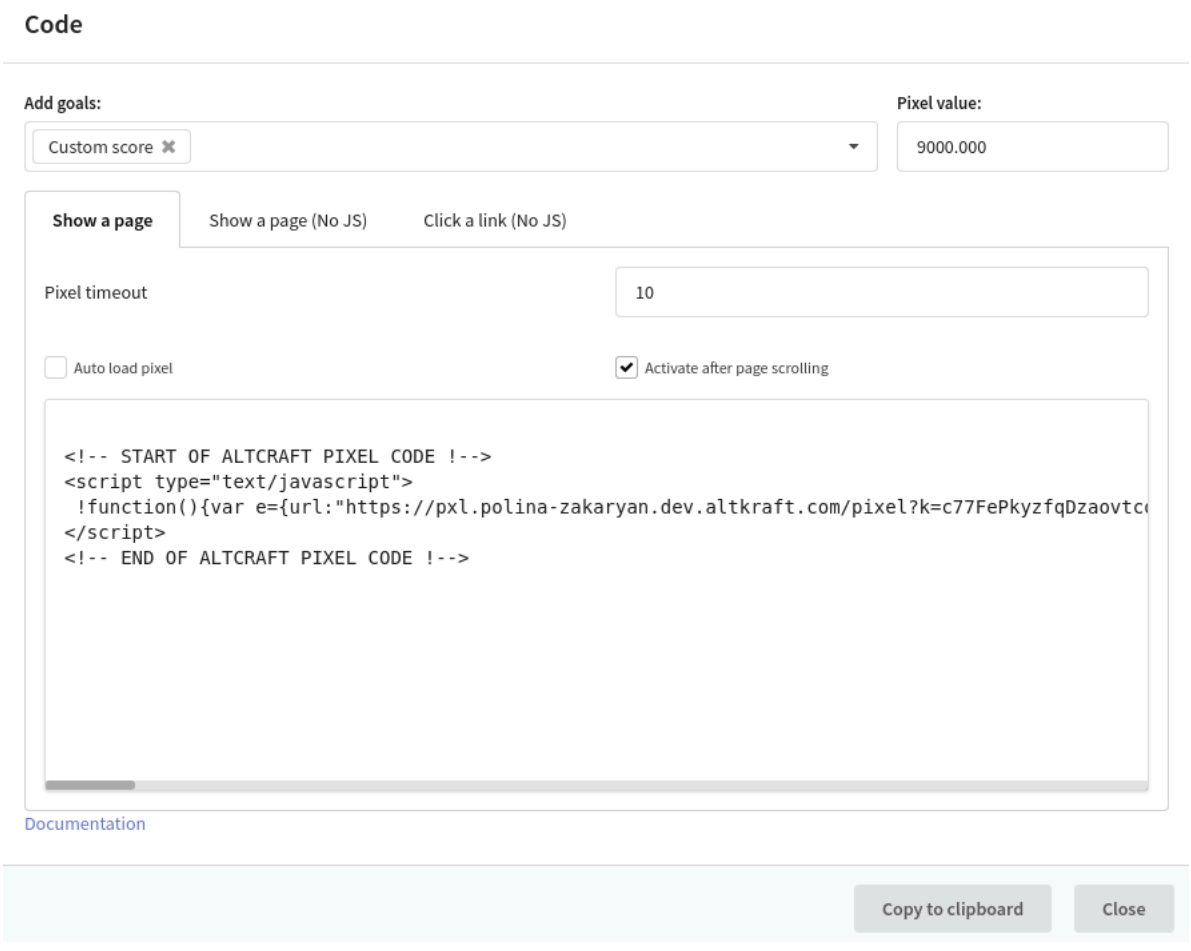
- **letters of the Latin and Russian alphabet (upper and lower case)**
- **spaces and tabs**
- **currency symbols**
- `! # $ % & * + - . , : ; '`
- `= ? @ ^ _ ( ) [ ] { } / \ |`

The maximum length is 128 characters.

Goal actions during viewing a web page

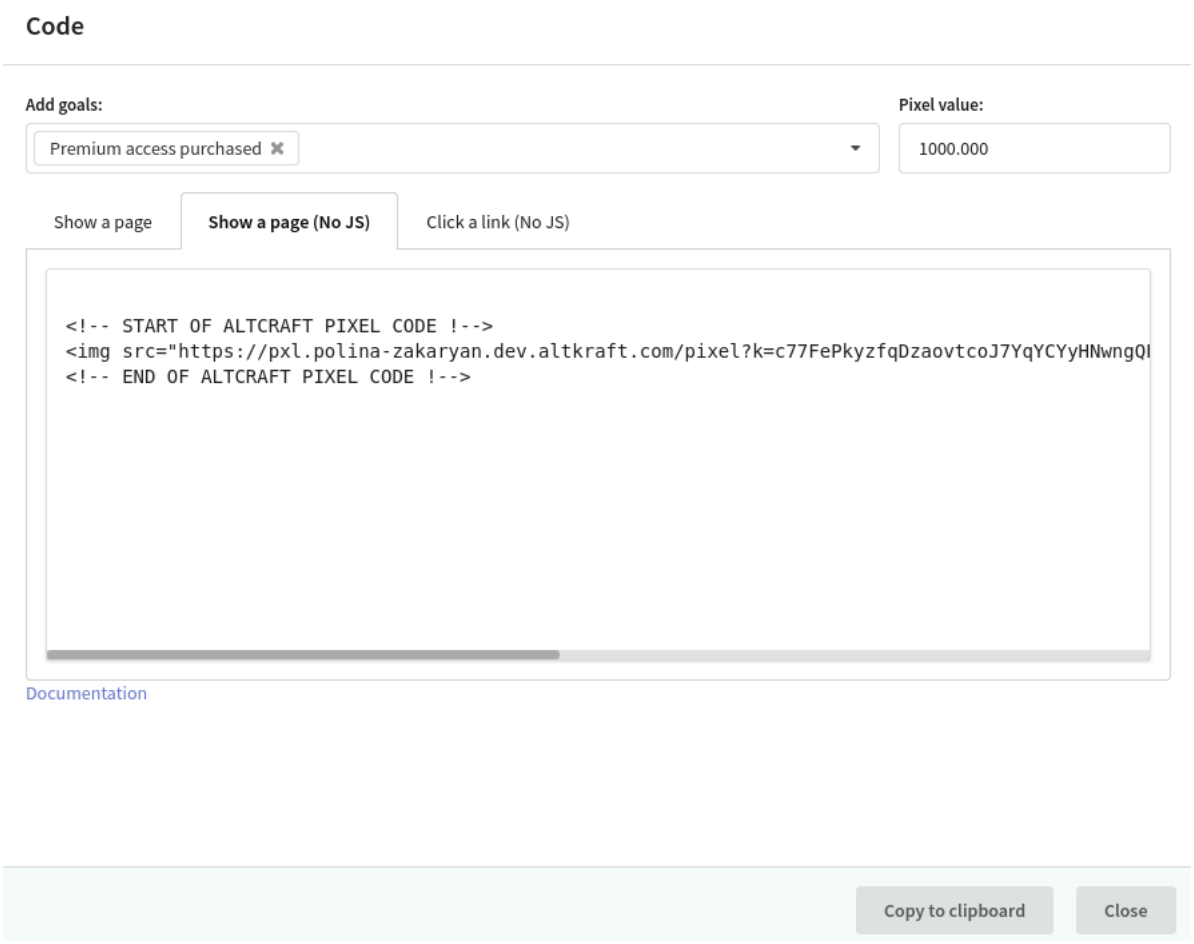
To register goal actions the visitors complete on a specific page when they view it, you can use two types of scripts:

- **Show a page — JavaScript is used for this kind of tracking. Script activation is configurable, for example, you can set it to trigger only after the user scrolls the page. Select the required pixel, click "Get pixel code" and in the appropriate tab configure the script parameters: Goals, Value, Timeout and scroll options.**



Then copy the code and give it to the developers of your website.

- **Show a page (No JS) — it is the simplest way. Here the pixel looks like a transparent image, that is stored on the platform's tracking server. You add the code to any part of the webpage. Every time the user loaded the page, the pixel registers the goal action and the value related to it.**



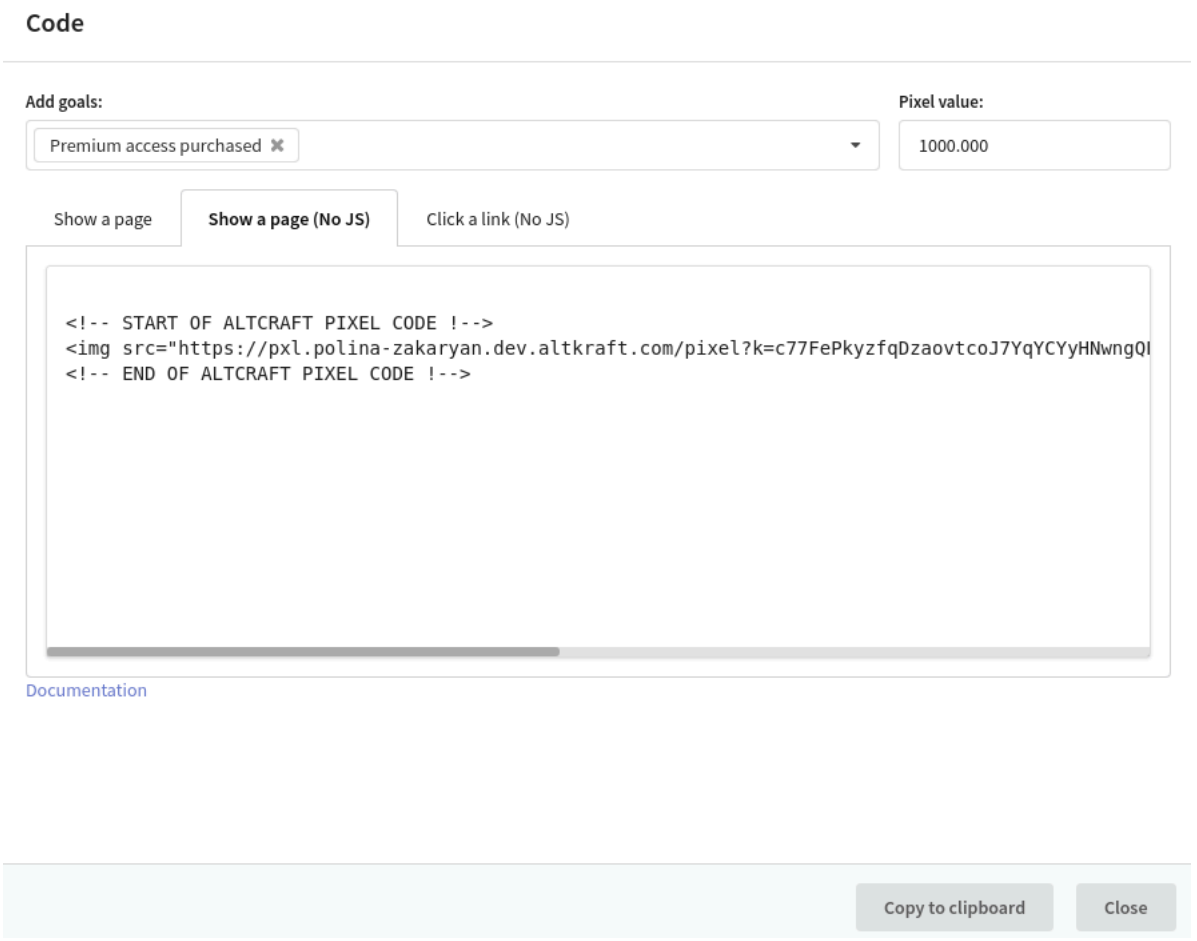
TIP

If you need to register several goals, each with its own value, or if you need a more advanced algorithm for registering goal actions, use the Goal Registration API.

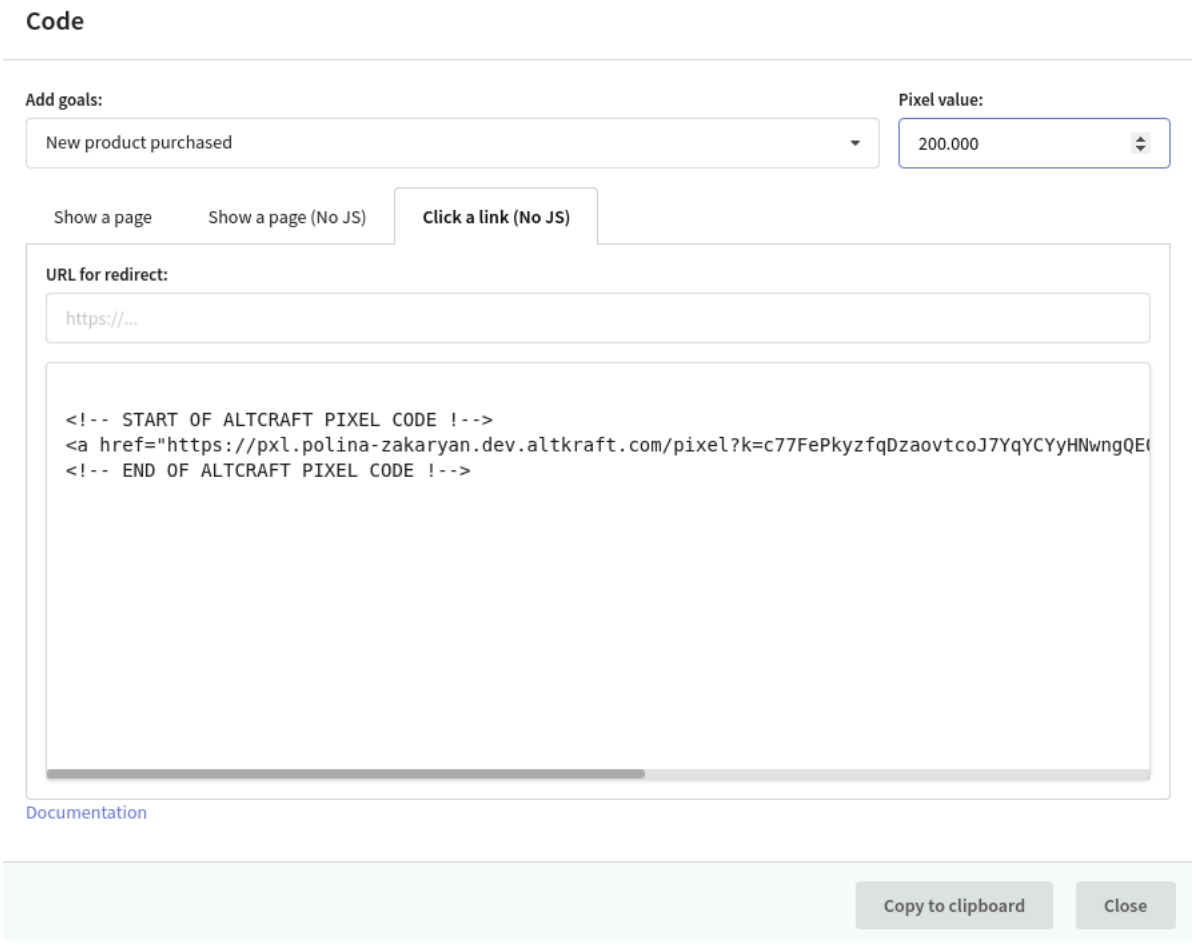
Goal actions in social networks and other public resources

You can use pixels to track the actions of customers on social networks, forums and similar web resources. There are two options:

- **Show a page (No JS) — a transparent image loaded from the platform's tracking server. It's added to a form that supports images from external sources. When the client views such a record, the goal action is registered.**



- **Click a link (No JS) — add a redirect to the platform's tracking server for any link you want to place in a publication - on a social network or on a similar resource. Clicks on such a link will register the specified goal action and its cost.**



Goal actions in desktop, mobile and web applications

In applications tracking of events is carried out by requests to the Goal and Goals Value Registration API. A distinctive feature of the API request is the grouping of goals and costs. Unlike ready-made scripts, “goal” + “value” pairs are supported in the request, so you can pass several goals, each of which will have its own value.

Pop-ups

	Pop-up is a small window that appears on top of the main content of a web page. It is used to attract user's attention to certain information or motivate them to take a targeted action. For example, using pop-ups you can offer to subscribe to a newsletter, tell about a discount or promotion, collect applications or provide technical support.

INFO

Pop-ups appear on the device screen when a certain event occurs — for example, when a page is loaded, when an element is clicked, or when the page is scrolled to a specified threshold. To set the conditions for a popup to appear, install Altcraft Tag Manager on your site.

In this section

Creating and publishing a pop-up

This guide contains instructions for creating and configuring pop-up windows for your website using the Altcraft platform. You will learn how to style the pop-up, set display rules, and analyze the results.

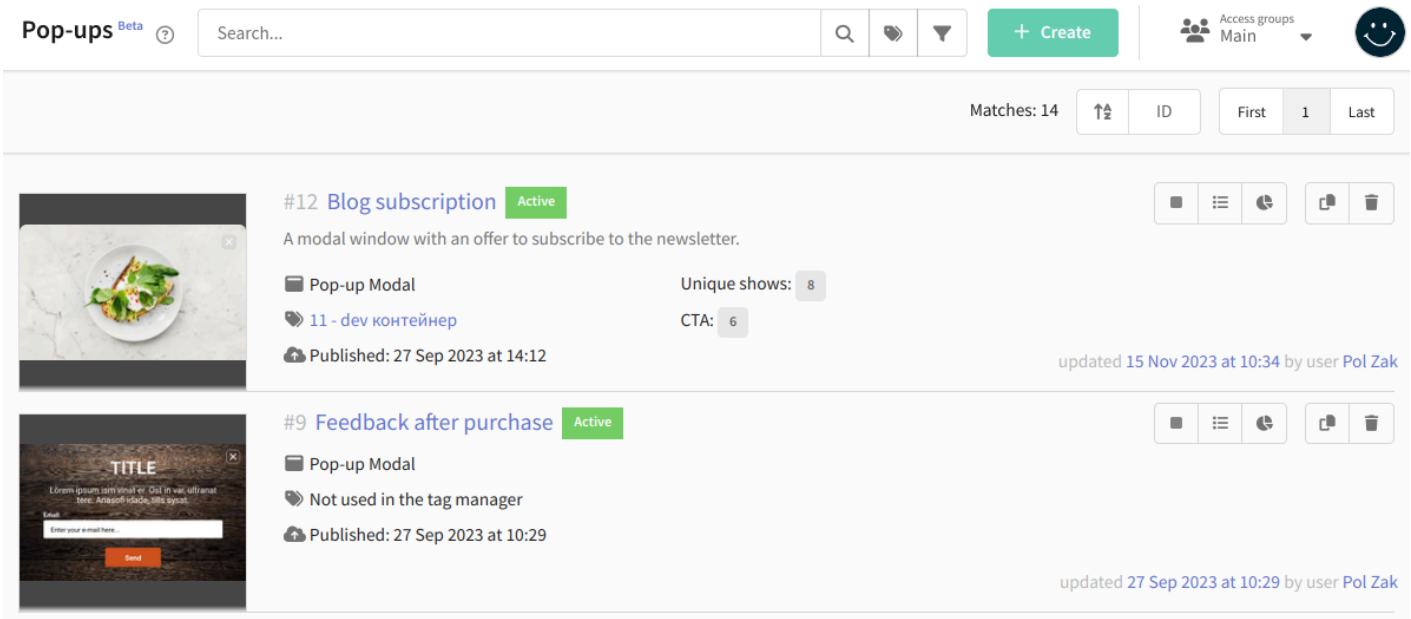
Full workflow for creating and configuring a pop-up

- **Step 1: Creating a pop-up**
  Choose the type of pop-up (modal, floating bar, slide-in, page takeover) and a base template for a quick start.
- **Step 2: Configuring content and design**
  Work in the visual editor: add text, media, forms, and adjust the appearance to match your brand.
- **Step 3: Configuring actions after form submission**
  Define what the user will see after submitting data (message, redirect), and configure notifications.
- **Step 4: Publishing and configuring display**
  Integrate the code into the site and set rules for display (time, triggers, target audience).
- **Step 5: Analytics and results**
  Monitor key metrics (conversion, impressions, clicks) to evaluate effectiveness and make decisions.

Each of these steps is described in detail below.

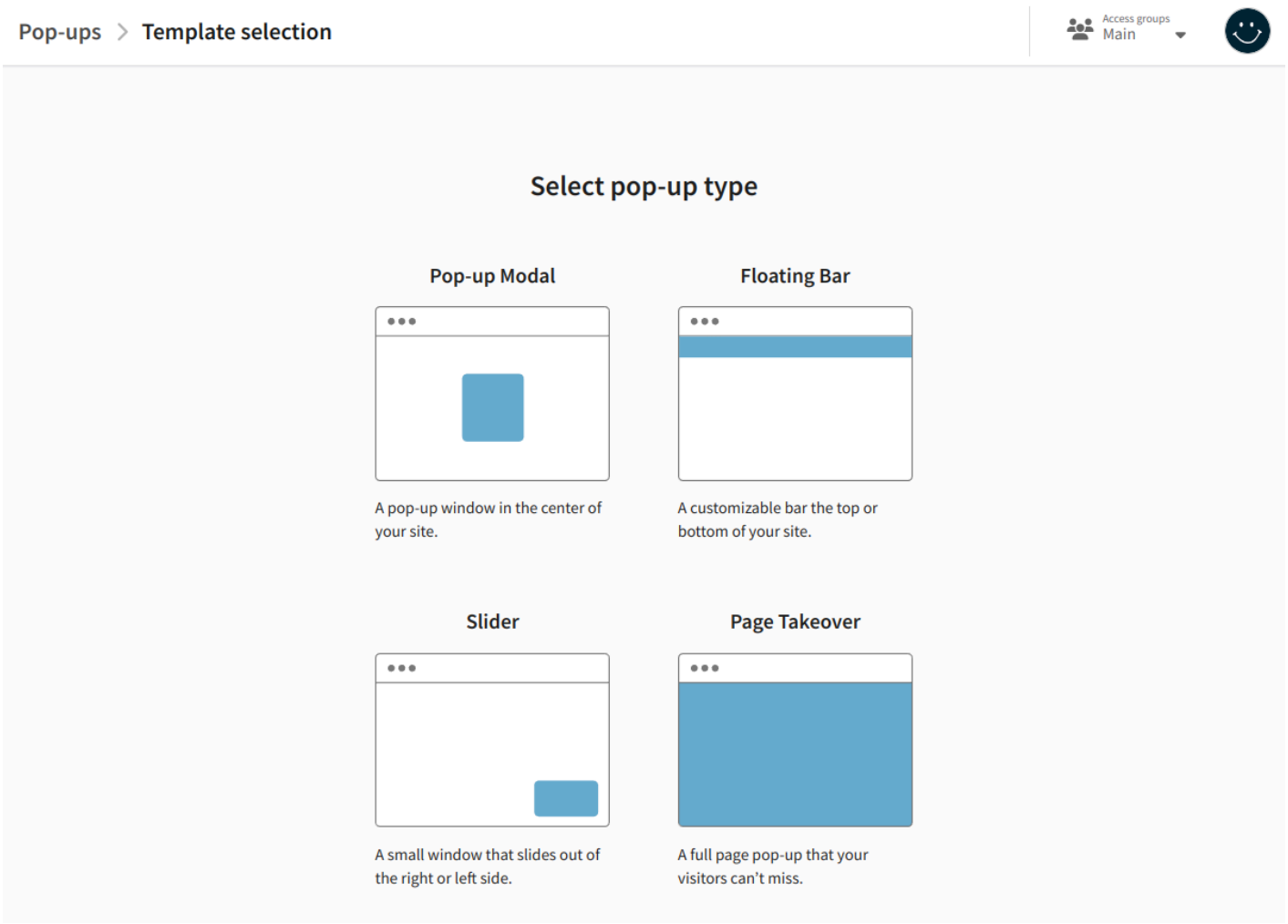
Creating a pop-up

Go to “Weblayers” then “Pop-ups” and click + Create at the top of the page:

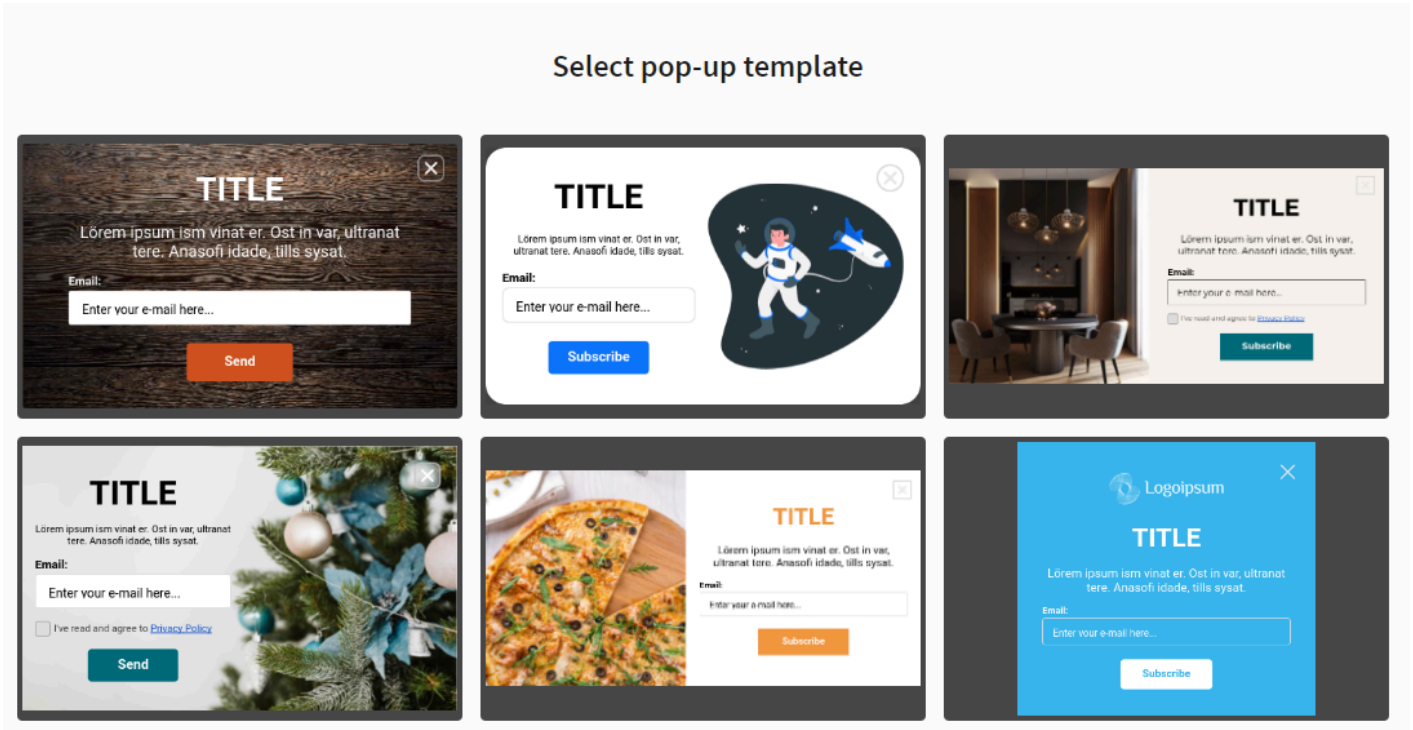


Select the type of pop-up:

- **Pop-up Modal — a pop-up window that appears in the center of the screen. It draws the user’s attention by dimming the background. Typically used to collect email addresses.**
- **Floating Bar — a customizable panel placed at the top or bottom of the page. It remains visible even when scrolling. Suitable for informing customers about news or limited offers.**
- **Slider — a small pop-up window that appears in the bottom-right corner of the page. It is less intrusive and does not interfere with viewing the page content. It can be used to display additional information, offers, or feedback forms.**
- **Page Takeover — a pop-up window that occupies the entire browser screen. This ensures visitors do not miss the information. However, full-screen pop-ups should be used carefully, as they can be perceived as intrusive advertising.**



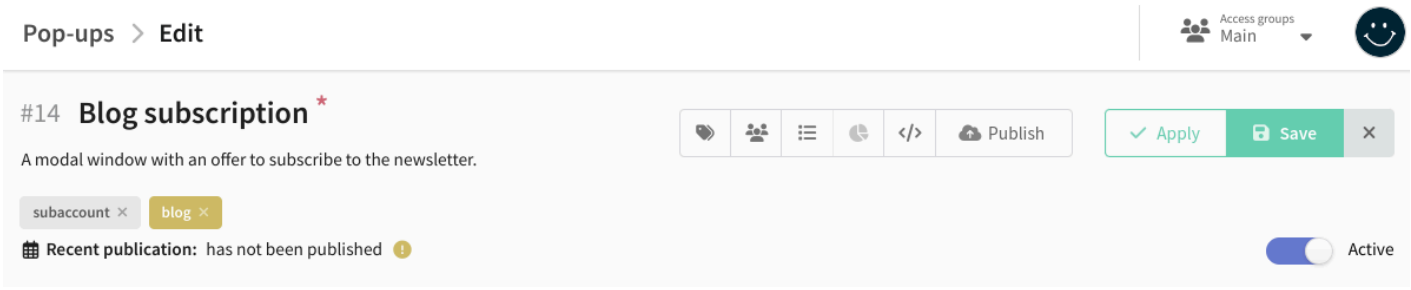
Choose a pop-up template. The platform already includes several ready-made templates. Use any of them as a base. At the next step, you can fully customize the template (text, styles, fields, etc.) for your tasks.



Set the system name of the pop-up and, if necessary, add a description. You can also set tags for quick search

and assign access groups

.



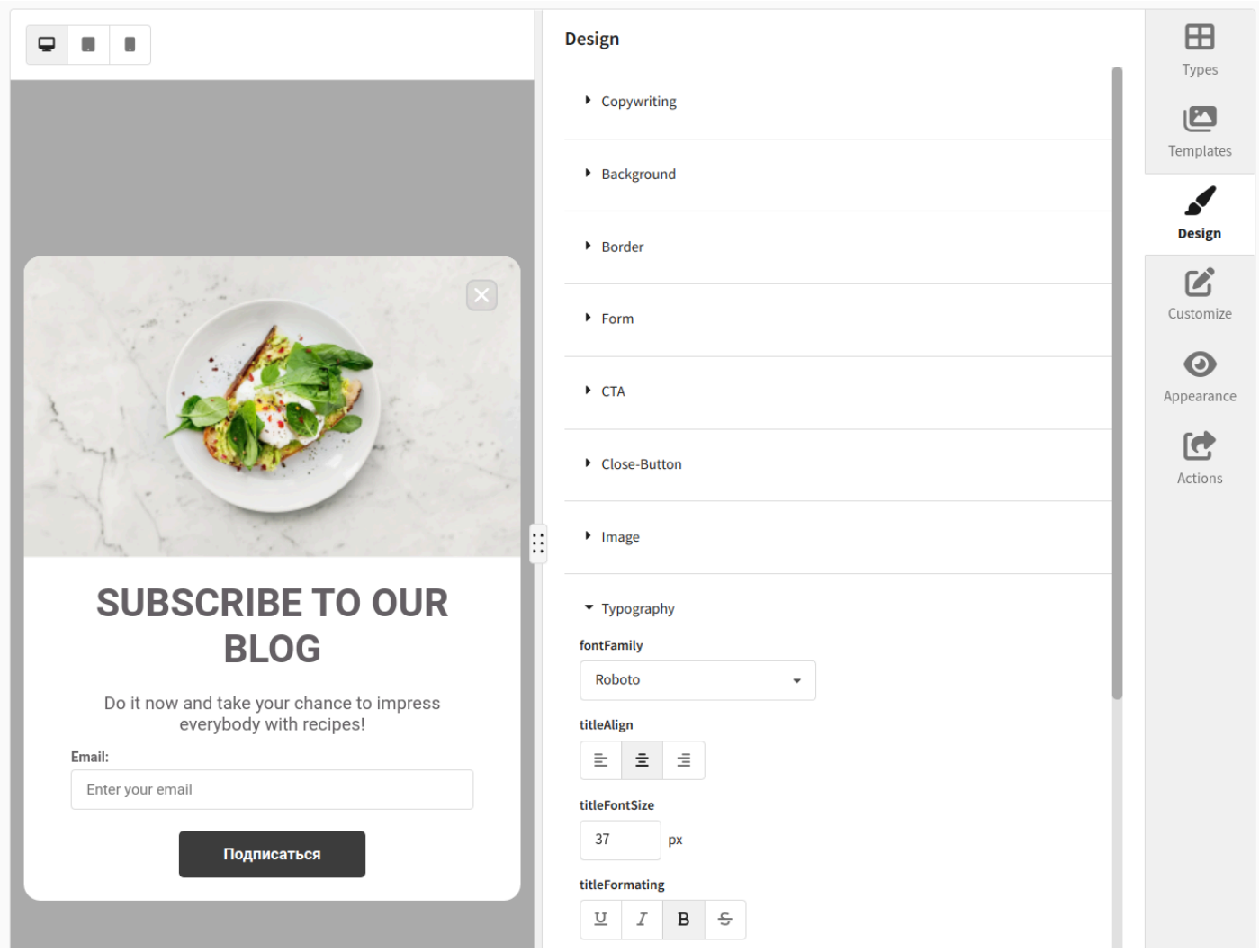
Configuring content and design

The appearance of a pop-up is defined by parameters. Parameters include text, images, and CSS styles. For example, `text` contains the main body text, and `backgroundColor` defines the background color.

Two tools are available for editing pop-up content:

- **Visual editor. Suitable for users without coding knowledge. Simply enter text, select images from the gallery, or use color palettes.**
- **Code editor (HTML, CSS, JS). Suitable for advanced users who want to edit the pop-up code directly.**

All changes are displayed in the preview window in real time. Use the buttons in the upper left corner to test on different devices.



Visual editor

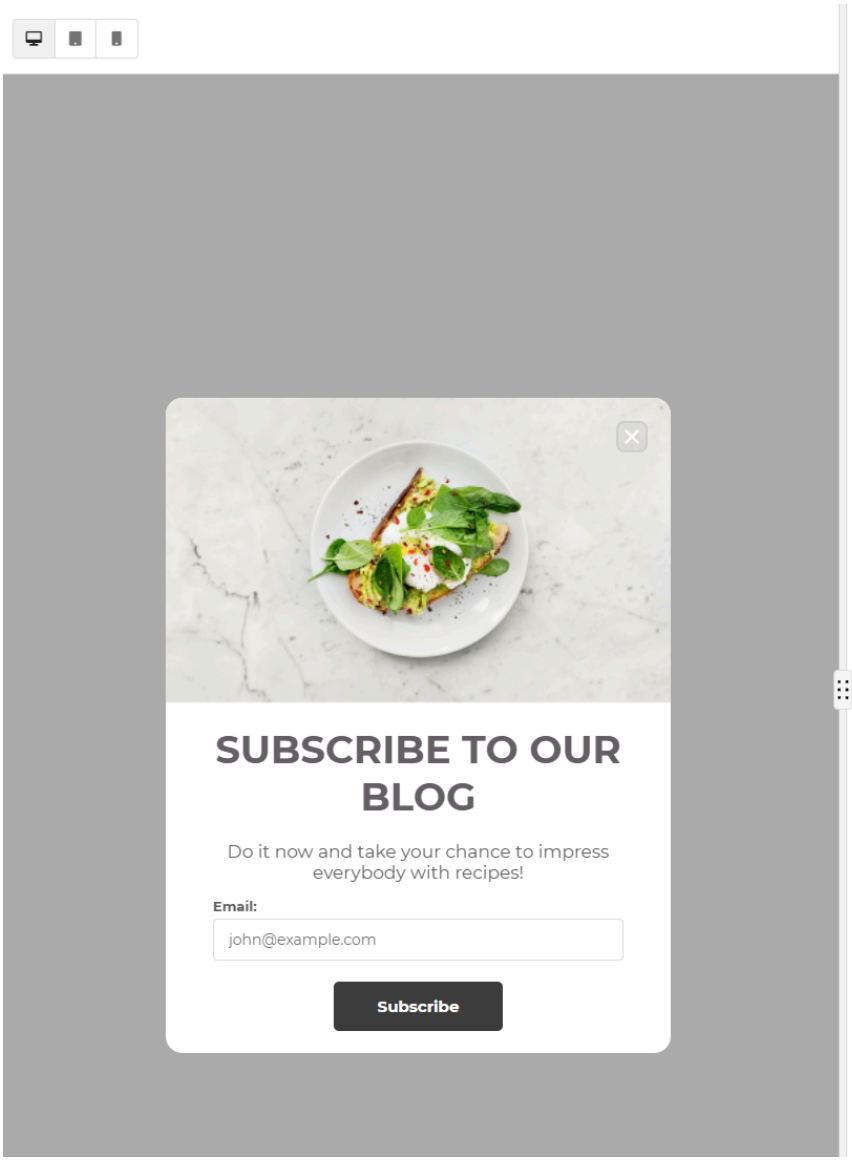
To customize text and design in the visual editor, go to Design. Parameters are grouped into categories:

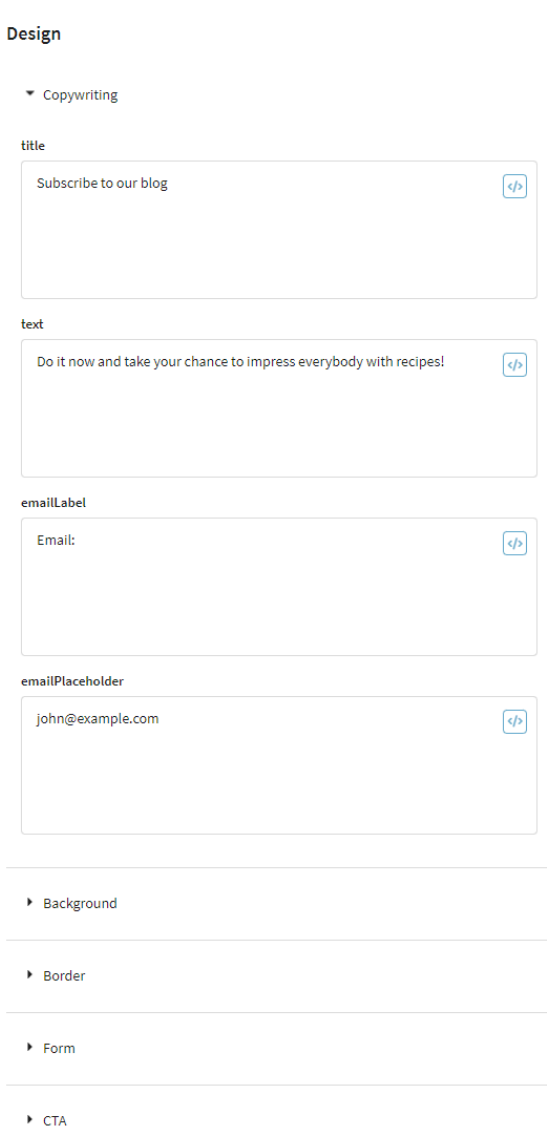
Copywriting

On the Copywriting tab, set the pop-up text. Available parameters:

- **title — heading;**
- **text — main text;**
- **<field>Label — label for form fields;**
- **<field>Placeholder — placeholder inside form fields.**

You can insert a `</>` variable into any text element to dynamically pull profile data. See [this article].

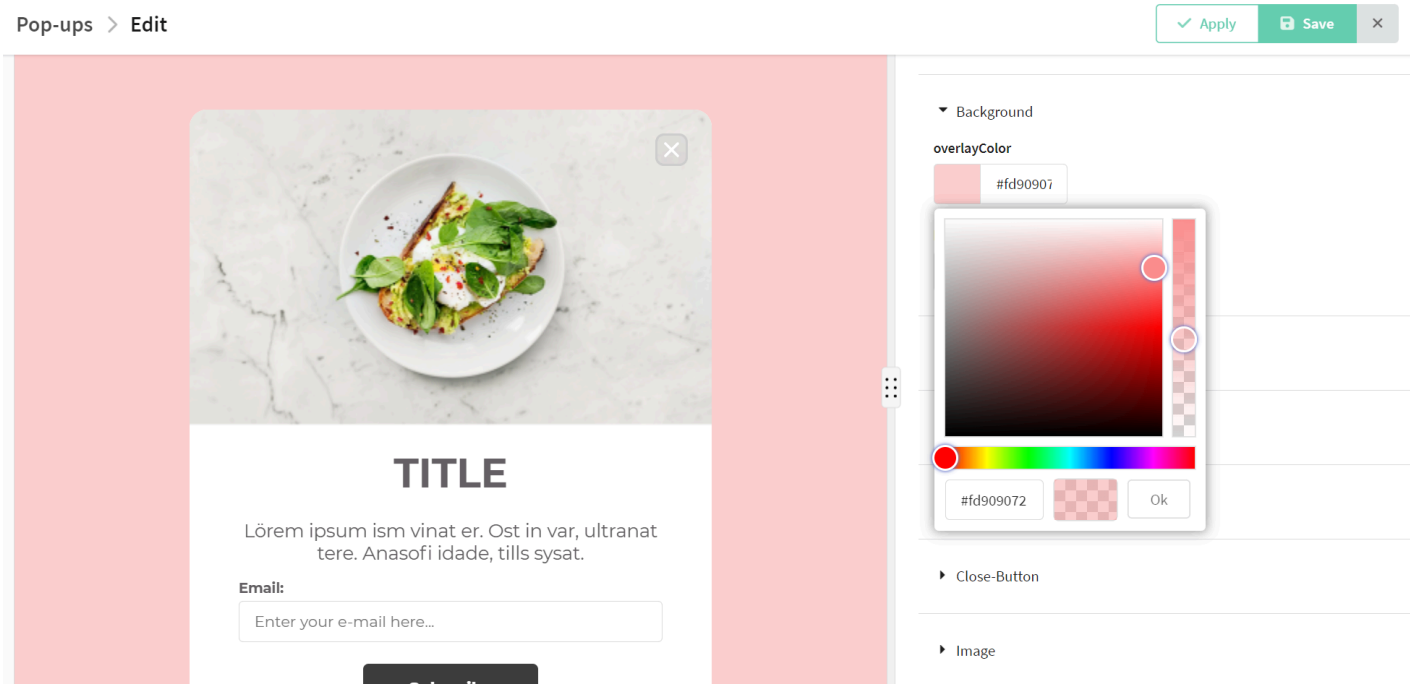




Background

- **backgroundColor — pop-up background color;**
- **backgroundImage URL — background image (upload to gallery**

  or specify URL);
- **overlay — background overlay for the page (color and opacity).**

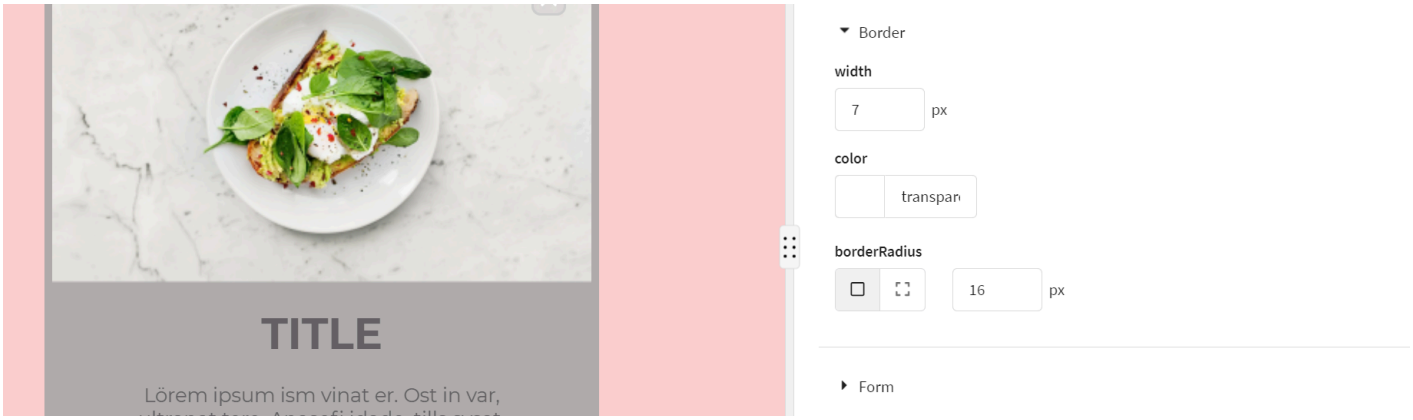


Border

Configure pop-up borders:

- **borderRadius — rounding (px, per corner)**

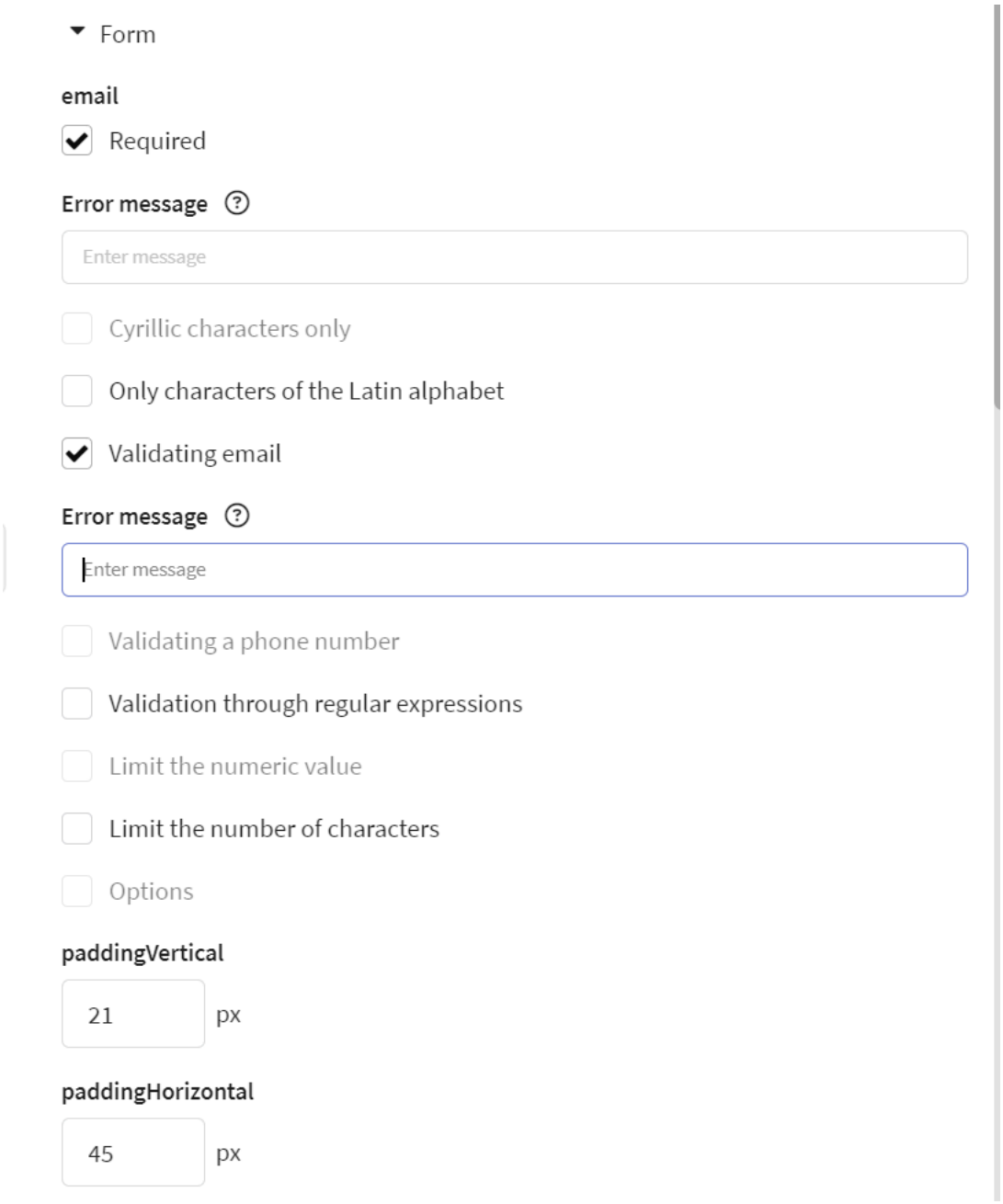
  ;
- **width — border thickness;**
- **color — border color.**



Form

Configure form fields:

- **set up validation rules;**
- **specify error messages shown to the user if input is invalid;**
- **edit CSS styles of form fields.**



Validation options:

- **Required — the user must fill in the field before submitting the form. If left empty, an error message will appear;**
- **Cyrillic only — only Cyrillic characters are allowed;**
- **Latin only — only Latin characters are allowed;**
- **Email — the value must match the email address format;**
- **Phone — the value must match the phone number format;**
- **Regex — you can apply your own custom regex validation rules;**
- **Numeric limits — define minimum and maximum values for numeric fields;**
- **Character limits — set the maximum number of characters allowed;**
- **Predefined answers — for fields with a limited set of options. The user must choose one of the allowed answers. If not, an error message will appear;**
- **Error message — the text shown to the user when input doesn't meet validation rules. Each field can have its own error message.**

Window styles:

- **paddingVertical — spacing between the button and the bottom border of the popup;**
- **paddingHorizontal — spacing between the fields and the left border of the popup;**
- **width — width of the popup;**
- **fieldsGap — spacing between form fields, including space between the last field and the button;**
- **borderWidth — border width of input fields;**
- **borderColor — border color of input fields;**
- **borderRadius — border radius of input fields. Radius can be set individually for each corner**

.

Input field styles:

- **inputPaddingVertical — vertical padding of the placeholder text inside the input field;**
- **inputPaddingHorizontal — horizontal padding of the placeholder text;**
- **inputFontSize — font size of the placeholder text;**
- **inputBackgroundColor — background color of the input field;**
- **inputColor — color of the user's text input;**
- **inputHoverColor — border color of the input field on hover;**
- **inputActiveColor — border color of the input field on click.**

Checkbox styles:

- **checkboxSize — size of the checkbox;**
- **checkboxLabelFormatting — formatting of the checkbox label text.**

Error styles:

- **errorColor — color of the error message;**
- **errorFontSize — font size of the error message;**
- **inputErrorBorderColor — border color of the input field when showing an error;**
- **inputErrorBackgroundColor — background color of the input field when showing an error.**

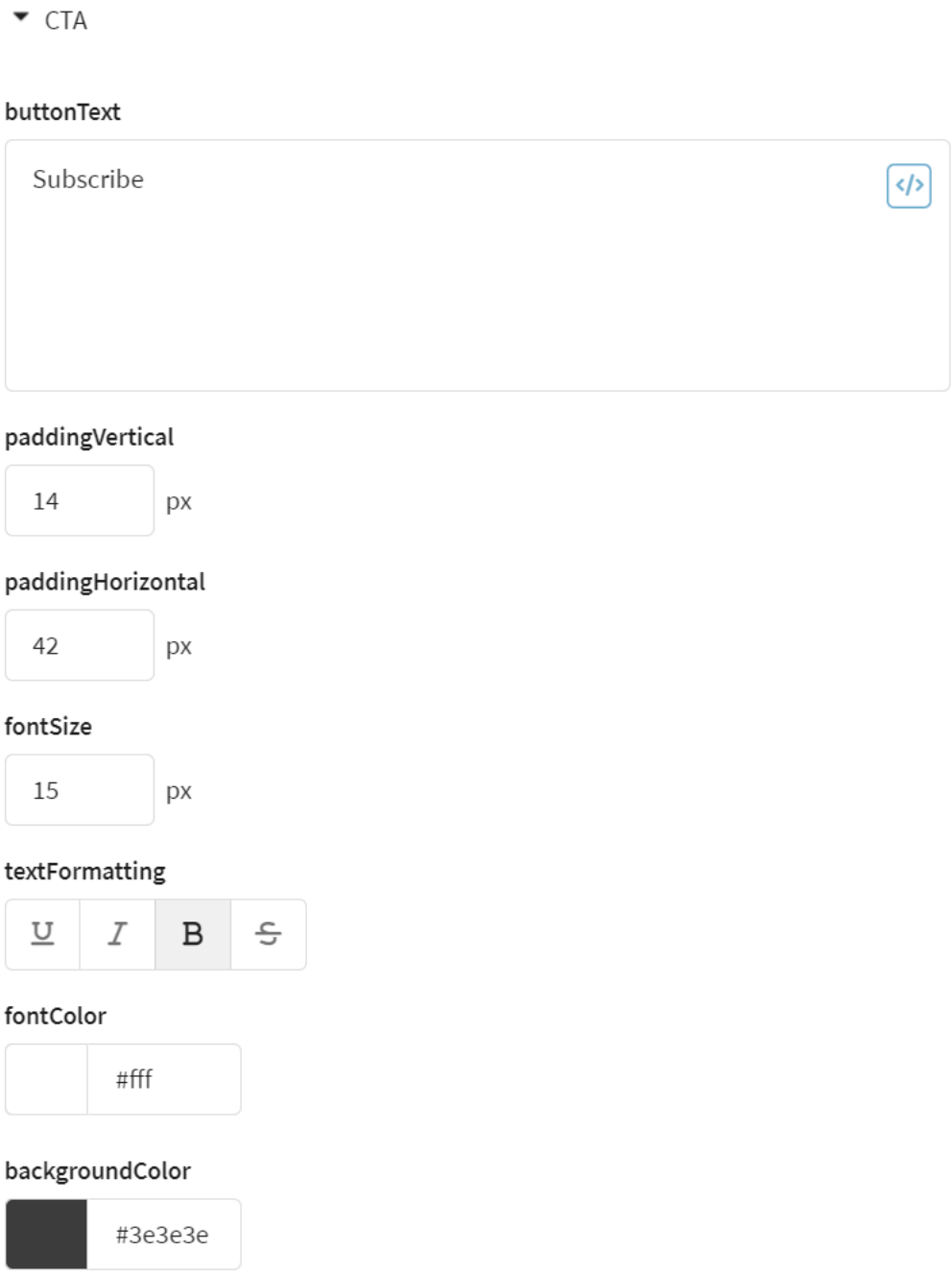
Note: only one form can be created in the visual editor. To add more, use the code editor.

Actions

Edit the submit button:

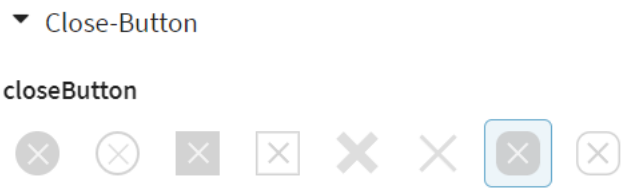
- **text — button text;**
- **textFormatting — formatting of the button text;**
- **borderRadius — rounding of the button corners. Radius can be set individually for each corner**

  ;
- **backgroundColor — background color of the button;**
- **hoverBackgroundColor — background color on hover;**
- **color — text color;**
- **fontSize — text size;**
- **paddingVertical — vertical padding inside the button;**
- **paddingHorizontal — horizontal padding inside the button.**



Close Button

Set the appearance of the close button (closeButton).

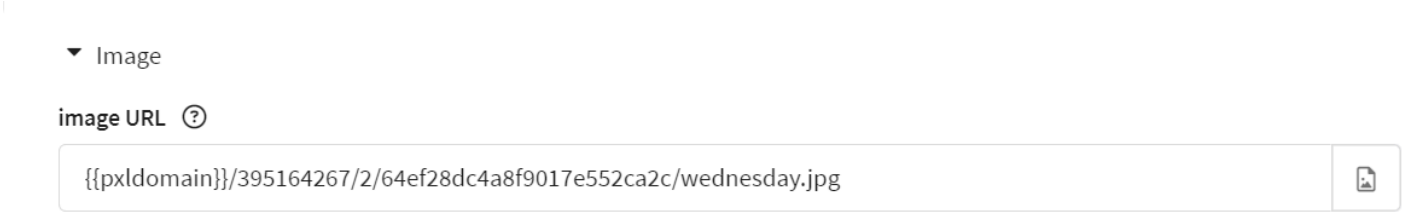


Image

Add an image (not all templates support it):

- **image URL — image URL or upload from the platform gallery**

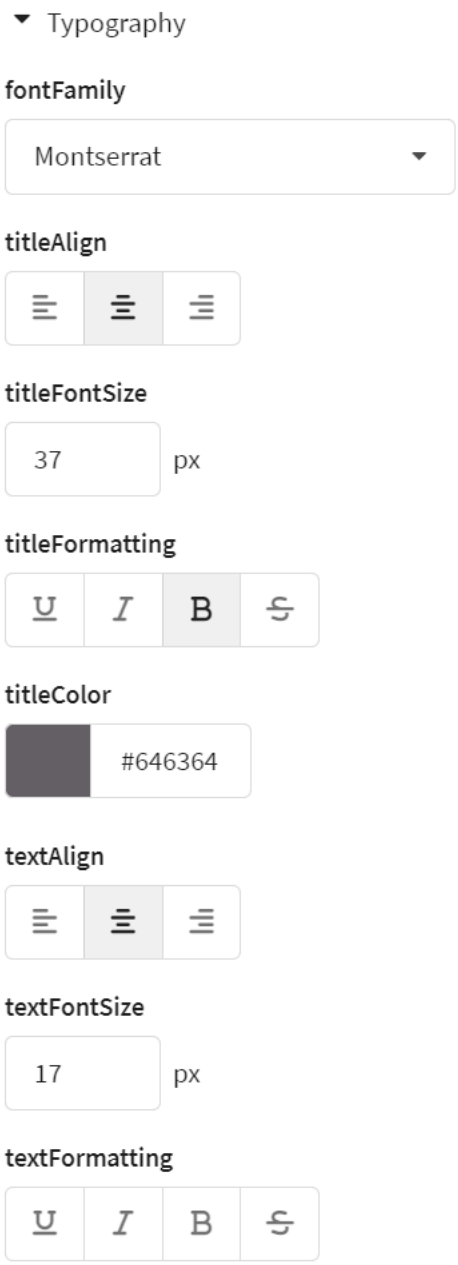
  ;
- **width — width in px, height scales proportionally.**



Typography

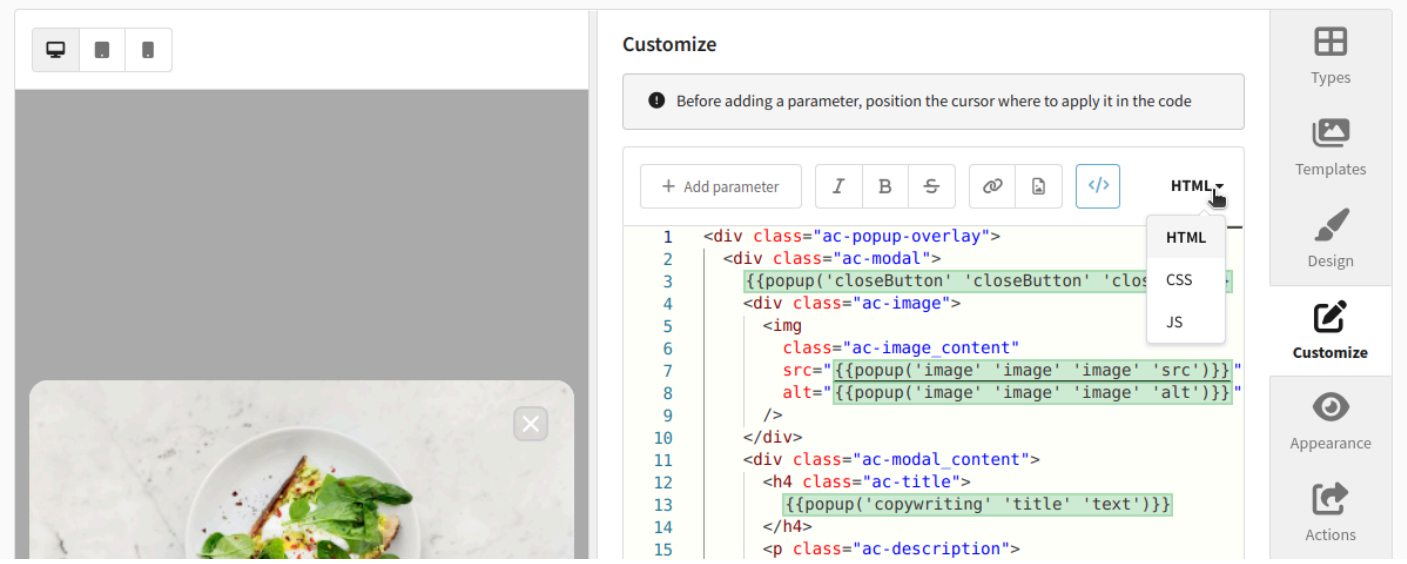
Change text styles for all text elements:

Category	Parameter	Description
Global font	**fontFamily**	**Font family used for all text elements in the popup.**
Title	**titleAlign**	**Alignment of the title (left, center, right)**
	titleFontSize	**Font size of the title**
	titleFormatting	**Text formatting of the title (underline, italic, bold, strikethrough)**
	titleColor	**Color of the title text**
Main text	**textFontSize**	**Font size of the main text**
	textAlign	**Alignment of the main text (left, center, right)**
	textFormatting	**Text formatting of the main text (underline, italic, bold, strikethrough)**
	textColor	**Color of the main text**
Label	**labelFontSize**	**Font size of the label text**
	labelFormatting	**Text formatting of the label (underline, italic, bold, strikethrough)**
	labelColor	**Color of the label text**



Code editor

Go to Settings to edit HTML/CSS/JS directly. See [this article](#).



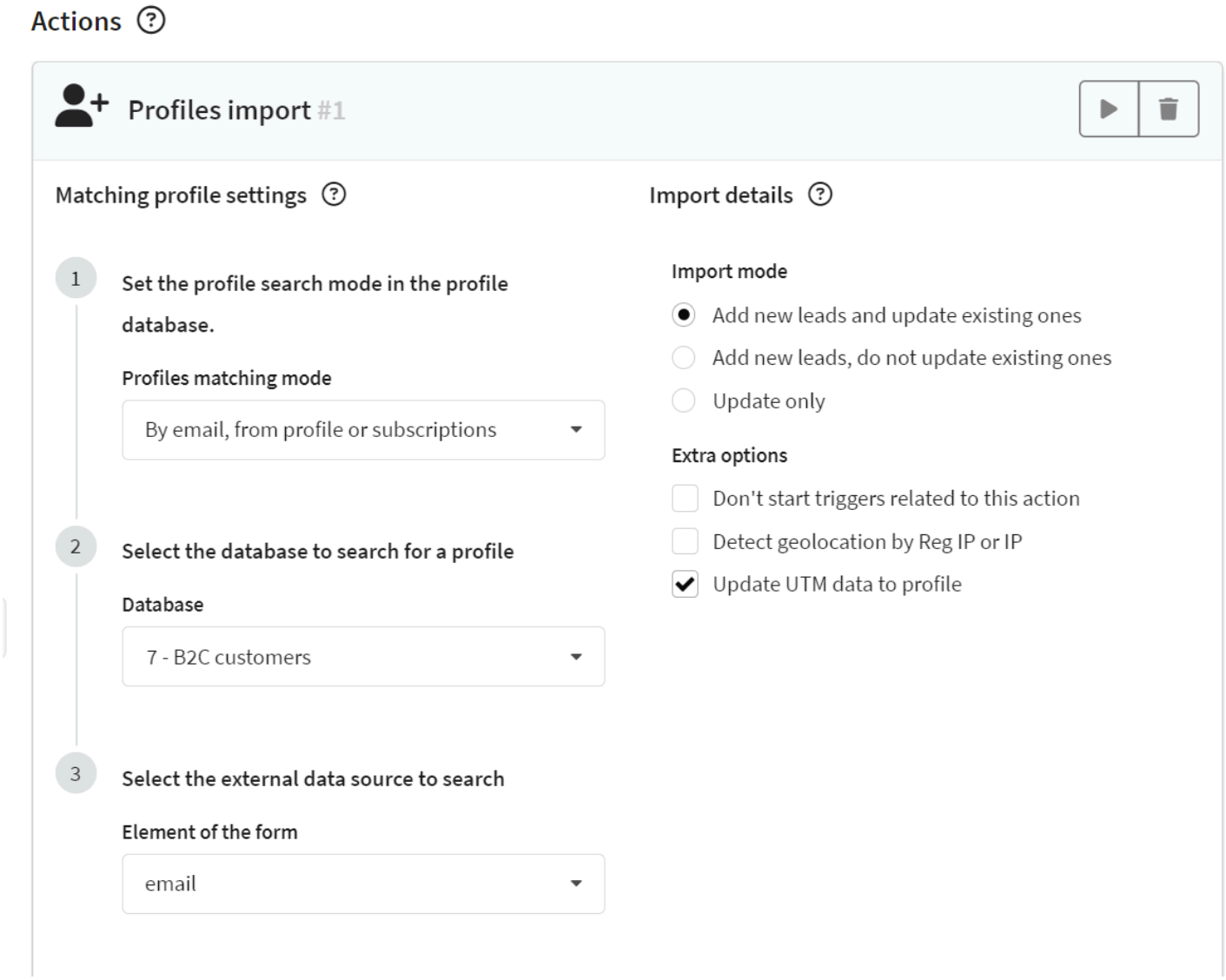
Configuring actions after form submission

Go to Actions. Define actions after the user submits the pop-up form:

- **Import profiles — save data into one or more databases.**
- **Send to email — send submission notifications.**

TIP

Post-submit actions for pop-ups are the same as for standard web forms. See [this article](#).



Configuring publication and display

Two display mechanisms:

- **Manual script insertion — direct control, implement display logic yourself.**
- **Tag Manager — easier: place container code once, configure rules in UI.**

Manual script insertion

Save settings, set status Active, click Publish:

In additional actions, click

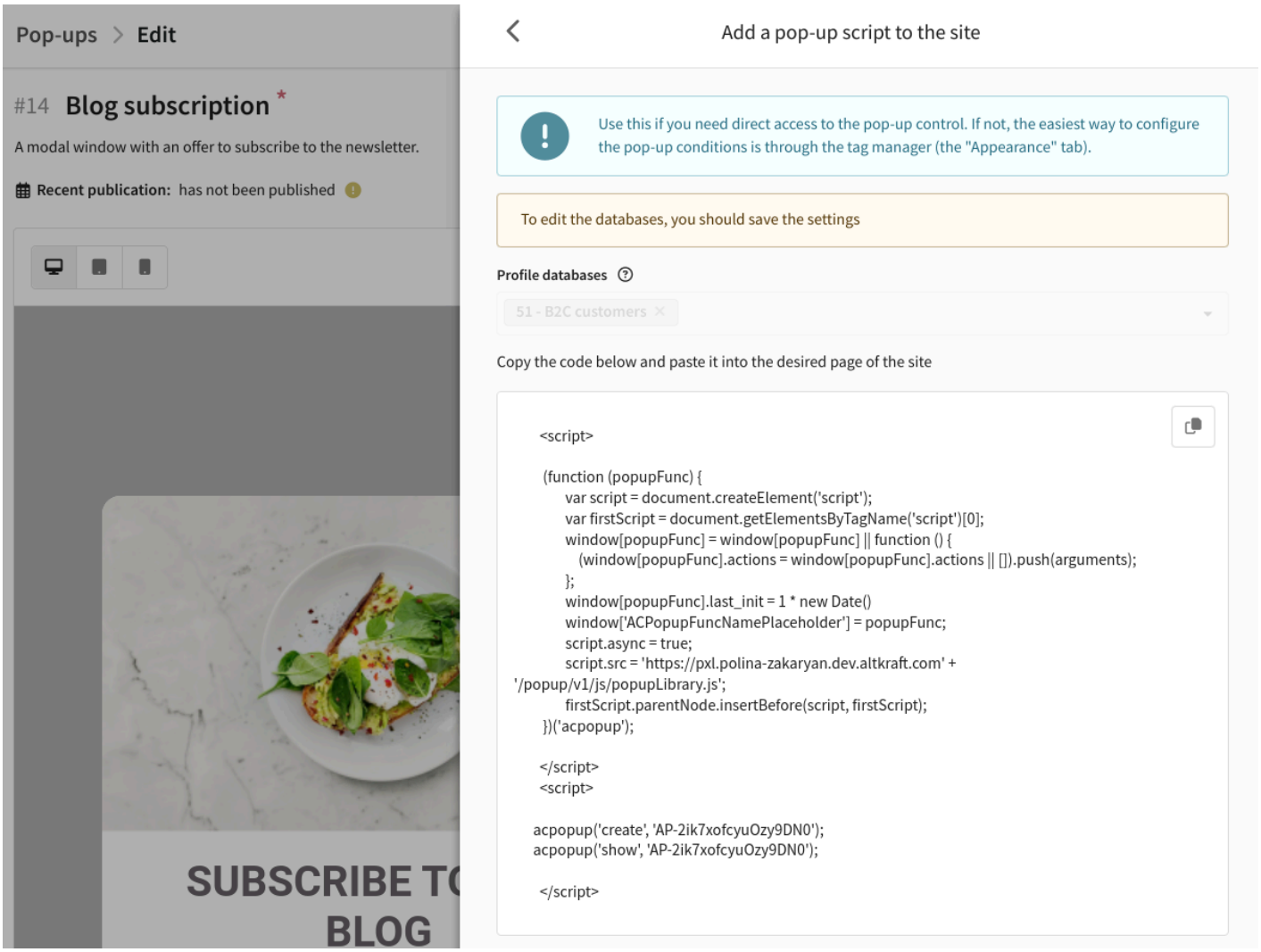
</>

. Select databases for profile search. Matching data can be passed with a server data request. If found, data is updated; otherwise, a new profile is created.

CAUTION

Databases here are only for searching existing profiles. Data from form submissions goes to the database specified in Actions.

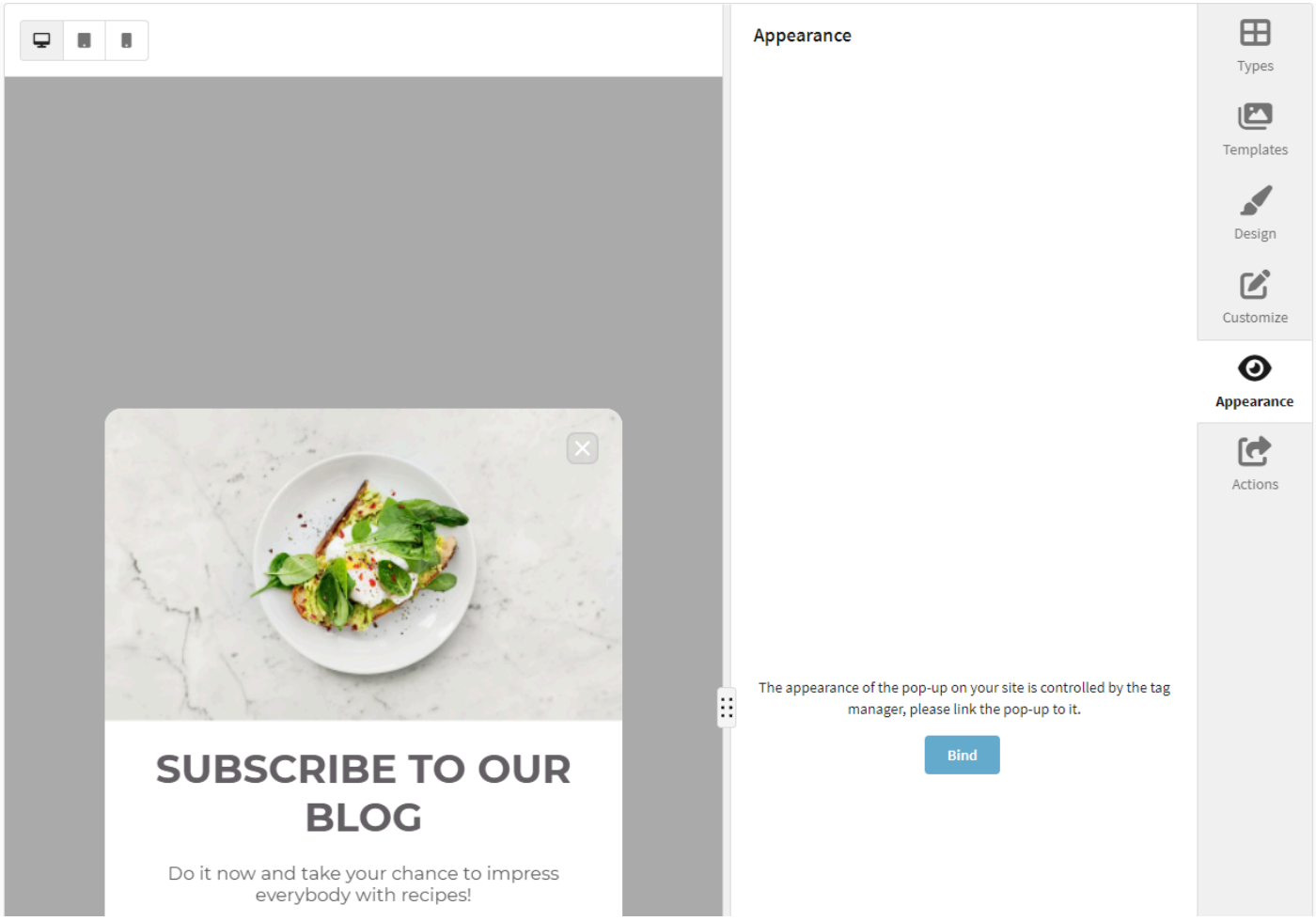
Click Show code then copy and pass to developers:



Tag Manager integration

Binding to a container

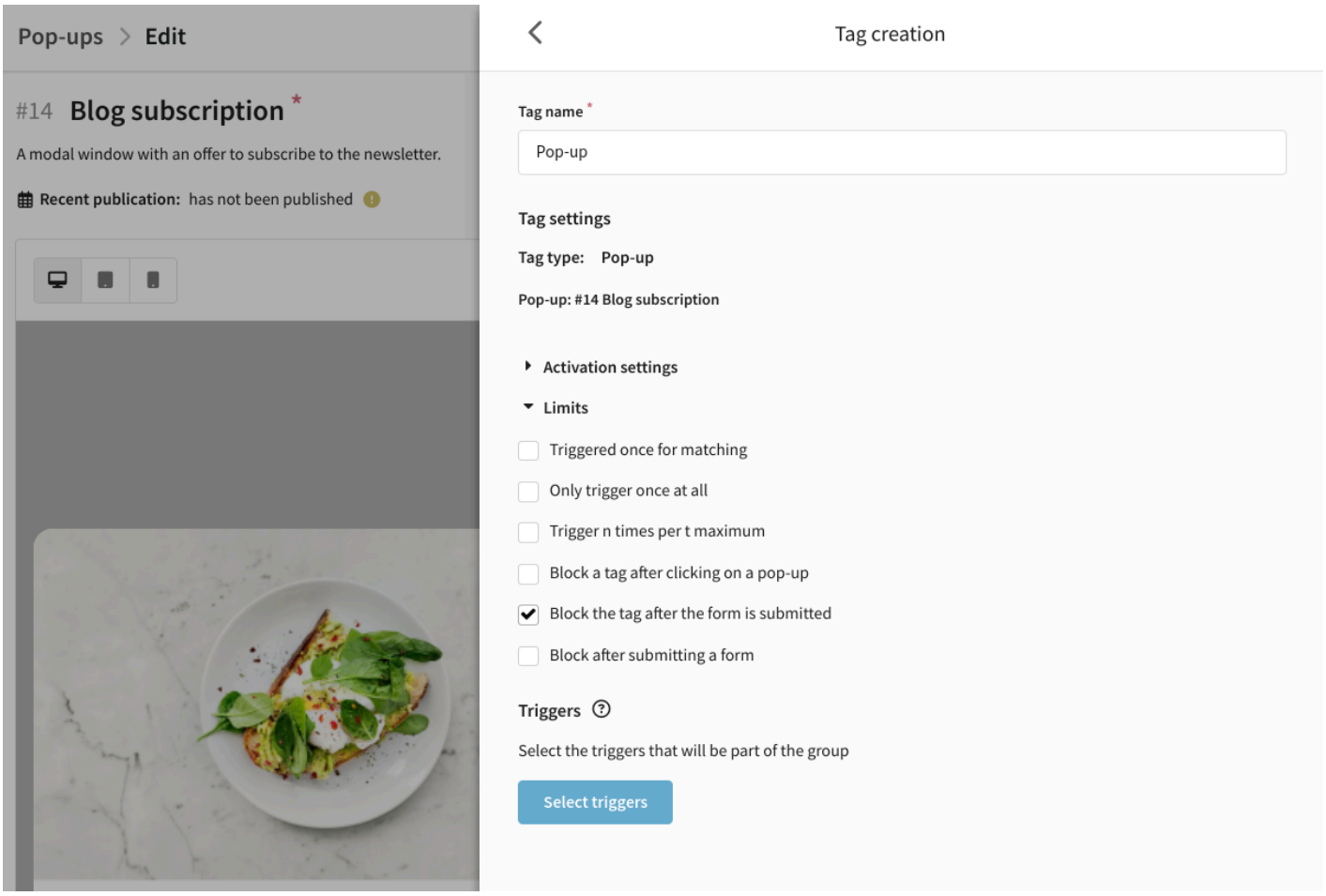
Save the pop-up. Go to Appearance then click Bind:



Select or create a container. Enter a tag name.

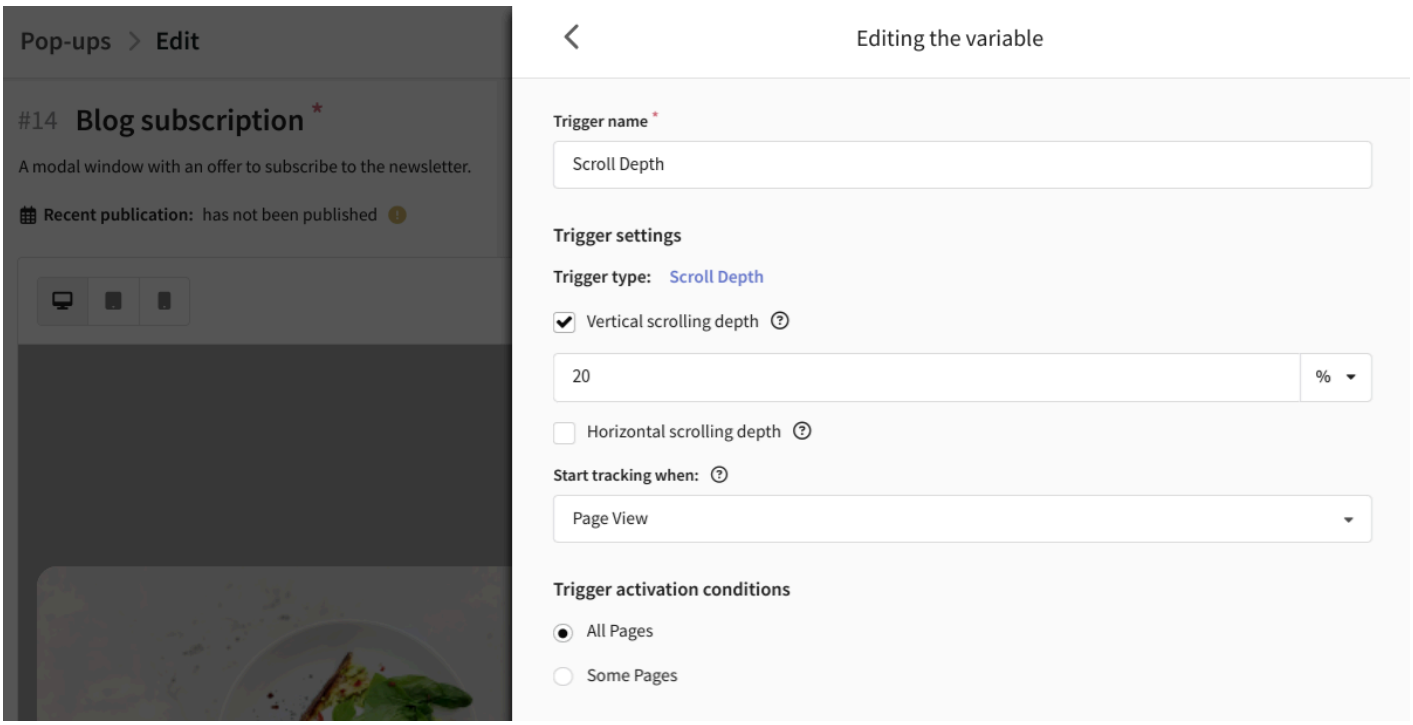
Tag activation settings

Set activation rules and limits:

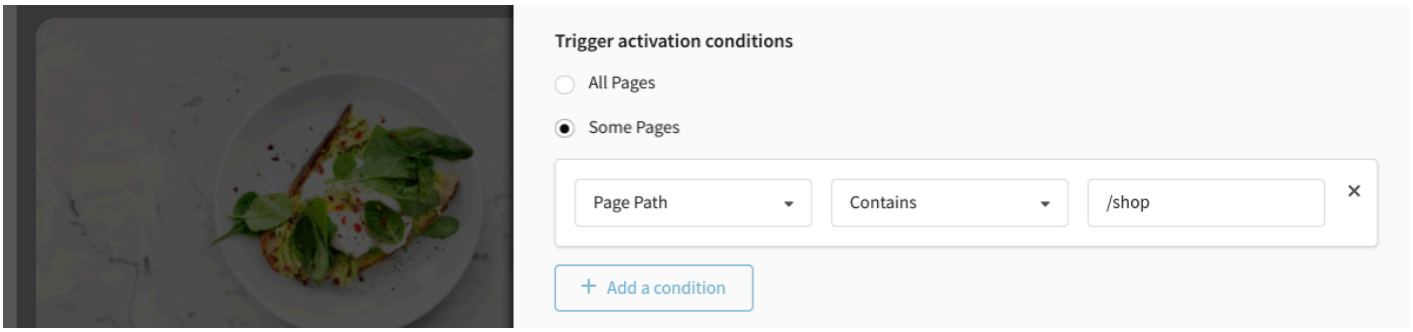


Configuring triggers

Select or create a trigger (page load, click, etc.):

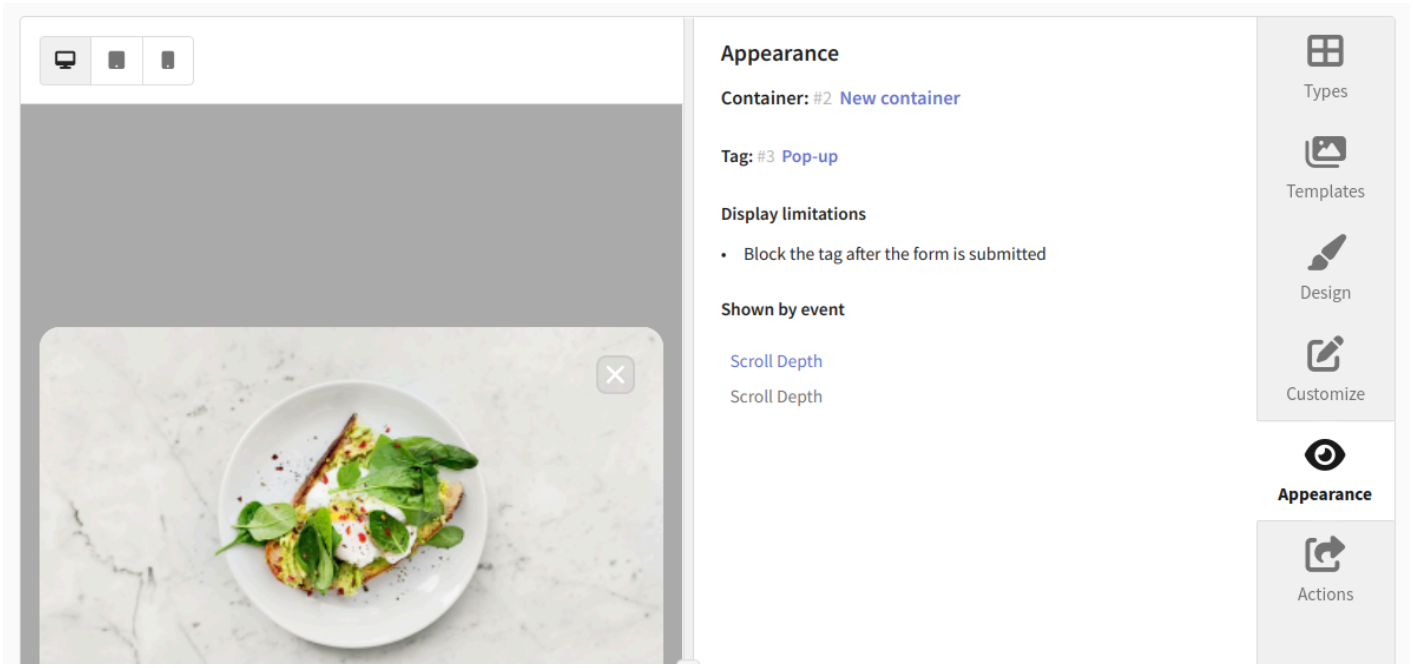


For detailed conditions, set a filter (Some events):

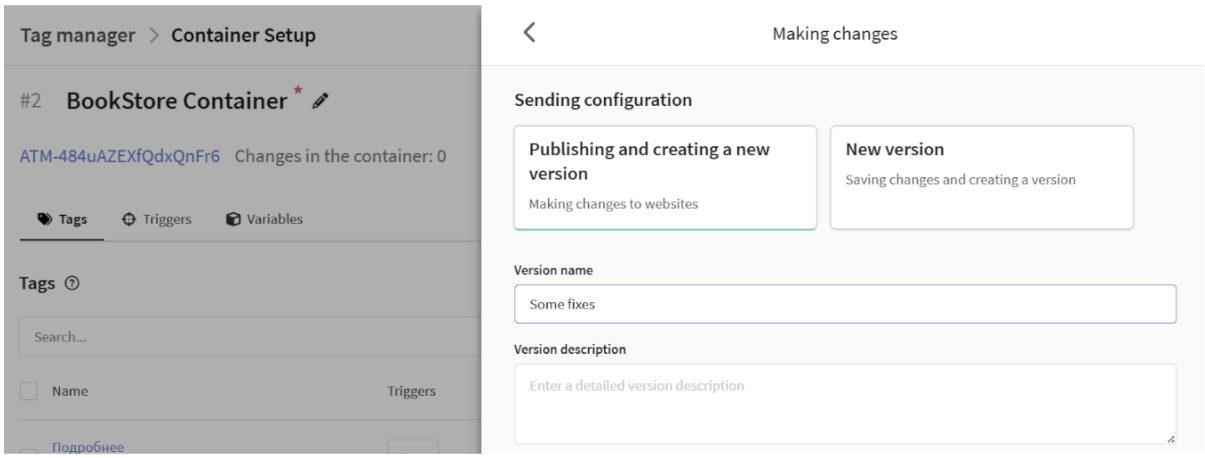


Publishing changes

Save the tag. It will appear in Appearance:



Save and publish the pop-up. Publish the new container version:



TIP

Learn more about containers, tags, triggers, and variables in this article.

Profile matching

You can identify a visitor manually with `acDataLayer.push()`. If a profile exists, its data will be updated.

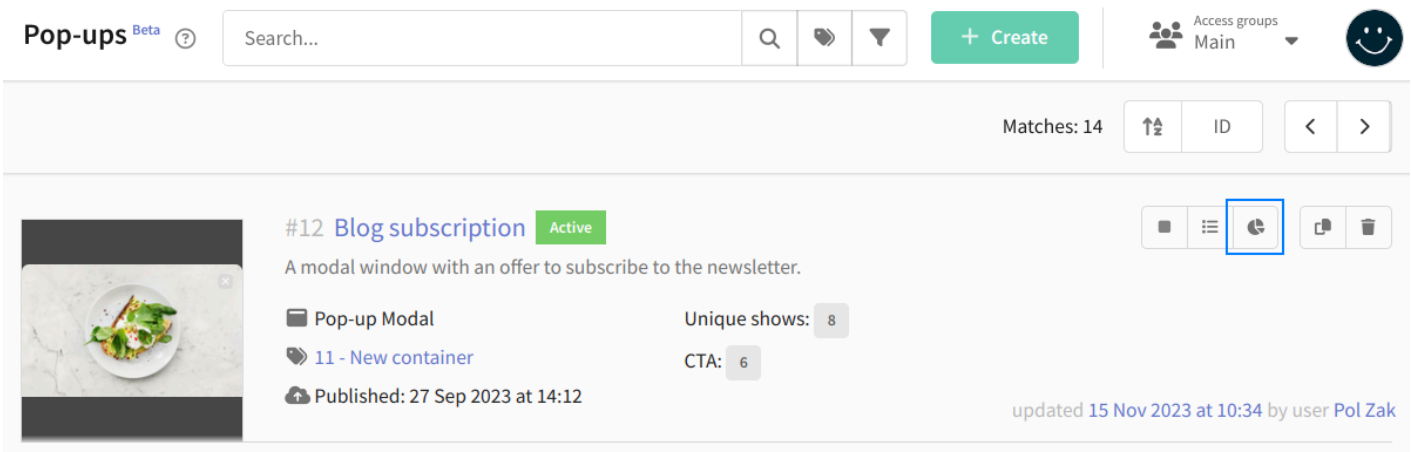
```
acDataLayer.push({
  matching: {
    matching_mode: "email",
    matching: { email: "test@example.com", db_ids: [1, 2] },
  },
});
```

- **matching_mode — search mode (“email”, “email_profile”, “phone”, “profile_id”, “custom”).**
- **matching — object with parameters for search and database IDs.**

Analytics and results

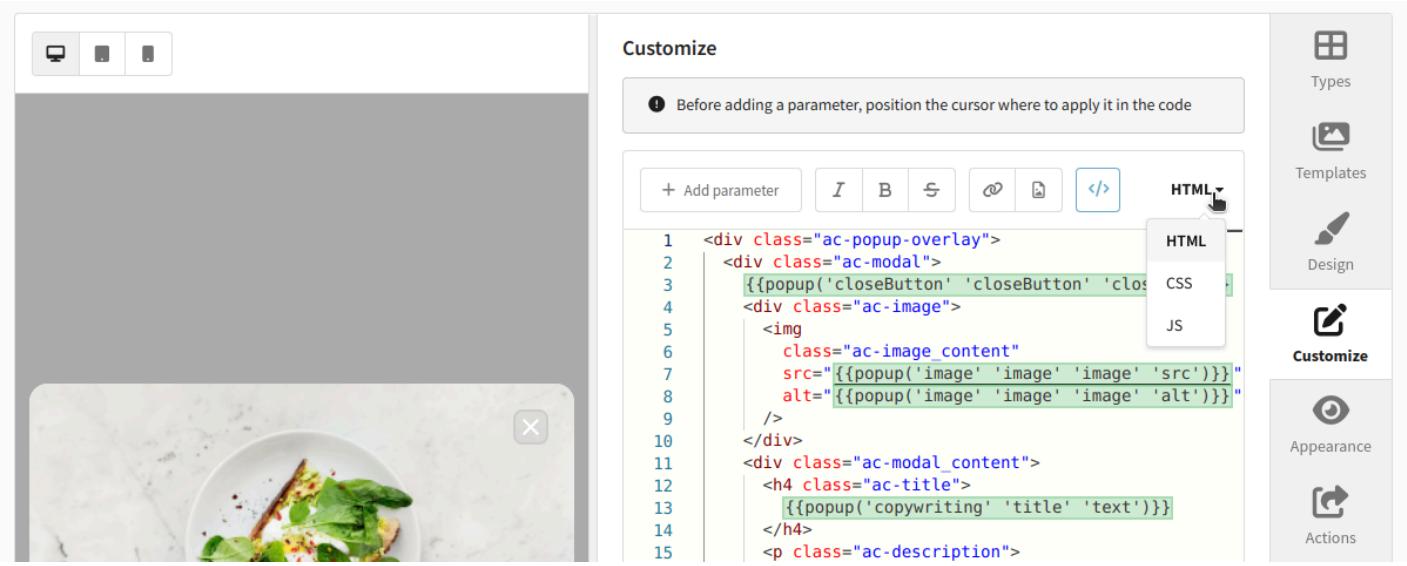
Analyze statistics in Analytics. Click the

icon in the editor or pop-up list:



Setting up a popup in the code editor

To work directly with the pop-up code, go to the Settings section. In the upper right corner you can switch between HTML, CSS and JavaScript code:



To make it easier to work in the editor, use the following functions in the toolbar:

B

— bold text;

I

— *italic*;

~~S~~

— ~~strikethrough text~~;

— adding or editing links;

— adding images from the platform gallery;

</>

— the list of [variables and functions](#) of the editor. Using this button you can add dynamic content to the popup.

INFO

You cannot add other HTML elements before the top-level `div` element, otherwise the template preview will become unavailable.

```
<em></em><div class="ac-popup-overlay">
    <div class="ac-modal">
        <div class="ac-image">
            <img
                class="ac-image_content"
                src="{{popup('image' 'image' 'image' '
                alt="{{popup('image' 'image' 'image' '
            />
        </div>
```

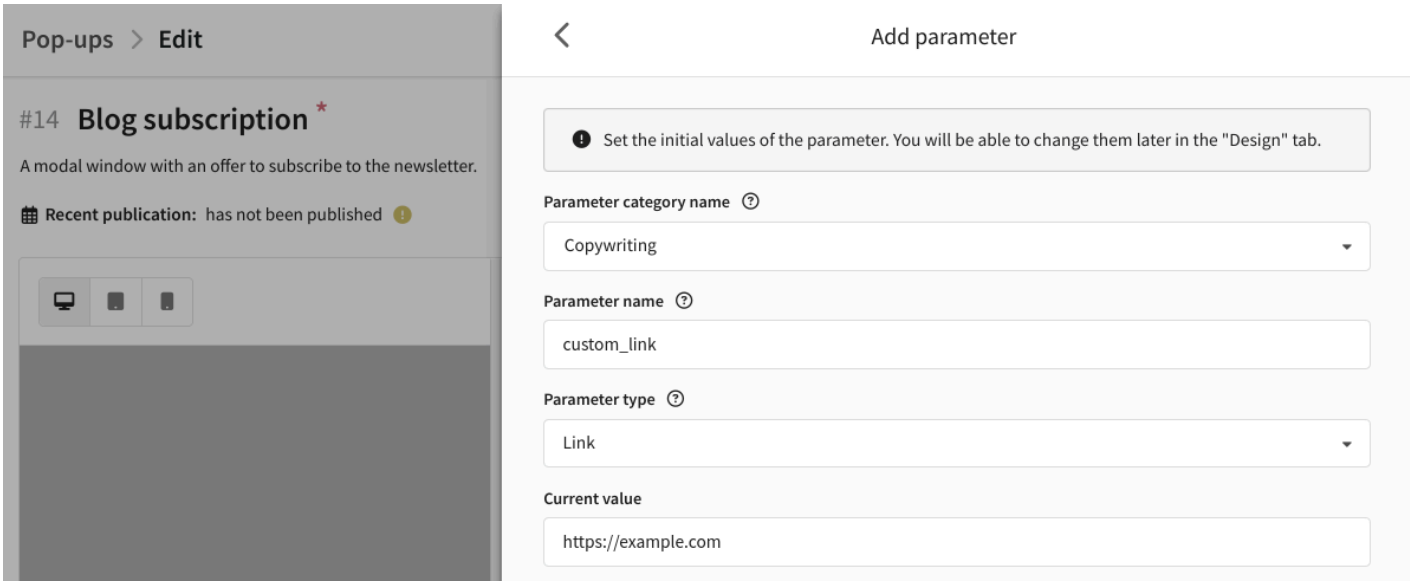
Creating a parameter

You can also create new settings in this section. When saved, they will appear on the Design tab. Thus, users who are not familiar with the code will be able to change the values of custom parameters in the visual editor.

To create a new parameter, place the cursor in the editor line where you want to add the parameter. Click the Add parameter button in the upper left corner of the editor. In the new window select:

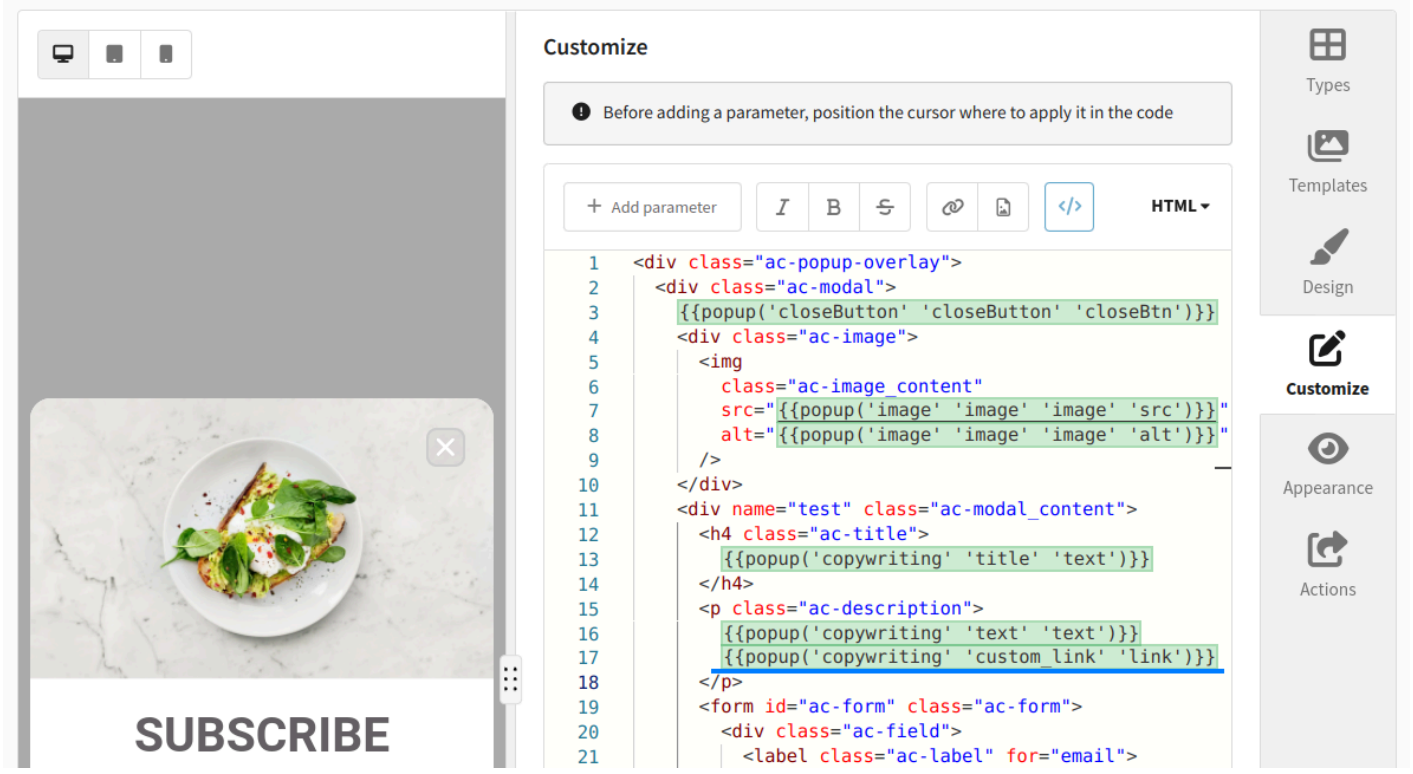
- **Parameter category name. Built-in parameters in the visual editor are grouped by category. You need to select one of the categories for the custom parameter. You can also create your own category. To do this, enter the category name and press Enter.**

- **Parameter name. Enter the name of the parameter to be used in the editor. The name must be unique within the category.**
- **Parameter type. Select the type of value that the parameter will contain (for example, text, numeric value, font family, input field options, etc.).**
- **Current value. Specify the default value for the parameter. This field is optional.**

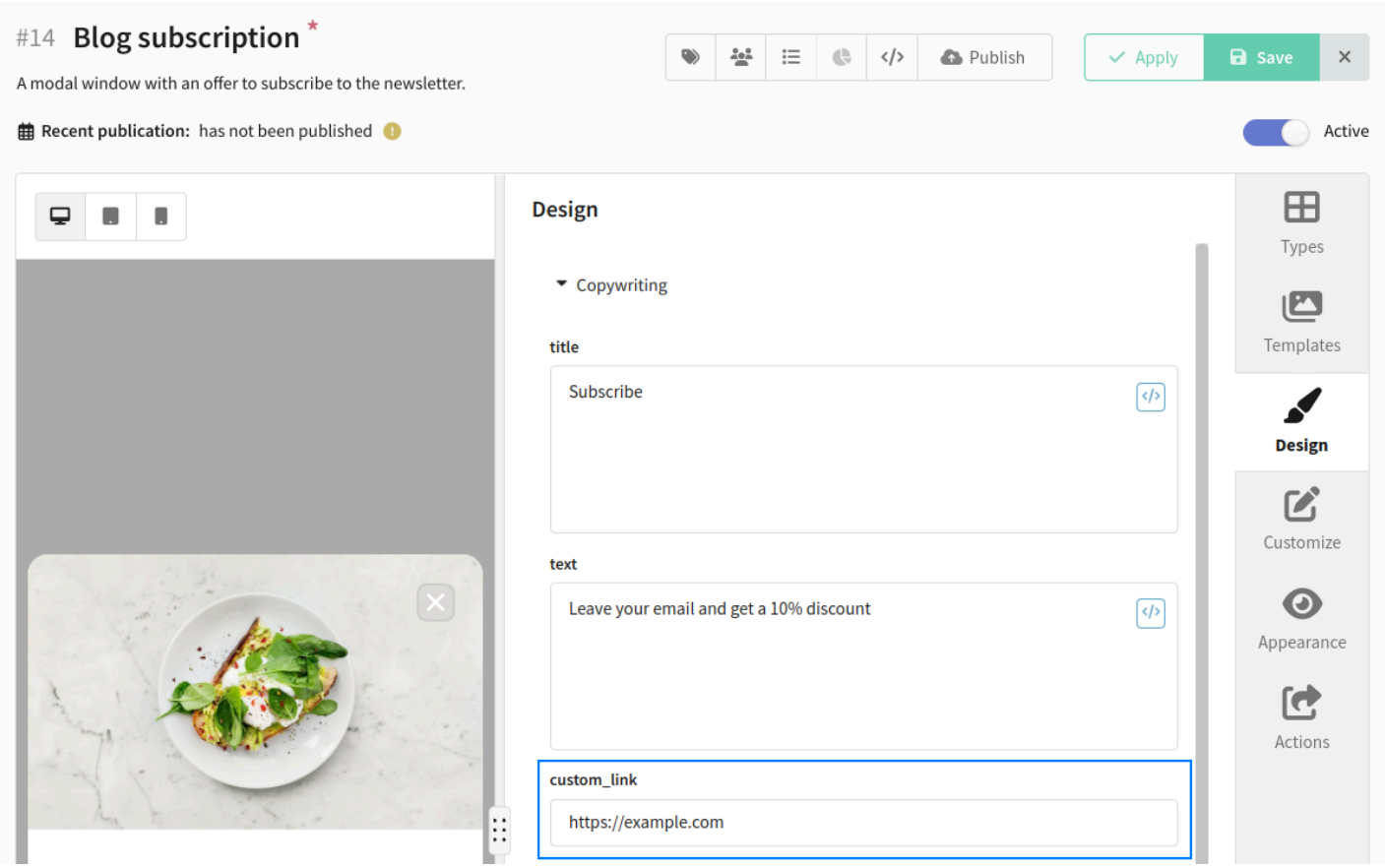


Once configured, click the Insert button at the bottom of the page. The parameter will be added in the following form:

```
{{popup('<category_name>' '<parameter_name>' '<parameter_type>')}}
```



The created parameter will appear in the visual editor in the selected category. After saving the popup, the value of the new parameter can be edited by other platform users:



Popup elements settings

When creating a pop-up, the platform offers many ready-made templates, but sometimes you need to make changes to an existing template. Below are recommendations on how to do this.

INFO

The field name (`name` attribute) before and after the expression should not contain extra characters:

Correct usage: `name="{{popup('form' 'week-field' 'fieldData')}}"` **Incorrect usage:** `name="{{popup('form' 'week-field' 'fieldData')}}-"`

Overlay layer

The overlay layer is designed to visually highlight the pop-up. Defined by an element with class `ac-popup-overlay`. By default, this element has a translucent background and occupies the entire available screen area.

Usage example:

```
<div class="ac-popup-overlay">
    <!--Popup contents-->
</div>
```

Text content

By default, two types of text are available: heading and paragraph.

Usage example:

```
<!-- Title -->
<h4 class="ac-title">
    {{popup('copywriting' 'title' 'text')}}
</h4>

<!-- Paragraph -->
<div class="ac-description">
    {{popup('copywriting' 'text' 'text')}}
</div>
```

Images

There are two ways to add an image to a popup:

- **as an element with the `<img>` tag**
- **using the `background-image` CSS property**

An example of adding an image using the `<img>` tag:

```
<div class="ac-image">
    <img class="ac-image_content img"
        src="{{popup('image' 'image' 'image' 'src')}}"
        alt="{{popup('image' 'image' 'image' 'alt')}}"
    />
</div>
```

An example of adding an image using the `background-image` CSS property:

```
background-image: url("{{popup('background' 'backgroundImage' 'image' 'src')}}");
```

Form

For the popup to work correctly, all input fields and the form submit button must be placed in an element with the `<form>` tag.

Usage example:

```
<form id="ac-form" class="ac-form">
    <!-- Form content -->
</form>
```

Input fields

For correct display, each input field must be inside an element with the `ac-field` class. To add a title, it is recommended to use an element with the `ac-label` class. Be sure to specify the `name` attribute, if it is not specified, the value of the input field will not be sent to the server and will not be saved in the platform.

If you intend to use validation rules, you must add an element with the class `ac-validation-msg`.

Example of an email input field:

```
<div class="ac-field">
    <label class="ac-label" for="email">
        {{popup('copywriting' 'emailLabel' 'text')}}
    </label>
    <input
        id="email"
        type="email"
        class="ac-input"
        name="{{popup('form' 'email' 'fieldData')}}"
        placeholder="{{popup('copywriting' 'emailPlaceholder' 'text')}}"
    />
    <div class="ac-validation-msg" for="email"></div>
</div>
```

Example of a phone number input field:

```
<div class="ac-field">
    <label class="ac-label" for="phone">
        {{popup('copywriting' 'phoneLabel' 'text')}}
    </label>
    <input
        id="phone"
        type="tel"
        class="ac-input"
        name="{{popup('form' 'phone' 'fieldData')}}"
        placeholder="{{popup('copywriting' 'phonePlaceholder' 'text')}}"
    />
    <div class="ac-validation-msg" for="phone"></div>
</div>
```

CAUTION

To correctly collect statistics for input fields, you should set the `type` attribute, the value of which will correspond to the type of input data.

The following field types are available:

- `<input type="text">` **- basic text field as a string.**
- `<input type="email">` **- field for email address. Before sending, it is checked that the input value contains either an empty string or one valid e-mail address.**
- `<input type="tel">` **- input field for a phone number; line breaks are automatically removed from the input value.**
- `<input type="number">` **- input field for a float format number.**
- `<input type="search">` **- one-line text field for entering search strings.**
- `<input type="url">` **- field for URL editing. The entered value must contain either an empty string or a valid absolute URL.**
- `<input type="color">` **- a color control element.**
- `<input type="date">` **- control element for entering date (year, month and day, no time).**
- `<input type="datetime">` **- control element to enter the date and time (hour, minute, second and fraction of a second) according to the UTC time zone.**
- `<input type="datetime-local">` **- control element for entering date and time without time zone.**
- `<input type="week">` **- control element for entering the date containing the number week-year and week number without time zone.**
- `<input type="month">` **- control element for entering month and year without time zone.**

Form submit button

To create a form submit button, it is recommended to use an element with class `ac-form_submit`, which contains a button element with type submit and class `ac-button`.

Usage example:

```
<div class="ac-form_submit">
     <button type="submit" class="ac-button">
         {{popup('cta' 'text' 'text')}}
     </button>
</div>
```

Close button

To create a button to close a popup, it is recommended to use the popup function, passing the appropriate parameters to it.

Usage example:

```
{{popup('closeButton' 'closeButton' 'closeBtn')}}
```

The element has several different options, which can be viewed on the Design —> “Close” button tab.

Additional event tracking

To track events, you need to add the `data-tracking-event="[event-name]"` attribute to the html element with which the event is associated. Replace `[event-name]` with the name of the event. The following events are supported:

- `click` **- click on an element**
- `close` **— closing the popup**

An example of registering clicks on a popup element:

```
<div data-tracking-event="click"></div>
```

TIP

If the popup contains an HTML form element, then when the form submit button is clicked, the action of filling out the form will be automatically registered. That is, in this case, to register an event, you do not need to add an additional attribute to the form button.

Managing pop-ups manually via script

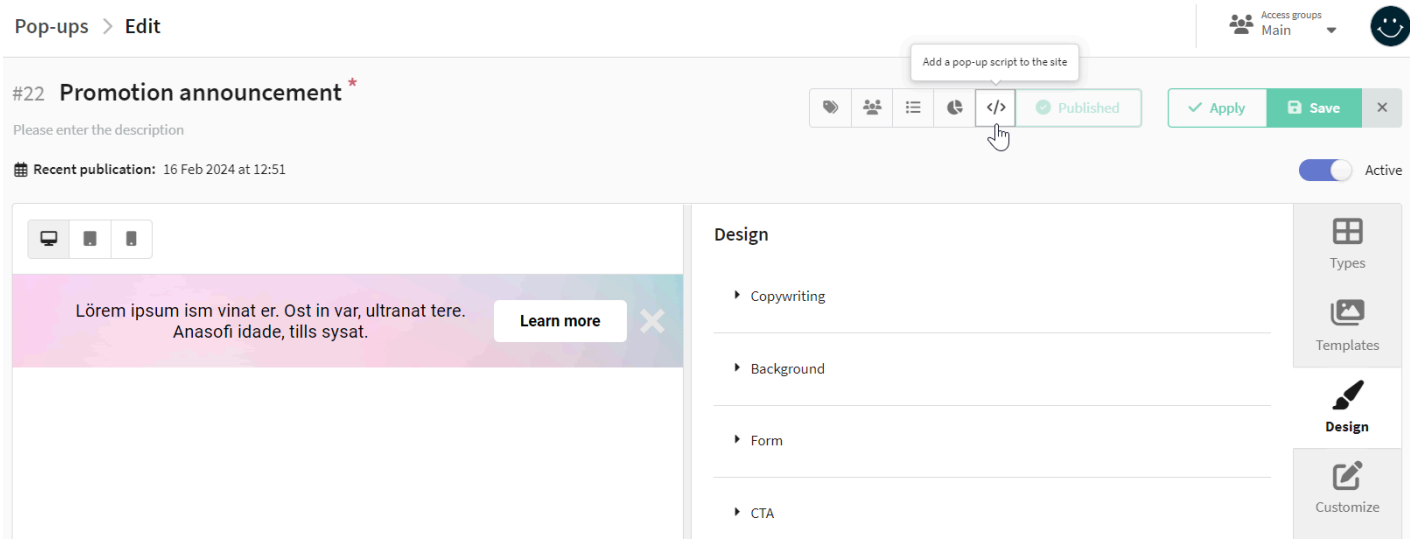
This guide is tailored for platform users who implement pop-ups on their site using scripts rather than the Altcraft Tag Manager.

Obtaining the script

To obtain the script, go to the pop-up editor and in the additional actions menu, click on the icon

</>

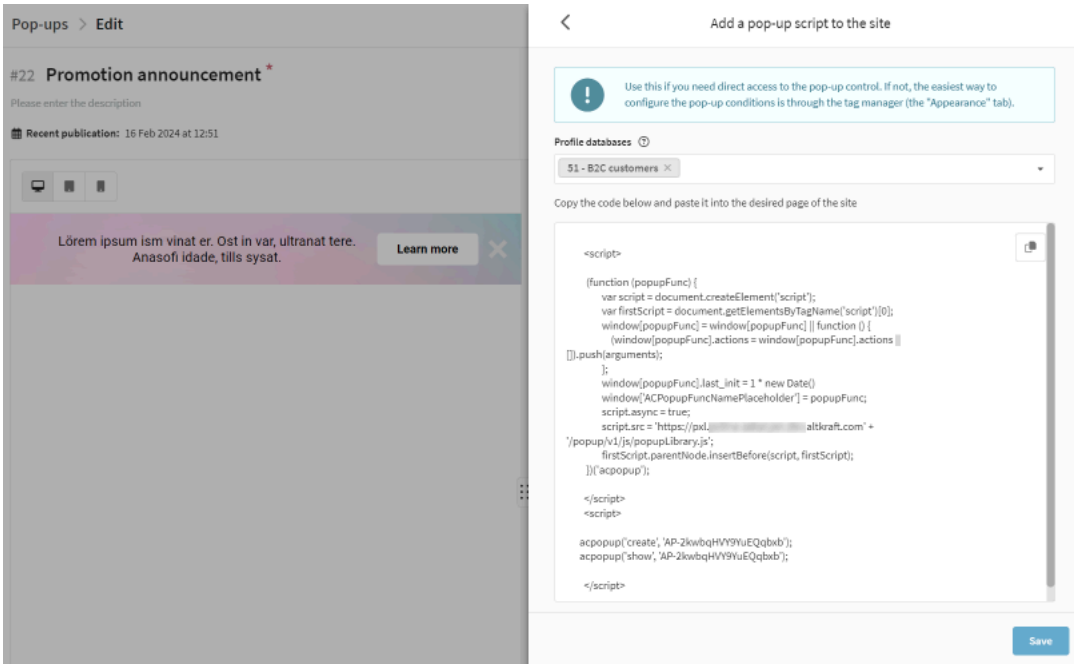
:



In the new window, select the databases for profile search. When a user fills out the pop-up form, their data is sent to the platform. Then the platform checks if there is a profile with such data in one of the selected databases. If there is, the data of the existing profile is updated; if not, a new profile is created in the database.

Note that here you are setting the databases to search for existing profiles. User data, filled out in the pop-up form, will be imported into the database specified in the Actions section of the pop-up editor.

Click the Show code button. This code needs to be placed on the pages of the site where you want to show the pop-up. Further details on configuring the pop-up appearance logic are provided below.



Managing a popup via the script

The following code is used to work with popups on the site:

```
<!-- Loading the script onto the page -->
<script>

  (function (popupFunc) {
        var script = document.createElement('script');
        var firstScript = document.getElementsByTagName('script')[0];
        window[popupFunc] = window[popupFunc] || function () {
            (window[popupFunc].actions = window[popupFunc].actions || []).push(arguments);
        };
        window[popupFunc].last_init = 1 * new Date()
        window['ACPopupFuncNamePlaceholder'] = popupFunc;
        script.async = true;
        script.src = 'https://pxl.altkraft.com' + '/popup/v1/js/popupLibrary.js';
        firstScript.parentNode.insertBefore(script, firstScript);
  })('acpopup');

</script>

<!-- Requesting pop-up data from the server + displaying the pop-up on the page -->
<script>

  acpopup('create', '<popup_id>');
  acpopup('show', '<popup_id>');

</script>
```

If you add this code to your site, the popup will be shown as soon as the page loads because of this script:

```
<script>
  acpopup('create', '<popup_id>');
  acpopup('show', '<popup_id>');
</script>
```

You'll need to set up how and when the pop-up appears.

Example 1. Pop-up appears 10 seconds after the page loads:

```
<script>
    window.onload = function() {
        setTimeout(function() {
            // Запрос данных попапа с сервера
            acpopup('create', '<popup_id>');

            // Показ попапа
            acpopup('show', '<popup_id>');
        }, 10000);
    };
</script>
```

Example 2. Pop-up appears when a button is clicked:

```
<button onclick="showPopupClick()">Show the pop-up</button>

<script>
    function showPopupClick() {
        acpopup('create', '<popup_id>');
        acpopup('show', '<popup_id>');
    }
</script>
```

To manage the pop-up, use the `acpopup` function. It has 3 modes:

- **create**
- **show**
- **sendEvent**

create — request pop-up data from the server

In the `acpopup` function you must pass the pop-up ID as the second argument. As the third argument you can pass* an object with information about matching* and *SMID*. The table describes the structure of this object:

Parameter	Type	Required	Description
`matching_mode`	**string**	**No**	**Profile search mode. It can be `"email"`, `"email_profile"`, `"phone"`, `"profile_id"` or `"custom"`.**
`matching`	**JSON object**	**No**	**An object with one or two parameters: `"search field": "field value"` and `"db_id": "database ID"` — the array of the databases in which the profile should be searched. `db_id` can be omitted if the profile database is selected in the pop-up settings.**
`smid`	**string**	**No**	**Send message ID**

Example:

```
acpopup('create', 'AP-6eVbo2dP2Nnh0wtBp', {
    matching_mode: 'email',
    matching: {
        email: 'john@example.com',
        db_ids: [1]
    },
    smid: 'send_message_id',
})
```

show — show a pop-up to users

In the `acpopup` function you must pass the pop-up ID as the second argument. The third argument can be object with `onShow` function, which will be called after the popup appears on the screen.

Example:

```
acpopup('show', 'AP-6eVbo2dP2Nnh0wtBp', {
  onShow: function () {
      console.log('Look at the screen')
  },
})
```

sendEvent — send a pop-up custom event

In the `acpopup` function you must pass the pop-up ID as the second argument. The third argument must be an object with custom event data. You can also pass matching data for a specific event and SMID to this object. The table describes the structure of the object:

Parameter	Type	Required	Description
`web_event_name`	**string**	**Yes**	**Event name**
`web_event_value`	**int**	**No**	**Numeric value of the event**
`web_event_data`	**JSON object**	**No**	**Object with optional data for the event**
`matching_mode`	**string**	**No**	**Profile search mode. It can be `"email"`, `"email_profile"`, `"phone"`, `"profile_id"` or `"custom"`.**
`matching`	**JSON object**	**No**	**An object with one or two parameters: `"поле для поиска": "значение поля"` and `"db_id": "ID базы"` — databases in which to search for a profile. `db_id` can be omitted if the profile database is selected in the pop-up settings.**
`smid`	**string**	**No**	**Send message ID**

Example:

```
acpopup('sendEvent', 'AP-6eVbo2dP2Nnh0wtBp', {
    web_event_name: 'new_event',
    web_event_value: 500,
    web_event_data: {
        status: 'active'
    },
    matching_mode: 'email',
    matching: {
        email: 'john@example.com',
        db_ids: [1]
    },
    smid: 'send_message_id',
})
```

TIP

If you’re adding multiple pop-ups to one page, you only need to place the coode for loading script onto the page once, and then you can manage them using the `acpopup` function.

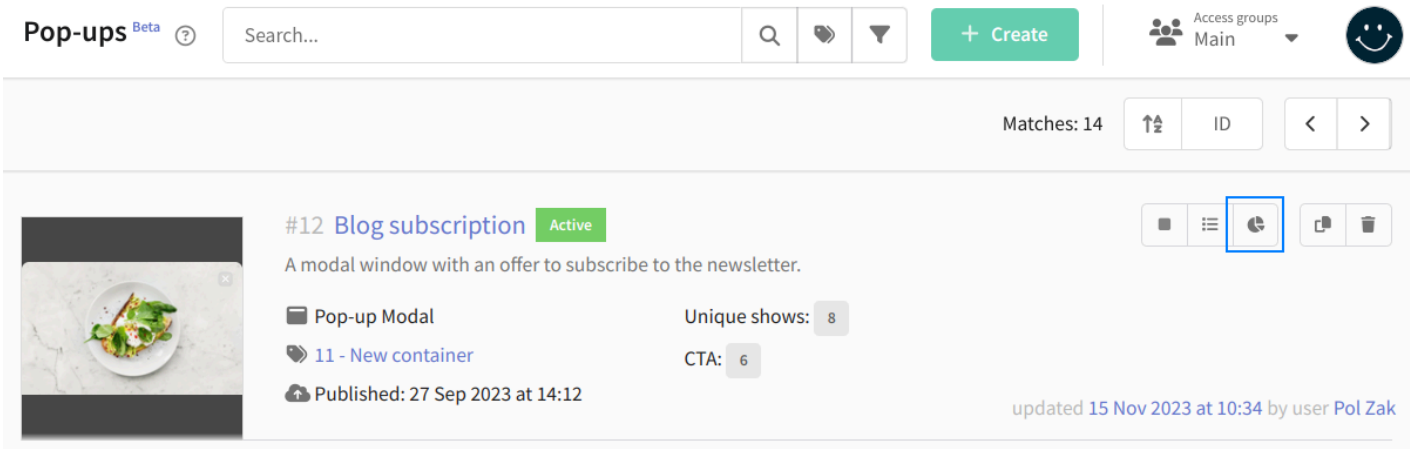
Popup analytics

Description

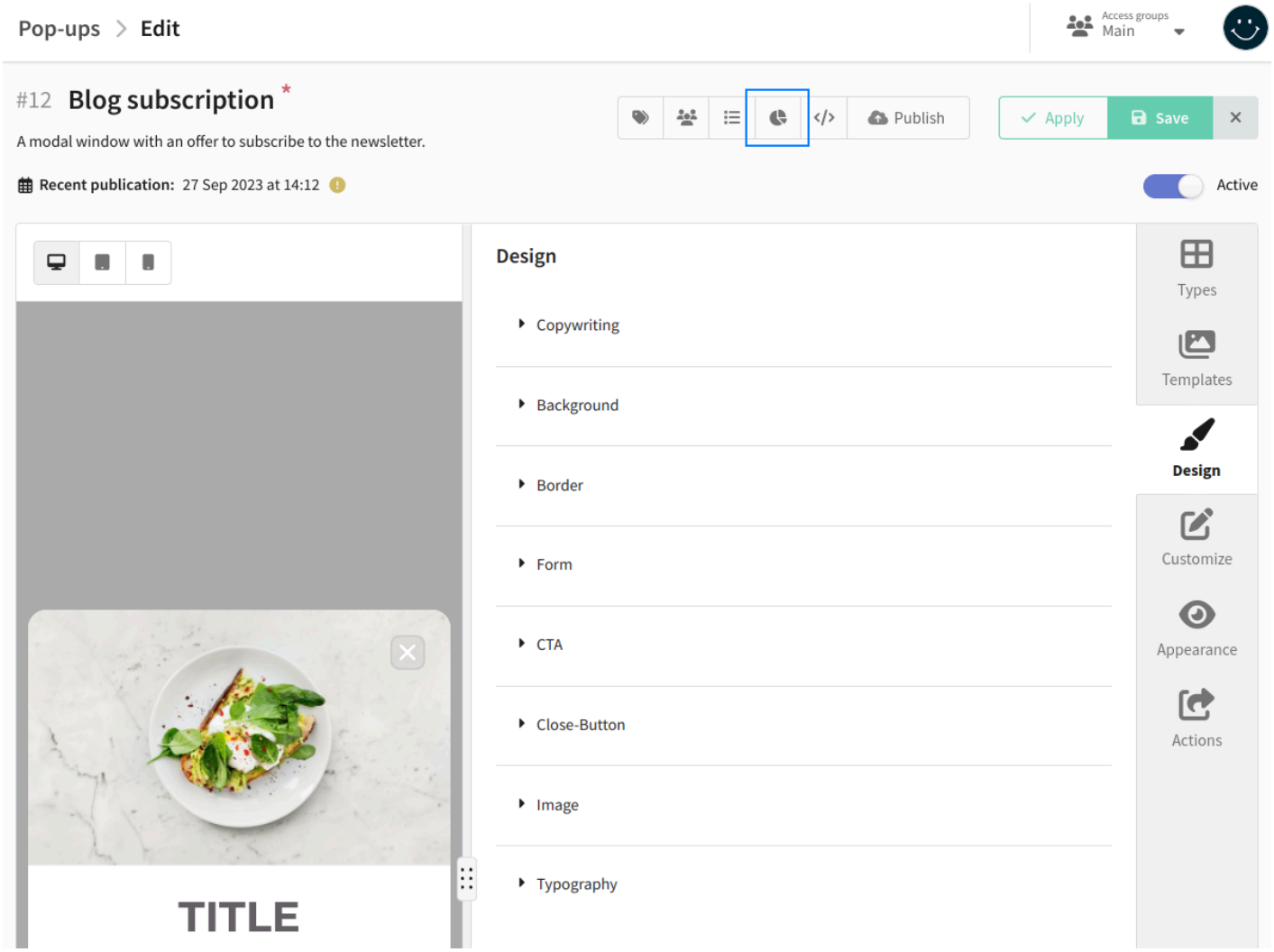
Analytical tools allow you to view key popup metrics (hits to targeting, shows, form submissions), as well as user responses. The data is presented in the form of a graph and a funnel. You can group any metric based on various criteria, such as user geo-data, browser, or UTM tags.

To go to Analytics, find the popup you need in the general list and click on the icon

:



You can also go to Analytics directly from the popup editing window:



There are two tabs available in analytics:

- **Popup statistics — general summary of the popup.**

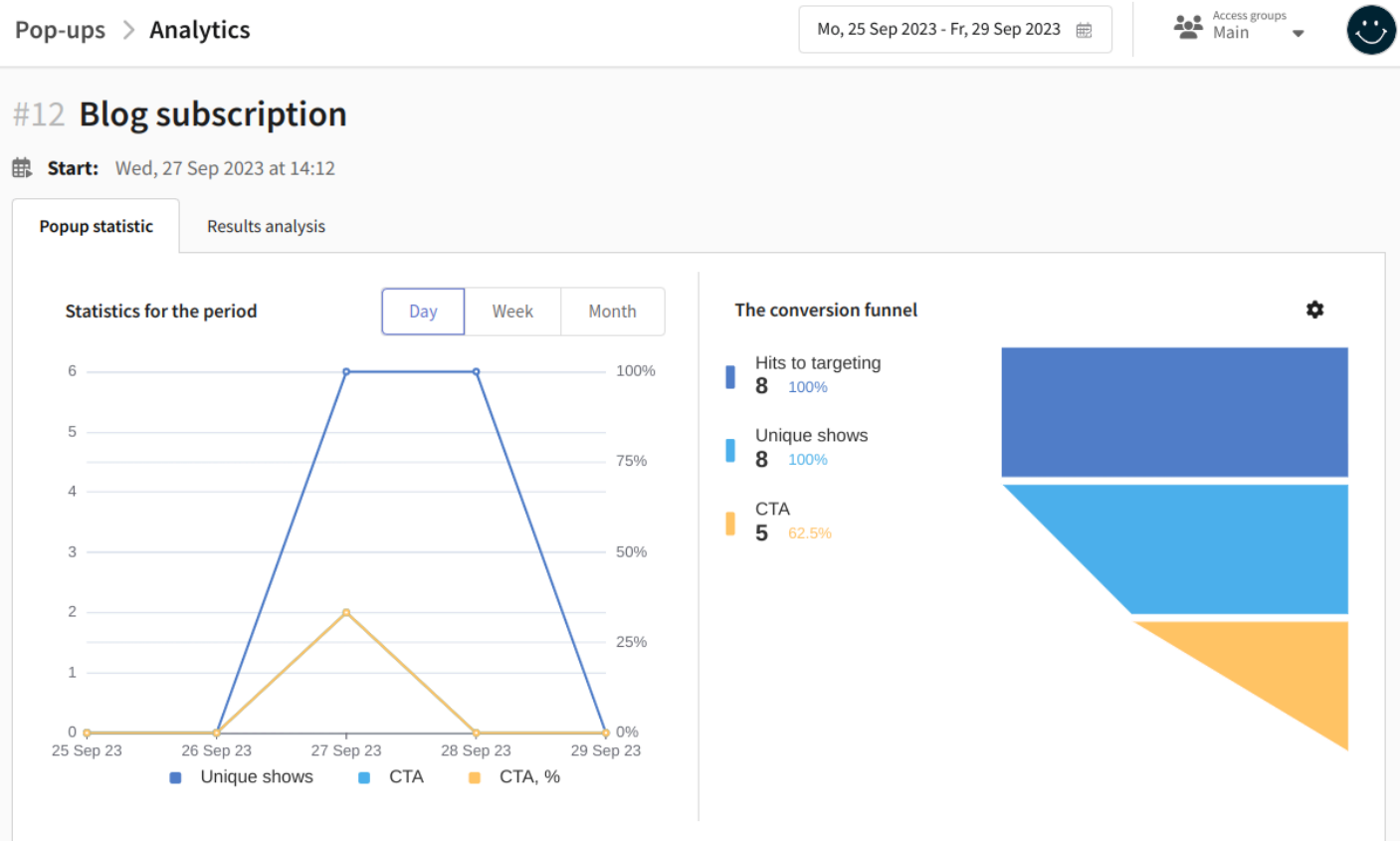
- **Analysis of results — statistics on filling out fields in the form.**

Popup statistics

This tab contains information about popup impressions and form fillings. On the top panel you can specify the period for which you want to receive data.

Popup statistics are divided into three blocks:

1. **Statistics for the period**
2. **Conversion funnel**
3. **Analytical slices**



Statistics for the period

The graph displays the dynamics of performance metrics over the selected period:

- **Unique shows — the number of times the popup is displayed to users. If the popup is shown to a user multiple times within one session, only one unique show is recorded.**
- **CTA — until a form submission event is registered, this metric reflects clicks on any form element. Once the form is filled out and submitted at least once, the metric reflects the number of submissions.**
- **CTA, % — (Unique shows / CTA) * 100%.**

Data can be grouped by days, weeks, or months. If you want to hide a metric from the graph, click on its name."

Conversion funnel

The funnel is a sequence of steps through which a group of website visitors passes. This group is defined by the first step of the funnel — hit to targeting. The report then shows how many users moved to the next stage. With each step, the number of users decreases.

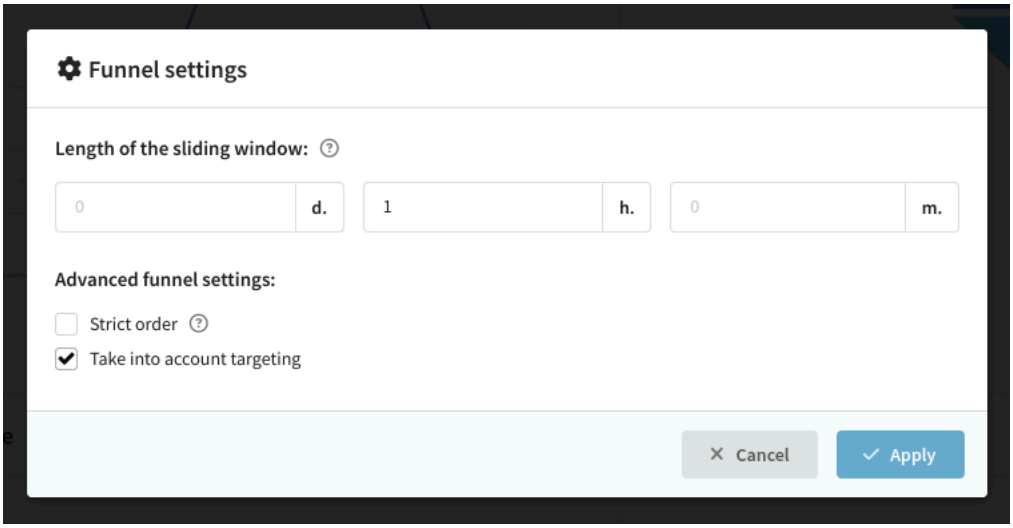
- **Hits to targeting — the number of times the conditions for the appearance of the popup, set in the trigger, were met. However, the popup might not have been shown to the user due to tag limits specified in the “Activation Settings” and “Limits” sections.**
- **Unique shows — the number of times the popup is displayed to users.**
- **CTA — until a form submission event is registered, this metric reflects clicks on any form element. Once the form is filled out and submitted at least once, the metric reflects the number of submissions.**

TIP

Events in the funnel are unique. For example, if a popup was shown to a user multiple times within one session, only one unique show will be counted.

Additionally, you can customize the funnel to suit your tasks. To open the settings window, click

in the top right corner.



Sliding window width is the time period between the first and last conditions in the funnel. Default is 1 hour. If more than 1 hour passes from the moment you hit the target until the CTA event is registered, then the last step of the funnel will not be counted.

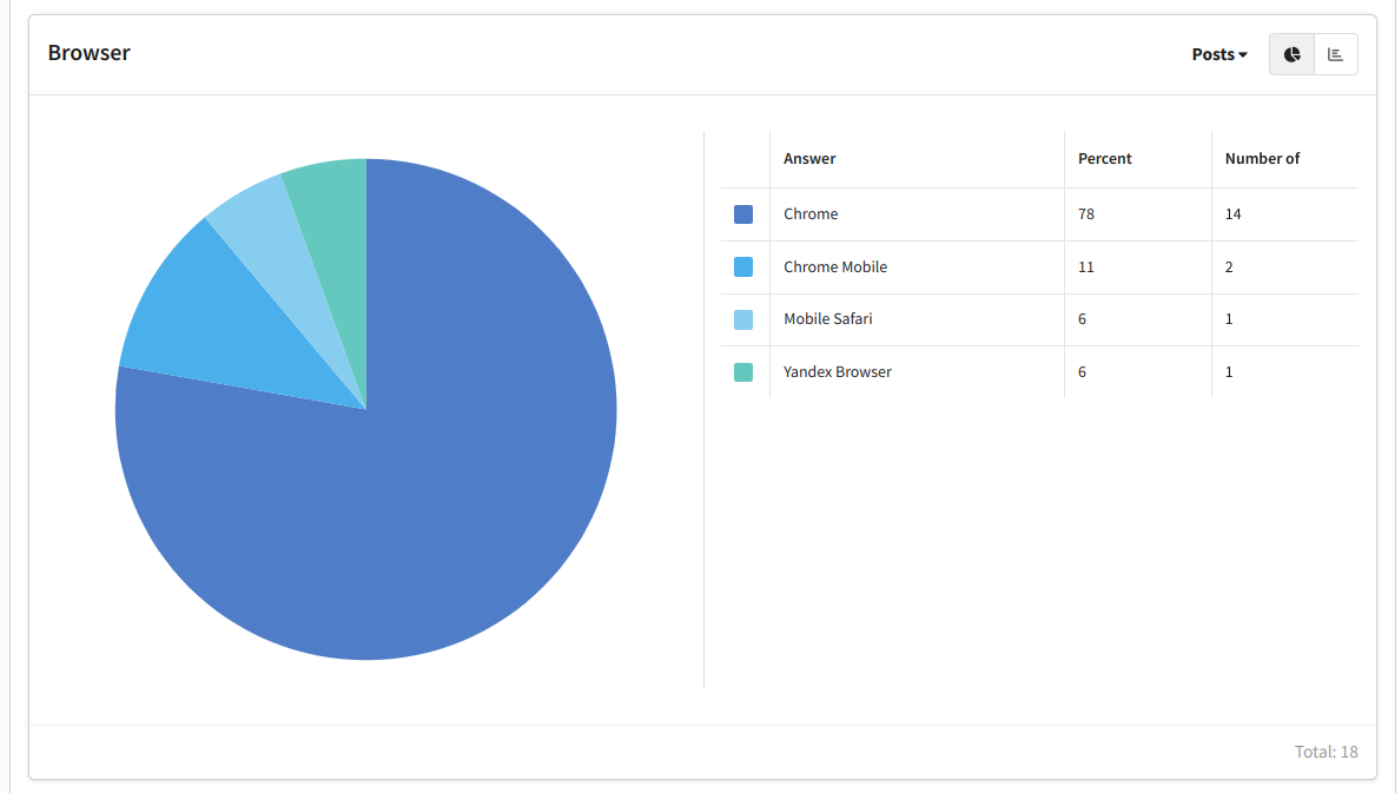
Strict order prohibits extraneous events in the desired sequence. For example, if in a funnel with the sequence *Targeting hit —> Unique. display —> CTA* between the first and second step one more event will be recorded, then all subsequent events will not be counted. In this case, the event that violated the order will also not be counted.

Take into account hitting the target — by unchecking the checkbox, you will remove hitting the target as a funnel step.

Analytical slices of the report

Below are pop-up indicators grouped by the following criteria:

- **By time zone**
- **By city**
- **By country**
- **By language (browser language)**
- **By OS (operating system on the user's device)**
- **By browser**
- **By referrer (URL of the page on which the popup was shown)**
- **By UTM tags (content, medium, source, mailing, term)**



For any slice, you can select a specific indicator:

- **CTA — while the form submit event is not registered, the metric displays clicks on any form element. If the form has been completed and submitted at least once, the indicator will change to “Completions”.**
- **Fillings — the user filled out and submitted the popup form.**
- **Impressions — the popup is shown to the user.**
- **Unique impressions — the popup is shown to the user. If a popup was shown to a user several times within one session, one unique impression will be counted.**
- **Close — click on the close button in the popup.**

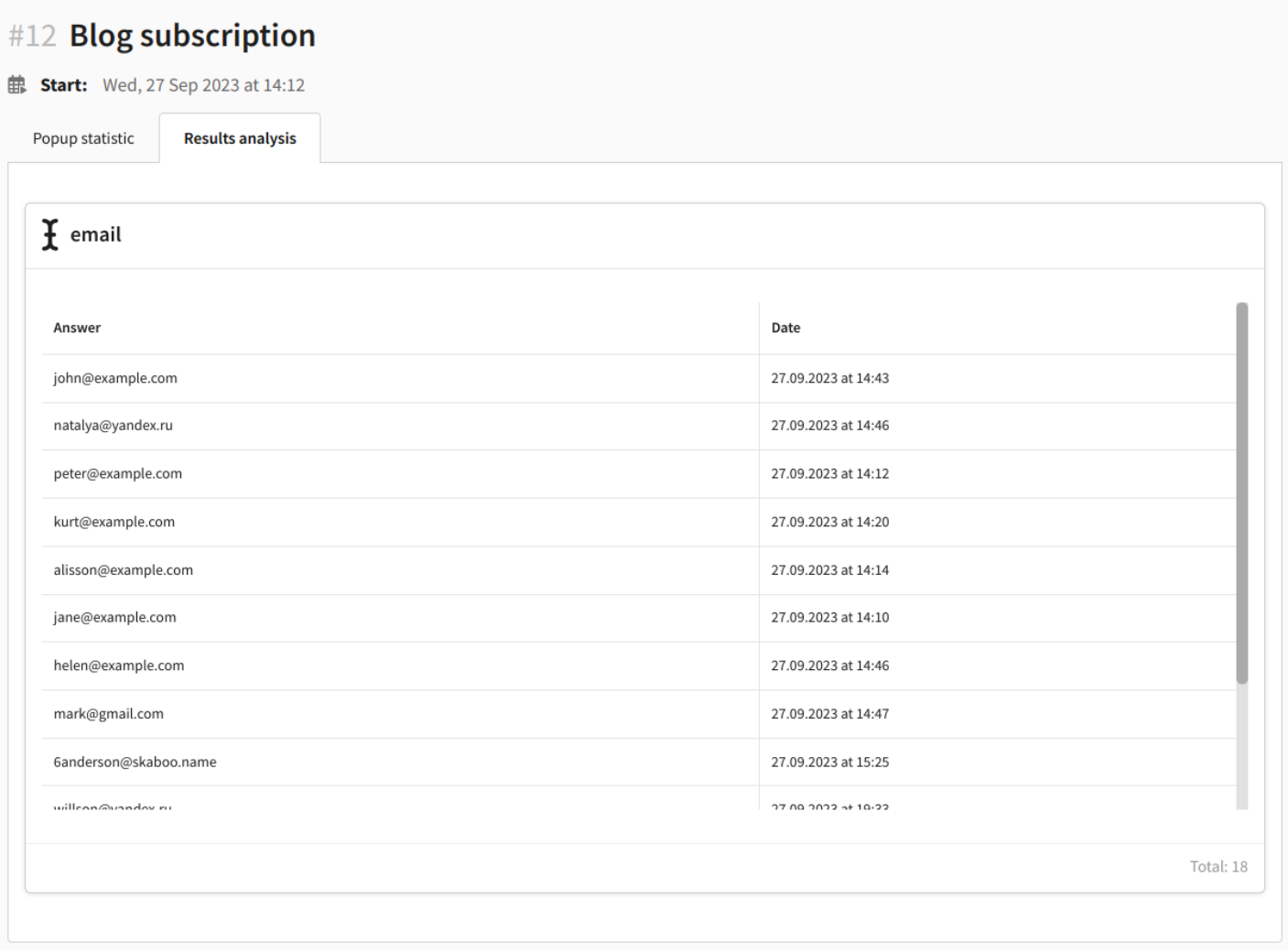
In the table on the right, the data is presented in percentage and quantity. The bottom of the block shows the total number of times the selected event was registered. Additionally, you can change the way the data is displayed:

— pie chart

— bar chart

Analysis of results

On this tab you can get information about filling out form fields in the popup:



Statistics are generated depending on the type of input field specified in the popup code: `<input type="<field_type">`. To collect statistics correctly, you should set the `type` attribute, the value of which will correspond to the type of input data. You can read more about input fields in [this article](#).

Guide: pop-up for push subscriptions

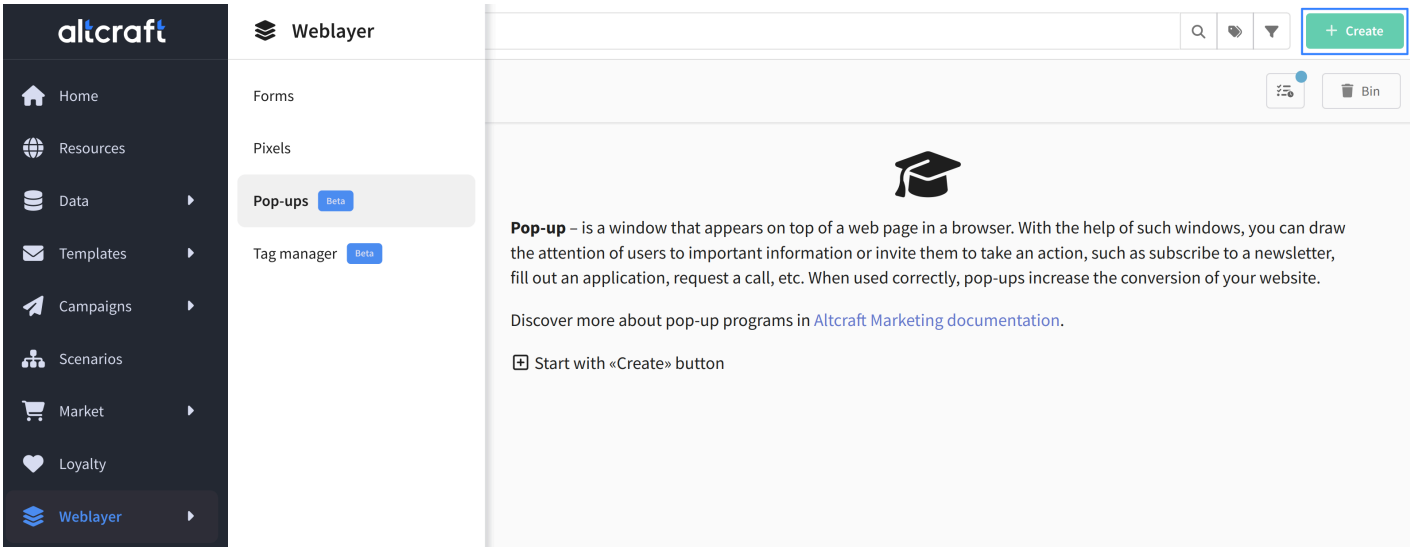
One of the most common types of pop-ups is the push notification subscription pop-up. This kind of pop-ups will help you to quickly subscribe your website visitors to the Altcraft platform resource. In this article, we will show you how to create and customize a pop-up that will appear on page load for those users who have not yet given permission to display push notifications on your site.

INFO

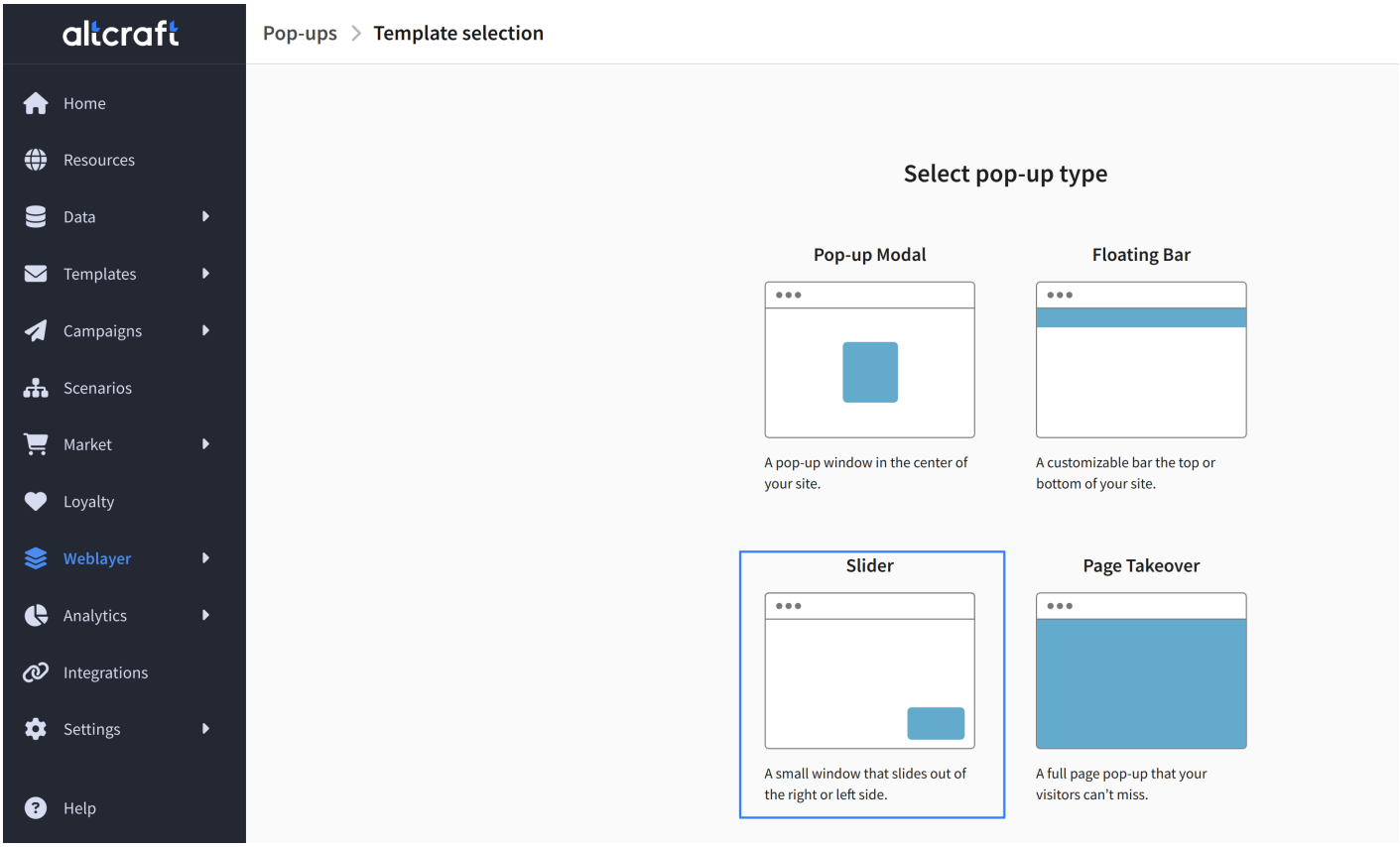
For push notifications subscription to work correctly, you must first set up a resource on the platform, as well as add the corresponding script to your site. You can learn more about setting up push notifications there.

Creating and publishing a pop-up

Go to "Weblayer" — "Pop-ups". Create a new pop-up:



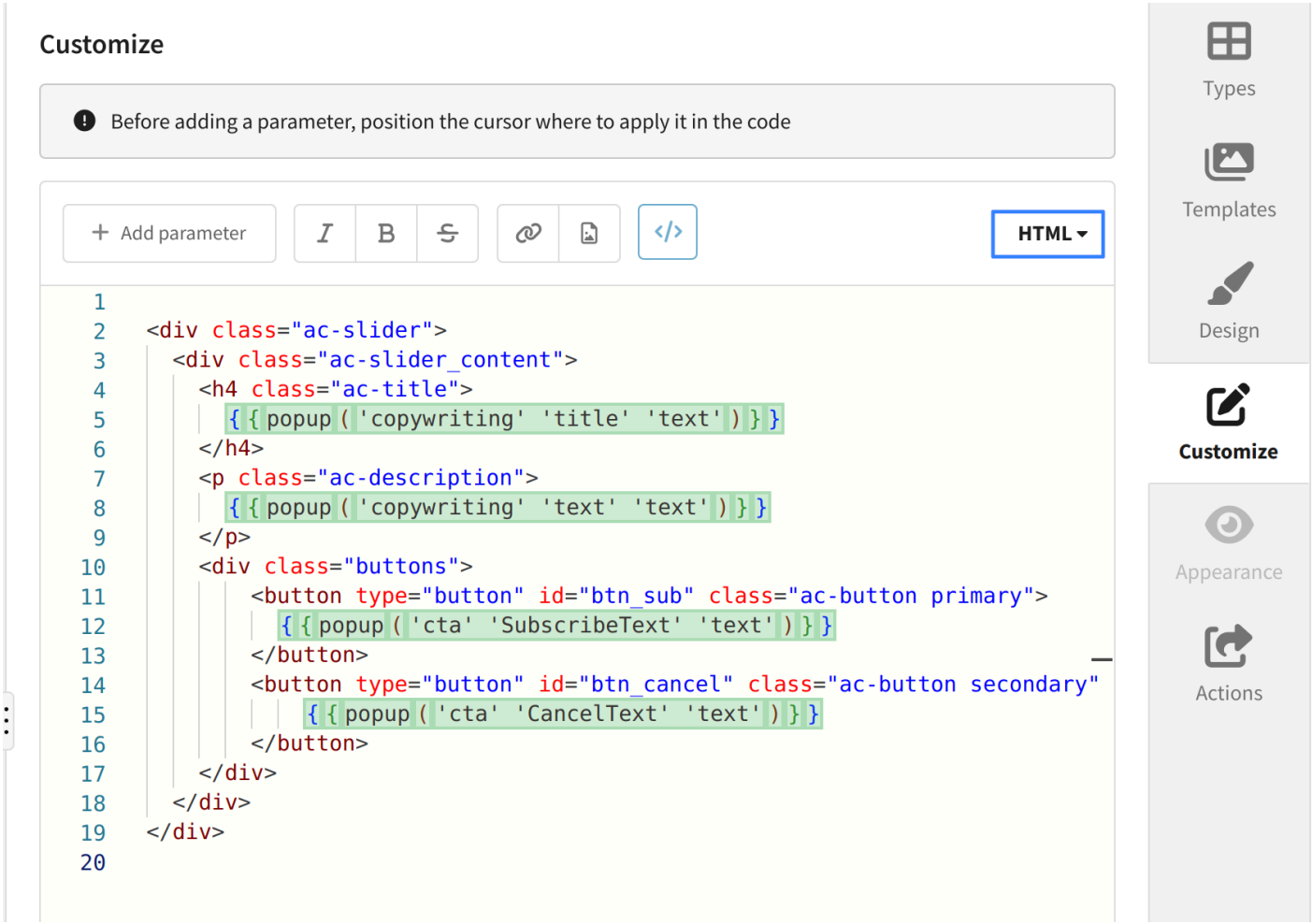
Select the Slider type:



Select pop-up template. Then go to the "Customize" section of the menu that opens on the right side of the screen. Paste the following HTML code there:

```
<div class="ac-slider">
  <div class="ac-slider_content">
    <h4 class="ac-title">
      {{popup('copywriting' 'title' 'text')}}
    </h4>
    <p class="ac-description">
      {{popup('copywriting' 'text' 'text')}}
    </p>
    <div class="buttons">
        <button type="button" id="btn_sub" class="ac-button primary">
          {{popup('cta' 'SubscribeText' 'text')}}
        </button>
        <button type="button" id="btn_cancel" class="ac-button secondary" data-tracking-event="close">
            {{popup('cta' 'CancelText' 'text')}}
        </button>
    </div>
  </div>
</div>
```

As a result, the body of the pop-up should look like this:



Change the format to JS and add code that will initialize the push subscription when the subscribe button is clicked:

```
const popupContainer = document.querySelector('[data-ac-popup-container]');

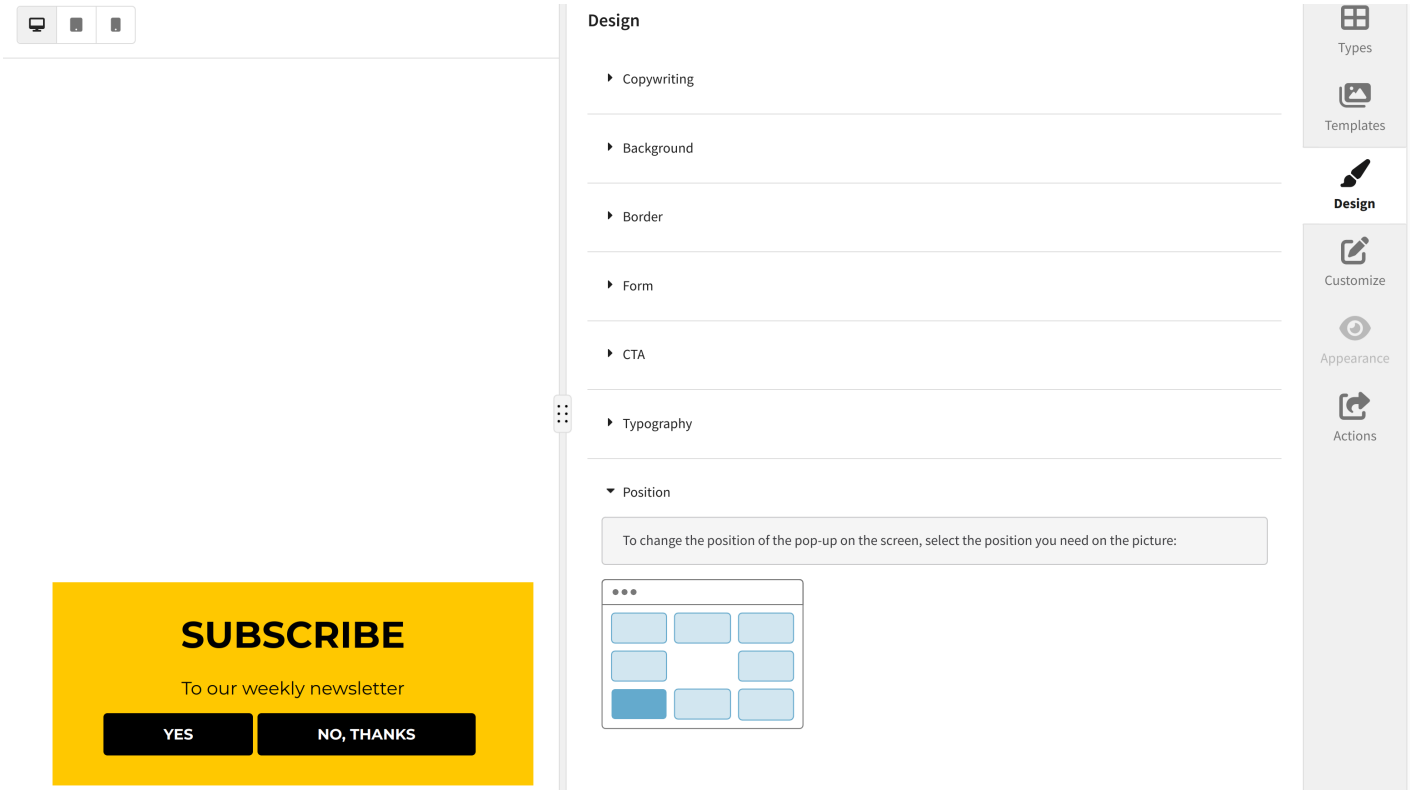
if (popupContainer) {
    const popupHost = popupContainer.shadowRoot || document;
    const subscribeBtn = popupHost.getElementById('btn_sub');

    if (subscribeBtn) {
      subscribeBtn.addEventListener('click', () => {
          // Search for a profile by email. It will also be written to the profile if no matches are found.
          // new AKPush().initSubscription({ email: 'example@email.com' });
          new AKPush().initSubscription();
          popupContainer.style.display = 'none';
      });
    }
}
```

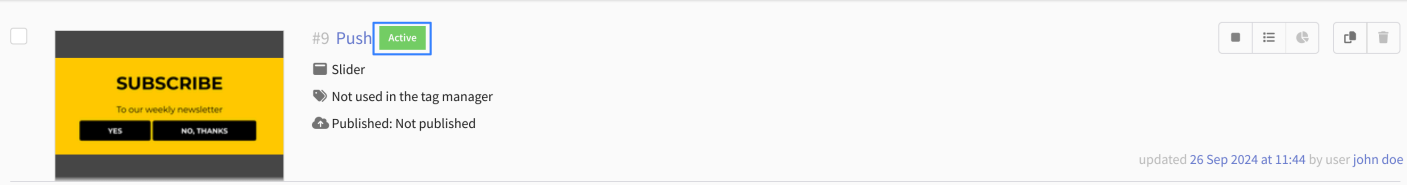
TIP

You can learn more about the layout and customization of pop-ups in the code editor in [this article](#).

In the Design section, you can adjust the display of elements, their color, the text of inscriptions, and the position of the window on the screen:



When all changes to the pop-up have been made, specify its name and then click Apply and Publish. In the list of pop-ups, check if it is active:



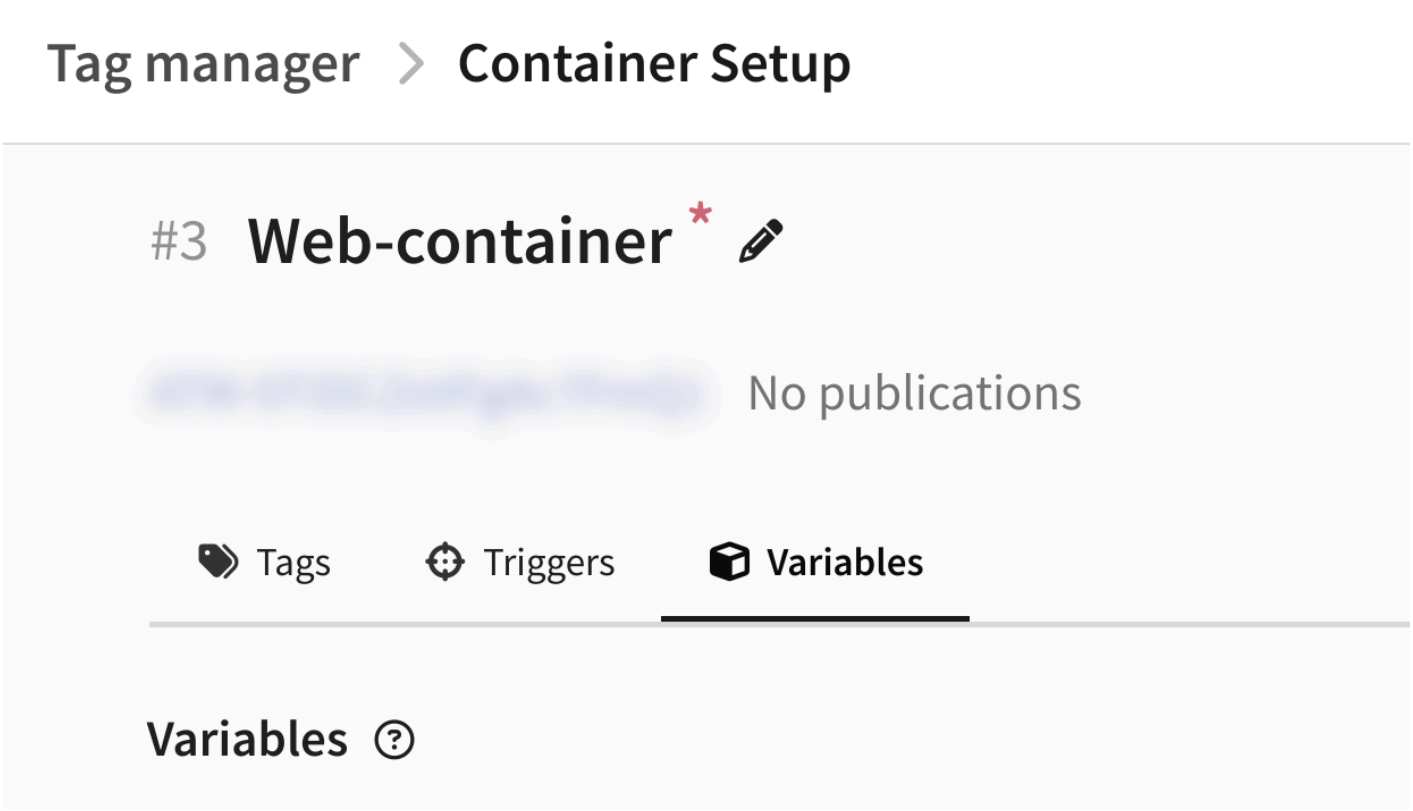
Setting up the Tag Manager

INFO

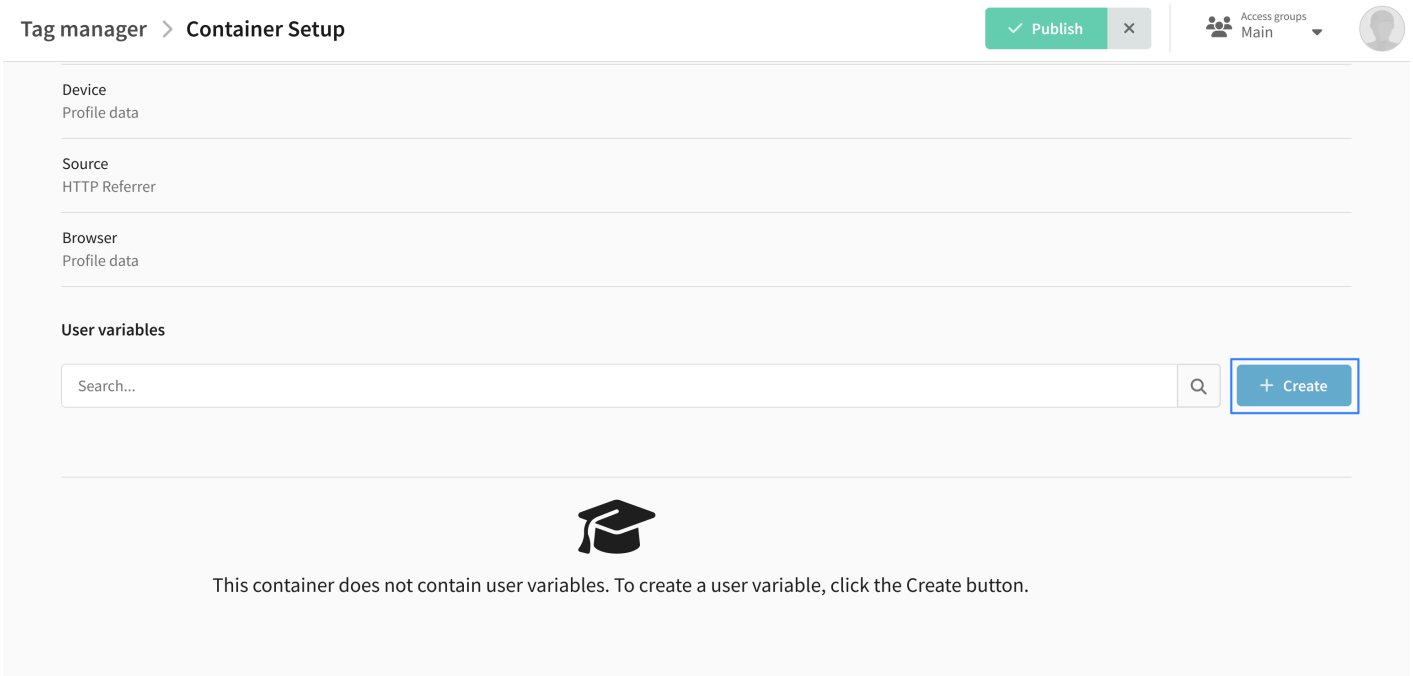
If you haven't worked with Tag Manager or customized the container before, you can read the instructions on how to customize these elements [there](#).

Variable creation

Go to “Weblayer” — “Tag Manager” menu. Open the container that you use for the site:



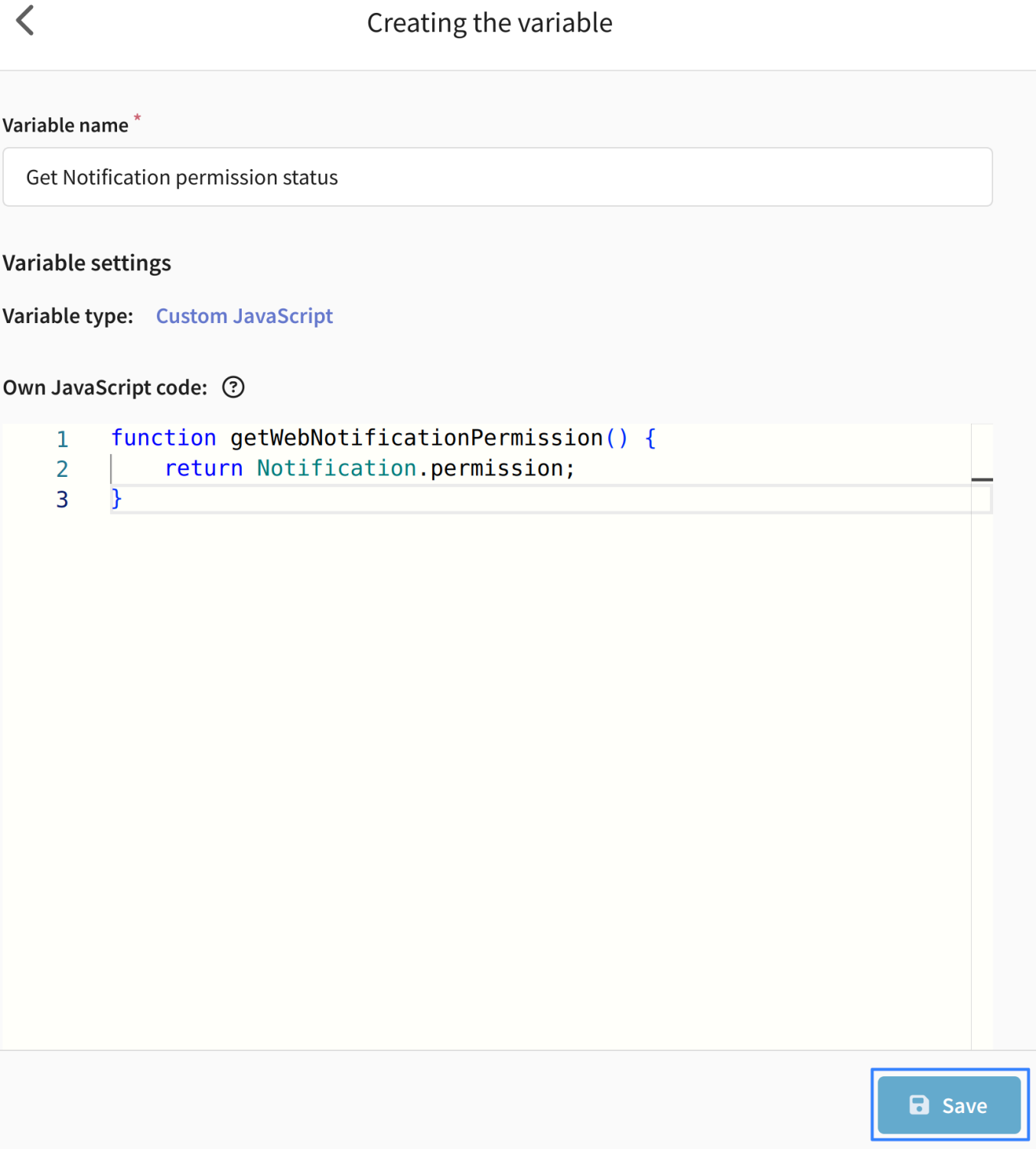
Select Variables and at the very bottom of the page, create a custom variable:



Specify the variable name, for example “Get Notification permission status”, then select the type Custom JavaScript. In the input field, paste the following code:

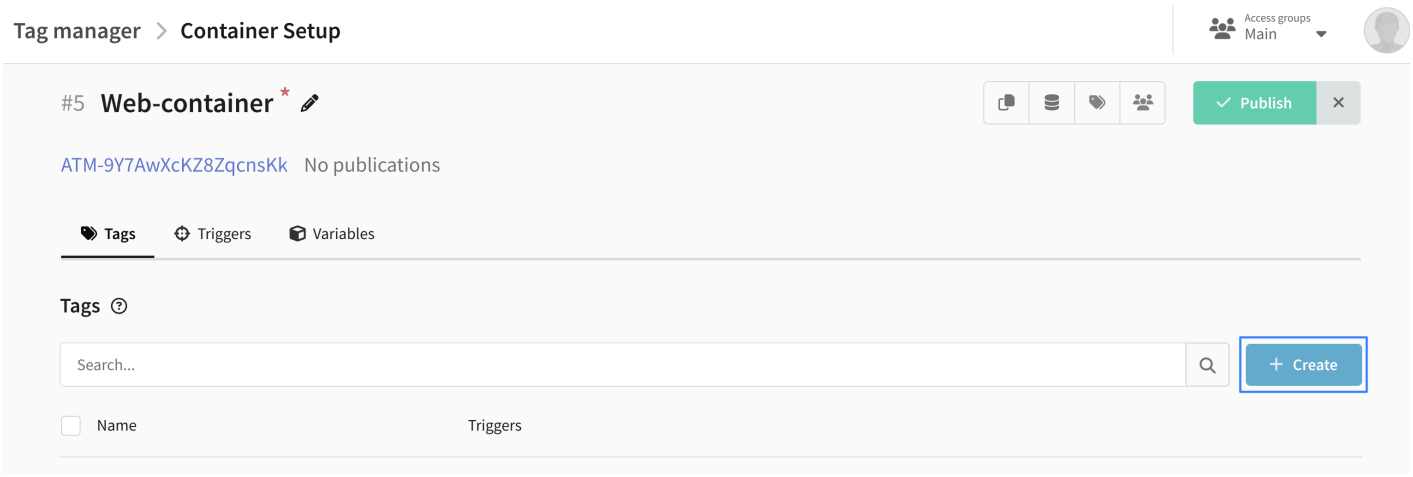
```
function getWebNotificationPermission() {
    return Notification.permission;
}
```

Save the created variable:

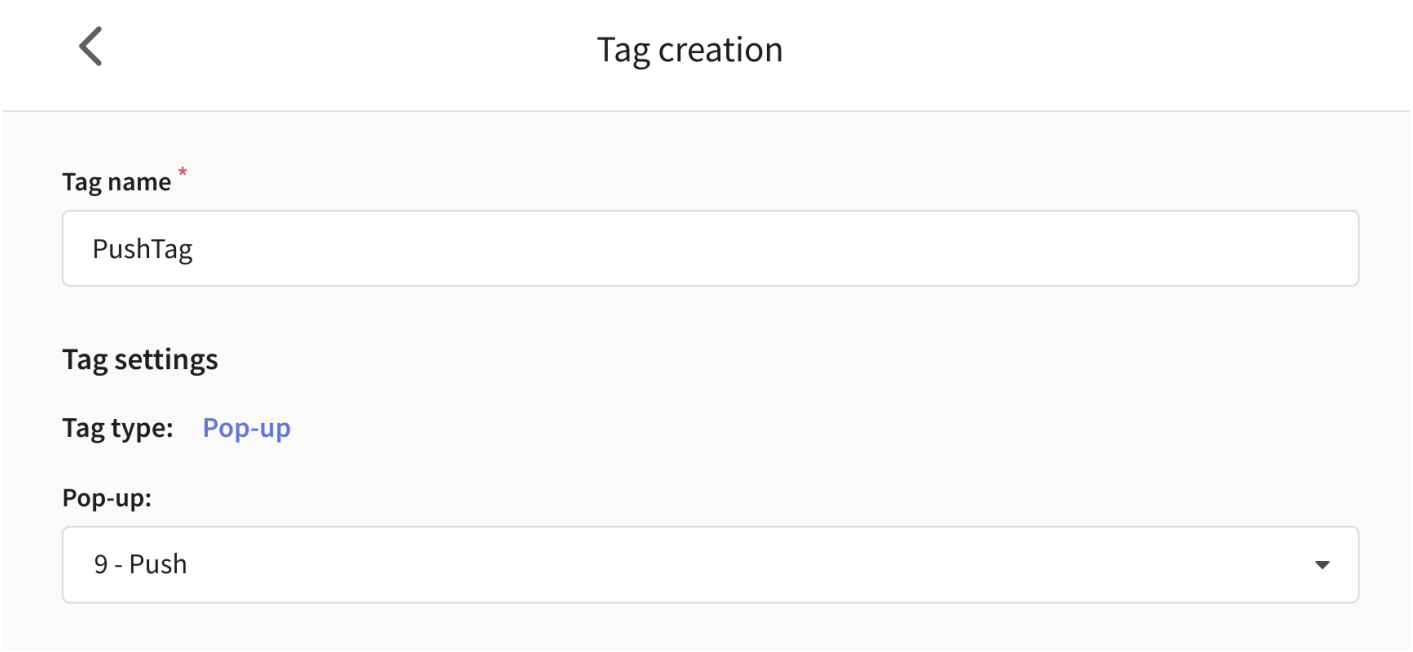


Tag creation

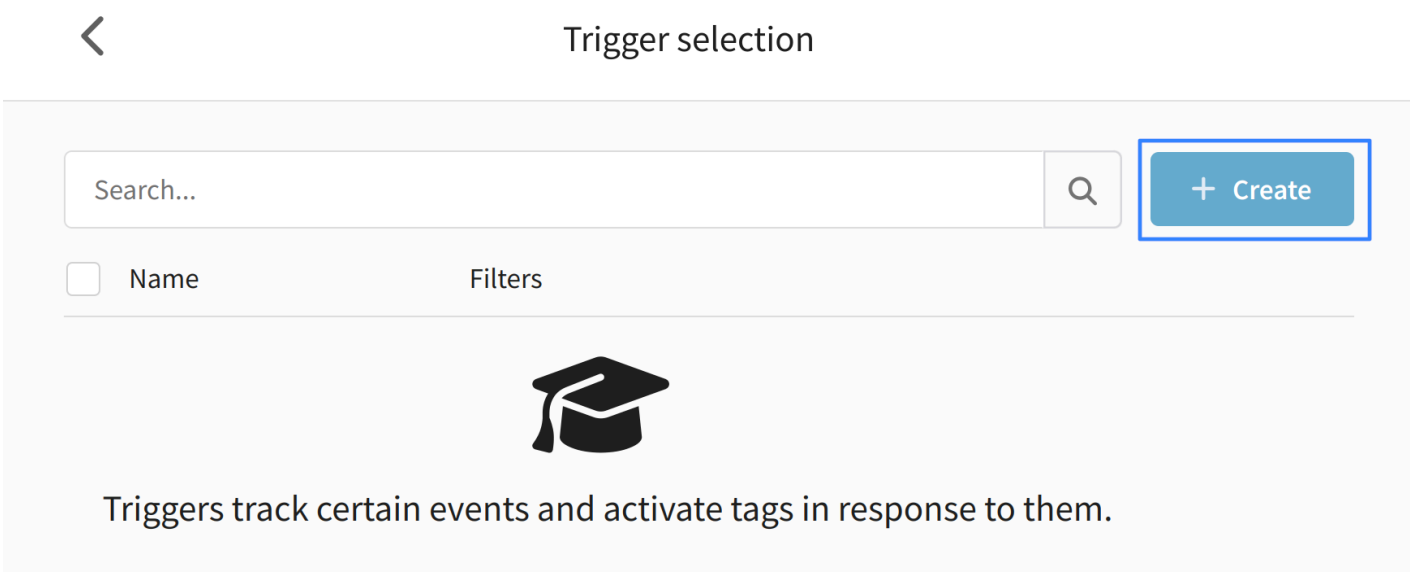
Go to Tags and create a new tag:



Name it, specify the type Pop-up and select the pop-up you created in the previous step:

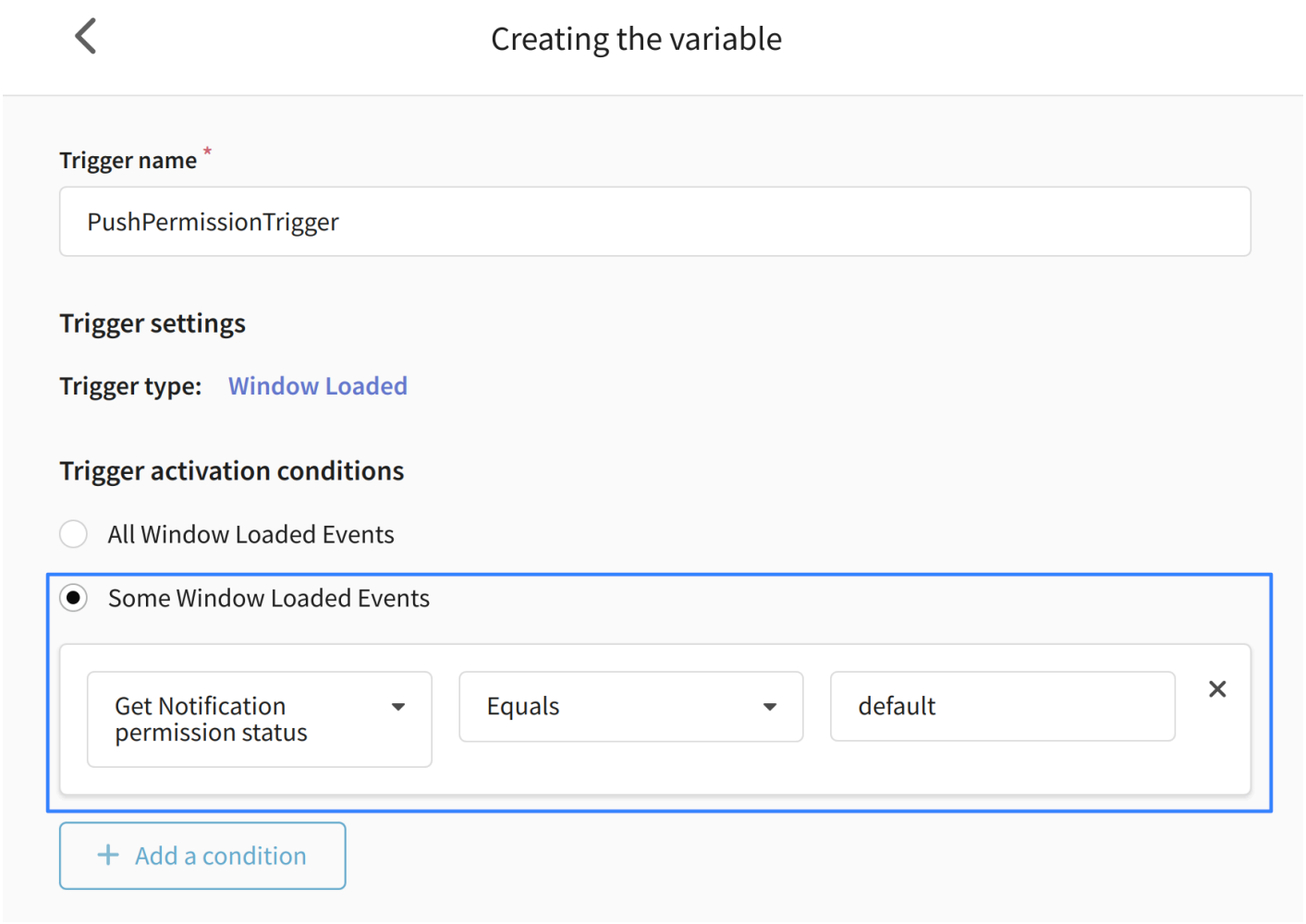


Customize the activation settings and limits if necessary. Then select triggers. If you have not already created triggers, you can create a new one directly from this window:

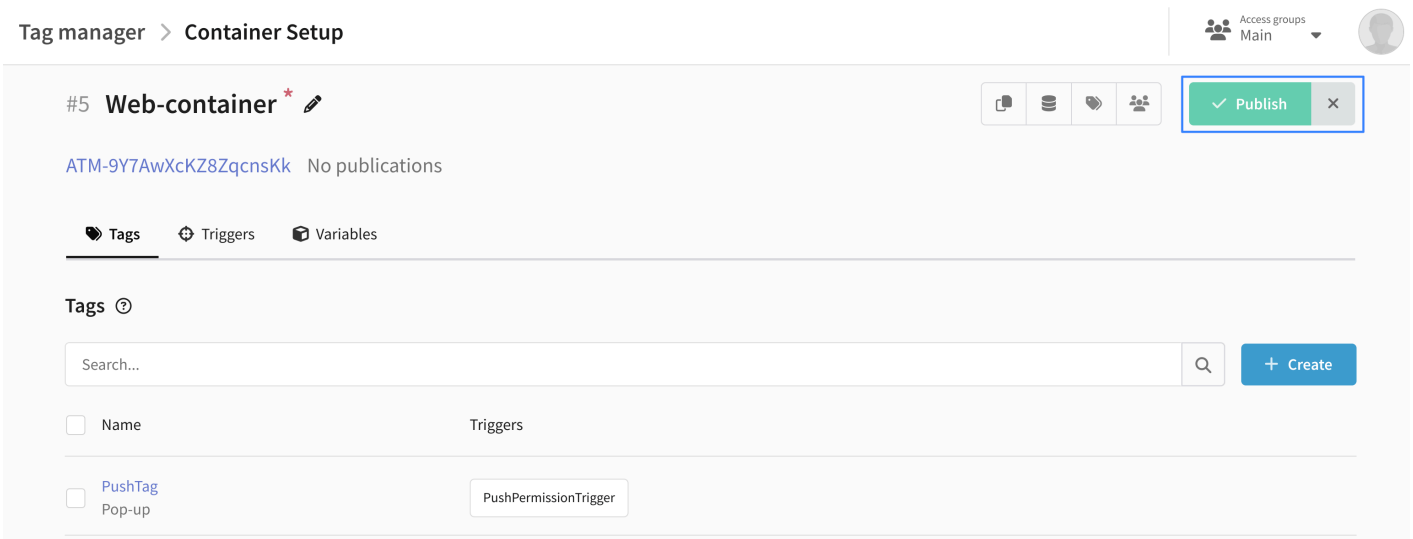


You can customize the trigger yourself for any action you need. In our example we use the “Window loaded” type.

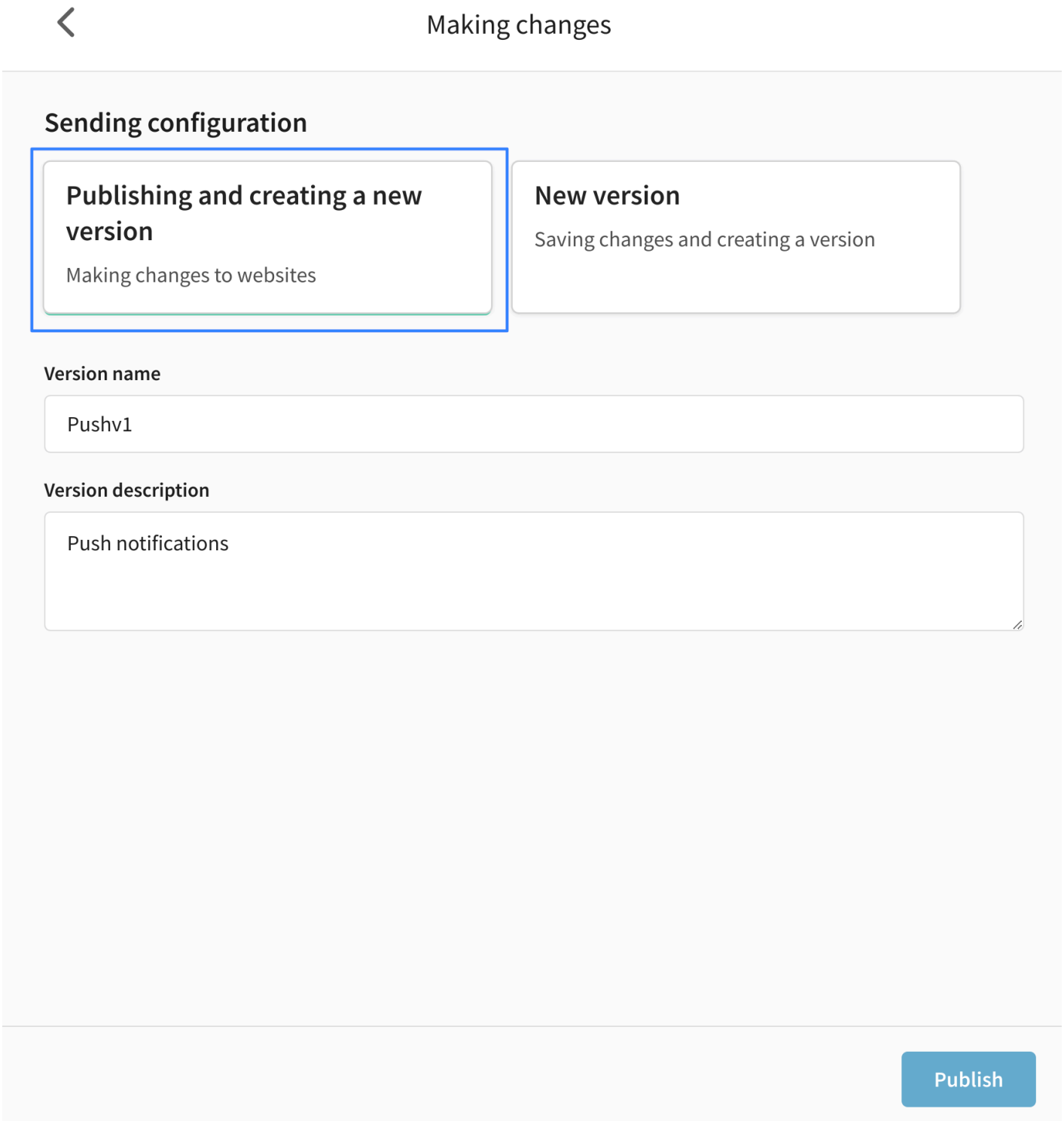
In the activation conditions specify Some events and the variable created earlier. It is responsible for the permission status for showing notifications. The variable will be set to `default` in cases where the browser has not yet requested permission to display notifications from the user. In such situations the pop-up will be triggered:



Come up with a name for the trigger and save it. Select it as the trigger to show your pop-up and save the tag. When the tag is saved and all changes to the container have been made, click Publish:



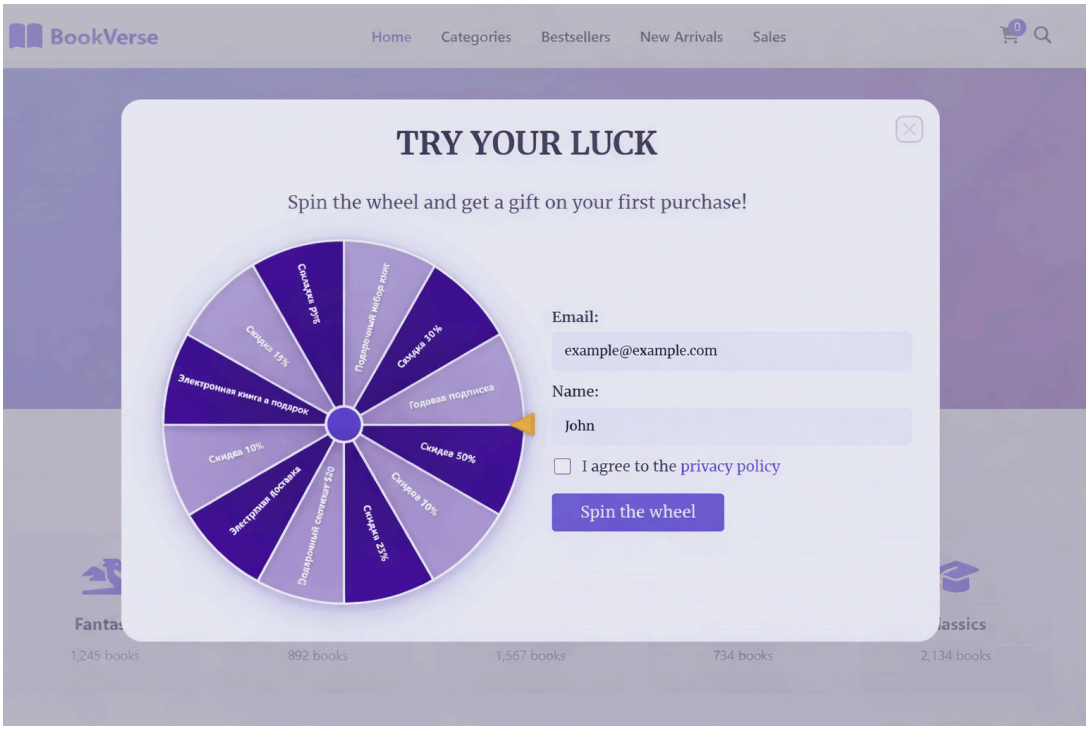
Select the "Publishing and creating a new version" option, provide a title and description for the version and publish it:



The work of the pop-up window has been set-up. Now when a user gets to the page, the window with an offer to subscribe to push notifications will be shown.

Case: Creating a pop-up with the “Wheel of Fortune” widget

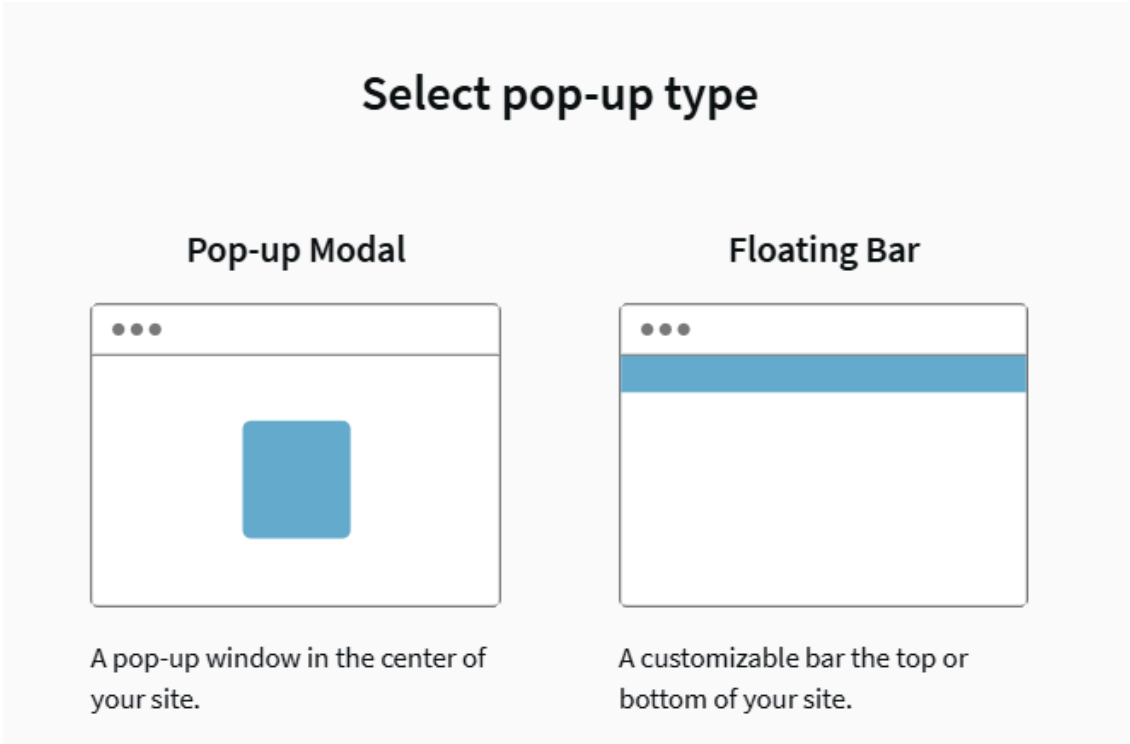
The special “Wheel of Fortune” widget lets you create an interactive pop-up that engages website visitors with a prize game. A user fills out a simple form and spins the wheel to receive a random gift: a discount, a promo code, or another bonus. This guide describes how to create such a pop-up in the Altcraft builder:



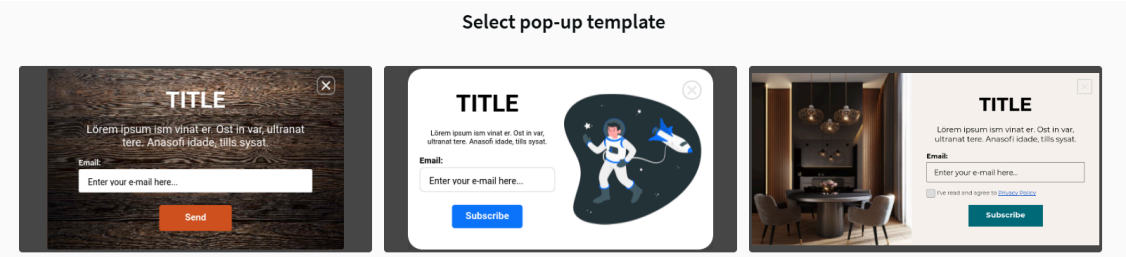
Before configuring the wheel, it’s recommended to create a field in the profiles database where the won prize will be saved. You can learn how to create custom fields in [this article](#).

Creating a pop-up and choosing a template

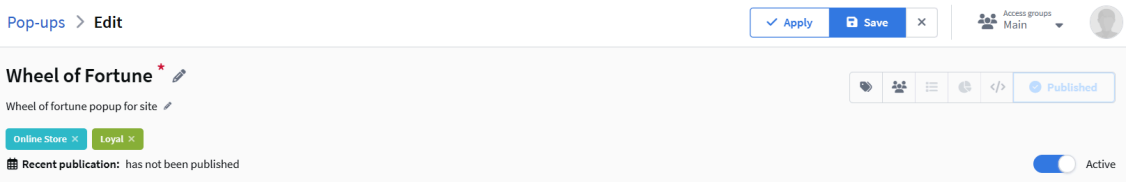
Start by creating a new pop-up in Web Layer — Pop-ups, and click Create. Choose the pop-up type. for a wheel of fortune, Modal window is the best fit—it appears in the center of the screen:



Choose a template. Use any ready-made template as a base. For easier customization, choose templates with a minimalist design (for example, one with an astronaut):



Set a name and description: specify the system name and, if needed, a description:



Integrating the widget code

The “Wheel of Fortune” widget is implemented with HTML, CSS, and JavaScript. To add it, replace the standard template code with the provided ready-made code.

CSS

Go to the CSS tab. Do not remove existing styles. Scroll to the very end of the editor and add the provided CSS code to style the wheel and its container.

```
.formWrapper {
    display: flex;
    align-items: center;
    justify-content: space-around;
    flex-wrap: wrap;
}

.ac-button:disabled {
    opacity: 0.6;
    cursor: not-allowed;
    background-color: #cccccc;
    color: #666666;
    transform: none;
    box-shadow: none;
}

.ac-button:disabled:hover {
    background-color: #cccccc;
    color: #666666;
    transform: none;
    box-shadow: none;
}
```

HTML

In Settings, open the HTML tab. Completely delete the current editor contents and paste the provided HTML code for the wheel.

```
<div class="ac-popup-overlay">
    <div class="ac-modal">
        <div class="ac-modal_content">
            <div style="display: none;" data-popupId="{{popup('copywriting' 'popupId' 'text')}}"></div>
            <div id="firstPage">
                <h4 class="ac-title">
                    {{popup('copywriting' 'title' 'text')}}
                </h4>
                <div class="ac-description">
                    {{popup('copywriting' 'text' 'text')}}
                </div>
                <div class="formWrapper">
                    <div class="canvas-container">
                        <canvas id="canvas" width="420" height="420"></canvas>
                    </div>
                    <form id="ac-form" class="ac-form" action>
                            <div class="ac-field">
                            <label class="ac-label" for="email">
                                {{popup('copywriting' 'emailLabel' 'text')}}
                            </label>
                            <input
                                id="email"
                                type="email"
                                class="ac-input"
                                name="{{popup('form' 'email' 'fieldData')}}"
                                placeholder="{{popup('copywriting' 'emailPlaceholder' 'text')}}"
                            />
                            </div>
                            <div class="ac-field">
                            <label class="ac-label" for="name">
                                {{popup('copywriting' 'nameLabel' 'text')}}
                            </label>
                            <input
                                id="name"
                                class="ac-input"
                                name="{{popup('form' 'name' 'fieldData')}}"
                                placeholder="{{popup('copywriting' 'namePlaceholder' 'text')}}"
                            />
                            </div>
                            <div class="ac-field">
                            <label>
                                <input type="checkbox" name="{{popup('form' 'privacyPolicy' 'fieldData')}}" id="privacyPolicy" required>
                                <span>I have read the <a href="{{popup('copywriting' 'policy' 'link')}}" target="_blank">privacy policy</a>
</span>
                            </label>
                            </div>
                            <div class="ac-validation-msg" for="email"></div>
                            <input type="text" style="display: none;" class="ac-input" id="prizeField" name="{{popup('form' 'prize'
'fieldData')}}">
                        <button class="ac-button" type="button" id="winBtn" disabled>
                            {{popup('copywriting' 'winBtn' 'text')}}
                        </button>

                        <div class="ac-form_submit" style="display: none;">
                            <button type="submit" class="ac-button" id="submitBtn">
                                {{popup('cta' 'text' 'text')}}
                            </button>
                        </div>
                    </form>
                </div>
            </div>
             <div id="secondPage" style="display: none;">
                <h4 class="ac-title">
                    {{popup('copywriting' 'title-congrats' 'text')}}
                </h4>
                <div class="ac-description">
                    {{popup('copywriting' 'text-before-prize' 'text')}}
                    <span id="prizeName"></span>
                    <div>
                        {{popup('copywriting' 'text-after-prize' 'text')}}
                    </div>
                </div>
            </div>
```

```
        </div>
        {{popup('closeButton' 'closeButton' 'closeBtn')}}
    </div>
</div>
```

JavaScript

Switch to the JavaScript tab. Completely delete the existing contents and paste the provided JS code. It handles wheel spinning logic, prize selection, and form behavior.

```
// Prize list. Length may vary
// id - Identifier sent to the server
// text - Label displayed on the wheel sector
const options = [
  {id: 1, text: '$100'},
  {id: 2, text: '$10'},
  {id: 3, text: '$25'},
  {id: 4, text: '$250'},
  {id: 5, text: '$30'},
  {id: 6, text: '$1000'},
  {id: 7, text: '$1'},
  {id: 8, text: '$200'},
  {id: 9, text: '$45'},
  {id: 10, text: '$500'},
  {id: 11, text: '$5'},
  {id: 12, text: '$20'}
]

// Wheel sector colors. If there are fewer colors than sectors,
// colors are repeated cyclically
const colors = ['white', '#3583FF']

// Initial wheel rotation angle
let currentAngle = 0;

// Wheel spin duration is randomized
// Minimum spin duration (seconds)
const minSpinTime = 4;

// Maximum spin duration (seconds)
const maxSpinTime = 7;

// Delay between wheel stop and prize message display (seconds)
const pageSwitchDelay = 2;

// Color of sector labels on the wheel
const textColor = "black";

let isPreview = true;
let myPopup = null;

const allPopups = document.querySelectorAll('[data-ac-popup-container]');

allPopups.forEach(item =>{
    const idContainer = item.shadowRoot.querySelector('[data-popupId={{popup('copywriting' 'popupId' 'text')}}]');
    if(idContainer){
        myPopup = item.shadowRoot;
        isPreview = false;
    }
})

if (!myPopup) {
  myPopup = document;
}

const winBtn = myPopup.getElementById('winBtn');

function initInputs(){
  let emailInput = myPopup.getElementById('email');
  let nameInput = myPopup.getElementById('name');
  let policyInput = myPopup.getElementById('privacyPolicy');

  function updateButtonState() {
      const isEmailEmpty = !emailInput.value.trim();
      const isNameEmpty = !nameInput.value.trim();
      const isPolicyChecked = policyInput.checked;

      console.log(myPopup.getElementById('winBtn'))
      myPopup.getElementById('winBtn').disabled = isEmailEmpty || isNameEmpty || !isPolicyChecked
  }

  emailInput.addEventListener('input', updateButtonState);
  nameInput.addEventListener('input', updateButtonState);
```

```
  policyInput.addEventListener('input', updateButtonState);
  myPopup.getElementById("winBtn").addEventListener("click", spin);

  updateButtonState();
}
if(isPreview) {
  setTimeout(()=>{initInputs()}, 1000);
} else {
  initInputs();
}

let isInitialized = false;

// Initialize wheel rendering once
function initializeWheel() {
    if (isInitialized) return;

    const canvas = myPopup.getElementById("canvas");
    if (!canvas) return;

    const ctx = canvas.getContext("2d");
    if (ctx && !canvas.dataset.initialized) {
        drawRouletteWheel();
        canvas.dataset.initialized = "true";
        isInitialized = true;
    }
}

const observer = new MutationObserver((mutations) => {
    for (let mutation of mutations) {
        if (mutation.type === 'childList') {
            mutation.addedNodes.forEach((node) => {
                if (node.nodeType === 1 && (node.id === 'canvas' || node.querySelector?.('#canvas'))) {
                    setTimeout(initializeWheel, 100);
                }
            });
        }
    }
});

if (document.body) {
    observer.observe(document.body, { childList: true, subtree: true });
}

document.addEventListener('DOMContentLoaded', initializeWheel);

let segmentAngle = Math.PI / (options.length / 2);
let spinTimeout = null;

let spinAngleStart = 0;

let spinTime = 0;
let spinTimeTotal = 0;

let ctx;

// Draw the wheel with all sectors
function drawRouletteWheel() {
  const canvas = myPopup.getElementById("canvas");

  if (canvas.getContext) {
    const centerX = 210;
    const centerY = 210;
    const outsideRadius = 200;
    const textRadius = 130;
    const insideRadius = 0;

    ctx = canvas.getContext("2d");
    ctx.clearRect(0, 0, 400, 400);

    ctx.strokeStyle = "black";
    ctx.lineWidth = 2;
    ctx.font = 'bold 12px Helvetica, Arial';
```

```
    ctx.strokeStyle = "white";
    ctx.lineWidth = 15;
    ctx.beginPath();
    ctx.arc(centerX, centerY, outsideRadius, 0, 2 * Math.PI);
    ctx.stroke();

    ctx.strokeStyle = "black";
    ctx.lineWidth = 2;

    for(let i = 0; i < options.length; i++) {
      const angle = currentAngle + i * segmentAngle;
      const segmentMiddleAngle = angle + segmentAngle / 2;
      ctx.fillStyle = colors[i % colors.length];
      ctx.beginPath();
      ctx.arc(centerX, centerY, outsideRadius, angle, angle + segmentAngle, false);
      ctx.arc(centerX, centerY, insideRadius, angle + segmentAngle, angle, true);
      ctx.stroke();
      ctx.fill();

      ctx.save();
      ctx.fillStyle = textColor;

      const textX = centerX + Math.cos(segmentMiddleAngle) * textRadius;
      const textY = centerY + Math.sin(segmentMiddleAngle) * textRadius;

      ctx.translate(textX, textY);
      ctx.rotate(segmentMiddleAngle);

      const text = options[i].text;
      ctx.fillText(text, -ctx.measureText(text).width / 2, 0);
      ctx.restore();
    }

    // Draw pointer
    ctx.fillStyle = "white";
    ctx.strokeStyle = "#666666";
    ctx.lineWidth = 2;

    ctx.beginPath();
    ctx.moveTo(centerX - 10, centerY - 15);
    ctx.lineTo(centerX + 30, centerY);
    ctx.lineTo(centerX - 10, centerY + 15);
    ctx.closePath();
    ctx.fill();
    ctx.stroke();
  }
}

// Animate wheel rotation
function rotateWheel() {
  spinTime += 30;
  if(spinTime >= spinTimeTotal) {
    stopRotateWheel();
    return;
  }
  const spinAngle = spinAngleStart - easeOut(spinTime, 0, spinAngleStart, spinTimeTotal);
  currentAngle += (spinAngle * Math.PI / 180);
  drawRouletteWheel();
  spinTimeout = setTimeout(rotateWheel, 30);
}

// Start wheel spin
function spin(e) {
  e.preventDefault();
  const form = myPopup.getElementById('ac-form');
  const isValid = form.checkValidity();

  if(!isValid){
    myPopup.getElementById('submitBtn').click();
    return;
  }
  winBtn.removeEventListener('click', spin);
  spinAngleStart = Math.random() * 10 + 10;
  spinTime = 0;
```

```
  spinTimeTotal = Math.random() * (maxSpinTime - minSpinTime) + minSpinTime * 1000;
  rotateWheel();
}

// Handle selected prize
function handlePrize(prize){
  myPopup.getElementById("prizeField").value = prize.id;
  myPopup.getElementById('submitBtn').click();
  myPopup.getElementById('prizeName').innerText = prize.text;
}

// Switch to prize screen
function showSecondPage() {
  myPopup.getElementById('firstPage').style.display = 'none';
  myPopup.getElementById('secondPage').style.display = 'block';
}

// Stop wheel rotation and determine winning sector
function stopRotateWheel() {
  clearTimeout(spinTimeout);

  const degrees = currentAngle * 180 / Math.PI;
  const segmentDegrees = segmentAngle * 180 / Math.PI;
  const index = Math.floor((360 - degrees % 360) / segmentDegrees);

  handlePrize(options[index]);

  const form = myPopup.getElementById('ac-form');
  const isValid = form.checkValidity();

  setTimeout(()=>{
    showSecondPage();
  }, pageSwitchDelay * 1000);

}

// Easing function for smooth deceleration
function easeOut(currentTime, startValue, changeInValue, duration) {
    const normalizedTime = currentTime / duration;
    const squaredTime = normalizedTime * normalizedTime;
    const cubedTime = squaredTime * normalizedTime;

    return startValue + changeInValue * (cubedTime + -3 * squaredTime + 3 * normalizedTime);
}

drawRouletteWheel();

// Override form submit handler
myPopup.getElementById('ac-form').submit = () => {}
```

Content and design setup in the visual editor

After integrating the code, go to Design. Here you can fill the pop-up with text and tune its appearance using the familiar visual interface—without editing code.

Pop-up width

First, set the pop-up width (Form — width). The wheel canvas size is fixed (420 px). So the pop-up width must be at least 420 px. If you want the wheel and the input form on the same horizontal row, set the width to at least 810 px.

Copywriting

On the Copywriting tab, fill in the key text parameters:

- `title` **— main headline on the first screen (for example, “Try your luck!”).**
- `text` **— explanatory text or campaign description.**
- `emailLabel` **and** `emailPlaceholder` **— label and placeholder for the email field.**
- `nameLabel` **and** `namePlaceholder` **— label and placeholder for the name field.**
- `policy` **— a required link to your privacy policy page. Without filling this field, you won’t be able to save the pop-up.**
- `winBtn` **— text on the button that starts the wheel spin (for example, “Spin the wheel!”).**
- `popupId` **— a required parameter. A unique text identifier for the pop-up (for example,** `onlinestore_wheel`**). It’s needed for correct work if there are multiple pop-ups on the page.**

The pop-up has two screens. Parameters for the “win message” screen:

- `title-congrats` **— congratulations headline.**
- `text-before-prize` **— text shown before the prize name.**
- `text-after-prize` **— additional text under the prize message.**

Appearance

Use the other tabs in Design for fine-tuning the look:

- **Background — change the pop-up background color and the page overlay opacity (** `overlay` **).**
- **Border — configure the pop-up corner rounding.**
- **Form — style input fields: spacing, border colors, background, and text.**
- **Actions — style buttons (color, rounding, font).**
- **Typography — choose the common font and styles for all text elements.**

Any changes in the visual editor are immediately shown in the preview pane on the right.

Wheel of fortune customization

The wheel’s look and behavior (prizes, sector colors, spin speed) are configured by editing variables in the JavaScript code.

Go back to Settings — JavaScript. At the beginning of the file, find and edit the following variables.

Prizes (options)

The `options` **variable defines the list of prizes on the wheel. It’s an array of objects, where each object contains an** `id` **and the prize label** `text`**.**

```
const options = [
    { id: 1, text: '$100' },
    { id: 2, text: '$10' },
    { id: 3, text: '$25' },
    { id: 4, text: '$250' },
    { id: 5, text: '$30' },
    { id: 6, text: '$1000' },
    { id: 7, text: '$1' },
    { id: 8, text: '$200' },
    { id: 9, text: '$45' },
    { id: 10, text: '$500' },
    { id: 11, text: '$5' },
    { id: 12, text: '$20' }
];
```

How to change:

- **Prize text: replace the** `text` **value with what you need (for example,** `'10% off'` **or** `'Free shipping'`**).**
- **Number of prizes: add new objects to the array or remove existing ones. The number of wheel sectors will update automatically.**

Sector colors (colors)

The `colors` **variable is an array of fill colors for the wheel sectors. Colors are applied in order, starting from the first sector.**

```
const colors = ['black', 'orange'];
```

How to change: Replace the colors in the array with your own. You can use:

- **English color names:** `'red'`, `'blue'`, `'green'`
- **HEX codes:** `'#4f46e5'`, `'#f59e0b'`, `'#10b981'`
- **RGB/RGBA values:** `'rgb(79, 70, 229)'`

Spin speed

Spin duration is set with two variables:

- `minSpinTime` — **minimum spin time (seconds).**
- `maxSpinTime` — **maximum spin time (seconds).**

The actual duration of each spin will be a random value within this range.

```
const minSpinTime = 4;
const maxSpinTime = 7;
```

How to change: Increase values to make the wheel spin longer, or decrease them for a faster result.

Reveal delay (pageSwitchDelay)

The `pageSwitchDelay` variable sets the pause (in seconds) between the wheel stopping and the automatic switch to the win message screen.

```
const pageSwitchDelay = 2;
```

How to change: Set it to the number of seconds you need.

Text color (textColor)

The `textColor` variable defines the color of prize labels on the wheel sectors.

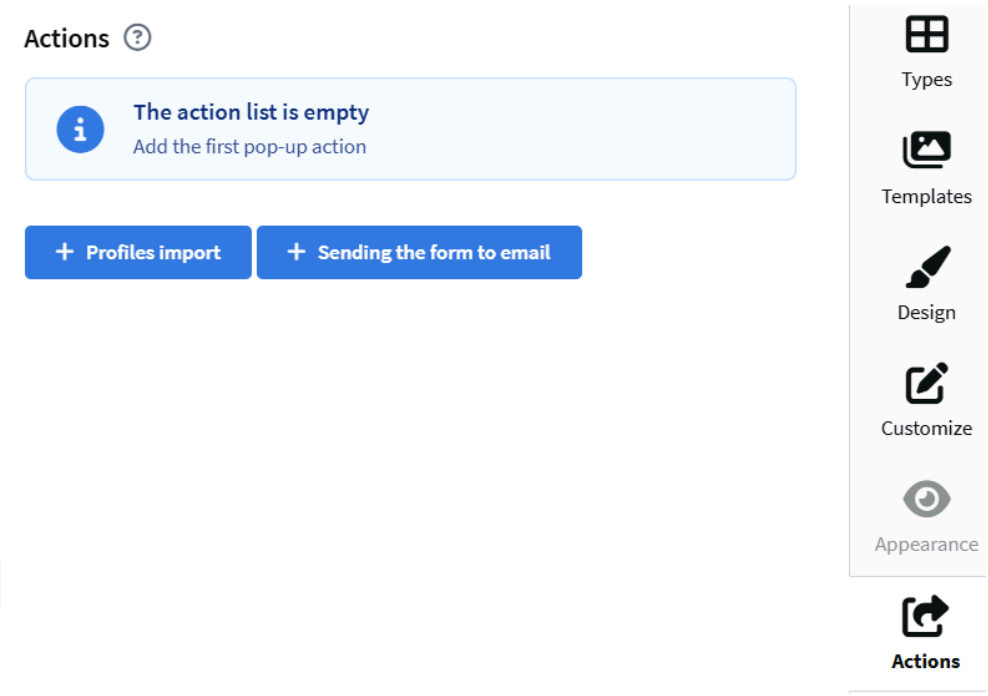
```
const textColor = "white";
```

How to change: Replace the value with any valid CSS color so the text remains readable on the sector background. For example, `"#ffffff"`, `"black"`, `"#1f2937"`.

Saving the won prize value

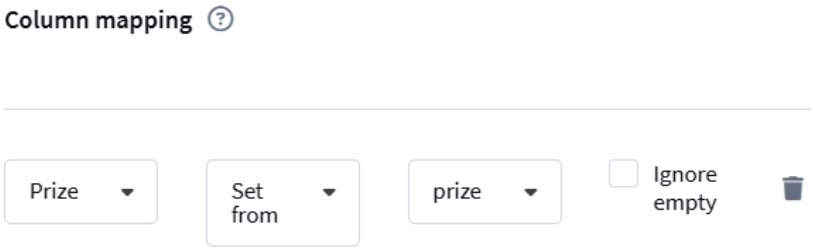
After a user spins the wheel and wins a prize, the result is automatically written into the hidden form field `prize`. To save this value into the user profile in Altcraft, configure the import action in Actions.

In the pop-up editor, open the Actions tab:



Configure profile import:

- **Follow the standard import setup flow described in the pop-up creation article.**
- **In import settings, map the `prize` field from the form to the required field in your Altcraft database.**



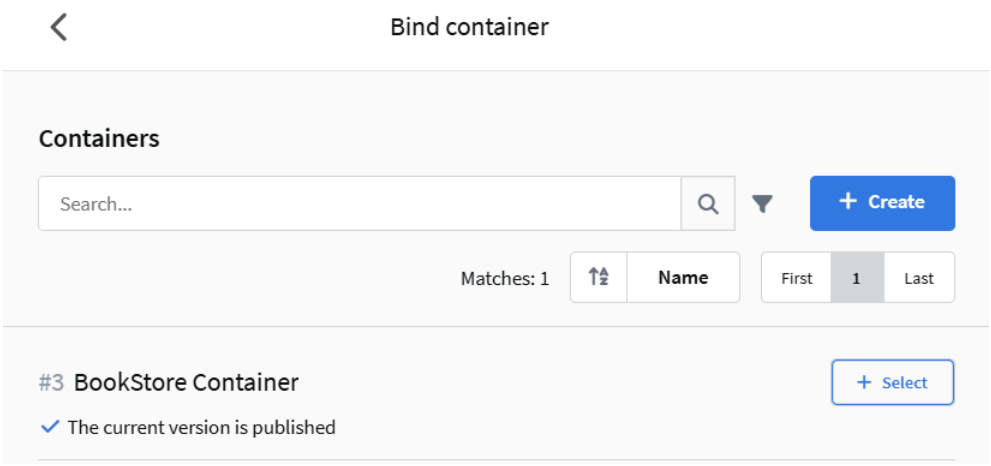
By default, the JavaScript code writes the prize ID into the `prize` field (`prize.id`). If you need to save the prize label text (`prize.text`) instead, edit the `handlePrize()` function in the JS code and replace `prize.id` with `prize.text`.

After configuring import, every form submission will save the prize data (text or ID), along with the user's email and name, into the user profile.

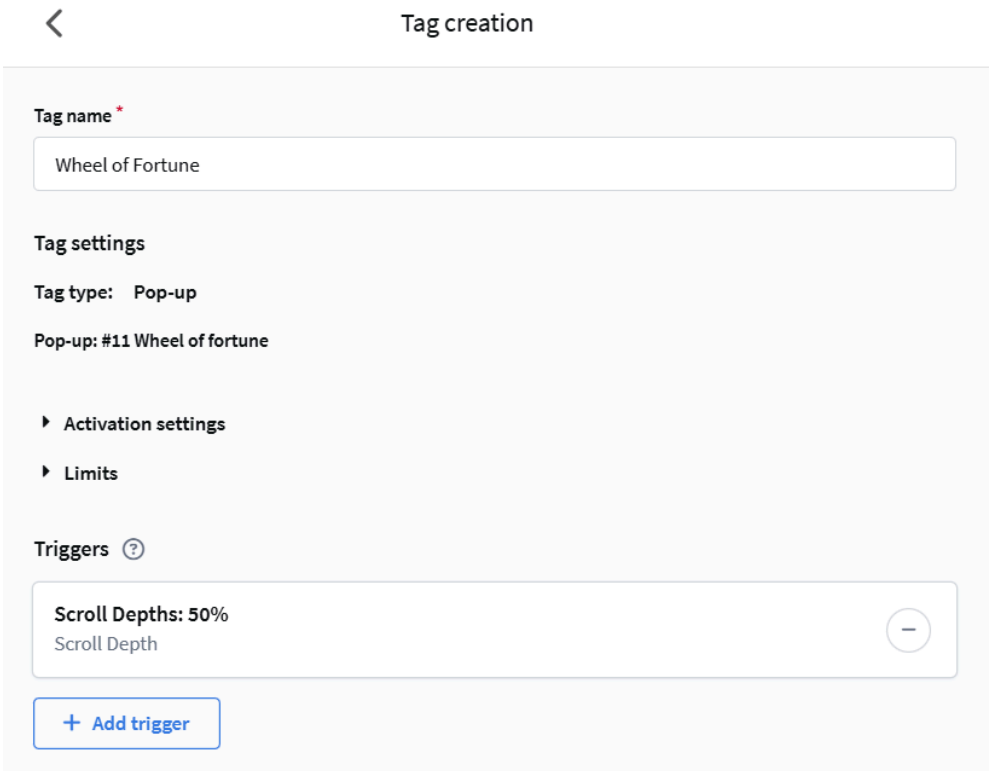
Publishing the pop-up

After you finish configuration, save the pop-up and make sure its status is Active.

In Appearance, bind the pop-up to a Tag Manager container:



Set the tag Name and configure the trigger — the condition that shows the pop-up (for example, Timer or Scroll depth):



More about triggers is in the dedicated article.

After you finish setup, save the pop-up and make sure the status is Active. Once published, the wheel-of-fortune pop-up will be ready and will appear to visitors according to your rules.

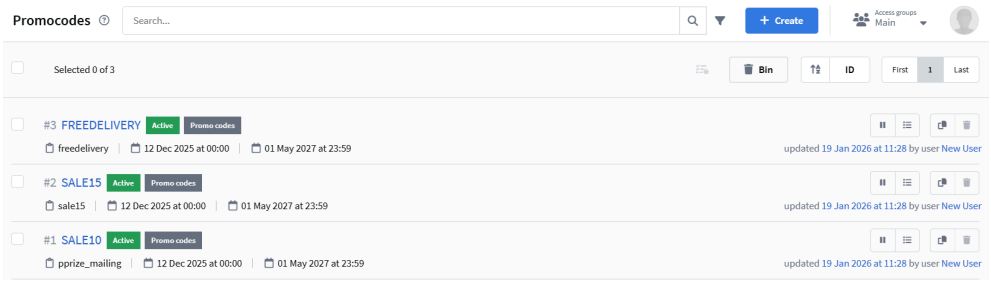
Automatic delivery of personalized prizes via scenarios

To automatically send promo codes that match the won prizes, create a scenario triggered by changes to the `Prize` field. You'll need separate promo codes for each prize type and conditional branching in the scenario.

Creating promo codes

Go to Loyalty — Create to create a new loyalty program. In the program settings, create promo codes for each win. For example:

- **10% discount — promo code `BOOK10`**
- **15% discount — promo code `BOOK15`**
- **Free shipping — promo code `FREESHIP`**
- **And so on for each prize**



You can read more about promo codes in this article.

Creating an email template

Create a universal email template that shows different text depending on the won prize. Use logical expressions to check the `Prize` field value. More about logical expressions and variables in templates is in this article.

The template checks what value the prize field (for example, `Prize`) contains in the user profile (`{lead.Prize}`), and shows the matching text. For each prize type, insert a promo code variable into the email body:

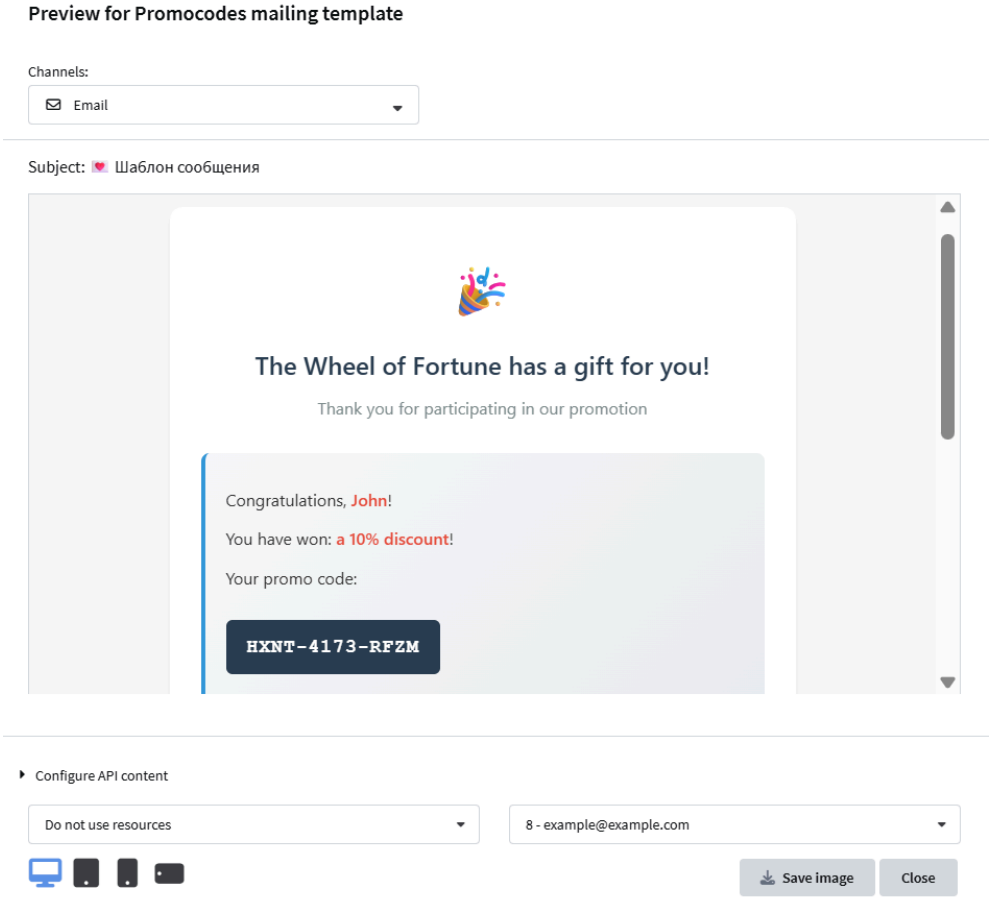
```
{if lead.Prize}

    {if lead.Prize equal "1" or lead.Prize equal "$100"}
        <p>Congratulations, {lead._fname}!</p>
        <p>You have won a cash prize of $100!</p>
        <p>Your promo code: {loyalty.prize100.promocode}</p>
        <p>The promo code is valid for all products in our store.</p>
    {end}

    {if lead.Prize equal "2" or lead.Prize equal "$10"}
        <p>Congratulations, {lead._fname}!</p>
        <p>You have won a cash prize of $10!</p>
        <p>Your promo code: {loyalty.prize10.promocode}</p>
        <p>The promo code is valid for all products in our store.</p>
    {end}

    {if lead.Prize equal "3" or lead.Prize equal "10% discount"}
        <p>Congratulations, {lead._fname}!</p>
        <p>You have won a 10% discount on your next purchase!</p>
        <p>Your promo code: {loyalty.discount10.promocode}</p>
        <p>The discount is valid for all products in our catalog.</p>
    {end}
{else}
    <p>Congratulations, {lead._fname}!</p>
    <p>You have won a prize in our promotion!</p>
    <p>Please contact your manager for details.</p>
{end}
```

As a result, you'll get a template where each condition has its own promo code:



Creating a scenario

Go to Scenarios — Create. Select the database where your campaign participants' profiles are stored. In the entry settings, enable Disable re-entry into the scenario for a customer profile. This ensures the scenario starts for each user only once per new win.

After creating the scenario, add an entry trigger. Choose Profile change as the trigger type, select the `Prize` field, and set the operator to Changed. This means the scenario will start each time the user's prize field appears or changes.

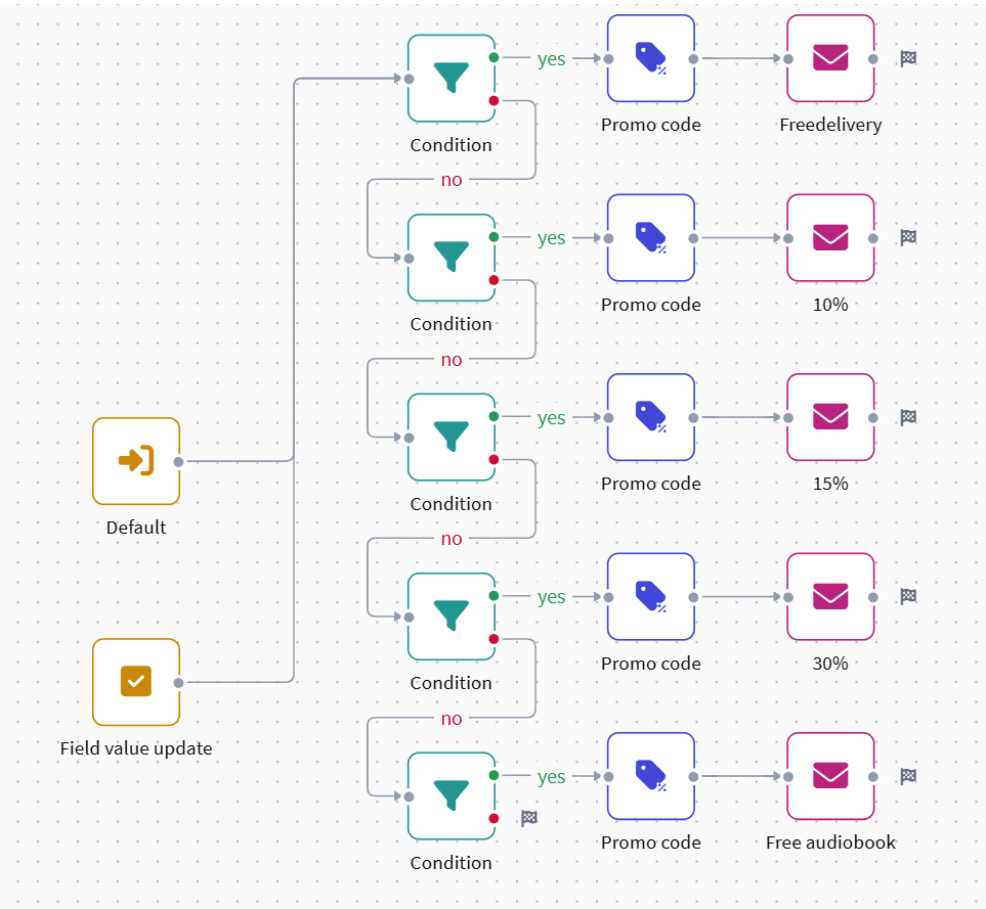
After setting the trigger, add a Condition element to the scenario. In the condition, check the `Prize` field value. For example, set Field to `Prize`, Operator to `equal`, and enter `1` (if you store the prize ID) or the text of the first prize, for example `$100`.

If the condition is true (the user won prize ID 1 or "$100"), add two nodes:

- **Promo code — select the created loyalty program and the specific promo code that matches this prize.**
- **Send email — select the created universal email template.**

If the condition is false, add the next Condition element to check the second prize. Configure it the same way: check whether the `Prize` field contains `2` or the text of the second prize.

Repeat this logic for each prize, creating a chain of conditions and matching actions to assign promo codes and send emails:



At the end, don’t forget to save and activate your scenario.

Basic cases of placing a popup via the Tag Manager

Configuring popup triggers

There are four main popup trigger scenarios available:

- **On page load**
- **On scroll**
- **On click**
- **After a certain time on the page**

INFO

To place a popup, you need to add it to the Altcraft Tag Manager and install the container code on your website.

Showing a popup on page load

To configure a popup to appear immediately after the page loads, create the corresponding trigger in the Tag Manager. The trigger type depends on your site's architecture:

Site architecture	Trigger
For MPA sites (with full page reload)	**Page View**
For SPA sites (without page reload)	**History Change**

To configure, create a trigger in the Tag Manager: go to the Triggers tab —> "Create".

Select the appropriate trigger type and set the activation conditions:

- **All page views — the popup will appear on any page with the container code.**
- **Some page views — for example:** `Page URL –> equals –> https://site.com/contact`**. The popup will only appear on the contact page.**

After configuration, click "Save" and publish the changes in the Tag Manager.

Showing a popup on scroll

The popup will appear when the user scrolls the page to the specified depth. To configure this, create a Scroll Depth trigger and set the value in percentages (for example, 50%). The trigger can fire on both vertical and horizontal scroll depth.

Set the activation conditions:

- **All page views — trigger on any page.**
- **Some page views — for example:** `Page Path –> contains –> /blog/`**. The popup will only appear when scrolling blog pages.**

After configuration, click "Save" and publish the changes in the Tag Manager.

Showing a popup on click

The popup will appear when clicking an element on the page. To configure, create a Click on element trigger.

Set the activation conditions:

- **All clicks — trigger on any click on the page.**
- **Some clicks — for example:** `Click ID –> equals –> subscribe-button`**. The popup will only appear when clicking the button with the ID "subscribe-button".**

After configuration, click "Save" and publish the changes in the Tag Manager.

Showing a popup after a time delay

With this trigger configuration, the popup will appear after the specified time has passed.

To do this, create a trigger in the Tag Manager. Go to the Triggers tab —> Create.
Enter the trigger name and select its type "Timer".

Trigger settings include:

- **Default event name —** `actm.timer`**. In most cases, you don't need to change the name. You can specify another name to create a trigger based on a custom event.**
- **Interval — the time between event firings (in milliseconds). For example, if you specify 5000, the event will fire every 5 seconds. The first firing will also occur after the specified time.**
- **Limit — the maximum number of firings. If you leave the field empty, the trigger will keep firing until the user leaves the page.**

For example, to show a popup 10 seconds after the page loads, set the interval to 10000 and the limit to 1.

After configuration, click “Save” and publish the changes in the Tag Manager.

Tag Manager

	Altcraft Tag Manager is a tool that allows you to manage HTML tags and JavaScript code without any need for developers. It is opportunity for you to create, add, and update code snippets on your site to track user activity and feed data into the platform.

TIP

Altcraft Tag Manager is fully compatible with Google Tag Manager (GTM). You can use both tools on the same website without the risk of conflicts.

Basic terms

Container is a wrapper that stores tags, triggers, and variables. It provides the code that needs to be placed on the site to activate the tags.

Tag is a piece of code that performs a specific function (for example, showing a popup to site visitors).

Trigger is a condition for executing a tag. Triggers monitor certain events on a site (such as clicks or page loads) and activate tags in response to them. To fire a tag, at least one trigger must be configured.

Variable is a parameter to which certain values are passed. Variables can be used in both triggers and tags. In triggers, they are used as an activation filter, they determine when a particular tag should be fired. In tags, variables are used to obtain dynamic values.

In this section

Configuring and installing Tag Manager

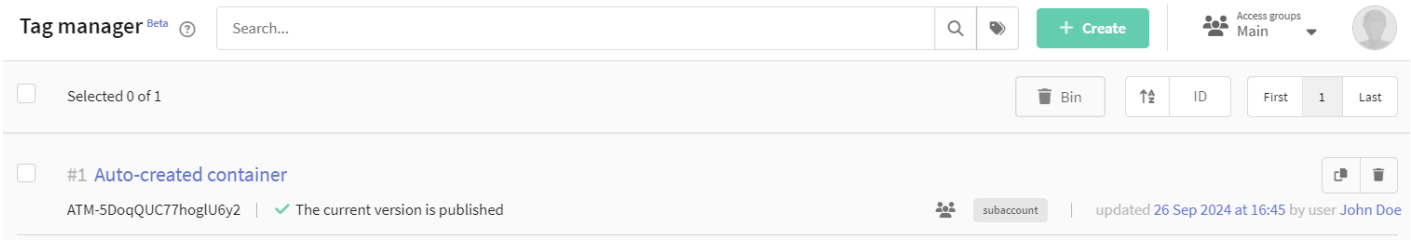
To get started with Tag Manager, go to Weblayer → Tag Manager.

Step 1. Creating a container and installing it on the site

TIP

A container stores tags, triggers, and variables. It is a piece of code that is installed on the site. After installation, all further tracking configuration is done in the platform interface without changing the source code.

By default, a container is already created in your account. You can work with it or add a new one. It is recommended to configure a separate container for each website or mobile app.



The following actions are available in the actions menu on the right side:

— clone the container: a new container is created and the settings, tags, triggers, and variables of the current container are copied to it (the version history is not copied).

— delete the container along with all created tags, triggers, and variables. Tags from this container will no longer be activated on the site.

Creating a new container

To create a new container, follow these steps:

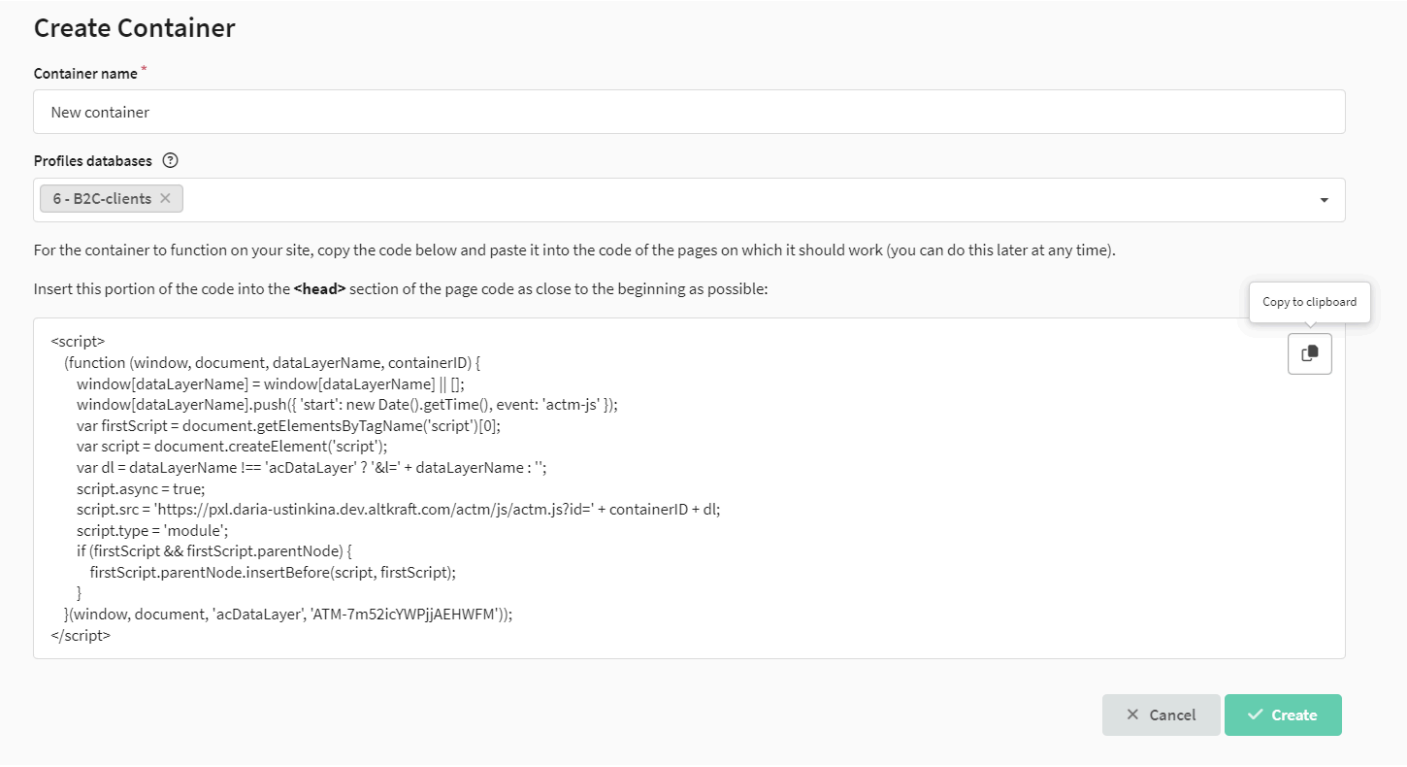
1. Click the + Create button in the upper right corner.

2. Enter a custom name for the container.

3. Select profile databases. Tags track the behavior of site visitors and send data to the Altcraft system. The collected data can be used to search for existing profiles in the database. Specify at least one database for the search.

4. Install the container on the site. To do this, copy the provided code and pass it to your developers. They must place the container code on every page of the site where you plan to use tracking tags. The container code should be added to the `<head>` section, as close to the opening tag as possible.

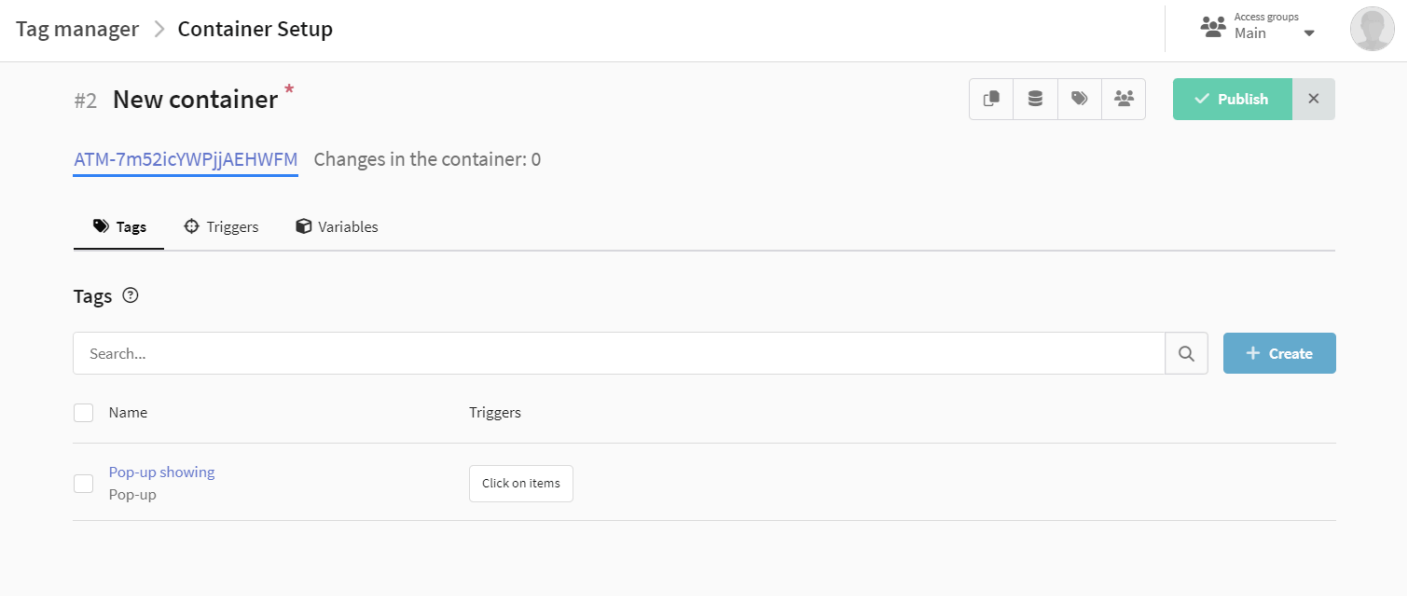
5. Click the Create button to save the container.



Working in a container

To edit an existing container, click its name. When you click the container identifier (ATM-xxxxxxx), a window with the code opens. If you have not yet placed the container on your site, pass the code to your developers.

To the right of the ID there is a counter that shows the number of unpublished changes. After a version is published, the counter is reset to zero.



For additional settings, a quick actions menu is available on the right:

— view version history;

— select databases for profile search;

— assign tags for quick search;

— configure access groups.

INFO

Any changes to the container settings (editing the name or access groups, configuring tags and triggers) are automatically saved in the platform, but they are not applied on the site until you publish the container.

Step 2. Adding tags and configuring triggers

TIP

A tag is a piece of code that runs on the site when a certain condition is met. A trigger is the condition that fires a tag.

Triggers monitor specific events on a site (such as clicks, page loads, or form submissions) and fire tags in response to those events.

To fire a tag, you must configure at least one trigger.

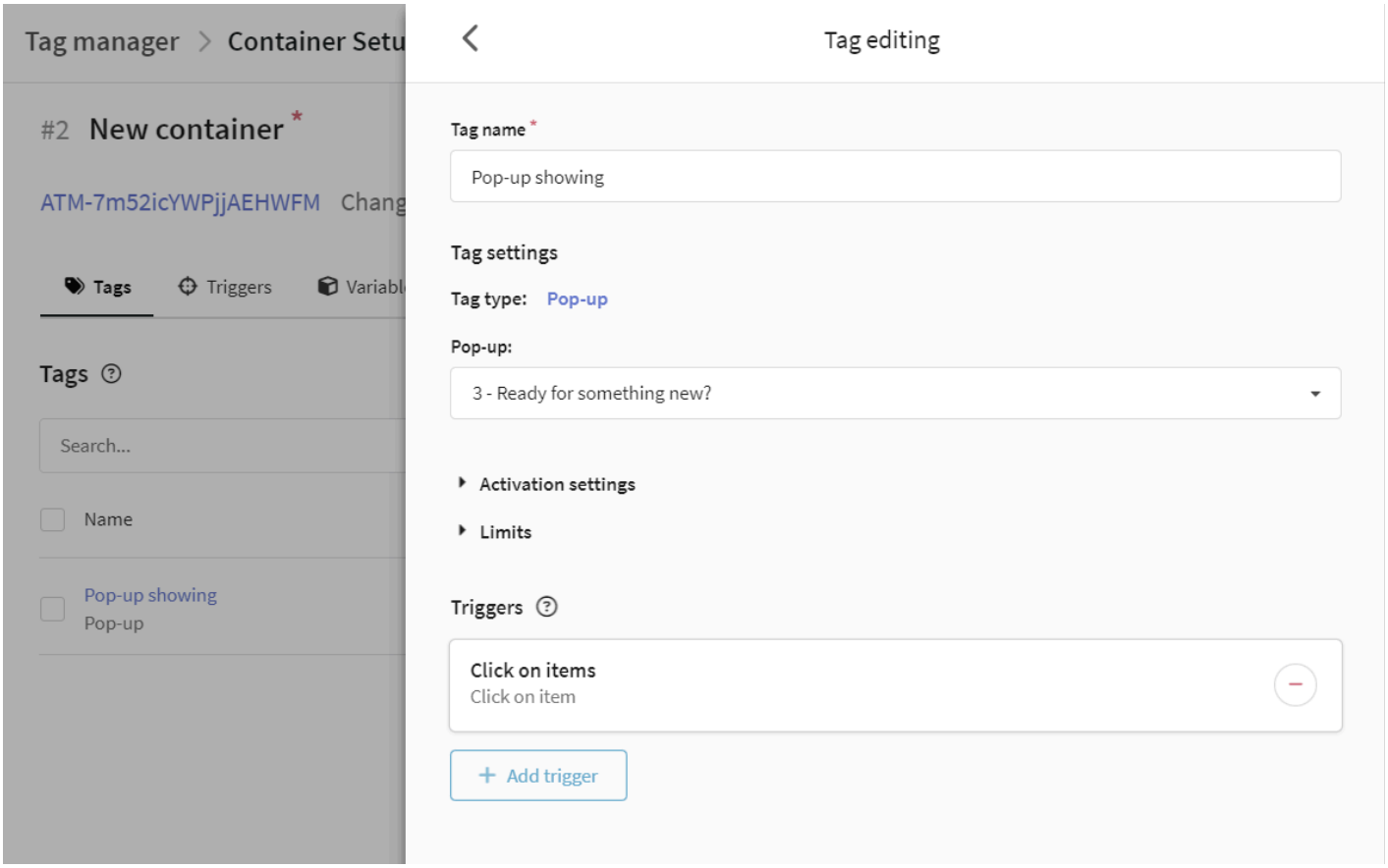
To create a tag and configure the conditions for firing it, follow these steps:

1. On the Tags tab, click + Create.

2. Enter the name of the tag.

3. Select the tag type:

- **Pop-up — used to display a pop-up window to website visitors. You must first create a pop-up and then select it in the Pop-up field. Read more about pop-ups in this article.**
- **Custom HTML — used to create a custom tag. Paste your HTML code into the editor window.**

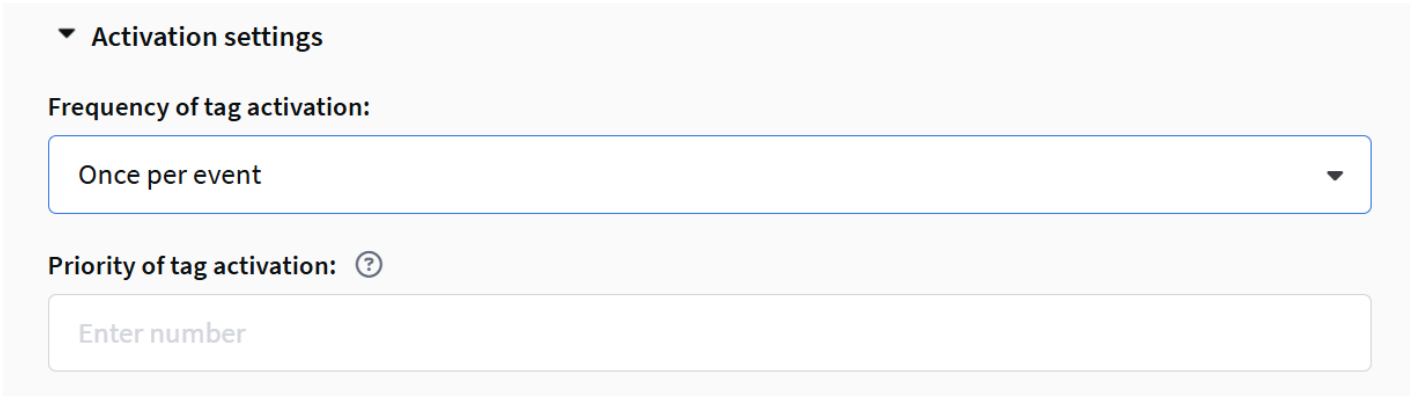


4. Configure activation settings and limits.

Tag activation settings

Frequency of tag activation:

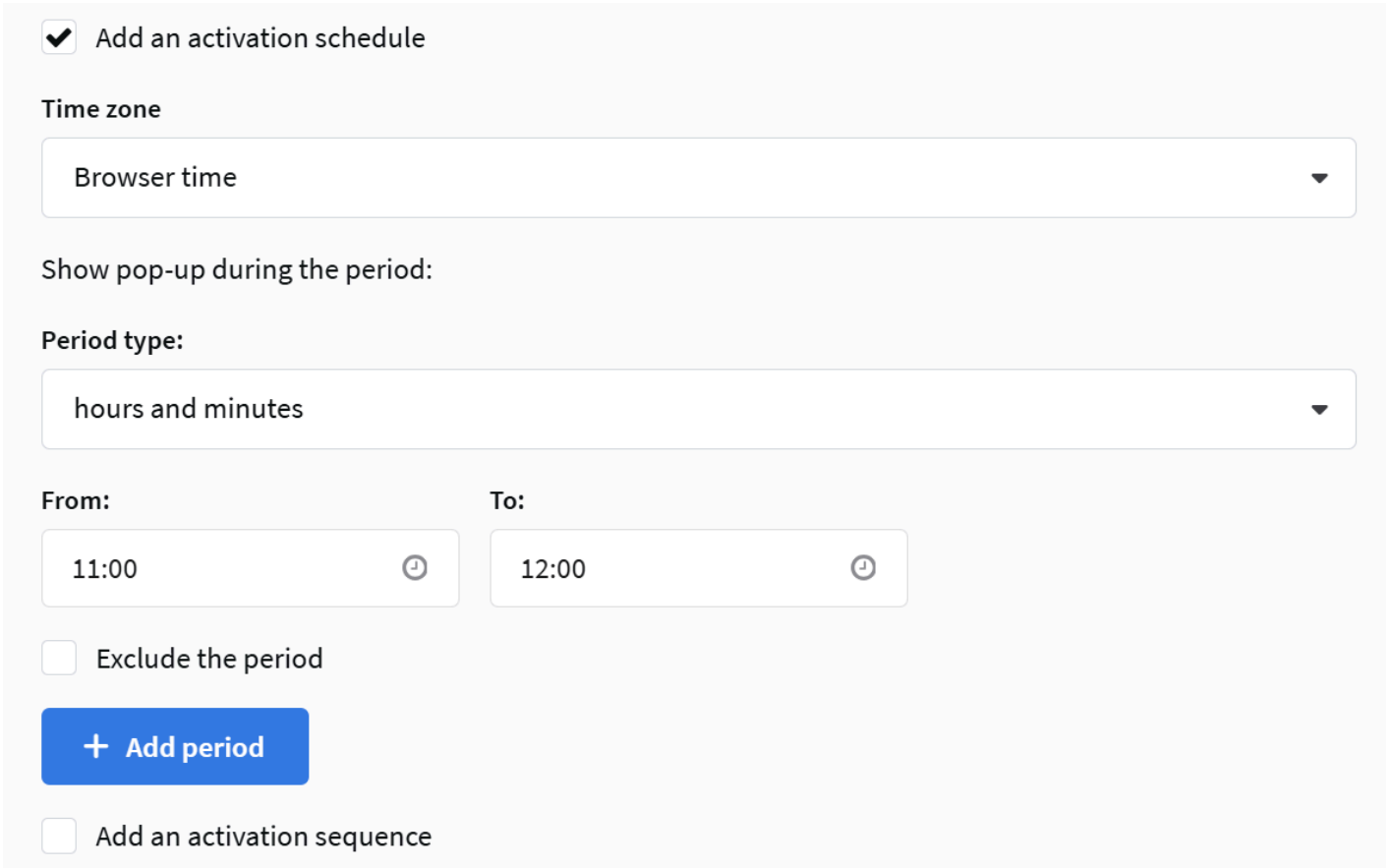
- **Once per event — the tag is activated only once when a specific event occurs.**
- **Once per page — the tag will fire only once per page, regardless of the number of events.**
- **No restrictions — the tag is activated each time the event specified in the trigger occurs.**



Priority of tag activation. Sets the order in which tags are fired. By default, the priority of a tag is 0. To change the priority, specify a positive or negative integer. For example, a tag with priority 2 will be fired before a tag with priority 1. Tags run asynchronously.

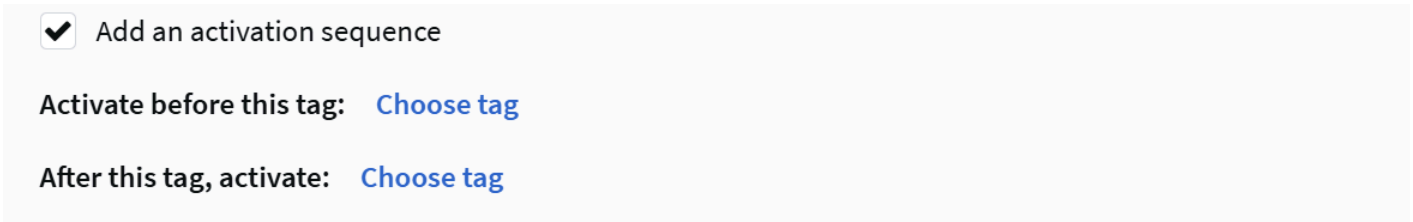
Add an activation schedule. Lets you configure tag activation during a given period. For example, you are running a promotion that lasts only one hour. In this case, the banner with information should be shown to site visitors only during this hour.

Select the period type, set the start and end date and time. If you want the tag to remain inactive during a certain period, select the Exclude period option. You can also combine several time periods — click the Add period button.



Add an activation sequence:

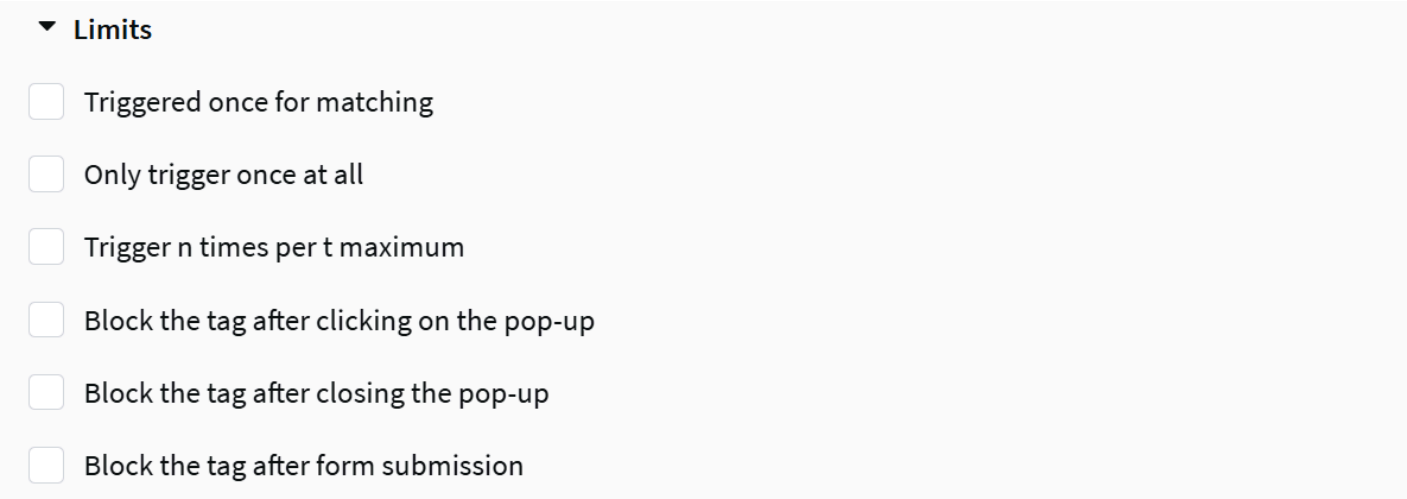
- **Activate before this tag. Select a tag that must be fired before the current tag. If the selected tag does not fire, the tag you are currently creating will not be activated.**
- **Activate after this tag. Select a tag that must be fired after the current tag. If the tag you are currently creating does not fire, the selected tag will not be activated.**



Tag limits

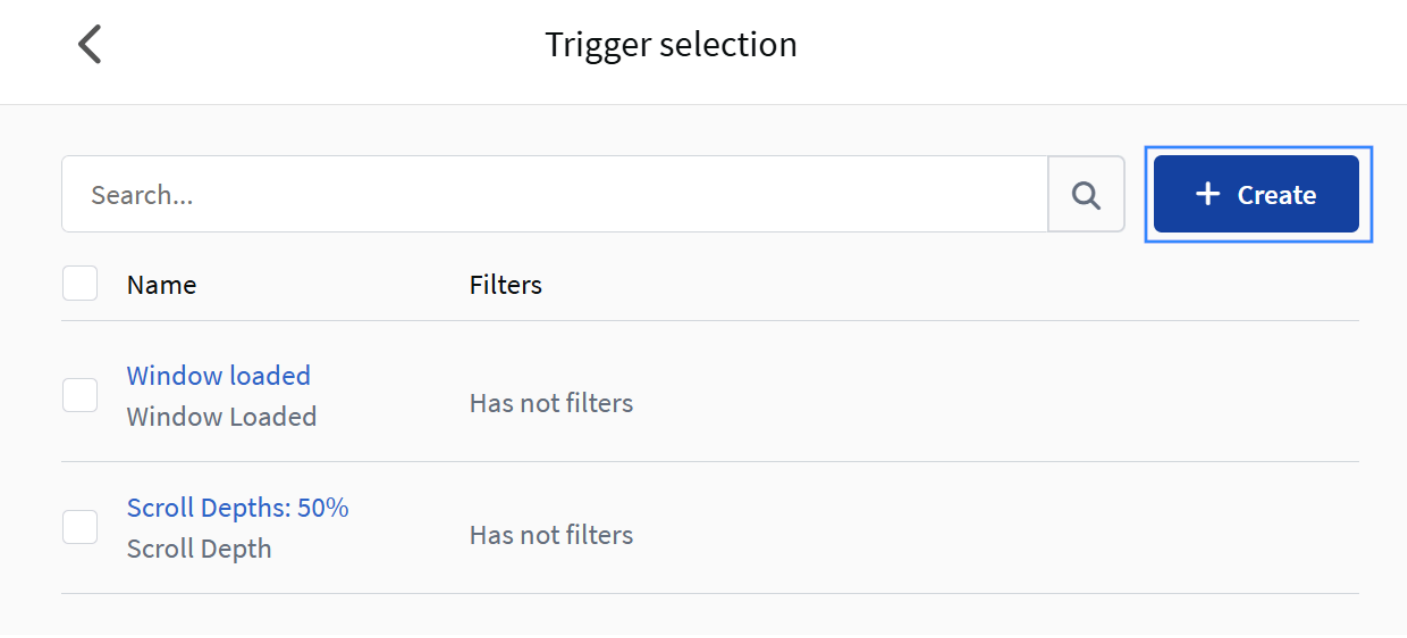
Limits

- **Trigger only once per match — the tag is executed only once each time a condition that matches its settings becomes true. If the condition becomes true again, the tag will not be executed again. This applies to conditions related to matching, such as page URL, page title, and other parameters that may change while the page is being viewed.**
- **Trigger only once — the tag is executed only once, regardless of which conditions are met. Use this limit to initialize scripts that must be executed only once.**
- **Trigger at most n times per period t — limits how many times the trigger can fire during a certain period of time.**
- **Block the tag after clicking in a pop-up — after clicking any element in the pop-up, the tag will not be activated again, even if the event it listens to occurs again. This prevents multiple tag activations on repeated clicks in the pop-up.**
- **Block the tag after closing a pop-up — after clicking the Close button in the pop-up, the trigger will not fire again, even if the event occurs again.**
- **Block the tag after submitting a form — after the form associated with the tag is submitted, the trigger will not fire again, even if the conditions for its activation are met again.**



5. Click Select triggers. A new window opens with all available triggers. If you have already created a trigger, you can go straight to step 9.

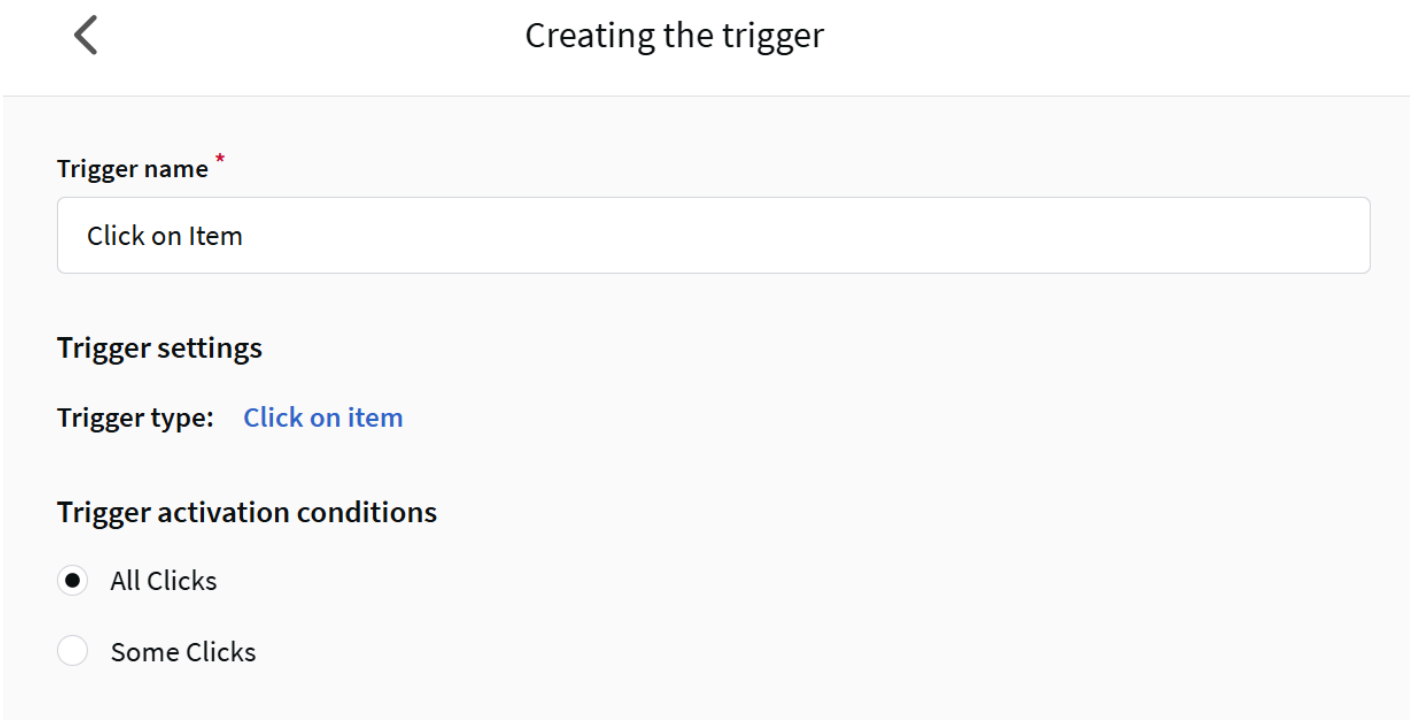
6. To set up a new trigger, click + Create:



7. Enter the name of the trigger and select its type.

TIP

Read more about trigger types in this article.



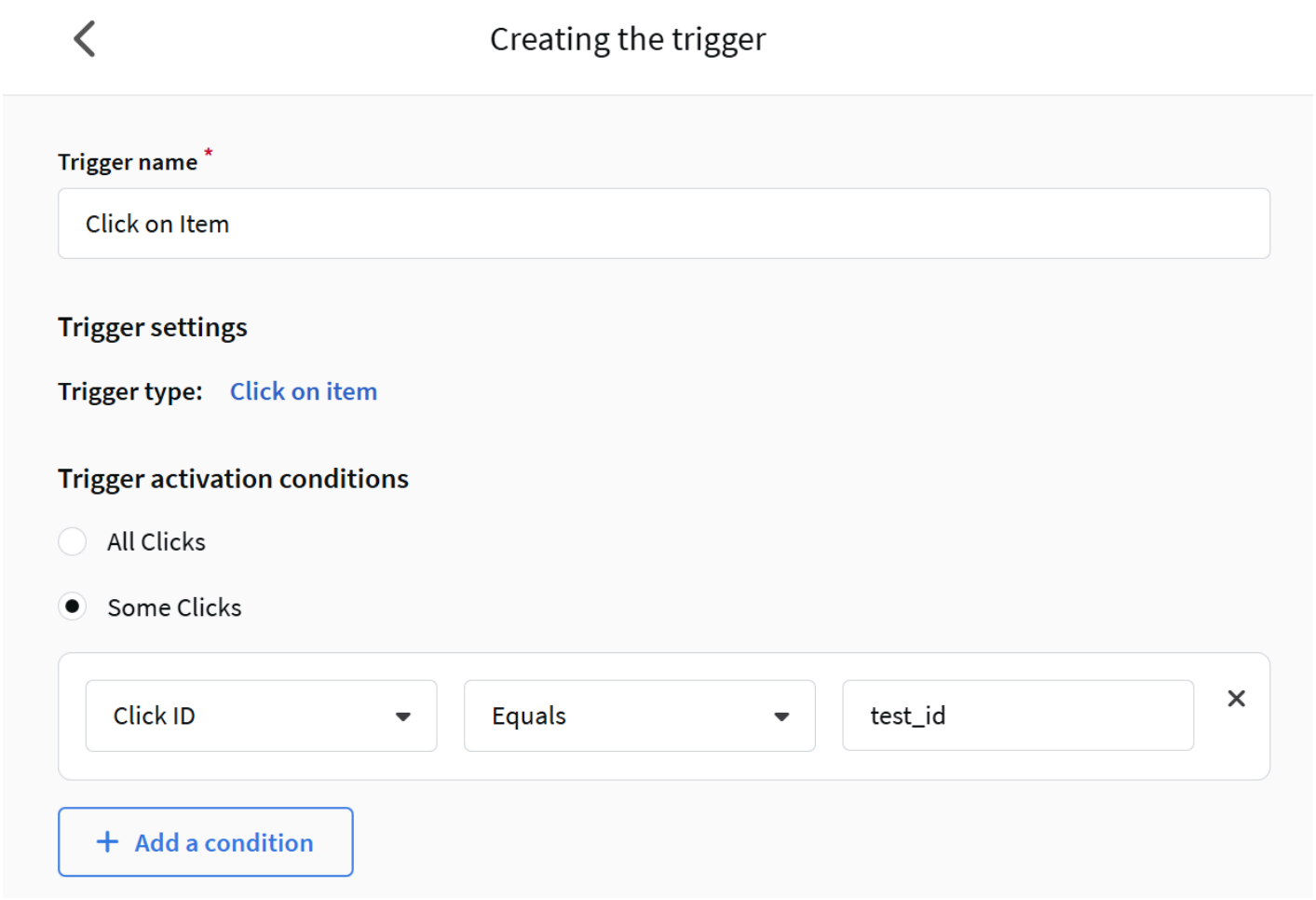
8. By default, triggers fire in response to all events of a certain type. To set additional trigger activation conditions, configure a filter. To do this, in the Trigger activation conditions block, select Some events. After configuring the filter, save the trigger using the button at the bottom of the page.

INFO

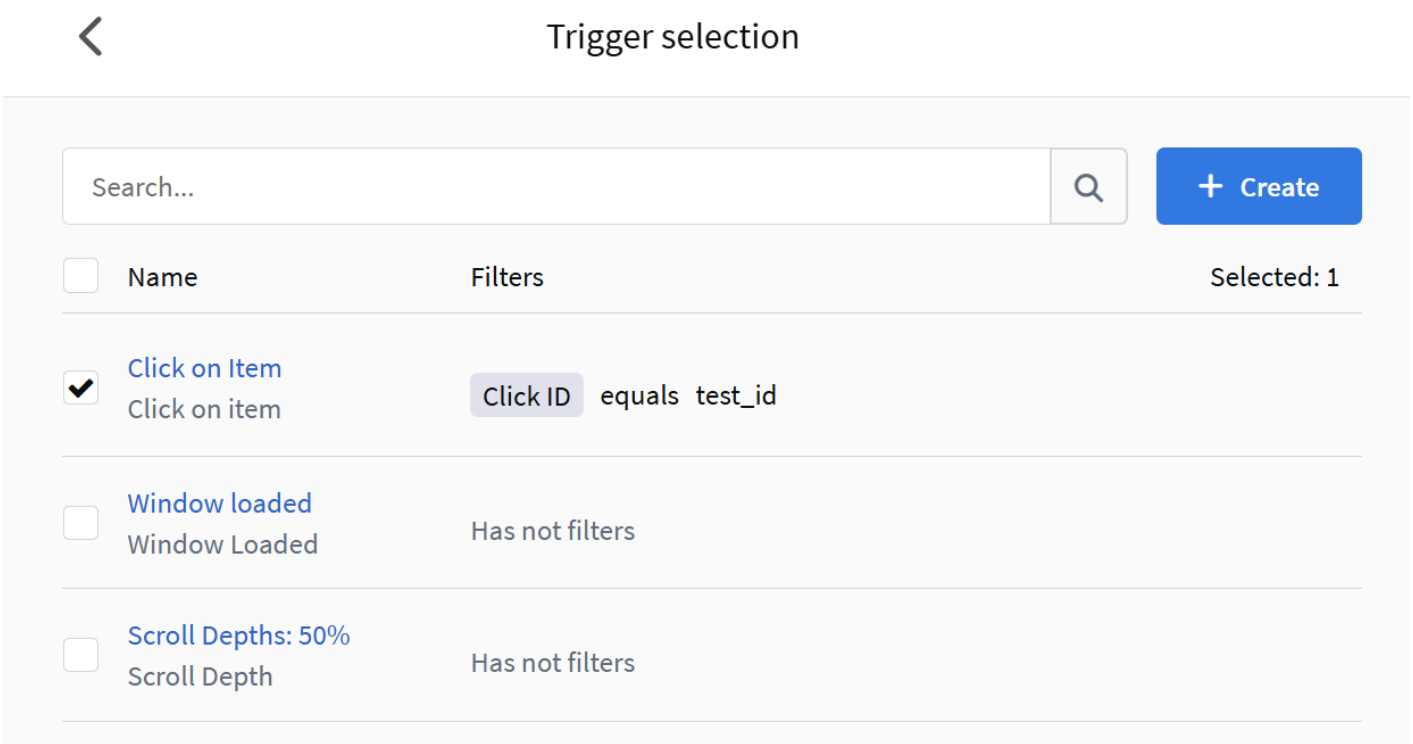
A filter consists of a variable, an operator, and a value. Select a variable (built-in or custom) and an operator. Then select or manually enter a text value for the variable.

Each event on the site is associated with a set of variables that contain information about it. When an event occurs, the trigger reads a specific variable. Tag Manager compares the value of this variable with the value specified in the filter. If the value of the variable matches the parameters specified in the filter, the trigger is activated and fires the tag.

Thus, using a filter, you can limit the audience for which the tag will be activated.



9. In the Trigger selection window, select one or more triggers and click the Select button at the bottom of the page. If you select multiple triggers, the tag will be activated when any of them fires.

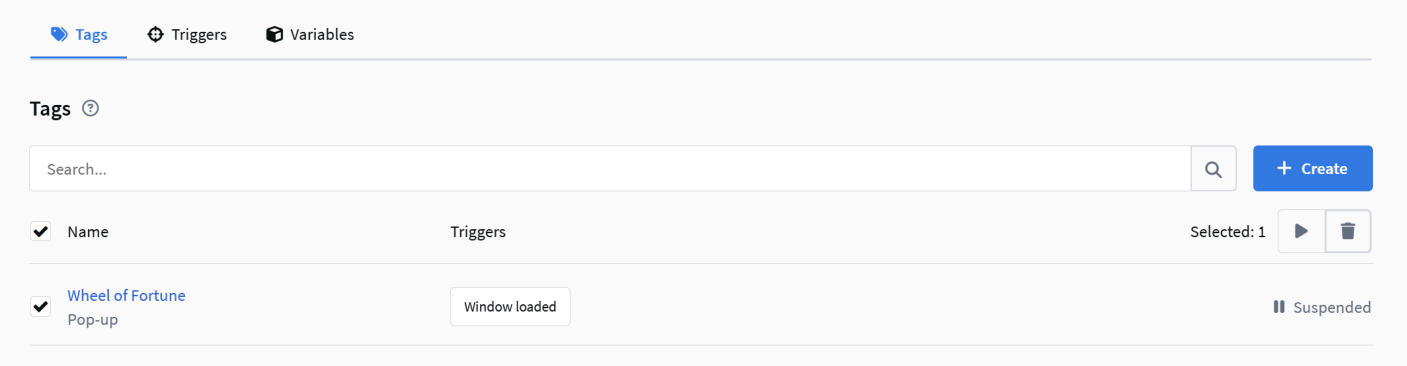


10. In the Tag creation window, check the settings and click Save.

The created tag appears on the Tags tab. To change its settings later, click its name. To temporarily stop tags from running on a page, select the tags and click

, and then publish a new container version. You can also remove unnecessary tags from the container using

.



Step 3. Publishing a container version

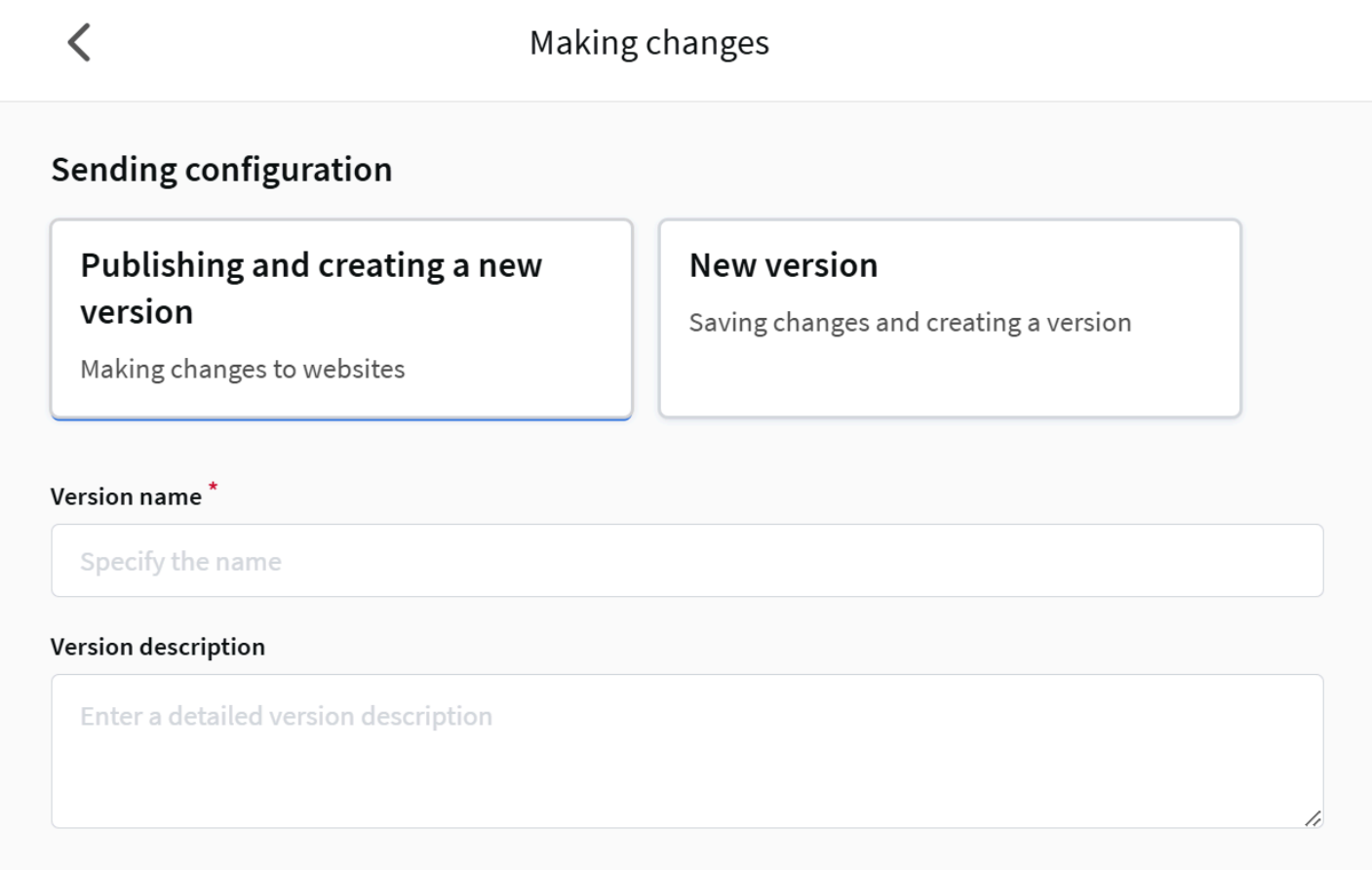
A version is a saved copy of the container at a specific point in time. If you make changes to the container (add a tag, change a trigger), you can save these changes as a version. To do this, in the Container setup window click the Publish button. Two sending configurations are available:

- **Publish and create a new version — a new version of the container is created and all changes are applied on the site.**
- **New version — a new version is created and all changes are saved, but the container is not published and no changes are applied on the site.**

CAUTION: Checking variables before publishing

Before publishing, the system checks that all variables used in HTML tags exist in the container. If unknown variables are found (for example, removed or renamed ones), publishing is canceled and an error message is shown.

It is recommended to provide a name and description for the version so you can see exactly what changes were made. Then confirm saving by clicking Publish at the bottom of the page.

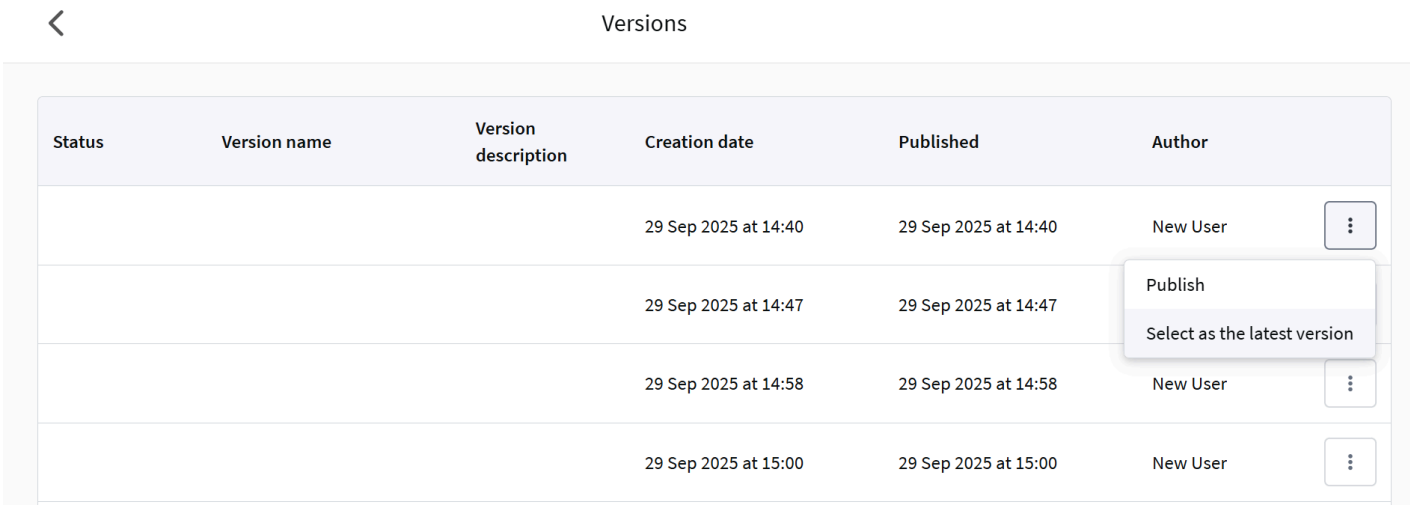


To open the container version list, use the

button in the quick actions menu. The table shows which user created or published each version and when.

You can switch to another version or publish it instead of the current one. To do this, open the additional actions menu

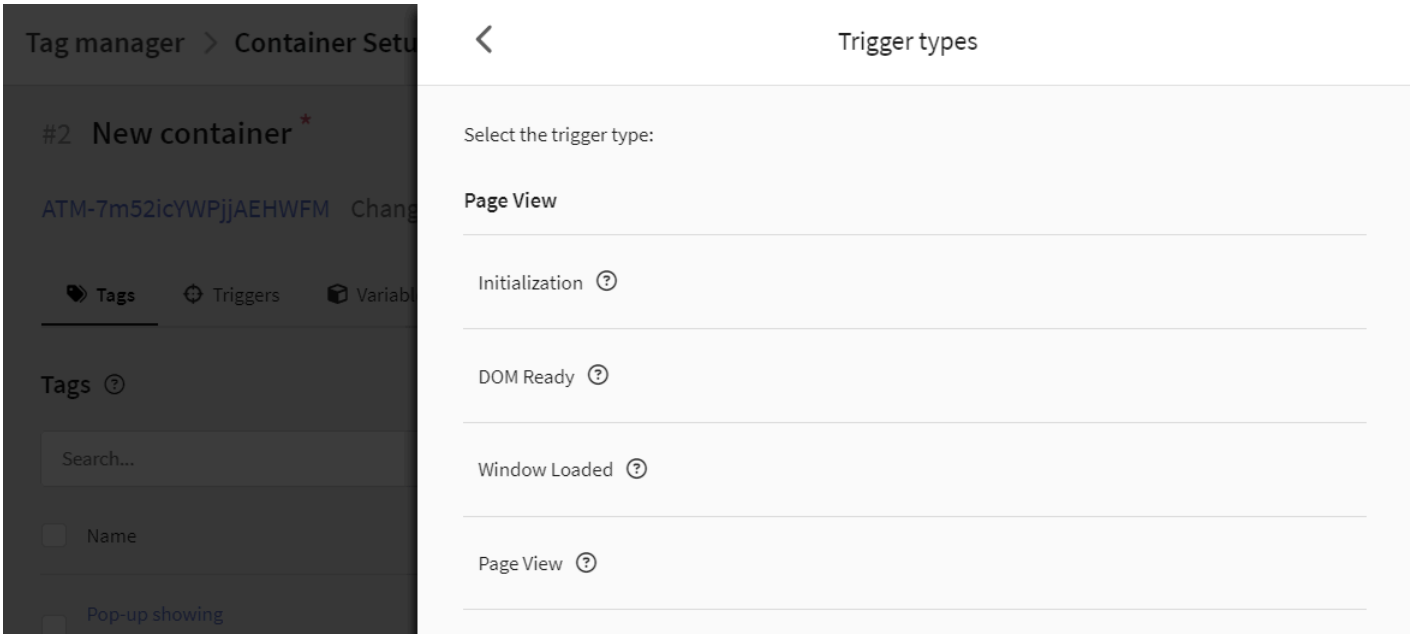
and click Select as the latest version:



The first version of the container has been published and Altcraft Tag Manager has been installed on your site. Don’t forget to check that the tag is working correctly.

Trigger types

Page view



Initialization

The trigger fires before all other triggers fire. Use this trigger to have it fire the tags you want before other triggers.

DOM ready

The trigger fires after the browser has completed processing the HTML code of the page.

The browser has fully loaded the HTML and the DOM tree has been built, but external resources such as `<img>` images and styles may not have been loaded yet.

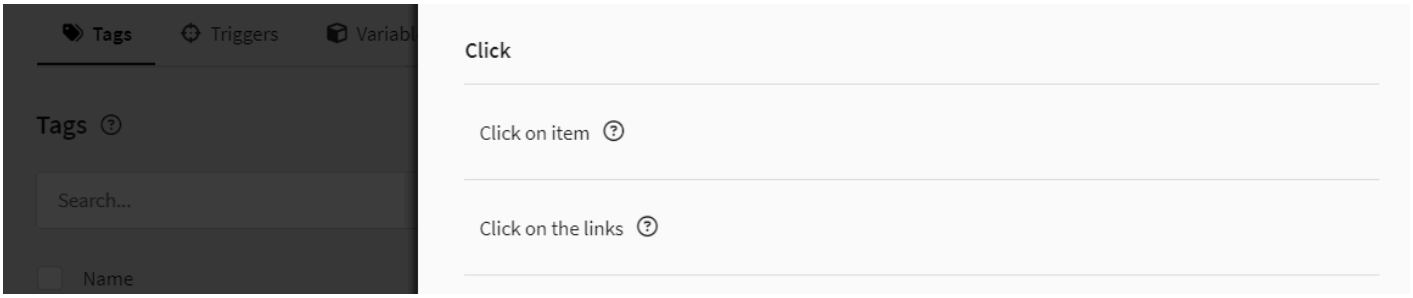
Window loaded

The trigger fires after the page has fully loaded, including images, scripts, and other embedded objects.

Page view

The trigger fires when the page begins to load. Used to obtain data only on page impressions.

Click



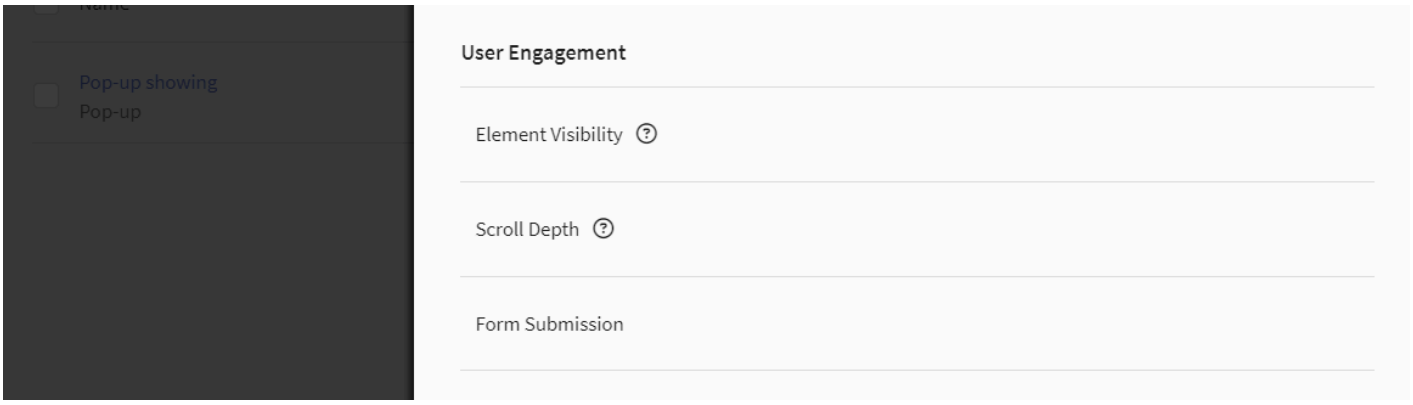
Click on item

The trigger fires when you click on any page item, including links, images, buttons, etc.

Click on the links

The trigger fires when HTML links are clicked, for example, `<a hrefs="https://example.com">Link </a>`

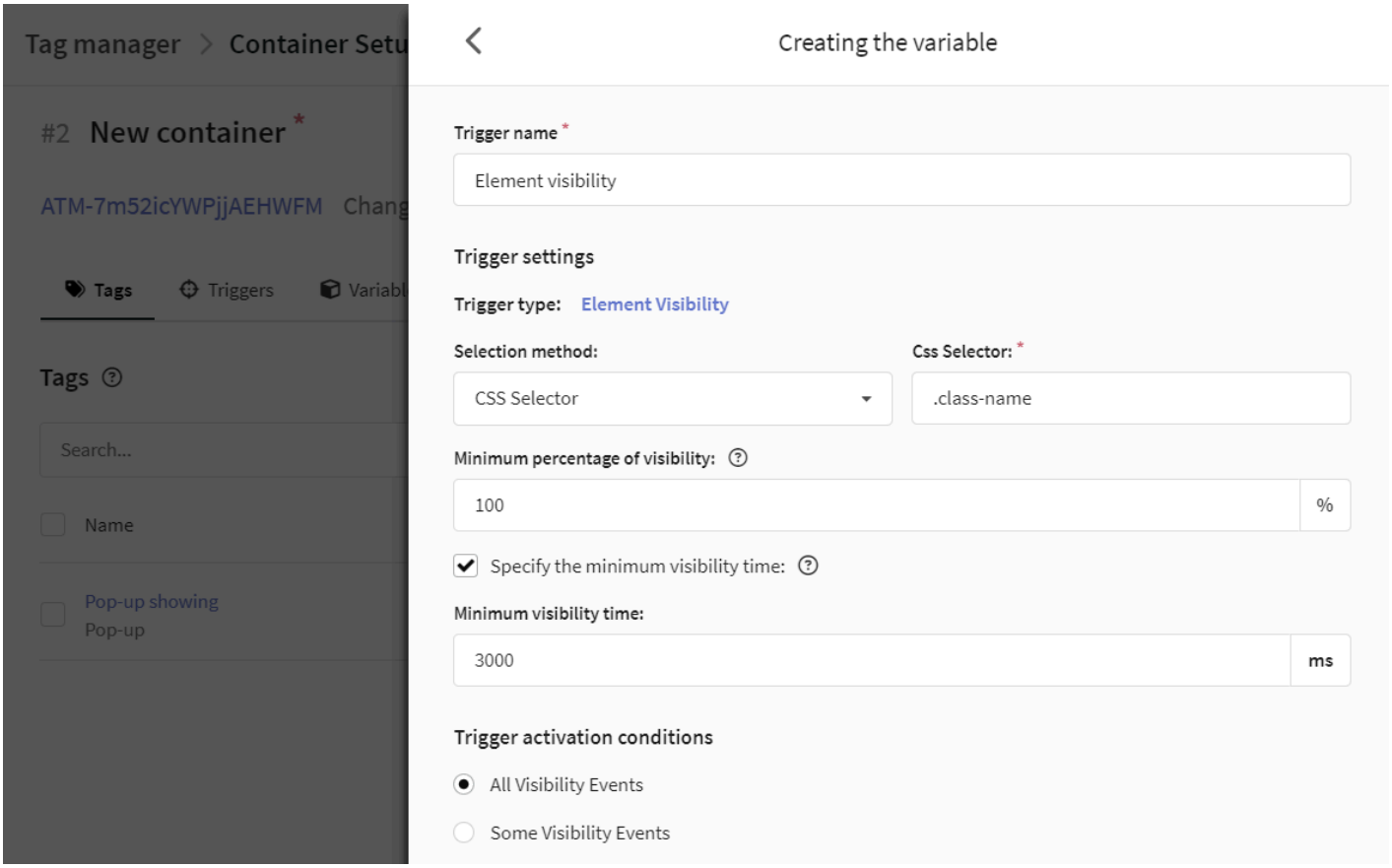
User engagement



Element visibility

The trigger fires when a specific element becomes visible in the viewport.

The visibility of an element can change as a result of various actions: loading the page, scrolling, switching to a background browser tab, changing the position of the element through code, etc.



Selection method:

- **Identifier is used to track the visibility of a single element with a specific `id` attribute value. For example, if you specify `test`, the element with `id="test"` will be selected.**
- **CSS selector is used to track the visibility of multiple elements that match a given CSS selector pattern. For example, `.class_name` will select elements on the page with `class="class_name"`.**

Minimum percentage of visibility - the percentage of the selected element that must be visible on the user's screen for the trigger to fire. The default is 100%, which means the element should be completely displayed on the screen.

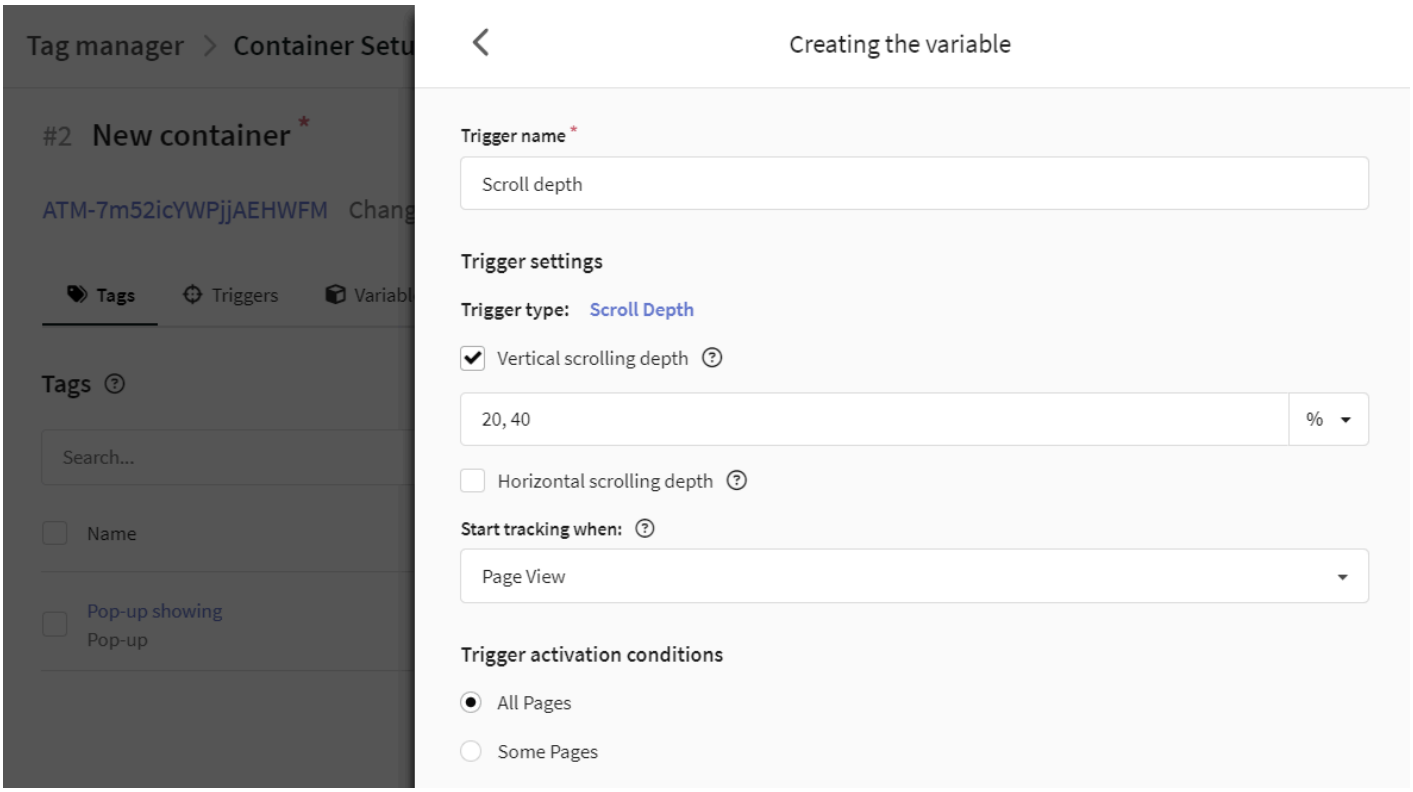
You can also specify the minimum visibility time. The minimum visibility time is the period of time an element must remain visible on the screen for the trigger to fire.

TIP

Please note that time periods are cumulative. If an element was visible for 5000 ms, then hidden and visible again for 3000 ms, then the visibility time would be 8000 ms.

Scroll depth

The trigger fires depending on how far the user scrolls the page down or to the right.



To configure the trigger, you must select at least one of the scrolling options:

- **Vertical scrolling depth — the trigger is activated based on how far the user has scrolled down the page.**
- **Horizontal scrolling depth — the trigger is activated based on data about how far the user has scrolled the page to the right.**

You can use both options at the same time. The scroll depth value is specified in pixels or as a percentage (of the page height and width). If you specify multiple values separated by commas, the trigger is activated when each of them is reached.

You can also specify at what stage of the page load you want to start tracking the event:

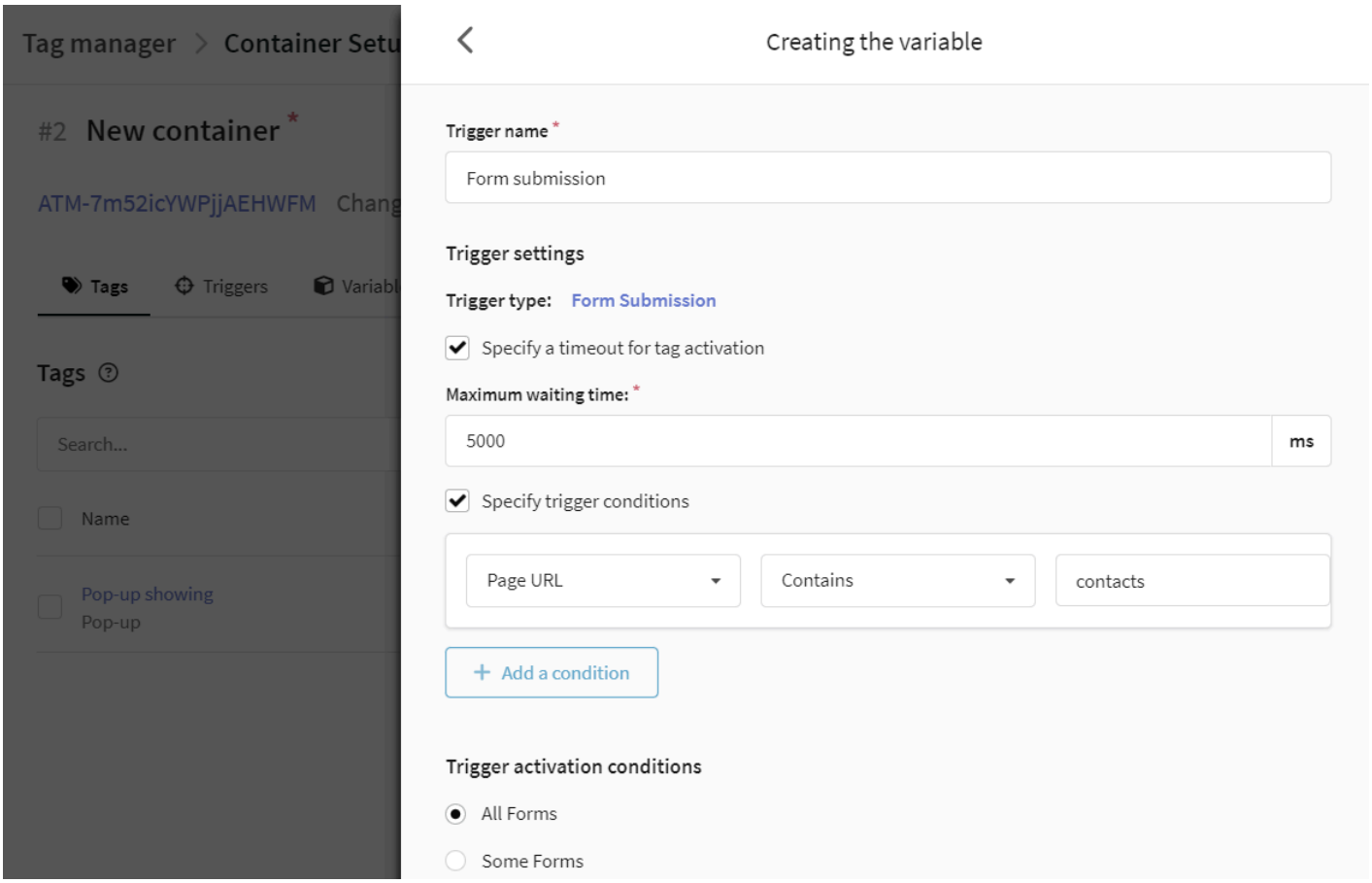
- **Window loaded;**
- **DOM model is ready;**
- **Page view.**

TIP

If the specified scroll depth enters the viewport when the page loads, the trigger fires without scrolling.

Form submission

The trigger fires every time the user tries to submit the form.



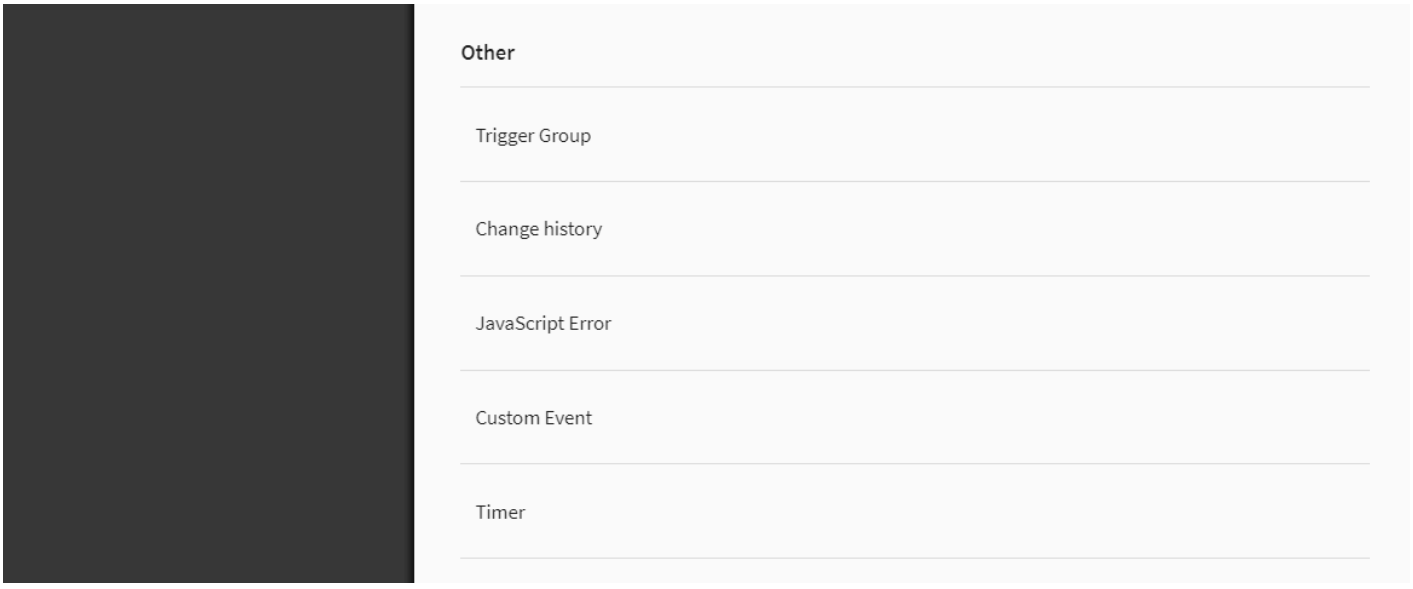
In trigger settings you can:

- **Specify a timeout for a tag activation: form submission is blocked until the specified timeout has expired or until all tags dependent on this trigger have been activated. The wait time is specified in milliseconds (5000 ms = 5 sec).**
- **Specify trigger conditions: use variables to select under what conditions the trigger should monitor events.**

TIP

It is recommended to use a filter in the trigger indicating the URL of the page on which the form is located.

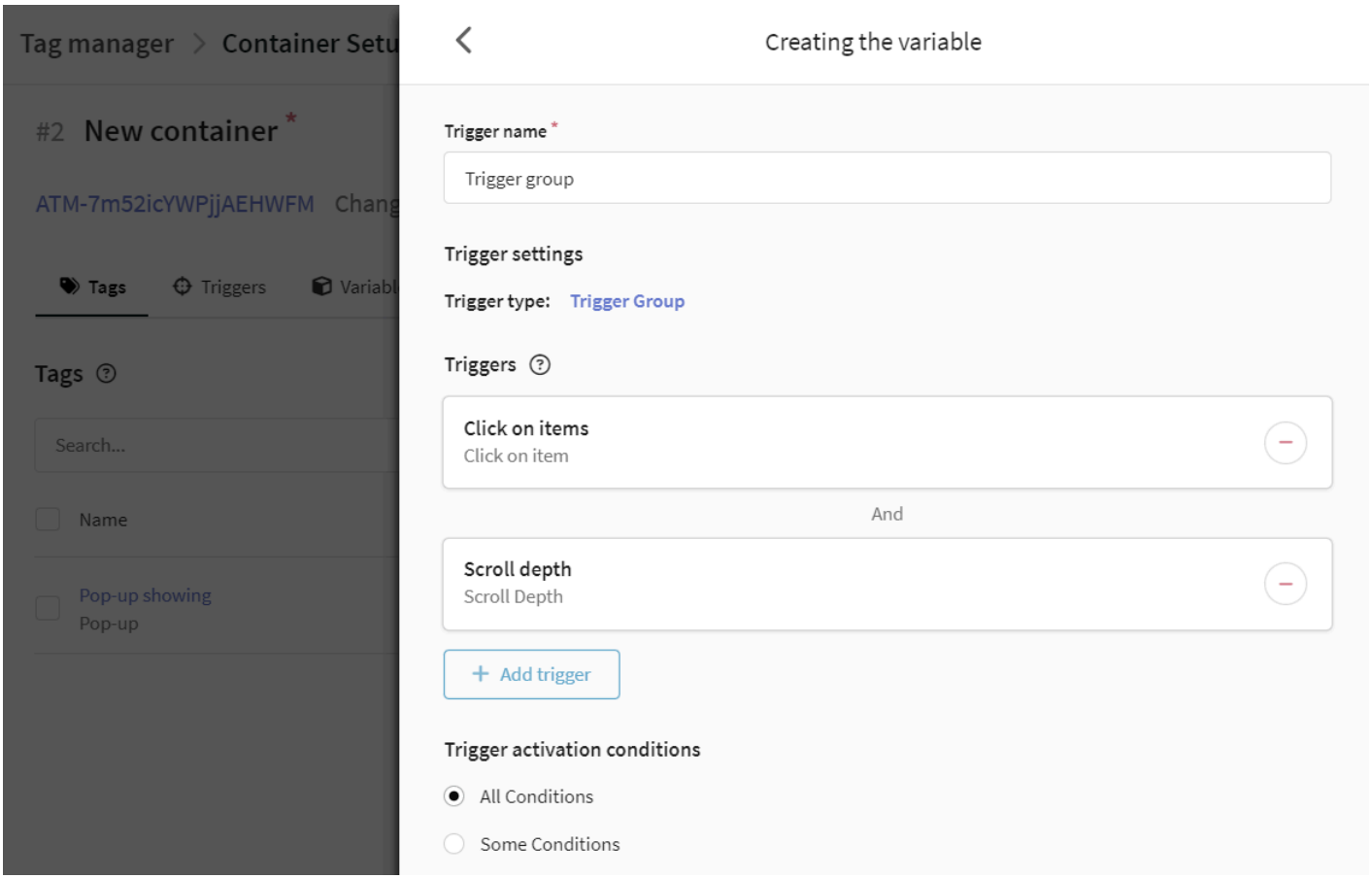
Other



Trigger group

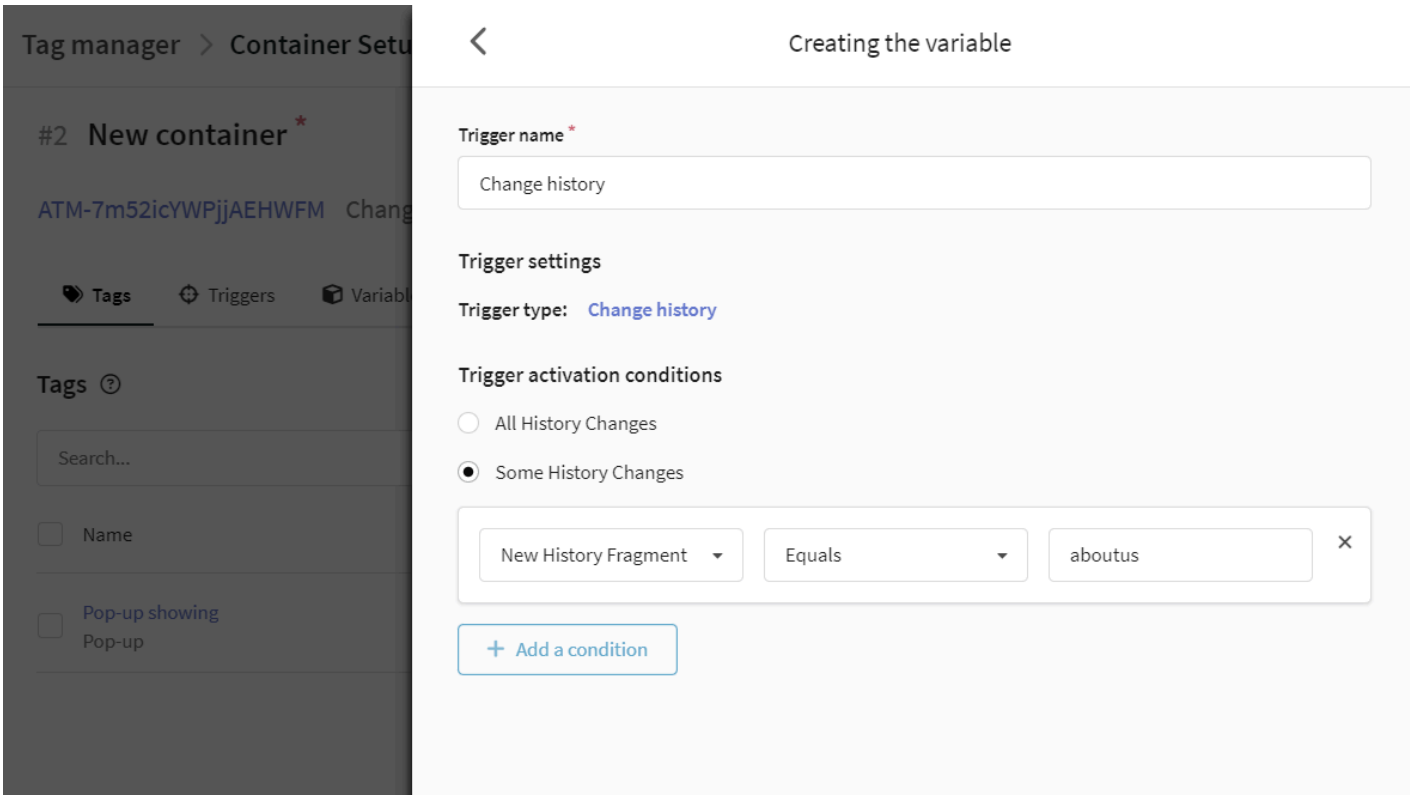
The trigger fires after all selected tags have been activated at least once.

Use the Add Trigger button to select the required triggers. The number of triggers in a group is not limited.



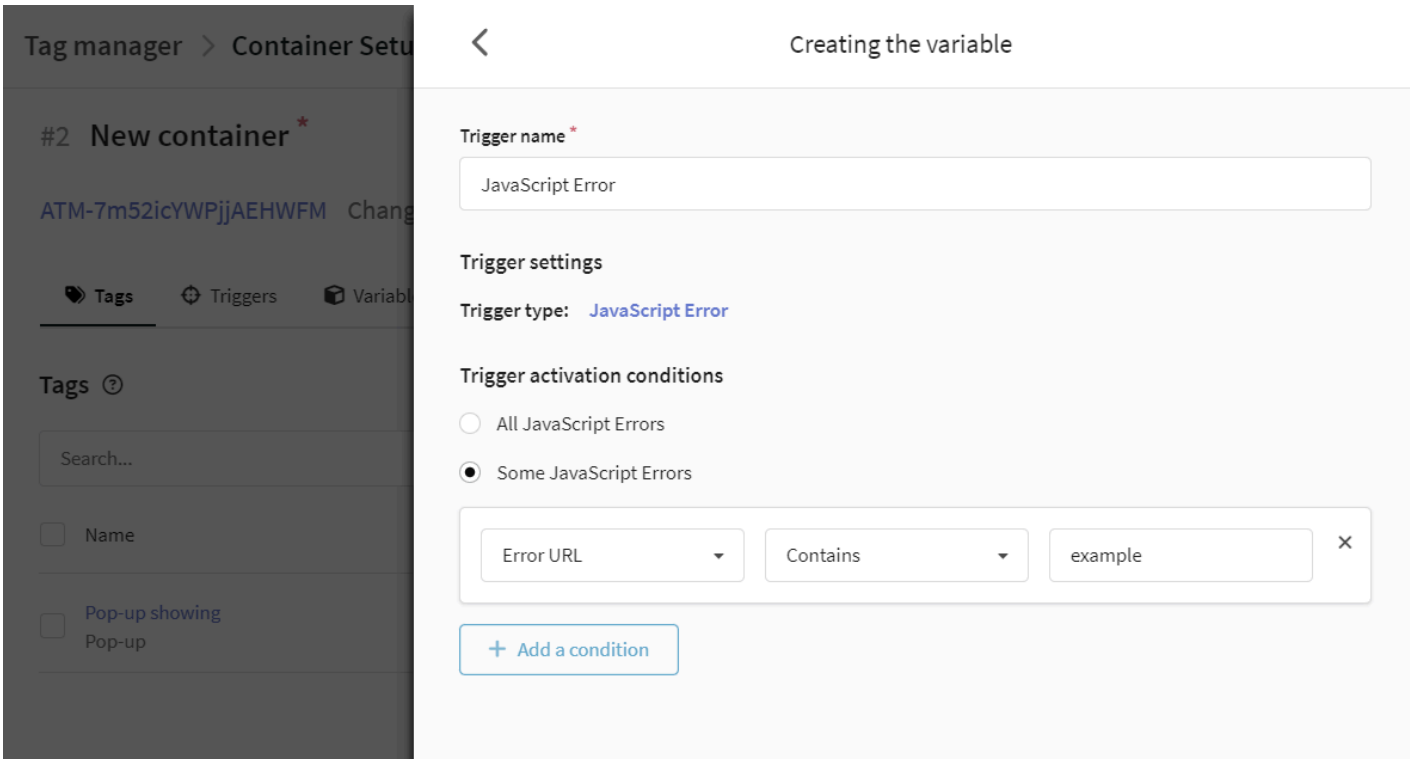
Change history

The trigger fires when there is a change in the URL fragment (hash) or when the pushState HTML5 API is used on the site.



JavaScript error

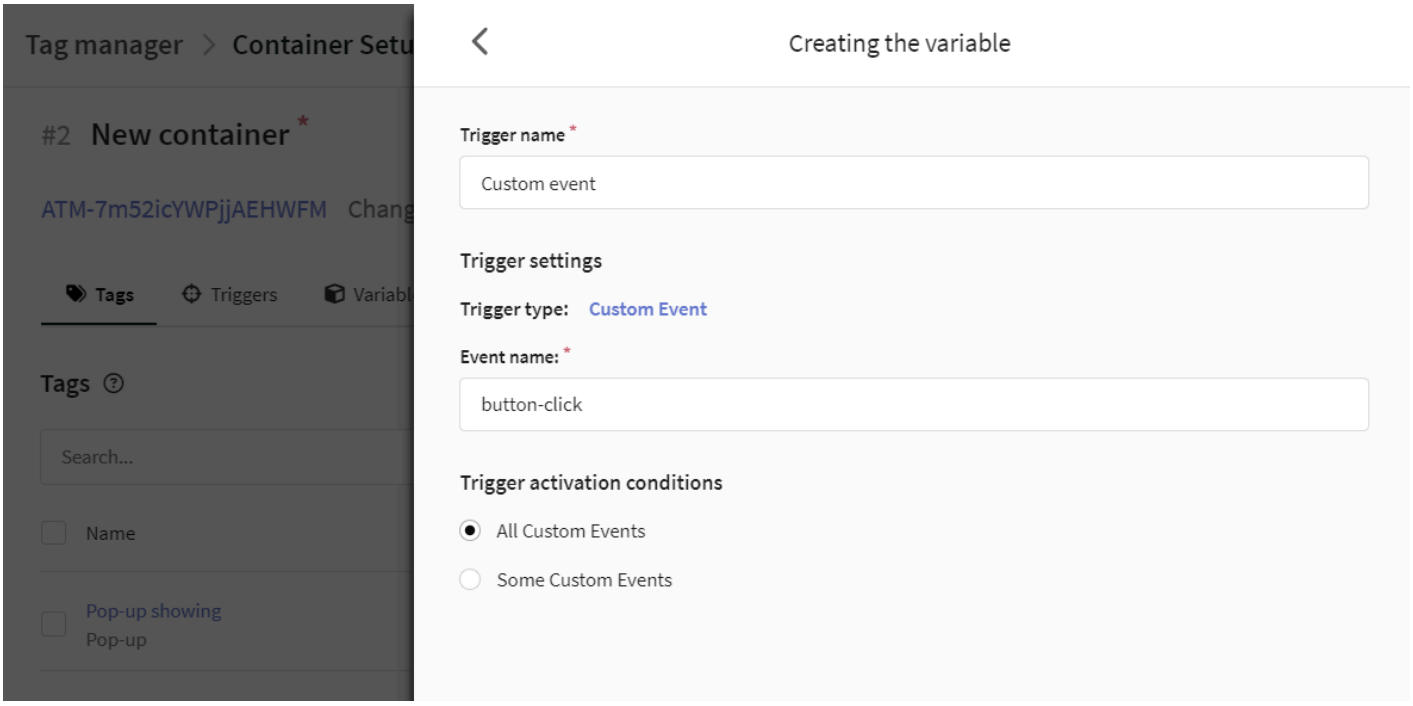
The trigger fires when a JavaScript error occurs on the site that was not caught by the window.onError event handler.



Custom event

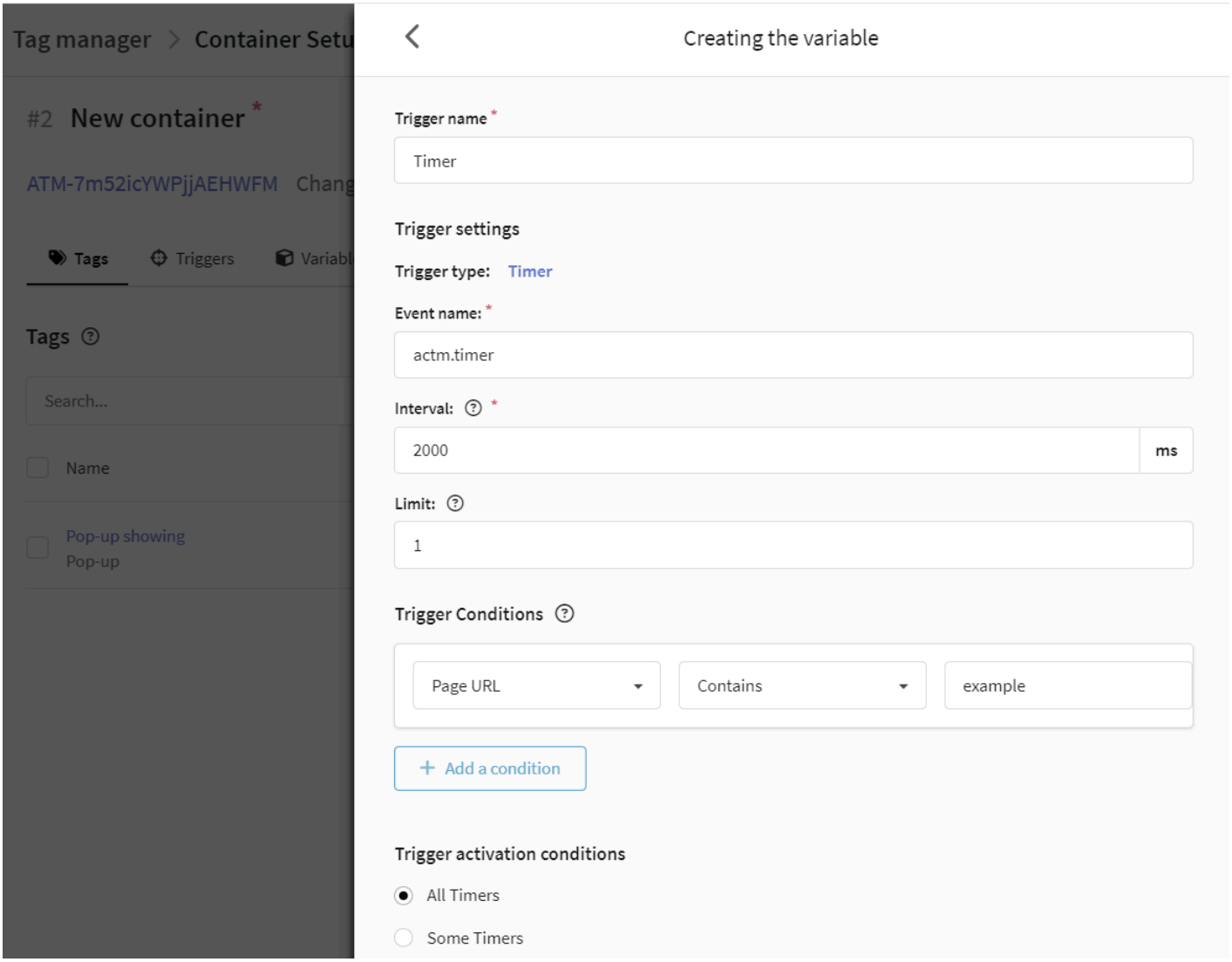
A trigger allows you to track interactions with a site or application for which standard methods are not suitable. For example, if you want to track the submission of a form for which you have overridden the behavior of the "submit" event in the browser.

To configure a trigger, you need to specify the name of the event, for example, `button-click`.



Timer

The trigger fires after a specified time interval.



Trigger settings include:

- **Event name by default — `actm.timer`. In most cases, you do not need to change the name. You can specify a different name to create a trigger based on a custom event.**
- **Interval — the time between event firings (in milliseconds). For example, if you set it to 2000, the event will trigger every 2 seconds. The first firing will also occur after the specified time.**
- **Limit — the maximum number of firings. If you leave this field empty, the trigger will fire until the user leaves the page.**

TIP

If you set the interval to 2000 ms and the limit to 1, the first and last firing will occur after 2 seconds.

Variables types

Description

A variable is an object that is given a name and can take on different values during code execution on the site (pair of “key:value”). The variable name is constant, but the value changes depending on what actions occur on the site. The variable can store data about the contents of a page. For example, the “Page URL” variable returns the URL of the current page.

Variables are used in both triggers and tags:

- **In triggers variables are used to set up filters that determine when certain tags will be triggered. In this case, variables serve for targeting — by setting conditions for activating a tag, they limit the audience to whom certain content on the site needs to be shown, for example, a pop-up.**
- **In tags variables are used to obtain dynamic values.**

To manage container variables, go to the “Variables” tab. Built-in and user’s variables are available in Tag manager. For most tag and trigger configurations, built-in variables are sufficient. However, you can also create your own variables to suit your needs.

> CAUTION
>
> **Please note that when setting up a trigger, it is not recommended to use the same variable several times.**

Using variables in tags

Built-in and user’s variables can be used in HTML and JavaScript tag code:

HTML

```
<div class="banner">
  <h3>{{Promotion name}}</h3>
  <p>Special offer for {{City}}</p>
</div>
```

JS

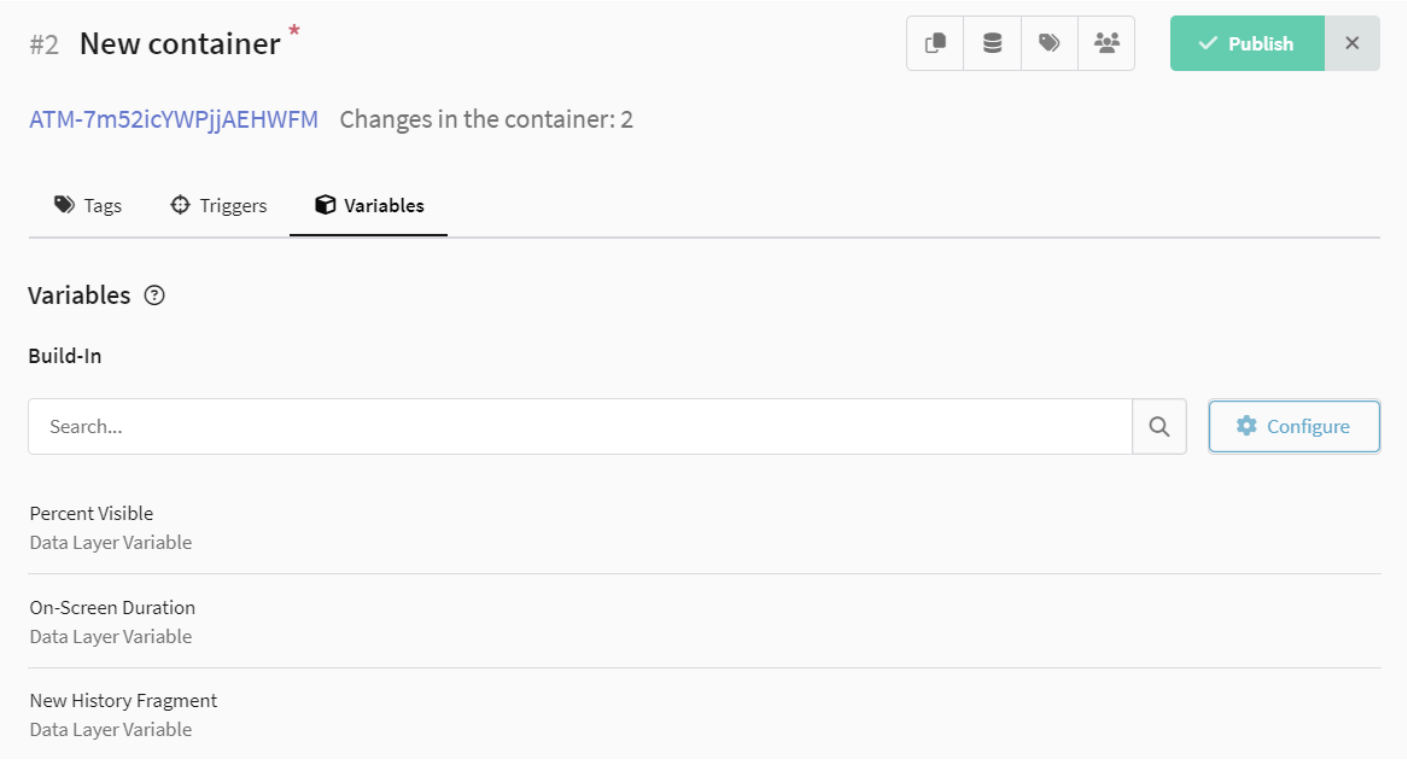
```
<script>
console.log('Current user:', {{User ID}});
analytics.track('page_view', { url: {{Page URL}} });
</script>
```

When you publish a container, the system checks that all used variables exist. If unknown variables are found (for example, a variable was deleted or renamed), the publication is canceled and an error message is shown.

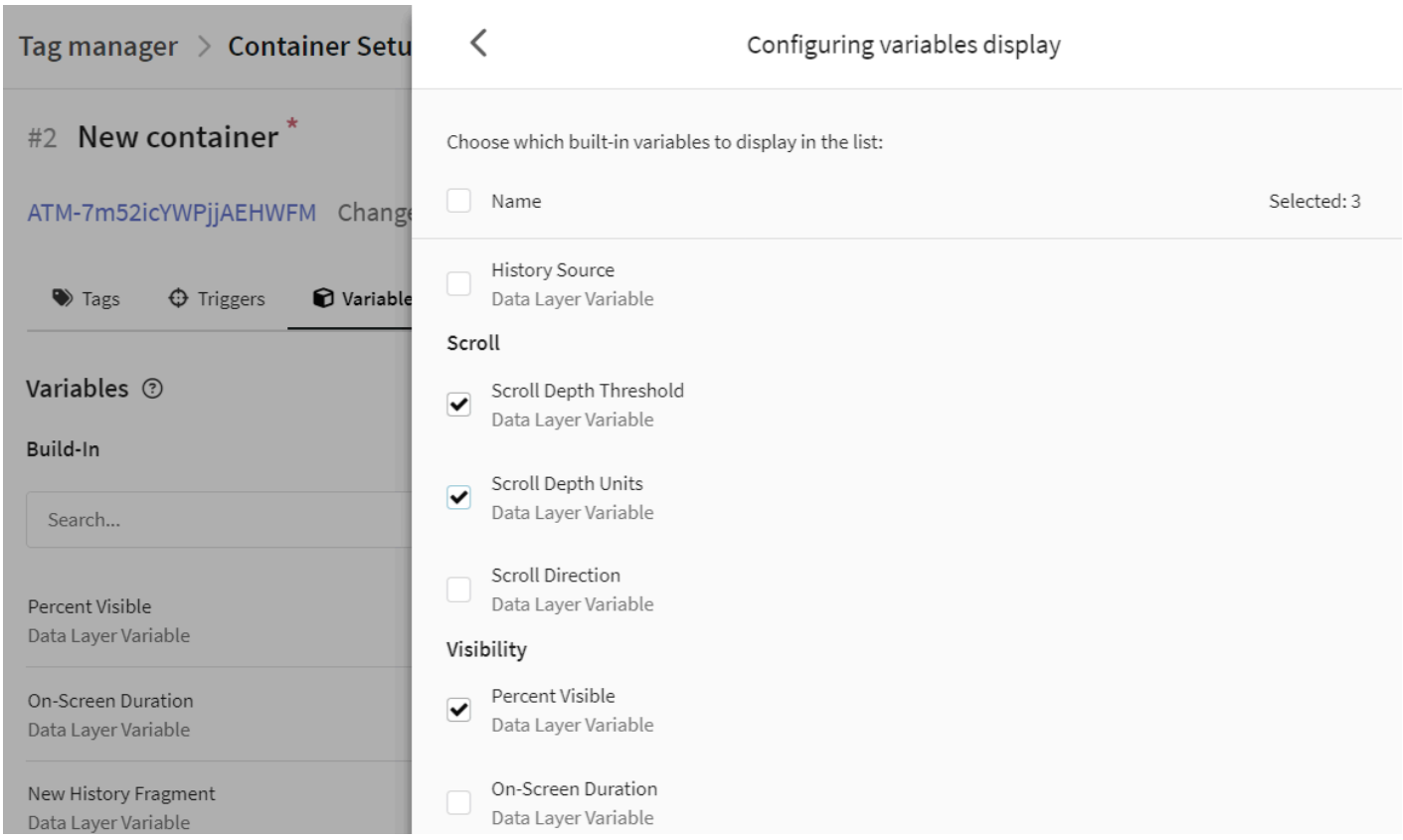
> CAUTION
>
> **If you change a variable name, you must update it in all tags where it is used.**

Built-in variables

Built-in variables are ready-made variables that are automatically added to a new container. Such variables cannot be changed.



By default, all built-in variables in a container are active. If you do not want to display a variable in the list, click the Configure button and in the window that opens, uncheck the box next to its name. Then click the Select button at the bottom of the page.



Types of built-in variables

Pages

- **Page URL. The variable returns the full URL of the current page, but without the hash. For example, if the user loaded the page `https://example.com/test?param=value#hash`, then the variable will return the value `https://example.com/test?param=value`.**
- **Page Hostname. The variable returns the domain name in the URL of the current page. For example, if the user loaded the page `https://example.com/test?param=value`, then the variable will return the value `example.com`.**
- **Page Path. The variable returns the path to the page in the current URL without taking into account GET parameters. For example, if the user loaded the page `https://example.com/test?param=value`, then the variable will return the value `/test`.**

- **Referrer. The variable returns the full URL of the page from which the user navigated to the current page.**

Utilities

- **Container ID. The variable returns the container number in the tag manager. For example, ATM-4ynTVvH4Ga0O.**

Errors

- **Error Message. The variable returns the error message sent by the JavaScript error trigger.**
- **Error URL. The variable returns the URL of the script in which the error was detected.**
- **Error Line. The variable returns the line number of the file where the error occurred.**
- **Debug Mode. The variable returns the value “true” if debug mode is activated in the Tag Manager.**

Clicks

- **Click Element. The variable returns the HTML element that the user clicked on.**
- **Click Classes. The variable returns the value of the `class` attribute for the HTML element that was clicked by the user.**
- **Click ID. The variable returns the value of the `id` attribute for the HTML element that was clicked by the user.**
- **Click Target. The variable returns the value of the `target` attribute for the HTML element that was clicked by the user.**
- **Click URL. The variable returns the value of the `href` attribute for the HTML element that was clicked by the user.**
- **Click Text. The variable returns the `textContent` value of the HTML element that the user clicked on.**

Forms

- **Form Element. The variable returns the JavaScript object for the submitted form. For example, class, ID attributes, data about parent and child elements.**
- **Form Classes. The value of the `class` attribute for the submitted form.**
- **Form ID. The variable returns the value of the `id` attribute for the submitted form.**
- **Form Target. The variable returns the value of the `target` attribute for the submitted form.**
- **Form URL. The variable returns the value of the `action` attribute for the submit.**
- **Form Text. The variable returns a text from the form, which was sent, and from form’s elements.**

History

- **New History Fragment. The variable returns a new hash value (#) of the website’s URL after the user event “Change in history” is committed.**
- **Old History Fragment. The variable returns the previous value of the URL hash before the user event occurred.**
- **History Source. The variable returns a string containing the event that led to the change of the history object.**

Scroll

- **Scroll Depth Threshold. The variable returns the numeric value of the scroll threshold at which the Scroll Depth trigger was fired (in percentage or pixels).**
- **Scroll Depth Units. This variable returns the unit of measure (pixels or percentages) for the threshold value that triggered the Scroll Depth trigger.**
- **Scroll Direction. The variable returns a parameter indicating the scroll direction (“vertical” or “horizontal”).**

Visibility

- **Percent Visible. The variable returns a numeric value between 0 and 100 that indicates what percentage of the selected item was visible when the Item Visibility trigger was fired.**
- **On-Screen Duration. The variable returns a numeric value that indicates how long the selected element was visible when the Element Visibility trigger was fired.**

Geo-variables

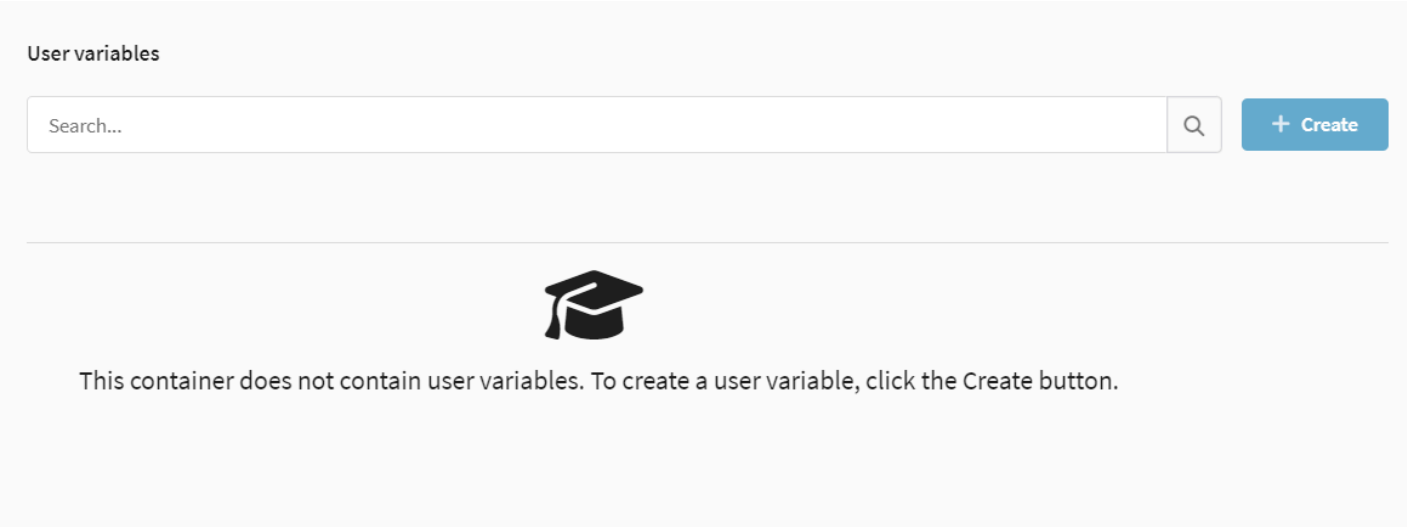
- **Country. The variable returns the country of the profile.**
- **City. The variable returns the profile city.**

Agent

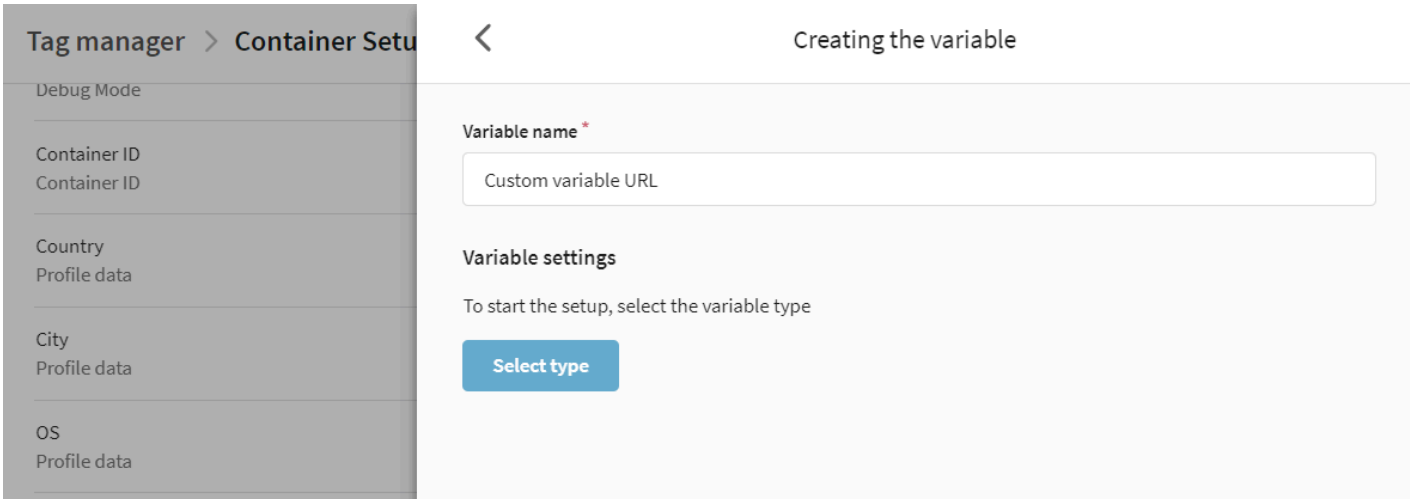
- **Device. The variable returns the profile device.**
- **Browser. The variable returns the profile browser.**

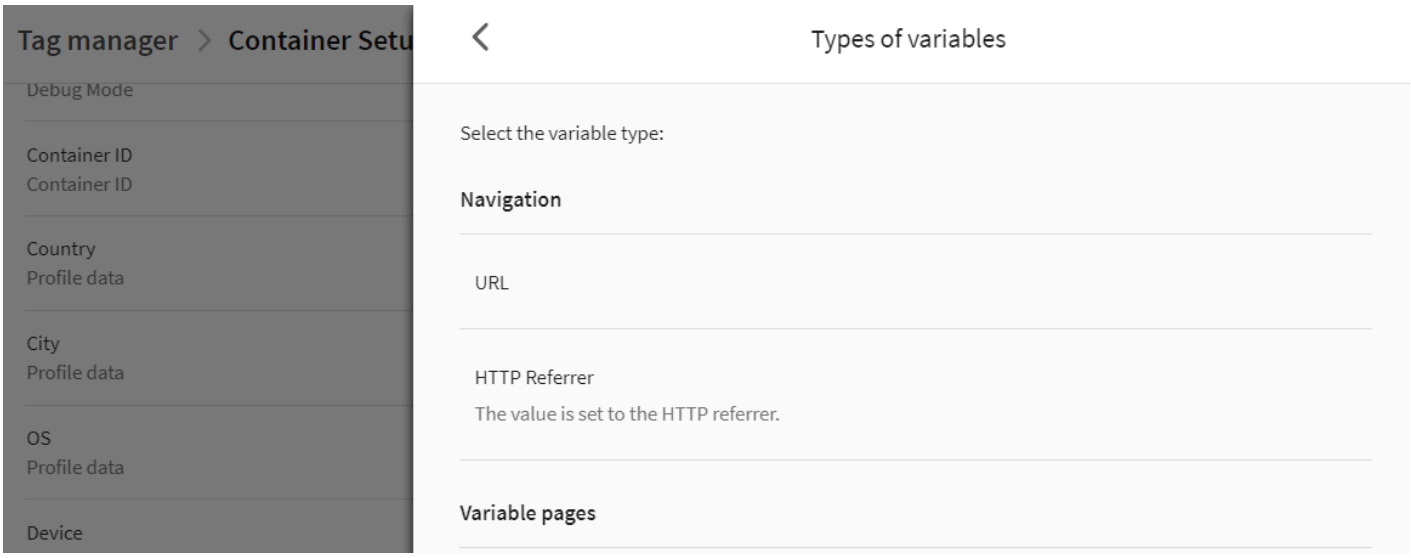
User variables

To create a new variable in the Container Setup window, go to the Variables tab. Scroll to the User variables block and click the + Create button:

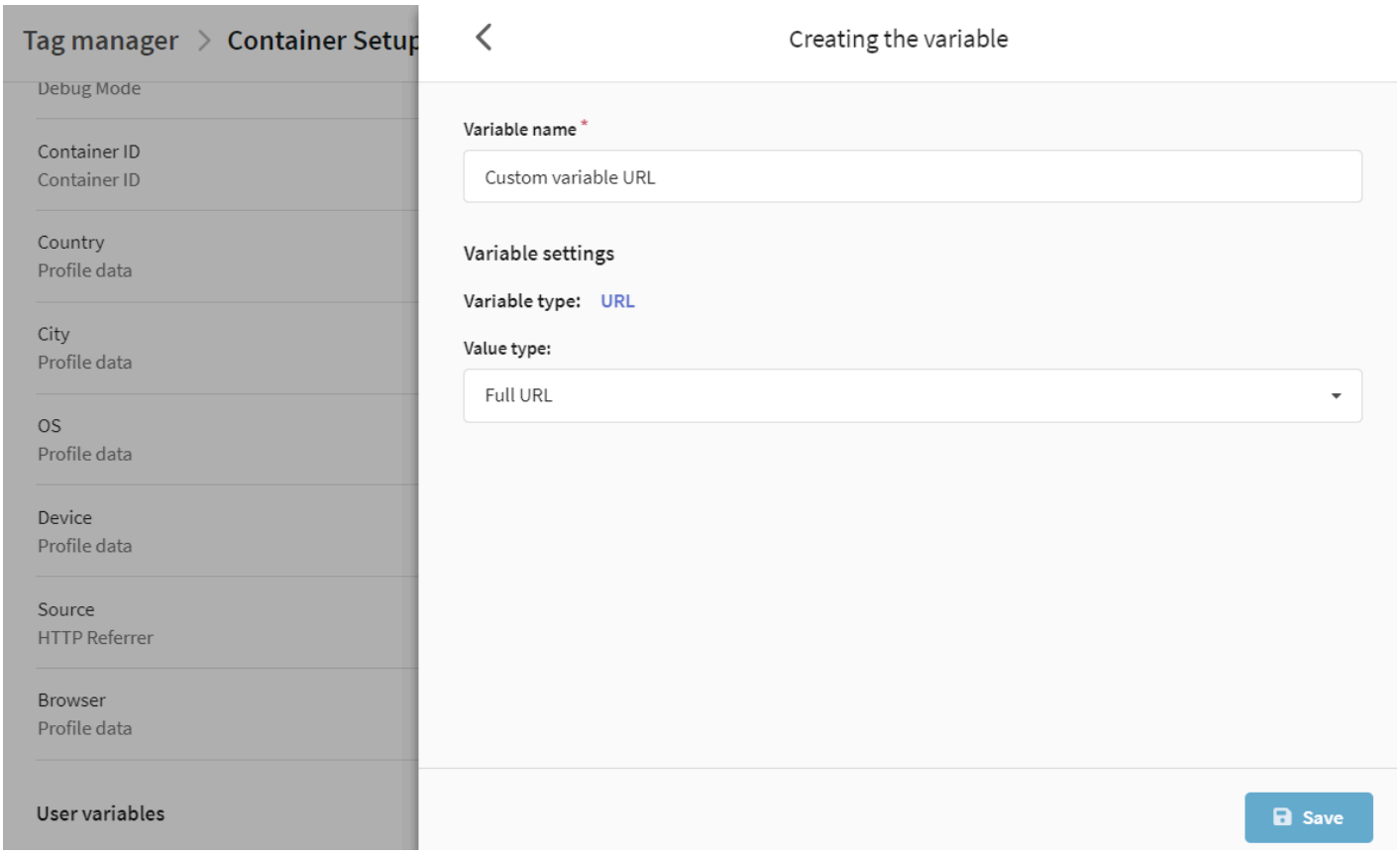


In the window that appears, specify a name for the variable and select its type:



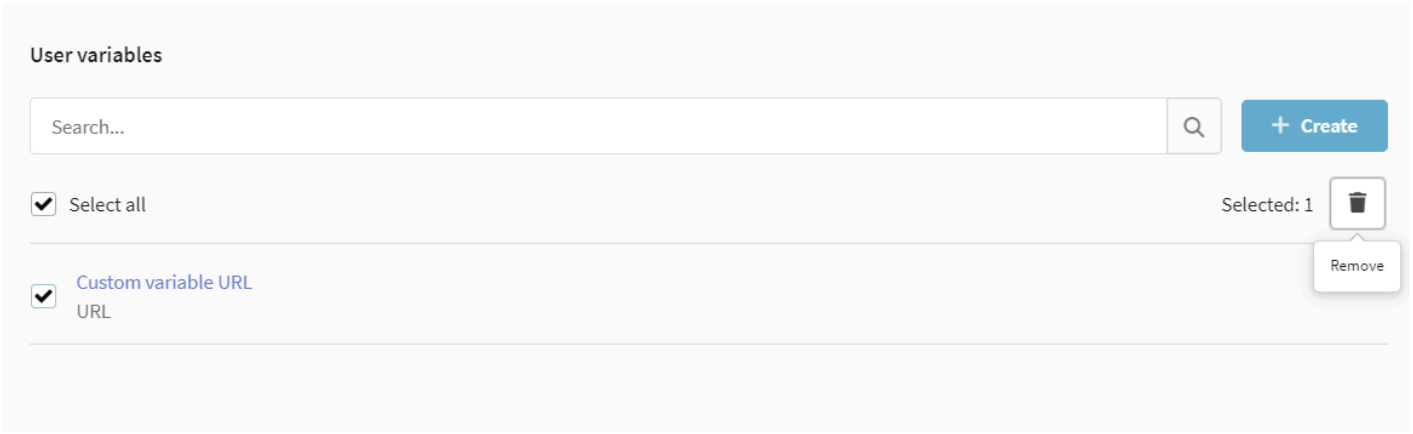


Set variable settings and save using the button at the bottom of the page.



After saving, the variable will appear in the list. To change the settings of a variable, click on its name. You can also delete unnecessary variables: select them and click

:

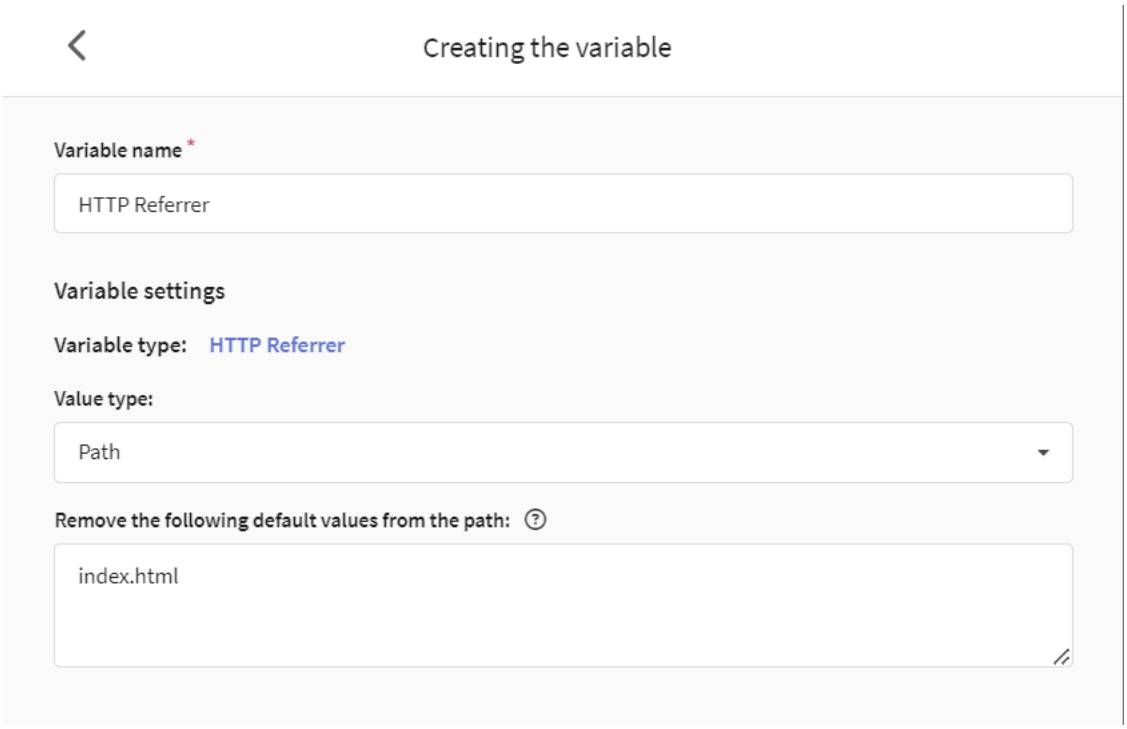


Types of user variables

Navigation

This type of variable allows you to access URL components:

- **URL — Analogue of the built-in variables Page URL, Page Hostname, Page Path. The input value for this type of variable is the URL of the page the user is on. The value is retrieved from** `document.location`
- **HTTP Referrer — Analogous to the built-in Referrer variable. The input value for this type of variable is the URL of the previous page. The value is retrieved from** `document.referrer`**.**



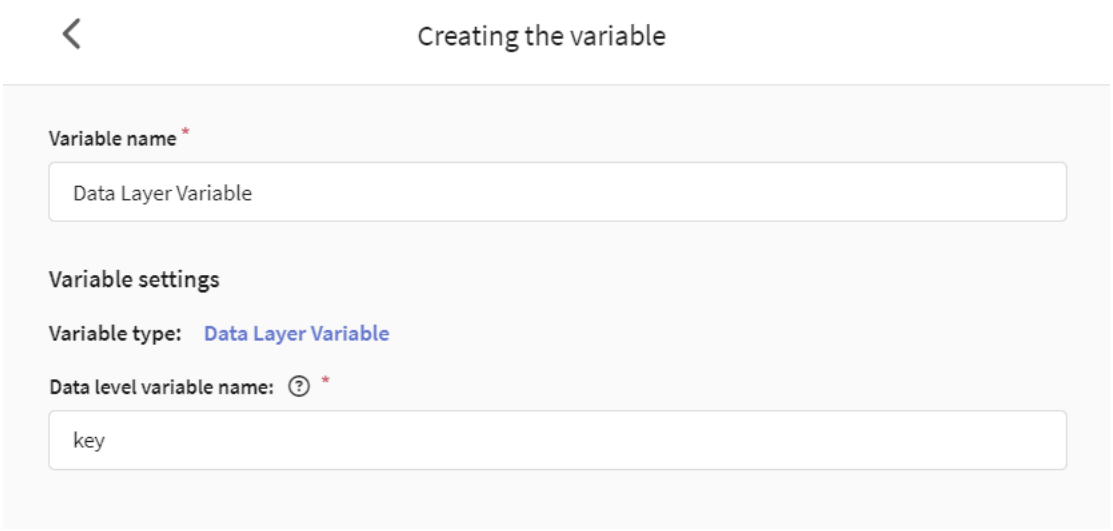
Value type (using URL example: `http://example.com:80/path/index.html?param1=value1&param2=value2#hash`):

Value	Example	Description
Full URL	**`http://example.com:80/path/index.html?param1=value1&param2=value2`**	**The variable returns the full URL without the hashing fragment.**
Hostname	**example.com**	**The variable returns the hostname of the URL without the port number.** If you enable the "remove www. from hostname" field, the variable will return the same value for www.example. com and example.com. </td>
Request	**param1=value1&param2=value2**	**The variable returns the query parameters (without the "?").** If you specify a query key, the variable will return only the value of that key. for example, for the key <code>param1</code> the variable will return the value <code>value1</code>. </td>
Path	**/path/index.html**	**The variable returns the path in the URL.** In the field "Remove the following default values from the path" you can specify the name of the index file - <code> index.html </code>. in this case, the pages <code> http://example.com/path/index.html </code> and <code>http://example.com/path/</code> will be considered as one page. You can also specify multiple default values - each value on a separate line. in this case, if the last segment of the path matches the default and is not a directory, it will be removed. </td>
Protocol	**http:**	**The variable returns the URL protocol.**
Port	**80**	**The variable returns the port number.**
File extension	**html**	**The variable returns the file extension.**
Fragment	**hash**	**The variable returns the hash fragment (without the "#").**

Page variables

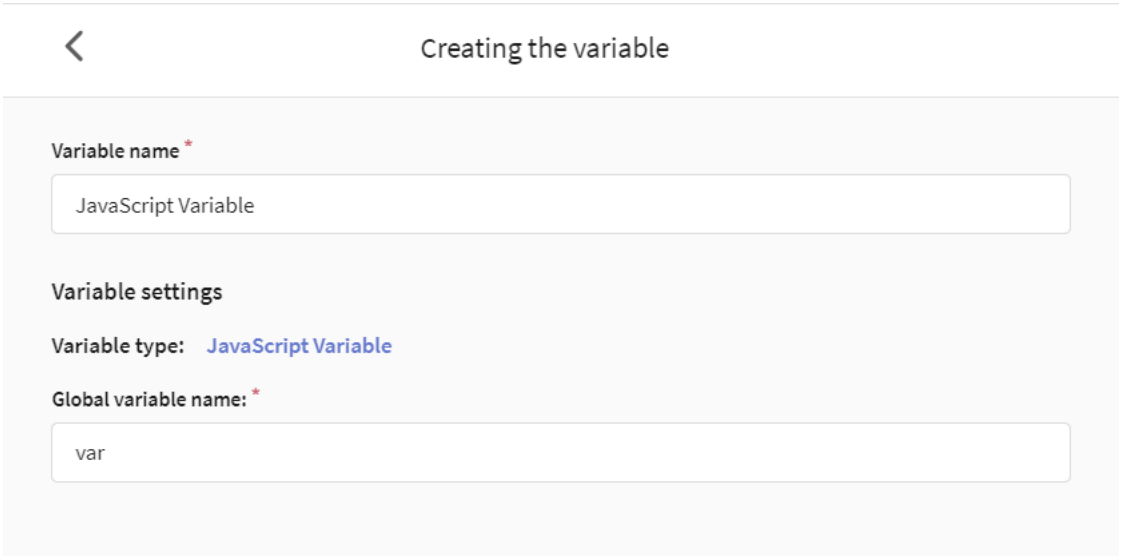
Data Layer Variable

The variable takes a value when information is passed to the data layer using the `acDataLayer.push({'var':'value'})` method. Information is transmitted in the `key:value` format, where `key` is the name of the variable, which is specified in the "Data level variable name" field, and `value` is the value being transmitted.



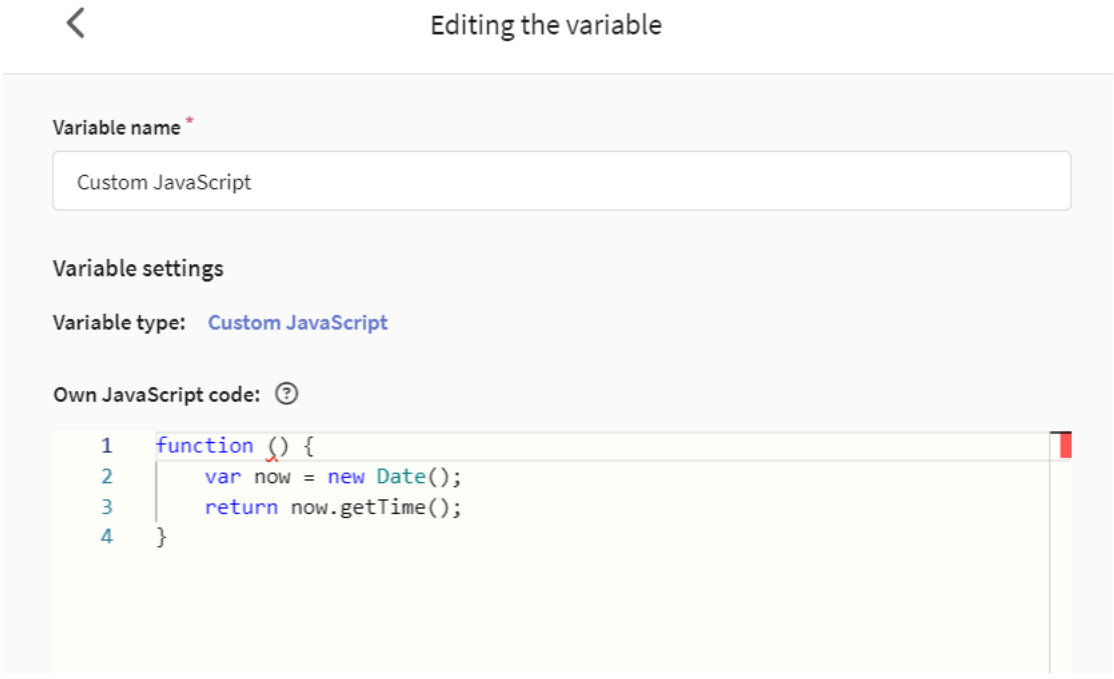
JavaScript Variable

The variable takes the value of a JavaScript global variable, the name of which is specified in the “Global Variable Name” field. If such a variable does not exist, the value undefined will be returned.



Custom JavaScript

The variable takes the value that is returned by the anonymous JavaScript function.



1st-Party Cookie

The variable takes the user's cookie value available for the current site. The value is retrieved from `document.cookie`. You must specify the name of the cookie whose value you want to transfer to the variable. For example, if the website has the cookie `"ak": "cookieValue"` installed, then in the "Cookie name" field you can specify `ak`.

The cookie value is encoded - enable the Enable URI cookie decoding field to decode.

Page elements

Element Visibility

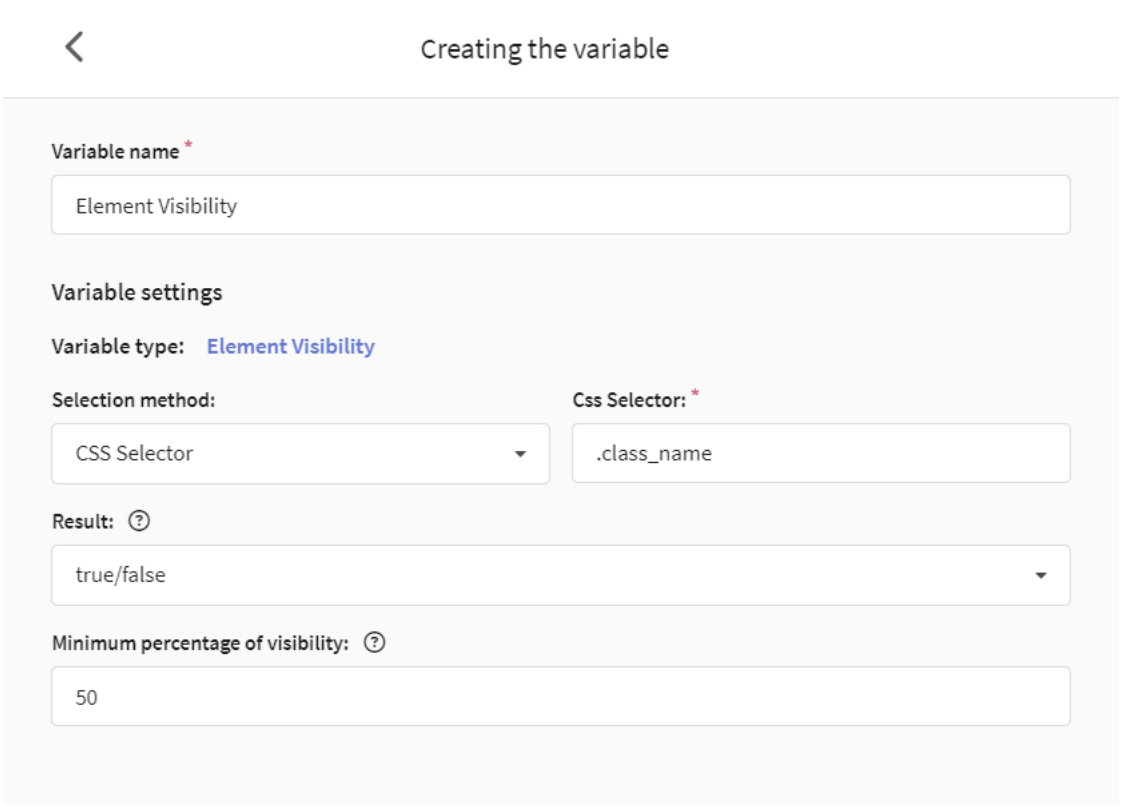
The variable allows you to determine which element was visible in the user's browser. First specify the selection method:

- **Identifier - tracks the visibility of a single element with a specific `id` attribute value. For example, if you specify `test`, the element with `id="test"` will be selected.**
- **CSS selector - tracks the visibility of multiple elements that match a given CSS selector pattern. For example, `.class_name` will select elements on the page with `class="class_name"`.**

Result defines the return value:

- **True/false - the variable will take the value true if the specified minimum element visibility percentage is reached. Otherwise - false.**
- **Visibility percentage - the variable will take a value equal to the visibility percentage.**

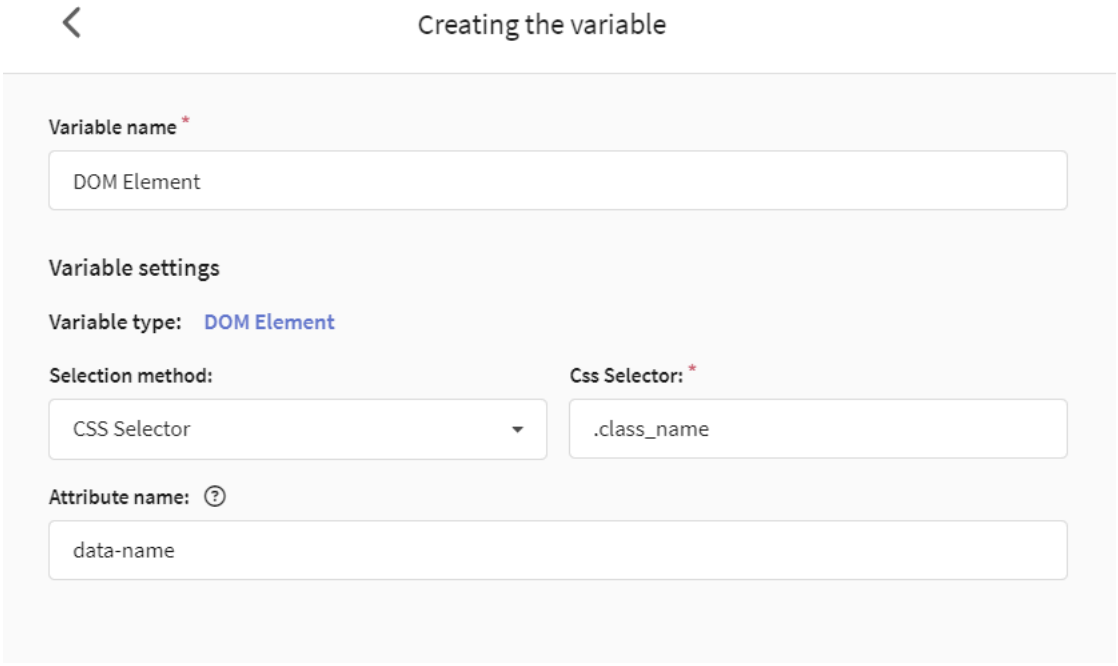
In the example below, the variable will return `true` when half of the element with class `class_name` is displayed on the user's screen:



DOM Element

The variable is used to retrieve the text content of any DOM element or to retrieve the value of any attribute of a DOM element. Determine which elements the variable should track. To do this, specify a selector method and specify an element ID or selector CSS template.

If you specify attribute name, the variable will be assigned the attribute value (for example, `data-name="value"`). If the attribute name is not specified, the variable will return the text of the DOM element.



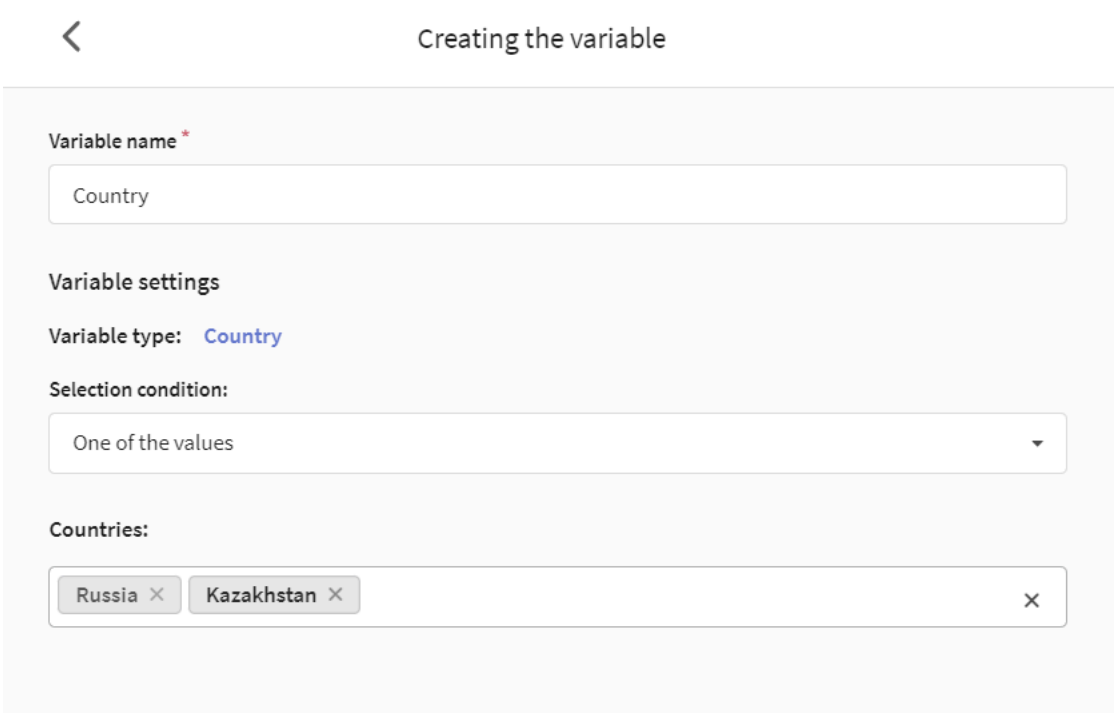
Other

Country

The variable takes the value `true` if the user's country matches the specified selection condition, and `false` if it does not.

Selection condition:

- **One of the values - the variable will return true if the user's country matches one of the selected ones.**
- **None of the values - the variable will return true if the user's country does not match any of the selected ones.**

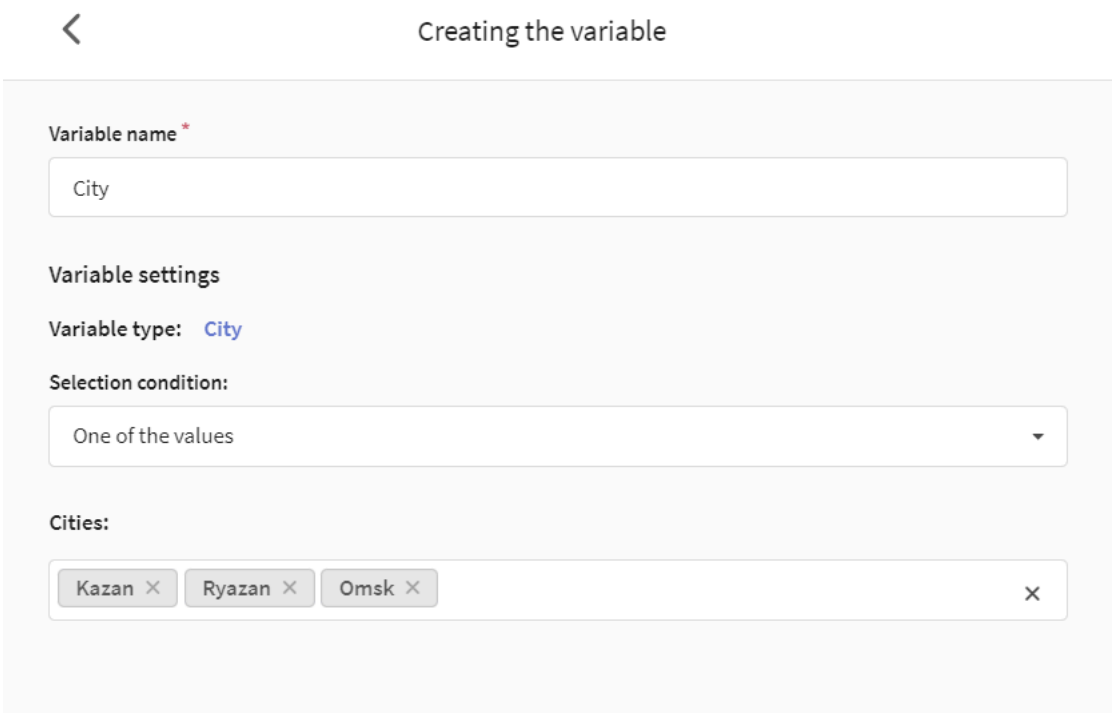


City

The variable takes the value `true` if the user's city matches the specified selection condition, and `false` if it does not.

Selection condition:

- **One of the values - the variable will return true if the user's city matches one of the selected ones.**
- **None of the values - the variable will return true if the user's city does not match any of the selected ones.**

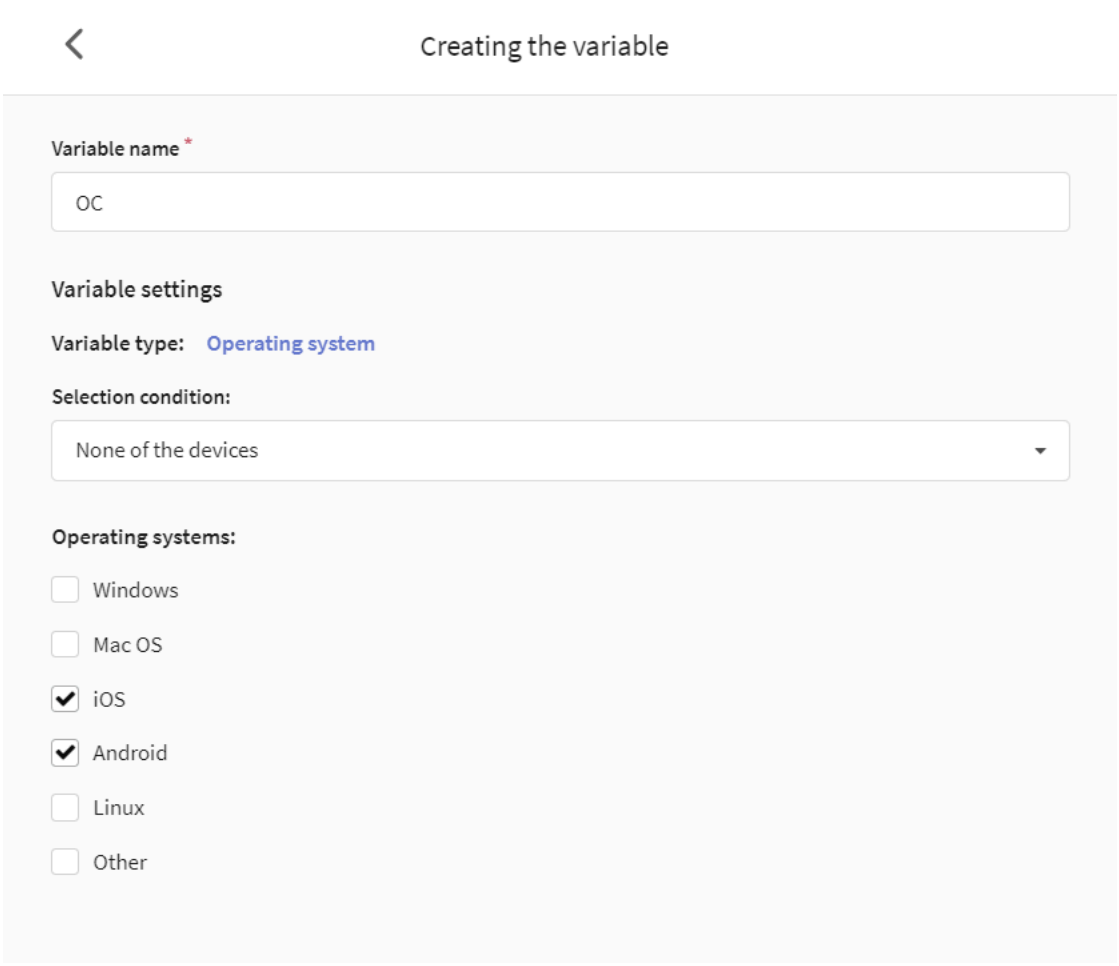


Operating system

The variable takes the value `true` if the user's OS matches the specified selection condition, and `false` if it does not.

Selection condition:

- **One of the devices - the variable will return true if the user is using one of the selected OS.**
- **None of the devices - the variable will return true if the user does not use any of the selected OS.**

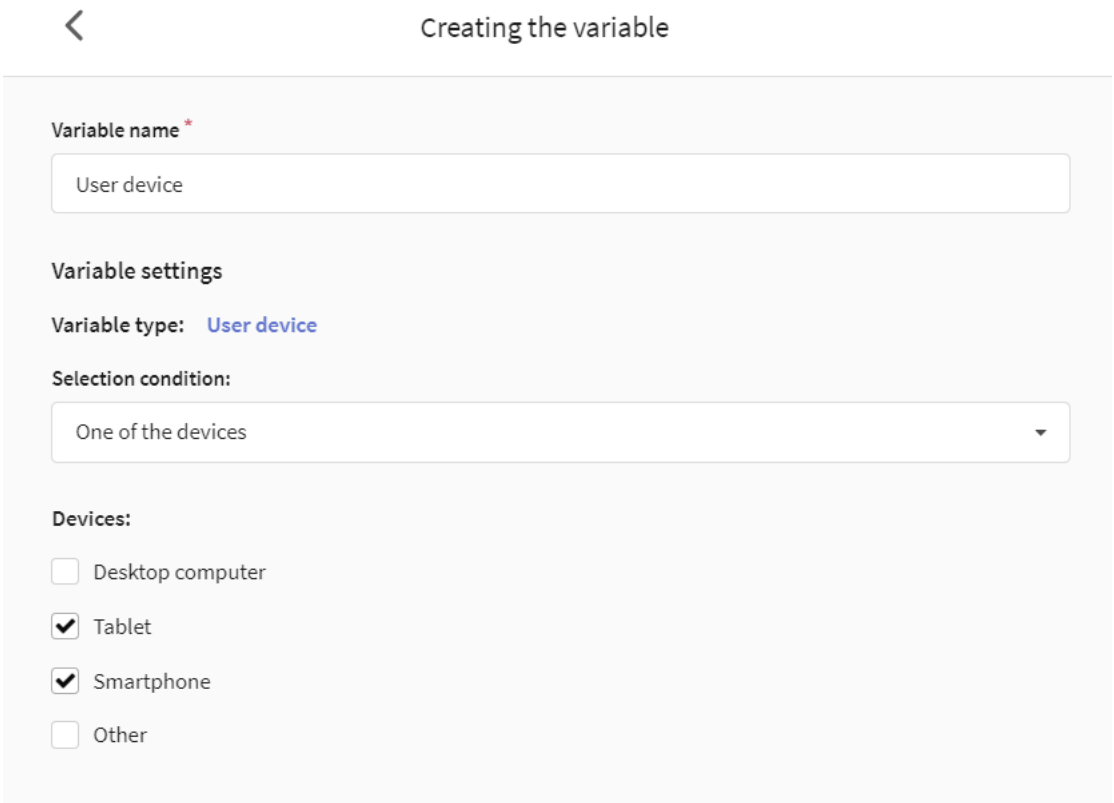


User device

The variable takes the value `true` if the user's device matches the specified selection condition, and `false` if it does not.

Selection condition:

- **One of the devices - the variable will return true if the user is using one of the selected devices.**
- **None of the devices - the variable will return true if the user does not use any of the selected devices.**

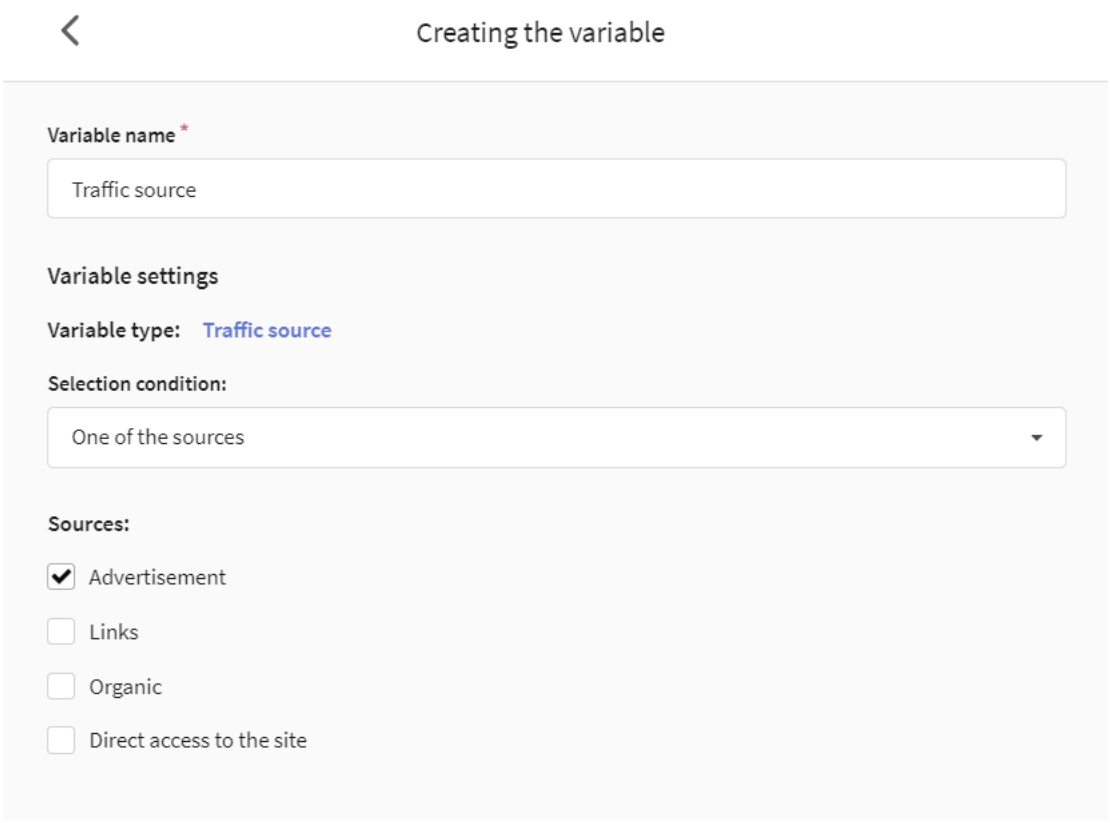


Traffic source

The variable takes the value `true` if the source of the user's transition to the site matches the specified selection condition, and `false` if it does not.

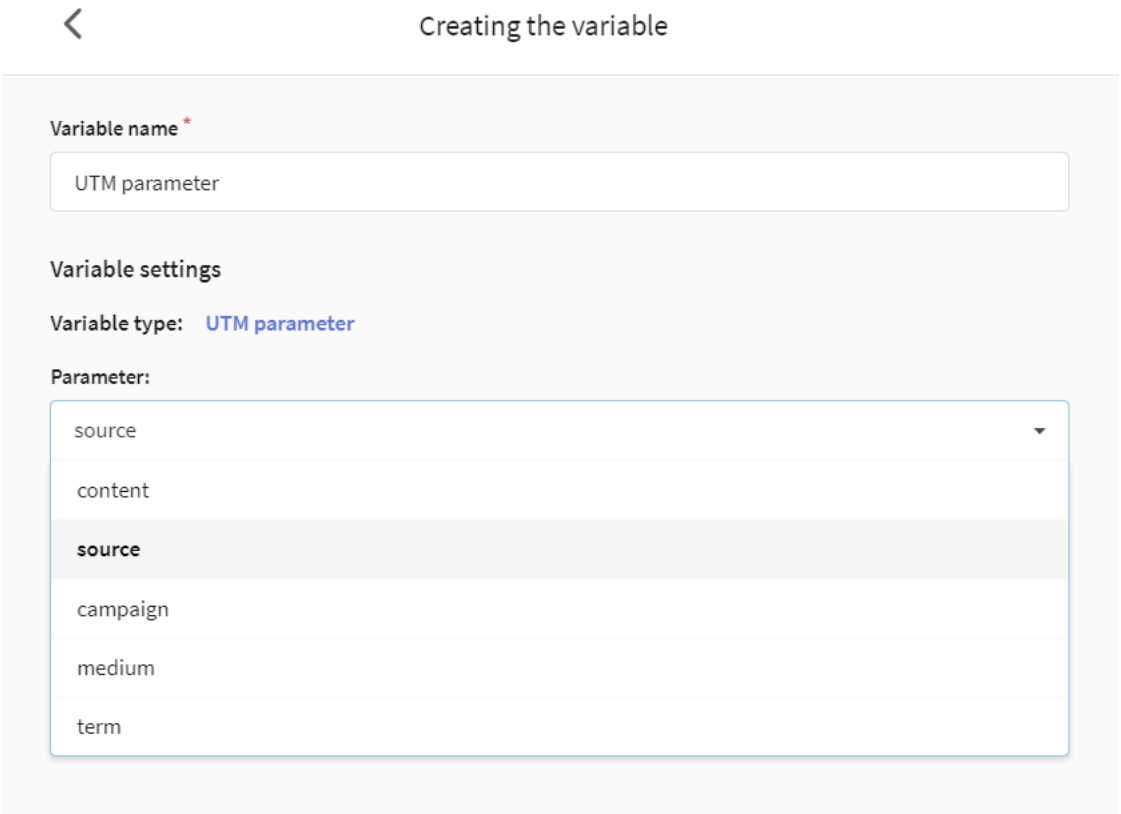
Selection condition:

- **One of the devices - the variable will return true if the traffic source matches one of the selected ones.**
- **None of the devices - the variable will return true if the traffic source does not match any of the selected ones.**



UTM parameter

The variable takes the value of the selected UTM parameter.



Linking a pixel and the Tag manager

Tag manager is a convenient way to place tracking pixels on your site.

When you use pixels as a standalone tool, you have to make changes to the site's source code every time you create a new pixel. This can be avoided by using Tag manager.

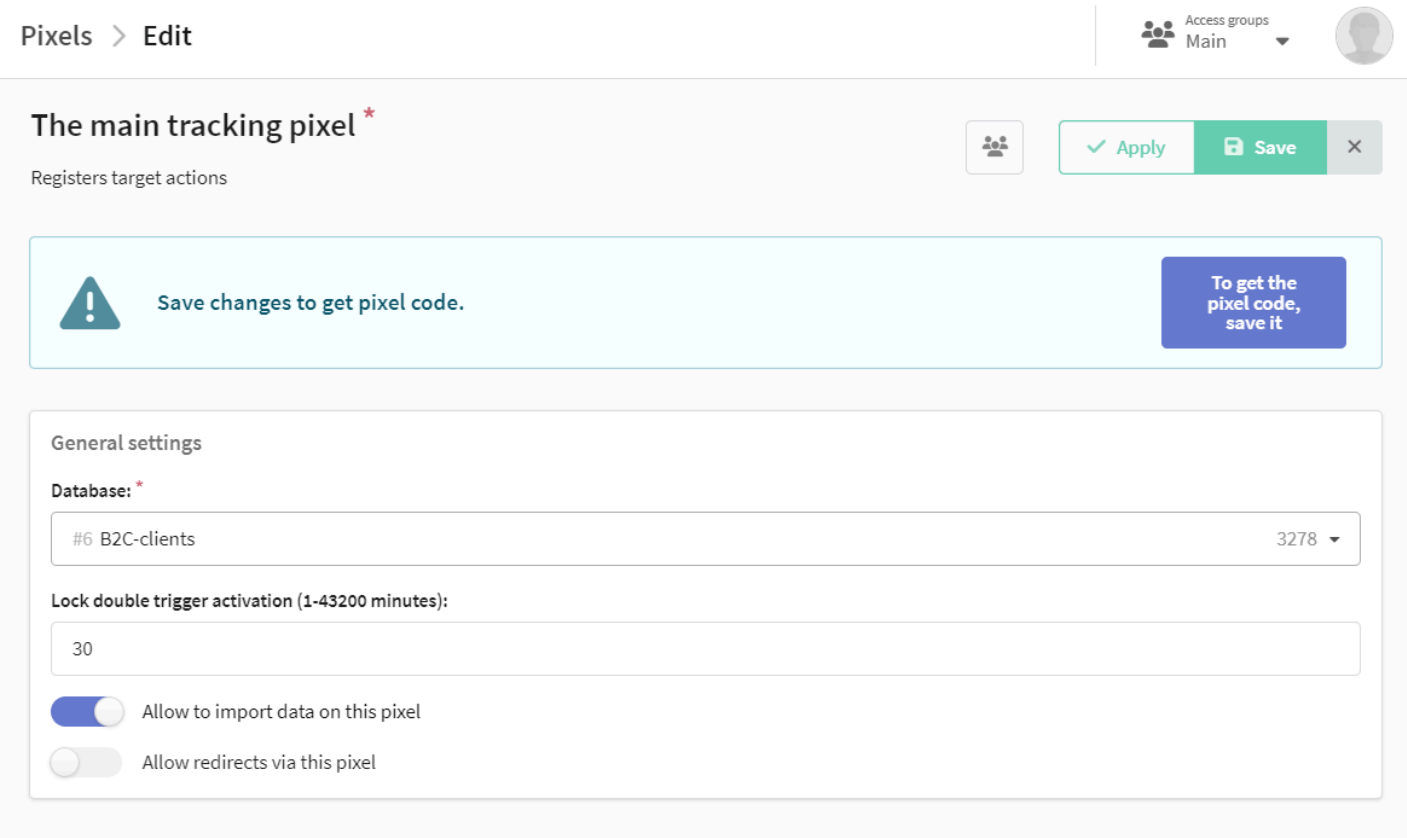
You create a container, add a tag with the HTML code of the pixel and triggers that determine the conditions for firing the pixel. Next, you place the container code on the site pages. If in the future you want to make some changes to the operation of the pixels (create a new pixel or suspend the operation of an existing one), it will be enough to edit the tag in the platform interface.

This guide will help you link Tracking Pixels and Tag manager in the Altcraft system.

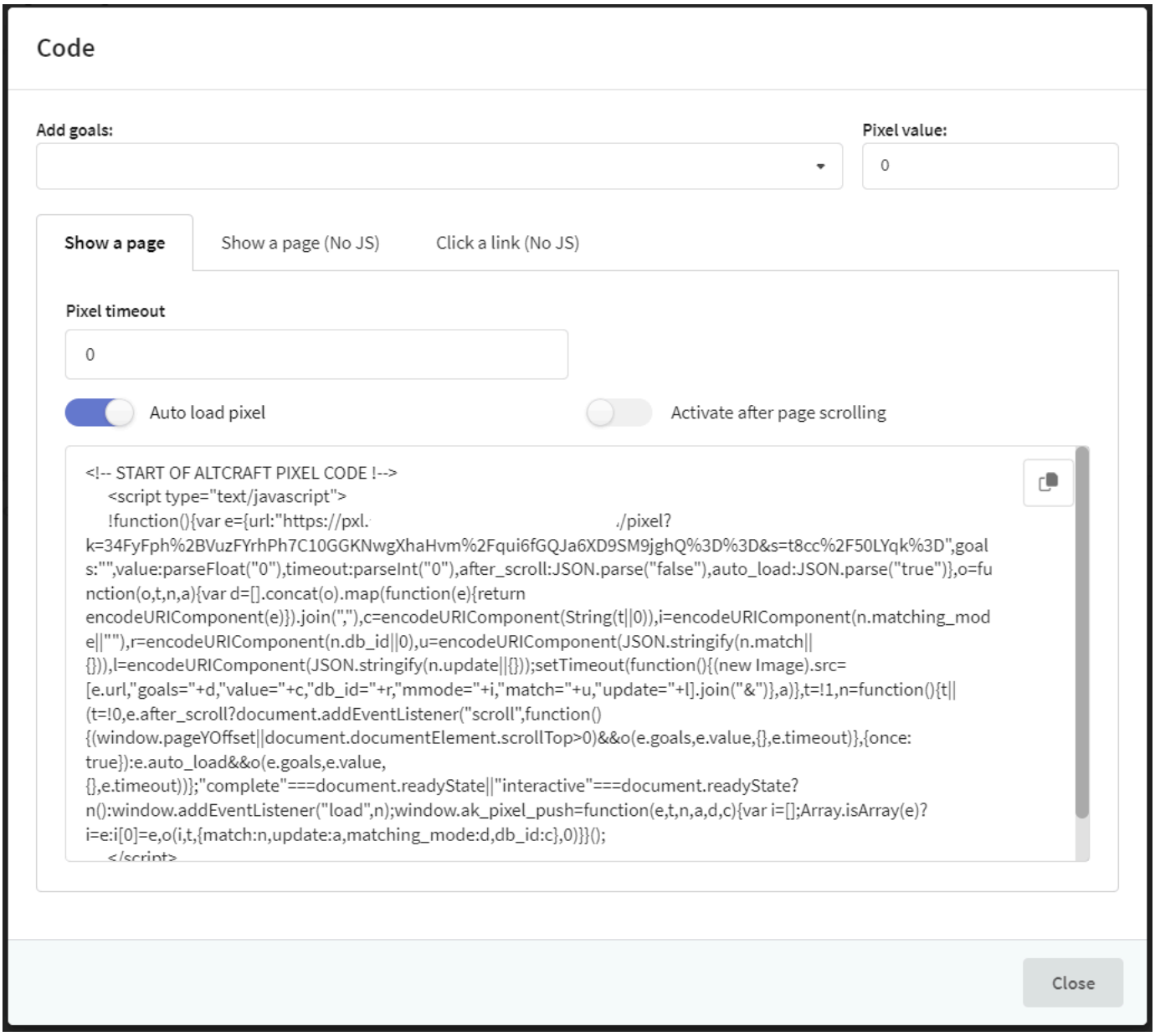
Step 1: Create a pixel

1. Go to the Pixels section and click Create.

2. Set up and save the pixel.



3. Click Get pixel code, add goals and copy the code.



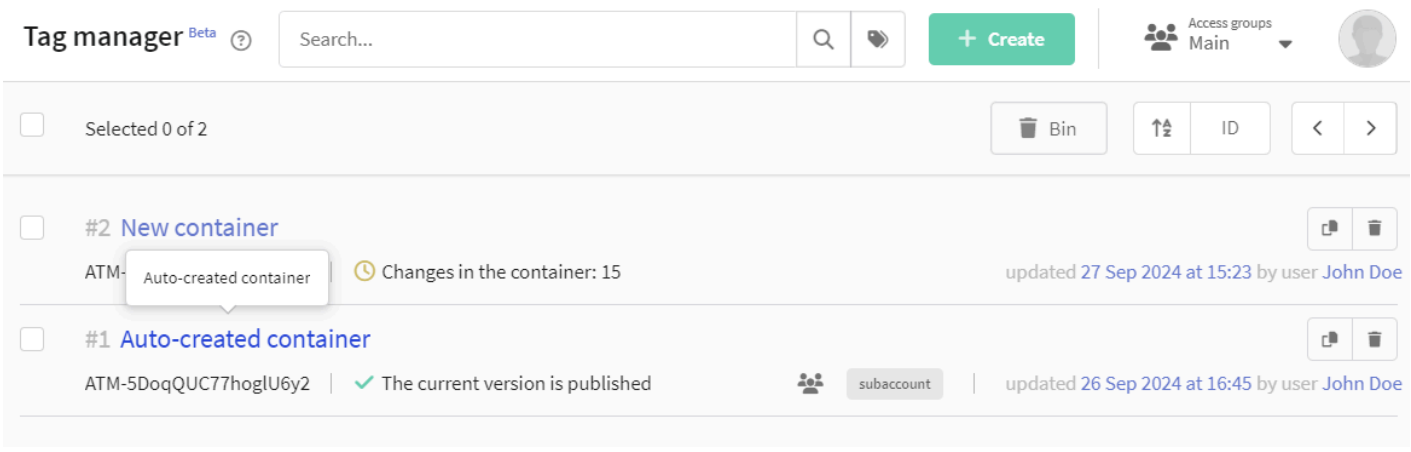
TIP

You can find more information about creating and configuring a pixel in this article.

Step 2: Create a tag in a container

1. Go to Weblayer —> Tag manager.

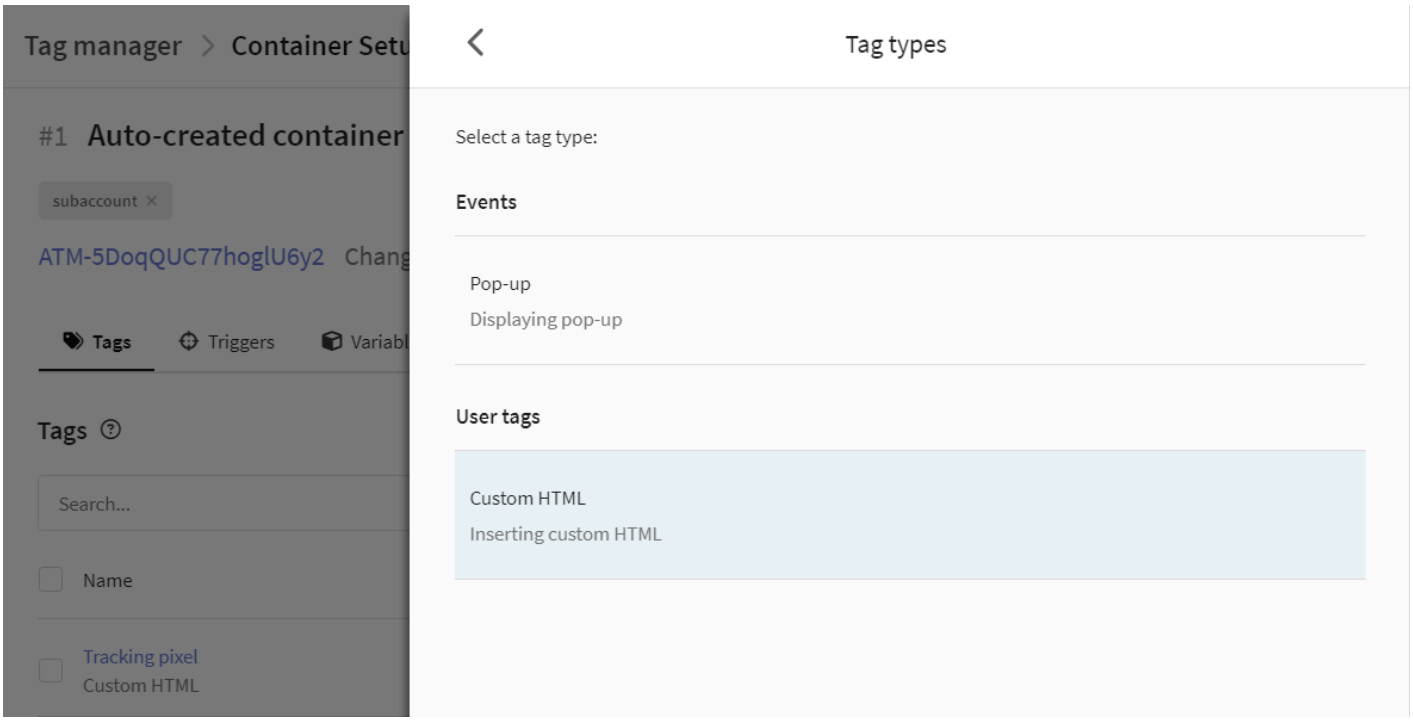
2. Open the container:



3. On the Tags tab, click + Create.

4. Enter a name of a tag.

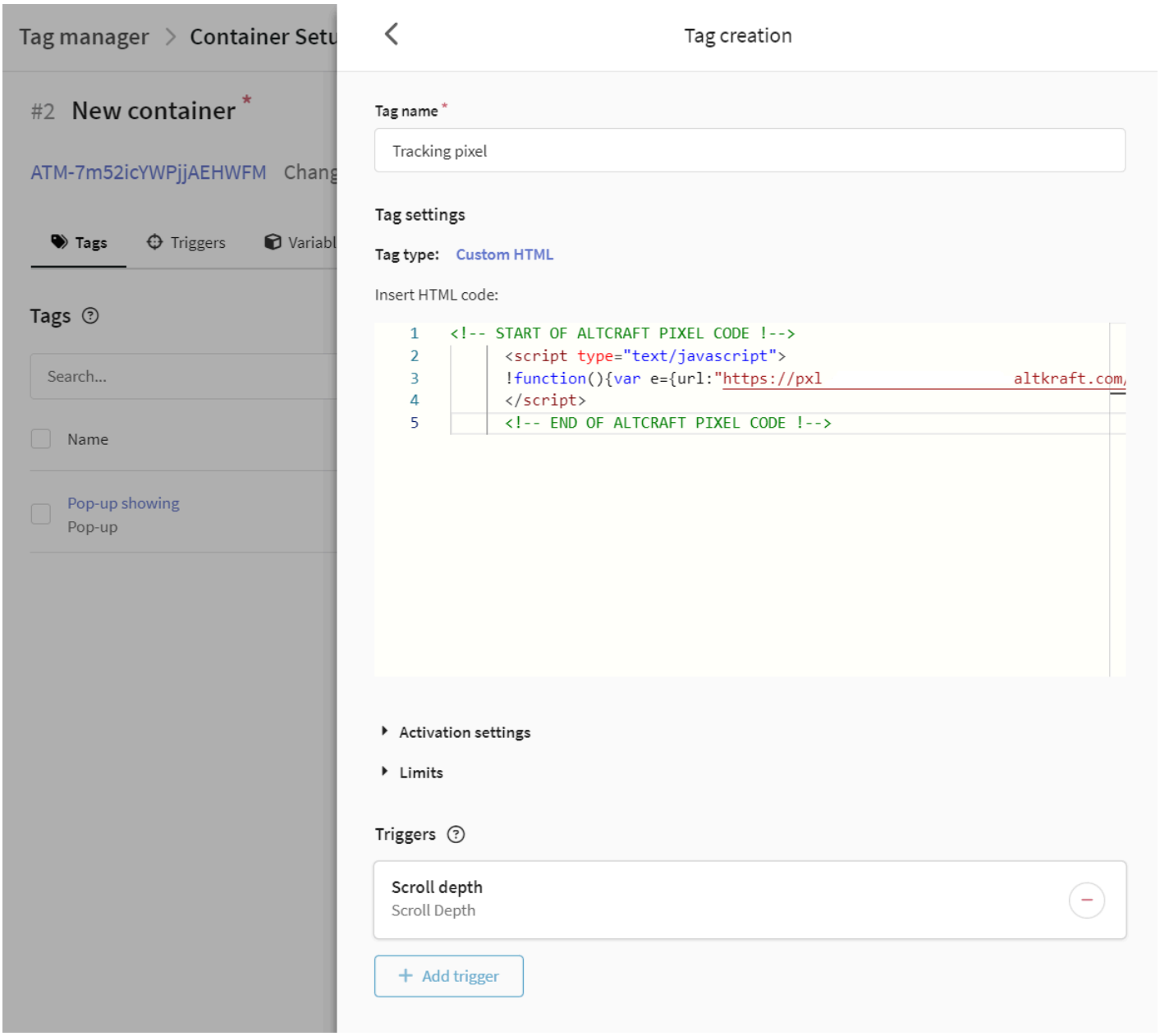
5. Select Custom HTML as the Tag Type.



6. Paste the pixel code into the editor window.

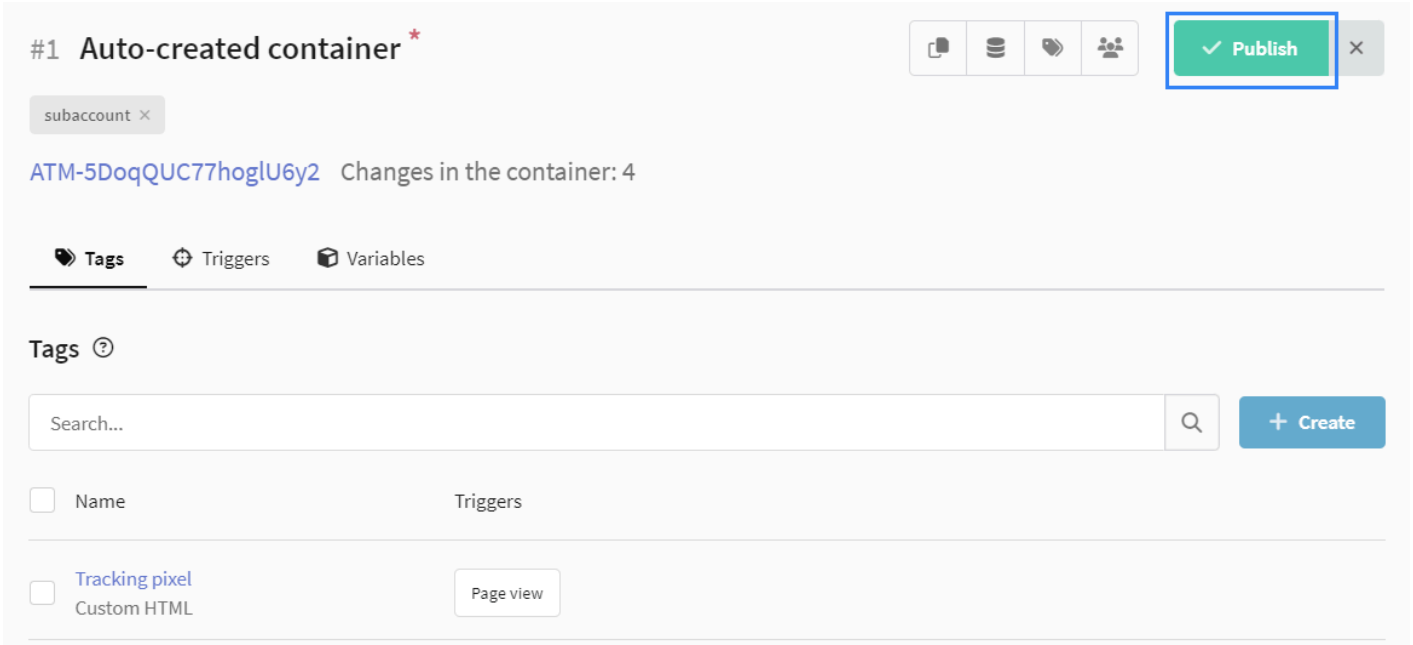
7. Set activation settings and limits for the tag.

8. Select and set up trigger. The tag will be activated when the conditions specified in the trigger occur. If you want to limit the audience of users for whom the trigger will work, additionally set up a filter.



9. Save the tag.

10. Click Publish —> Publishing and creating a new version. All changes to the container will be made to the site, and the pixel will begin tracking user actions.



TIP

More information about creating a container, setting up tags and triggers can be found in this article.

Settings

Settings menu

> **NOTE**
>
> **Altcraft MP account master-users have access to Settings section of the main menu containing the following pages:**

Section contents

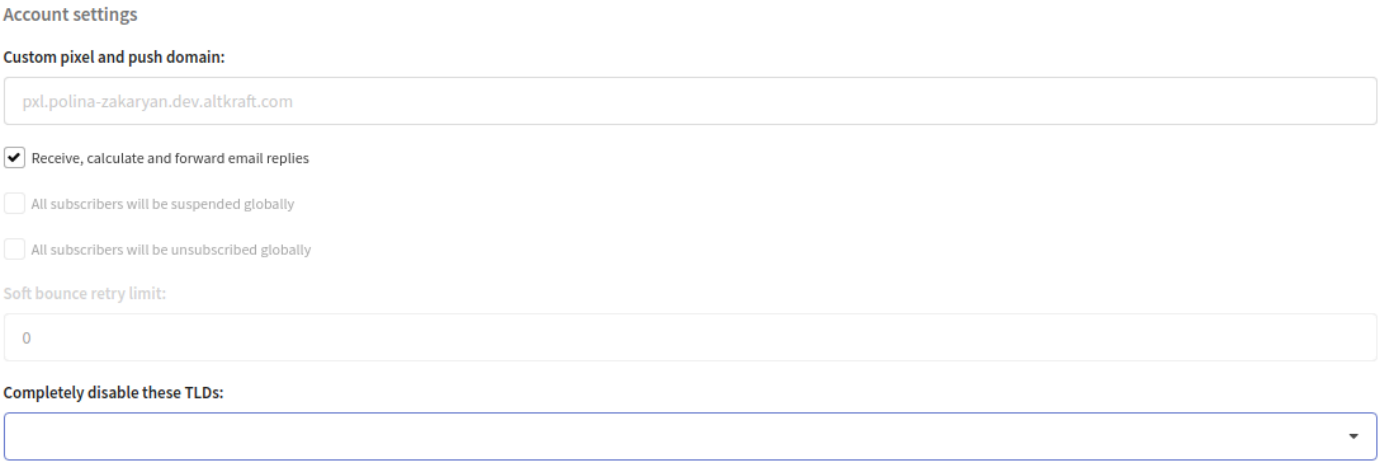
Account settings

Description

	The "Account settings" section includes configuration settings that are applied globally: to all processes and objects in the account. Settings affect different functionality.

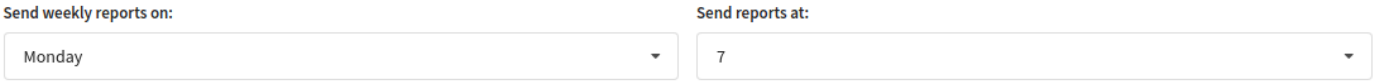
Administration

In this tab, you'll find the settings of email sending and subscriptions. You also can specify your custom pixel and push domain.



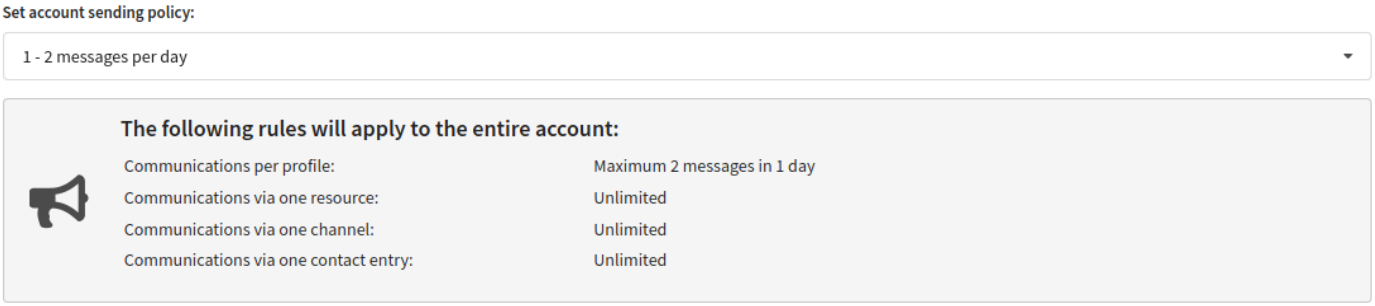
Report settings

In this tab, you can set the day of the week and hour for sending reports to email of platform users. The content of reports is configured for each user separately. Read more in this article.



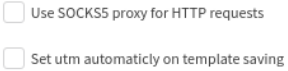
Global sending policy

In the next tab, you can set the global sending policy. It will apply to all resources and mailings that do not have their own policy:



Additional options

At the bottom of the page there are options for proxying tracking requests and auto-setting utm tags when saving a message template:



Custom links

Description

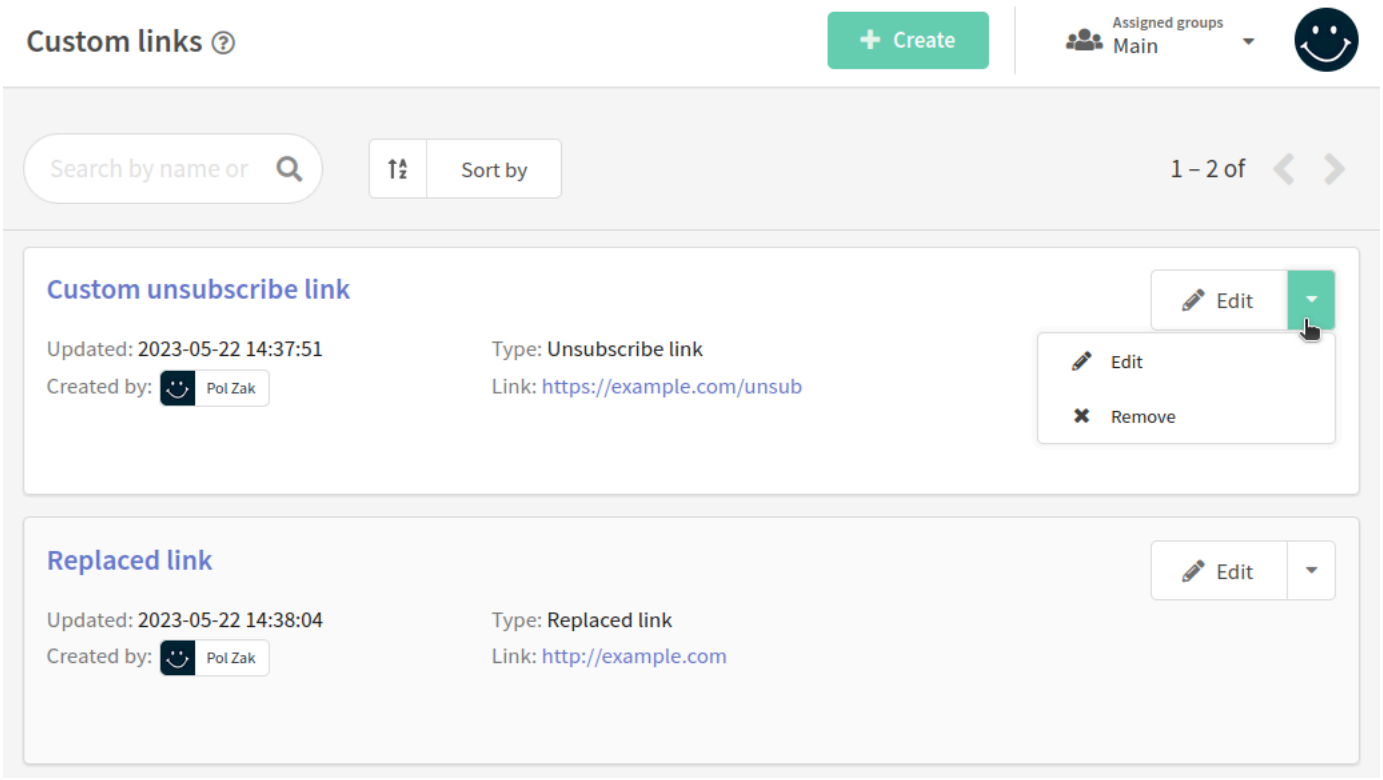
There are two types of custom links available: unsubscribe link and replaced link. With a custom unsubscribe link, you can redirect email recipients to their personal account on your site or to any web page when they click on the “Unsubscribe” button. Replaced link allows you to replace the URLs of links in messages that have already been sent.

Managing custom links

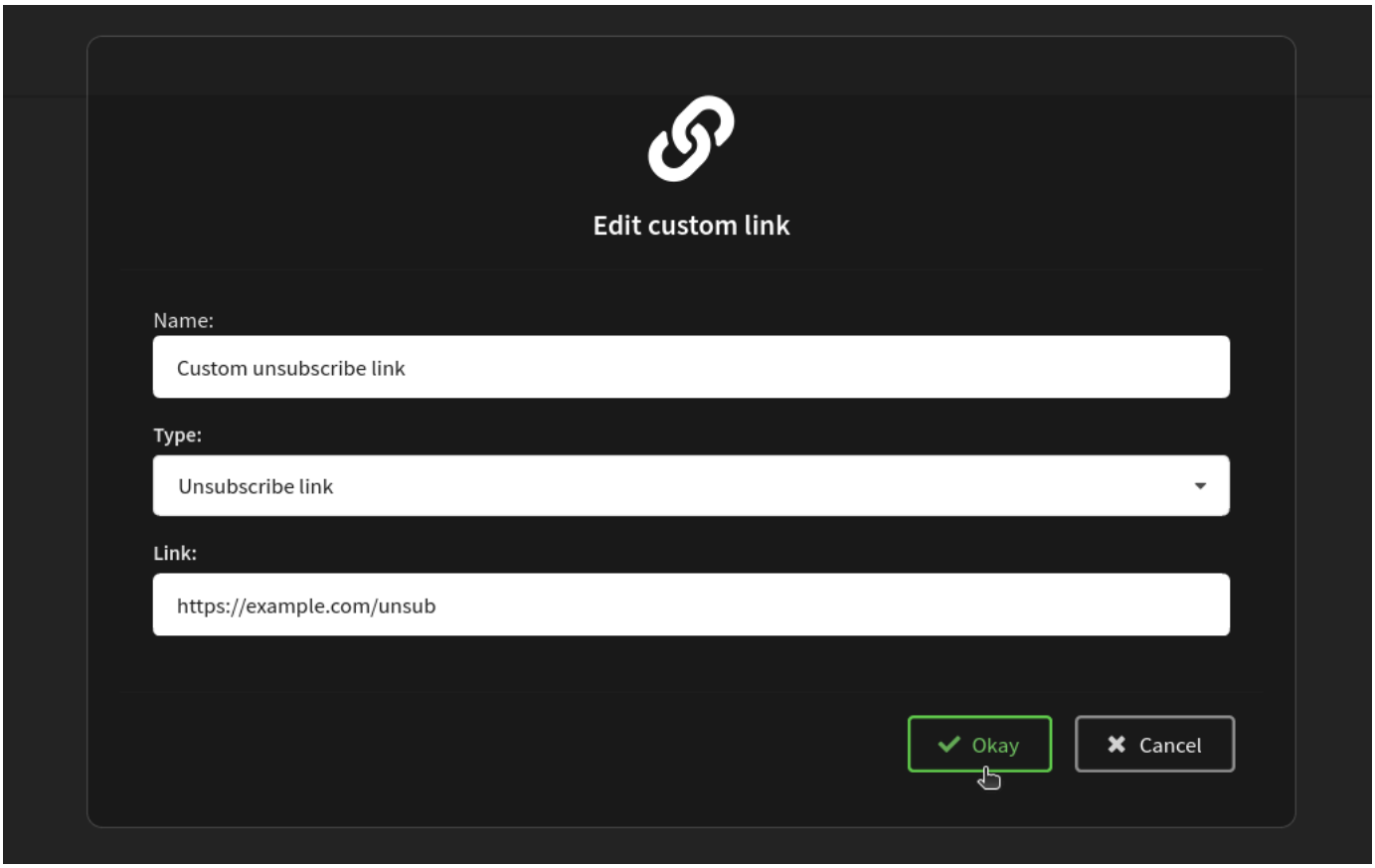
CAUTION

To create custom links you would like to have master useraccount.

In the main menu select Settings —> Custom links and start with + Create button:

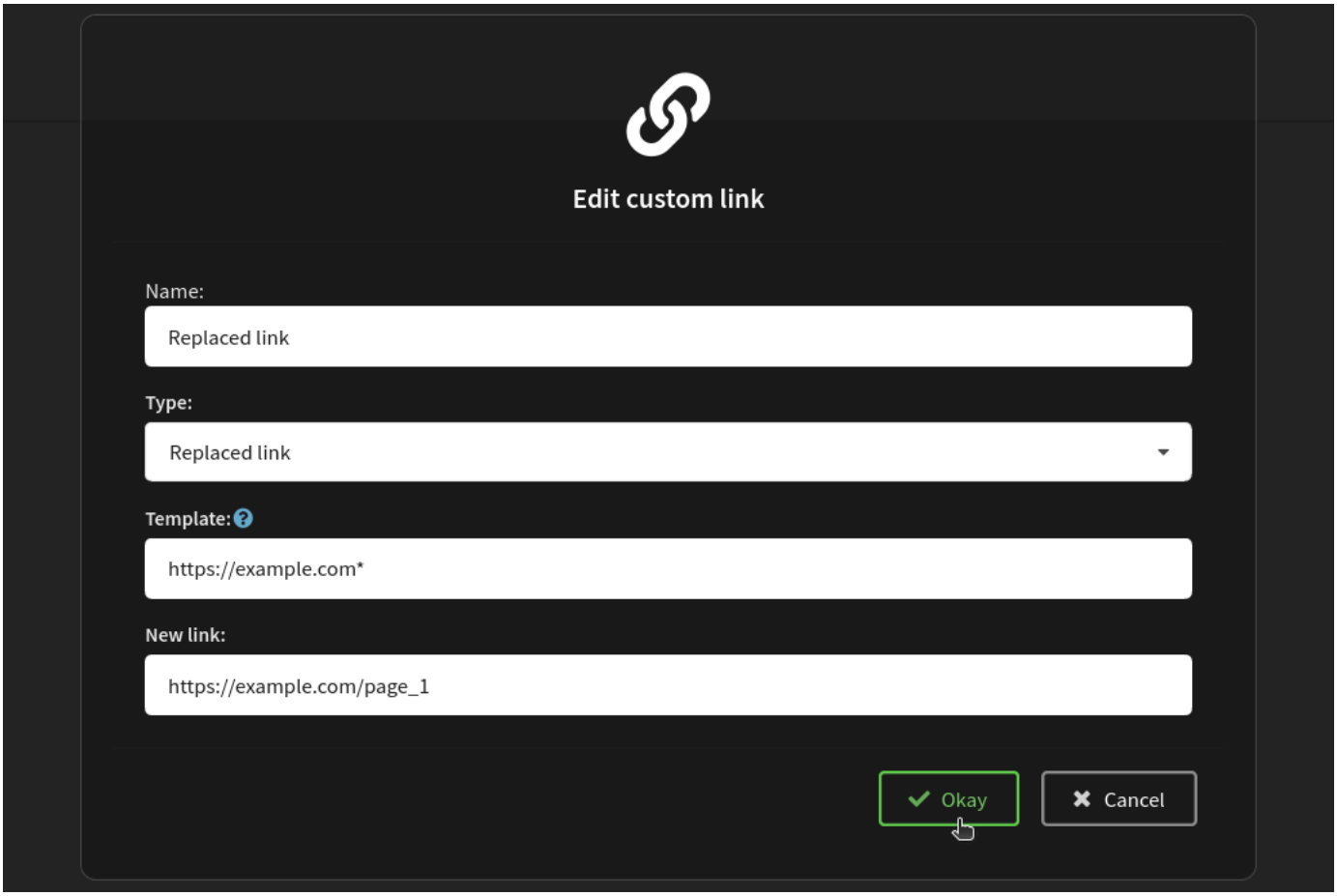


In the editing form enter the new link name, select its type (Unsubscribe or Replaced) and enter the URL for redirection.



For a replaced link specify:

- **Template — an old link or a part of a link. Wildcard (symbol *) can be used to search for multiple URLs that contain a specific string. For example, if you specify `https://example.com/*`, the platform will replace all links in message templates that start with `https://example.com`.**
- **New link — new redirect address for your link.**



Virtual senders

Overview

A virtual sender is a set of rules for real senders and a powerful tool for fine messaging tuning.

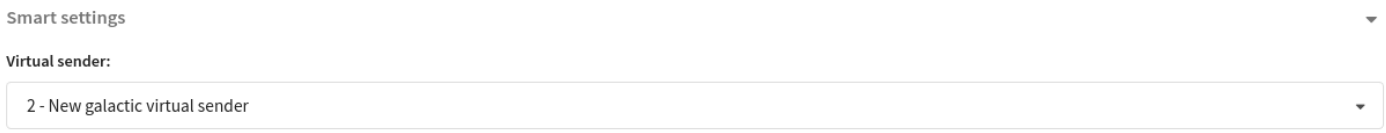
Your delivery rate can greatly depend on it — especially with large amounts of messages to be sent.

These rules specify real sender usage proportions and custom From and Reply-to email fields for messages sent.

> NOTE
>
> **Virtual senders can be created and managed only by a master user or an Altcraft platform administrator.**

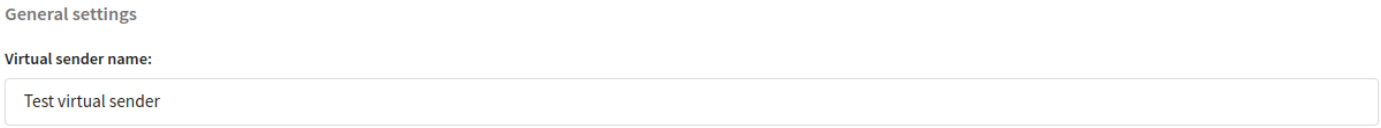
If you need to manage the virtual senders of your account, in the main menu, select Settings —> Virtual senders. Click + New to add a sender.

You can enable virtual senders for mailings if Smart settings are available for your account. If Smart settings are not available, the first valid virtual sender will be used.

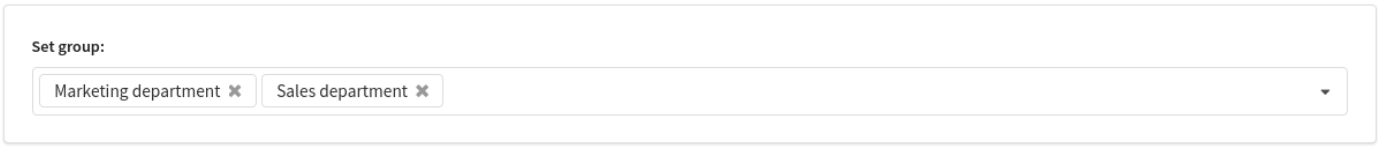


General settings

First of all, you need to enter a unique name for each sender:



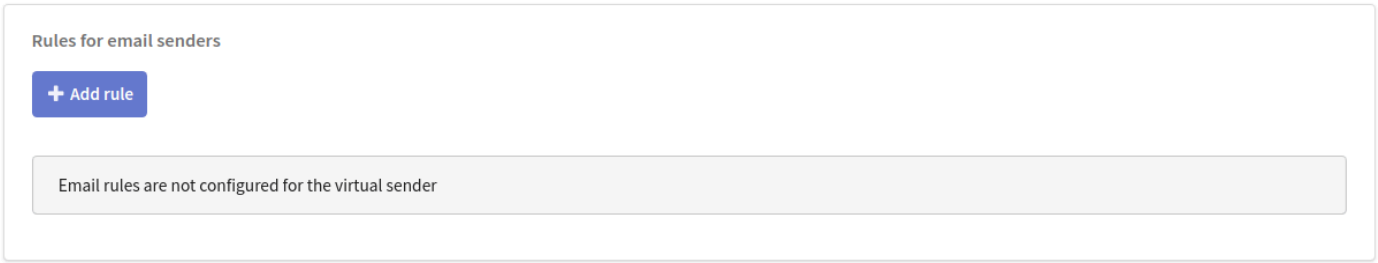
You can set groups in the bottom of the page. These groups will get access to the virtual sender.



Sending rules for different communication channels

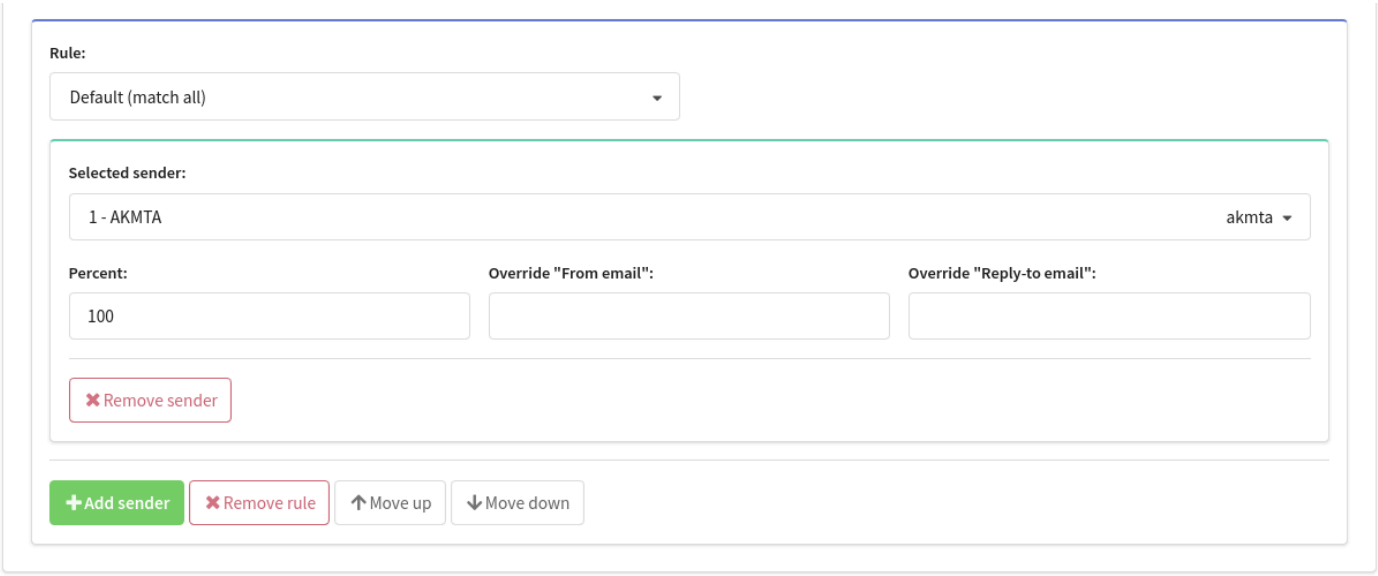
Rules for email senders

The rules for email senders determine which real senders and in what proportions will be used under certain conditions. To add a rule, click + Add Rule.



Select the appropriate condition from the drop-down menu When the rule is applied:

- **Match by group — the rule will be applied to mailings within a group specified for matching.**
- **Match by "From domain" — the rule is applied when sending from a specific domain.**
- **Match by "To domain" — the rule is applied when sending to specific email domains.**
- **Match by "From email" — the rule is applied when sending from a specific email address.**
- **Default (match all) — no conditions are required for this rule to be activated. It will be applied by default whenever a current virtual sender is set for a mailing.**



You can create multiple rules and sender options. The priority of applying rules is set by their position on the page: from top to bottom. The Default rule (match all) must be the last one in the list. The priority of other rules is set using the Move Up and Move Down buttons.

CAUTION

The Default rule (match all) is not possible to delete or move up in priority, since it handles all cases that do not fall under the other rules.

Using the Add sender and Remove rule buttons, you can add and remove senders within the sending rule. The following fields are configured for each sender:

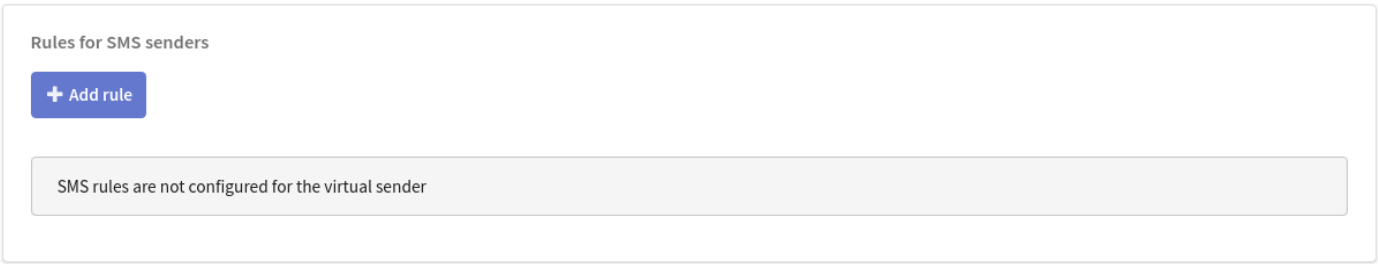
- **Selected sender — select a real sender for sending.**
- **Percent — set for the sender the percentage of messages sent within the rule.**
- **Override "From Email" — override the "From email value" for outgoing messages.**
- **Override "Reply-to Email" — override the "Reply-to email" value for outgoing messages.**

NOTE

For one rule the sum of senders usage percent must be 100.

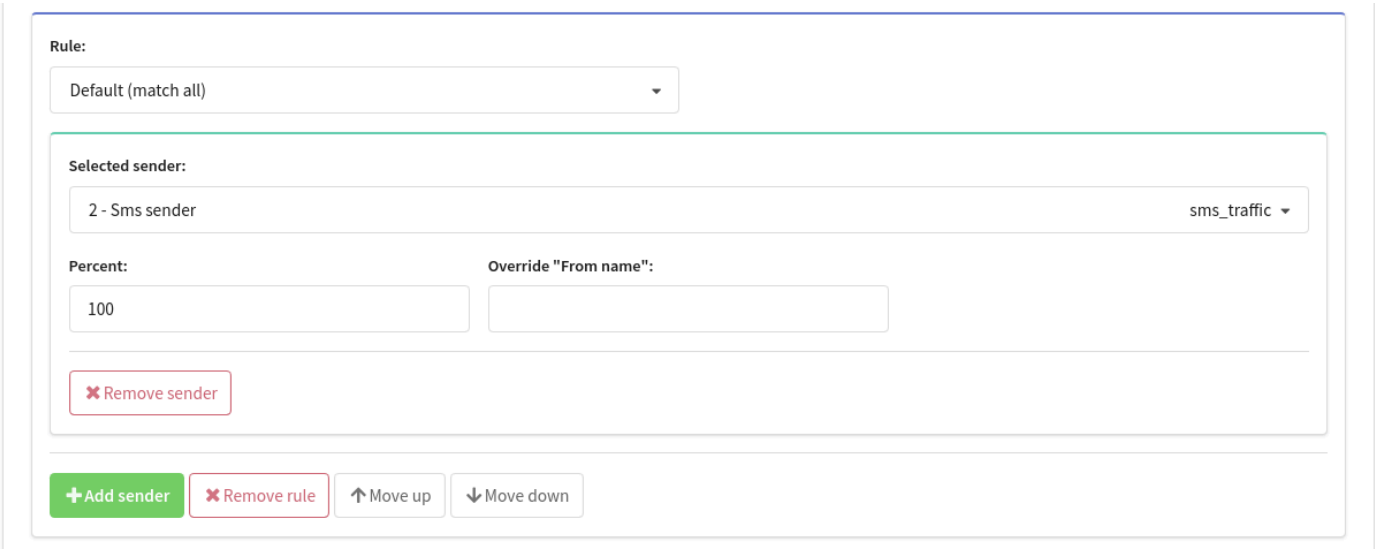
Rules for SMS senders

The rules for SMS senders determine which real senders and in what proportions will be used under certain conditions. To add a rule, click + Add Rule.



Select the appropriate condition from the dropdown menu with rules:

- **Match by group — the rule will be applied to mailings within a group specified for matching.**
- **Default (match all) — no conditions are required for this rule to be activated. It will be applied by default whenever a current virtual sender is set for a mailing.**



You can create multiple rules and sender options. The priority of applying rules is set by their position on the page: from top to bottom. The Default rule (match all) must be the last one in the list. The priority of other rules is set using the Move Up and Move Down buttons.

CAUTION

The Default rule (match all) is not possible to delete or move up in priority, since it handles all cases that do not fall under the other rules.

Using the Add sender and Remove rule buttons, you can add and remove senders within the sending rule. The following fields are configured for each sender:

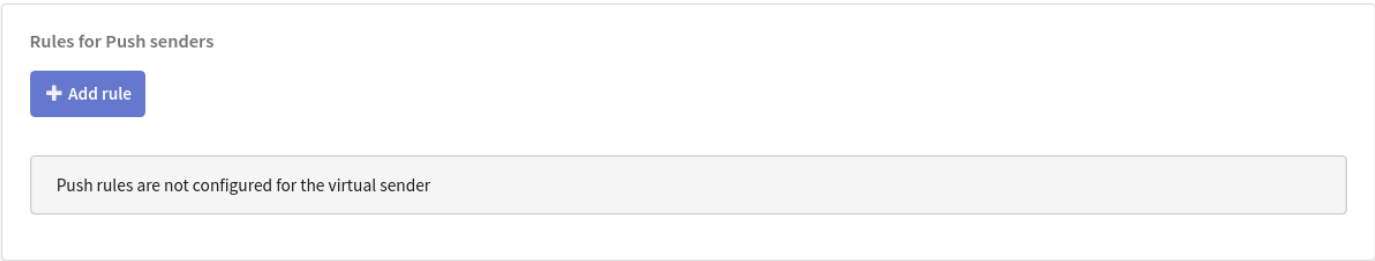
- **Selected sender — select a real sender for sending.**
- **Percent — set for the sender the percentage of messages sent within the rule.**
- **Override "From name" — override the "Caller_ID" value for outgoing messages.**

NOTE

For one rule the sum of senders usage percent must be 100.

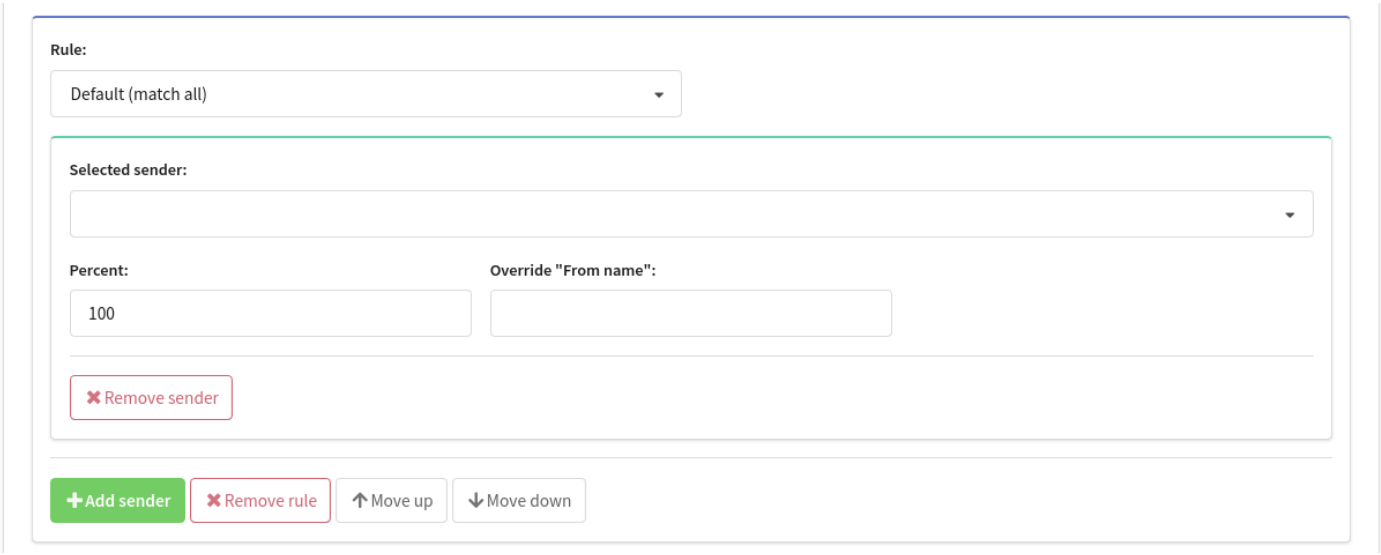
Rules for push senders

The rules for SMS senders determine which real senders and in what proportions will be used under certain conditions. To add a rule, click + Add Rule.



Select the appropriate condition from the dropdown menu with rules:

- **Match by group — the rule will be applied to mailings within a group specified for matching.**
- **Default (match all) — no conditions are required for this rule to be activated. It will be applied by default whenever a current virtual sender is set for a mailing.**



You can create multiple rules and sender options. The priority of applying rules is set by their position on the page: from top to bottom. The Default rule (match all) must be the last one in the list. The priority of other rules is set using the Move Up and Move Down buttons.

CAUTION

The Default rule (match all) is not possible to delete or move up in priority, since it handles all cases that do not fall under the other rules.

Using the Add sender and Remove rule buttons, you can add and remove senders within the sending rule. The following fields are configured for each sender:

- **Selected sender — select a real sender for sending.**
- **Percent — set for the sender the percentage of messages sent within the rule.**
- **Override "From name" — override the "From name" value for outgoing messages.**

NOTE

For one rule the sum of senders usage percent must be 100.

Sending policies

Overview

Sending policies prevent from sending too much messages to a single recipient. Such policy can be applied to:

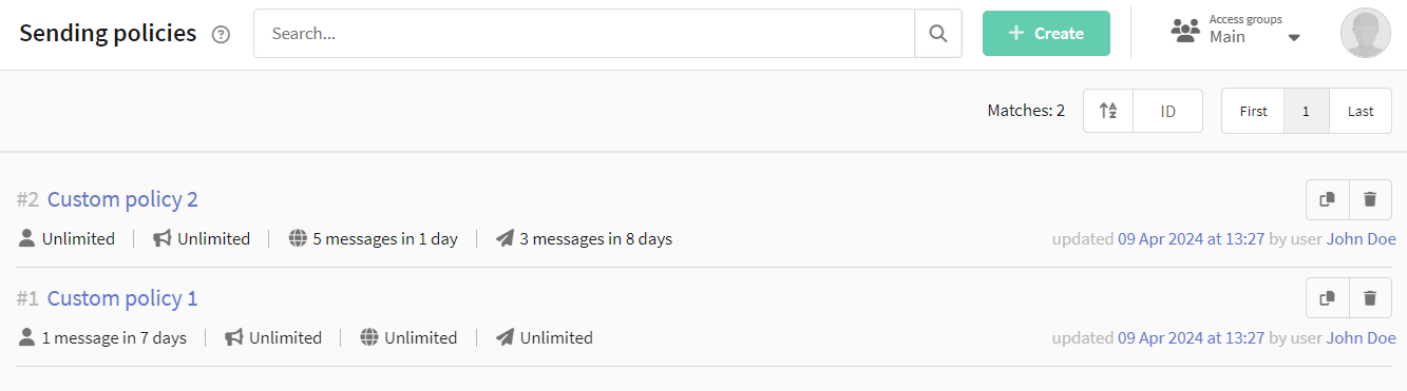
- **Account — a policy will be applied globally to all account mailings.**
- **Resource — a policy will affect all mailings within a Resource, overriding Account policy.**
- **Mailing — a policy will affect only the mailing it was applied to, overriding both Account and Resource policies.**

> CAUTION
>
> **The sending policy cannot be set for a specific profile. The policy is configured and valid for all profiles of the account/resource/company.**

Managing sending policies

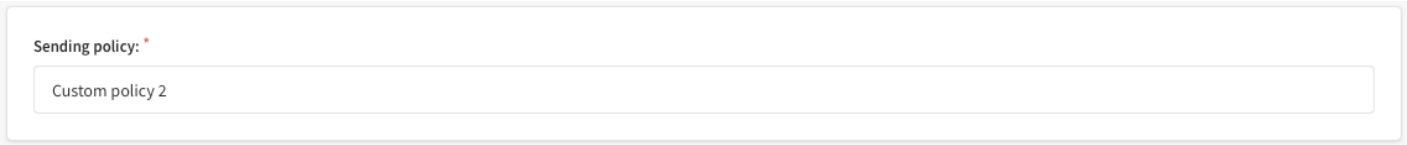
> CAUTION
>
> **Only master-users can create, edit or delete sending policies.**

To start managing sending policies in the main menu select Settings —> Sending policies. The list you will see contains the existing sending policies:



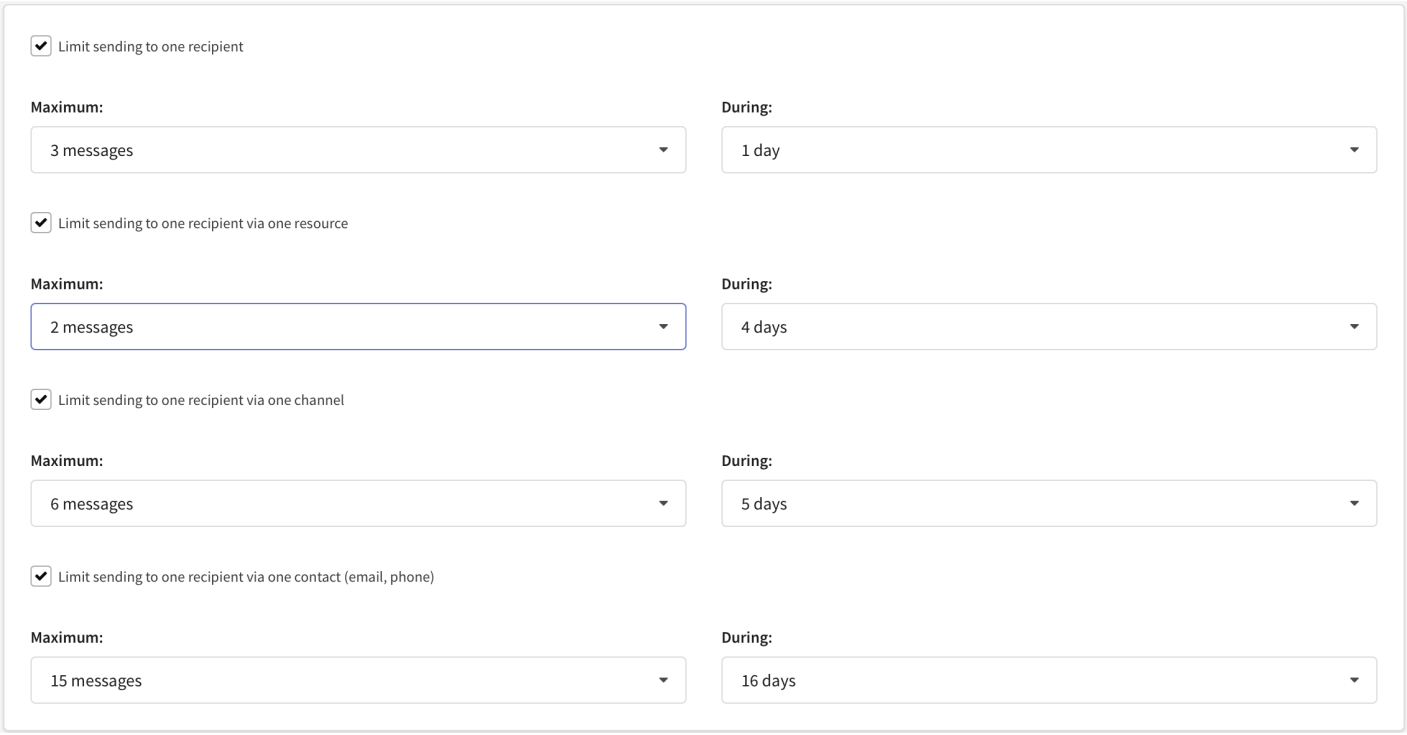
You can clone or delete a policy in a context menu on the right side of the list. To edit a policy you can as well click on its name.

To create a new policy click + Create in page header. Every policy has three sections of settings. In the first one specify policy name:



In the second section specify messaging limitations. Set maximum number of messages between 1 and 15 and time period between 1 and 16 days:

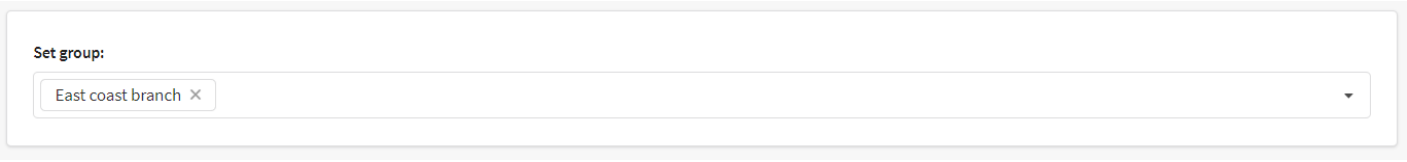
- **Limit sending to one recipient — will limit messaging from all resources and channels for one subscriber profile.**
- **Limit sending to one recipient via one resource — will limit messaging from all channels of any single resource for one subscriber profile.**
- **Limit sending to one recipient via one channel — will limit messaging via one channel (email, sms, push) from all resources for one subscriber profile.**
- **Limit sending to one recipient via one contact (email, phone) — will limit messaging for a subscription containing specific contact data. The subscription can be associated with multiple subscriber profiles.**



TIP

A sending policy with no restrictions specified is unlimited. Thus you can set restrictions for the whole account and then override them for a single mailing or a resource applying an unlimited policy.

In the third section you can optionally set access group to the sending policy. Thus it will be available only in Resources and Mailings of the groups you specify:

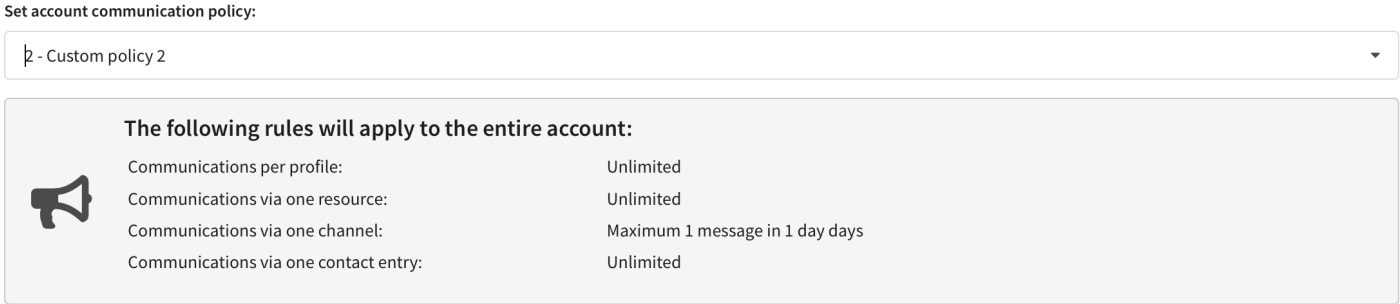


Applying sending policies

You can apply a sending policy to the whole account, to a specific resource - or to a specific mailing.

Global sending policy

To apply a global account policy open Settings —> Account settings and select a sending policy in this dropdown menu:

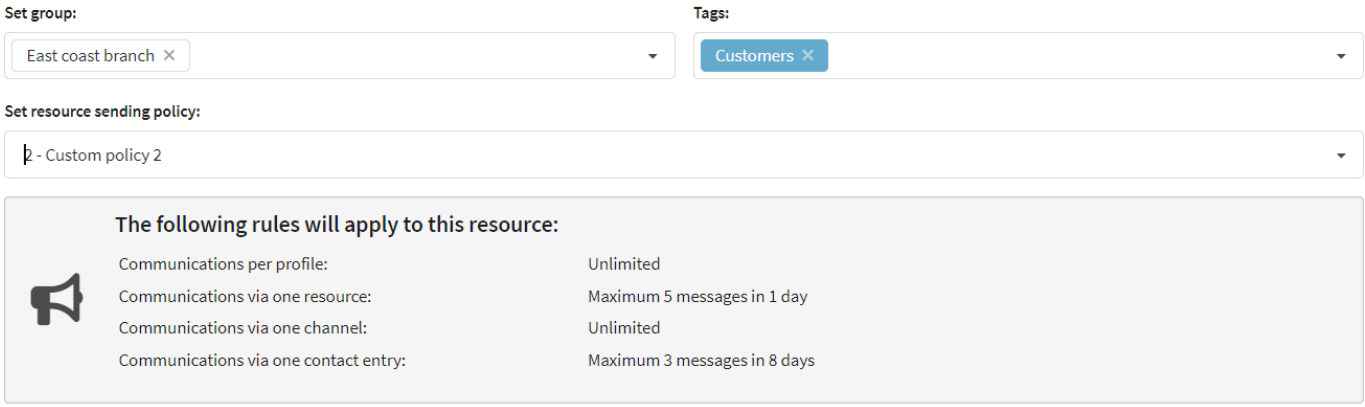


CAUTION

Only master-users have access to Account settings and thus only they can set global sending policies.

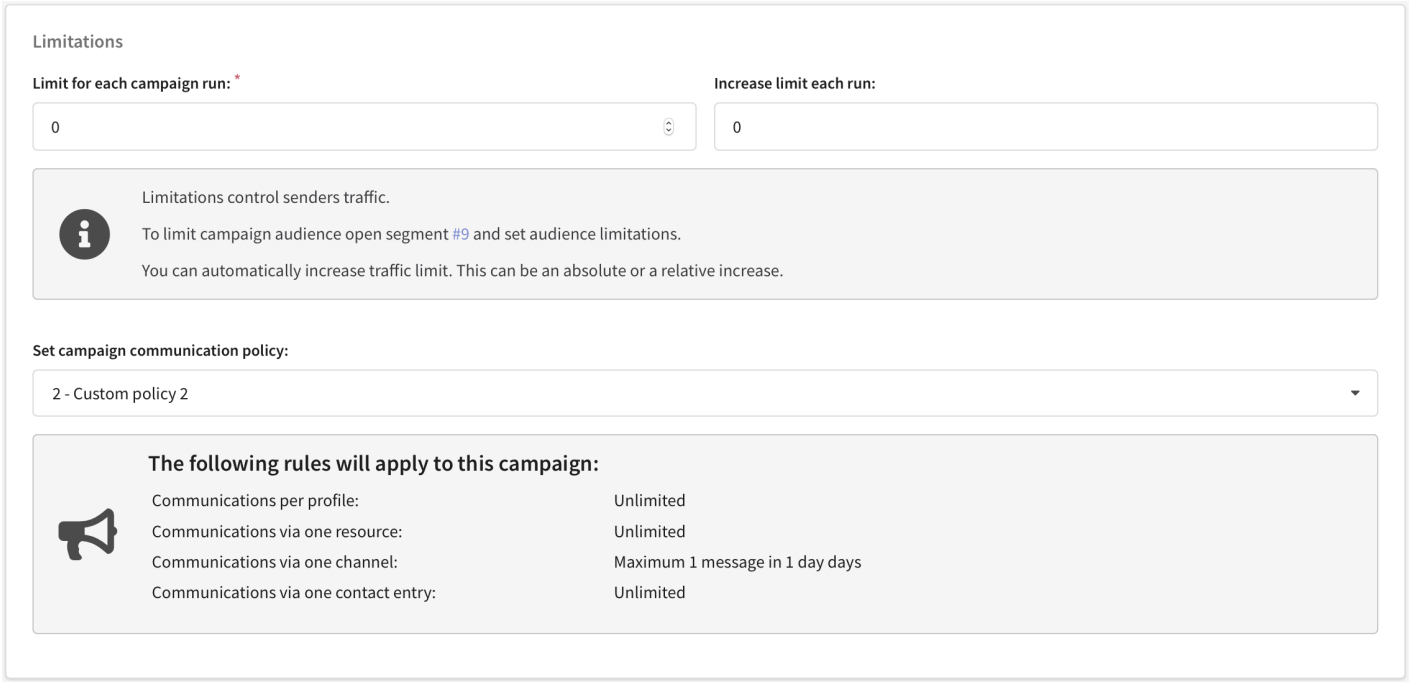
Resource sending policy

A policy applied to a Resource will override global Account settings. Open resources in the main menu and in General settings section of a resource select a sending policy:



Mailing sending policy

A policy applied to a specific mailing will override both account and resource sending policies. Open a mailing and set a sending policy in Limitations section (Set mailing communication policy).



INFO

Messages that fall under restrictions will form entries in the mailing log of the type Sending is restricted by the policy applied.

Audit journal

Description

Audit journal is an important security and monitoring element that records user actions in the platform. Only users with master privileges have access to the audit. To go to the audit, open Settings → Audit journal.

User action audit is a very important security and monitoring feature, capturing Altcraft platform user actions. Only users with master privileges can access audit log by opening Settings —> Audit journal.

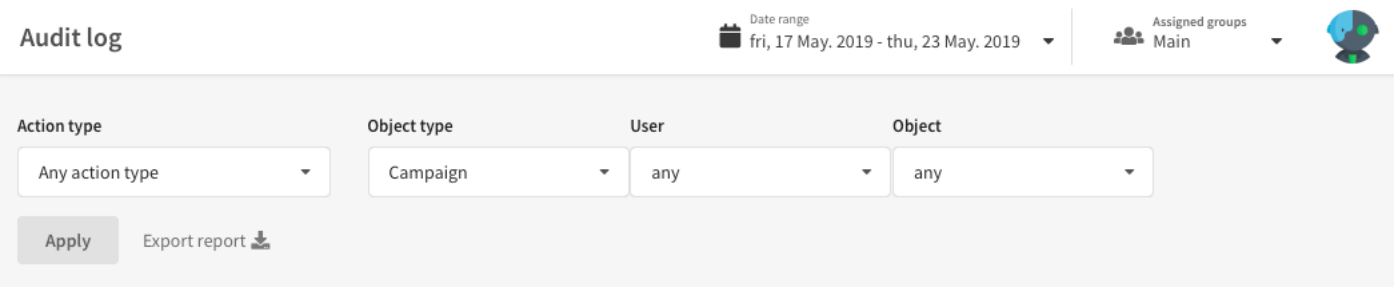
The audit log is stored for 90 days. To change this value, you need to add the entry `"AUDIT_LOG_TTL" : 90` to the `main.json` configuration file, changing the value as needed.

Audit journal settings

You can configure the audit query according to your current tasks:

- **Period — time frame for displaying recorded actions.**
- **Action type — type of event recorded by the system. Actions are related to authorization in the platform, creation, editing and activation of objects in the account.**
- **Object type — allows you to view actions related to a specific type of entities in Altcraft. These can be users, groups and access roles, databases and segments, resources and mailings, campaigns, templates and their parts, as well as other objects.**
- **Object — you can select a specific account object and get information about actions with it.**

To view the audit report, click Apply. To export the audit report to .XLS or .CSV, click Export report:

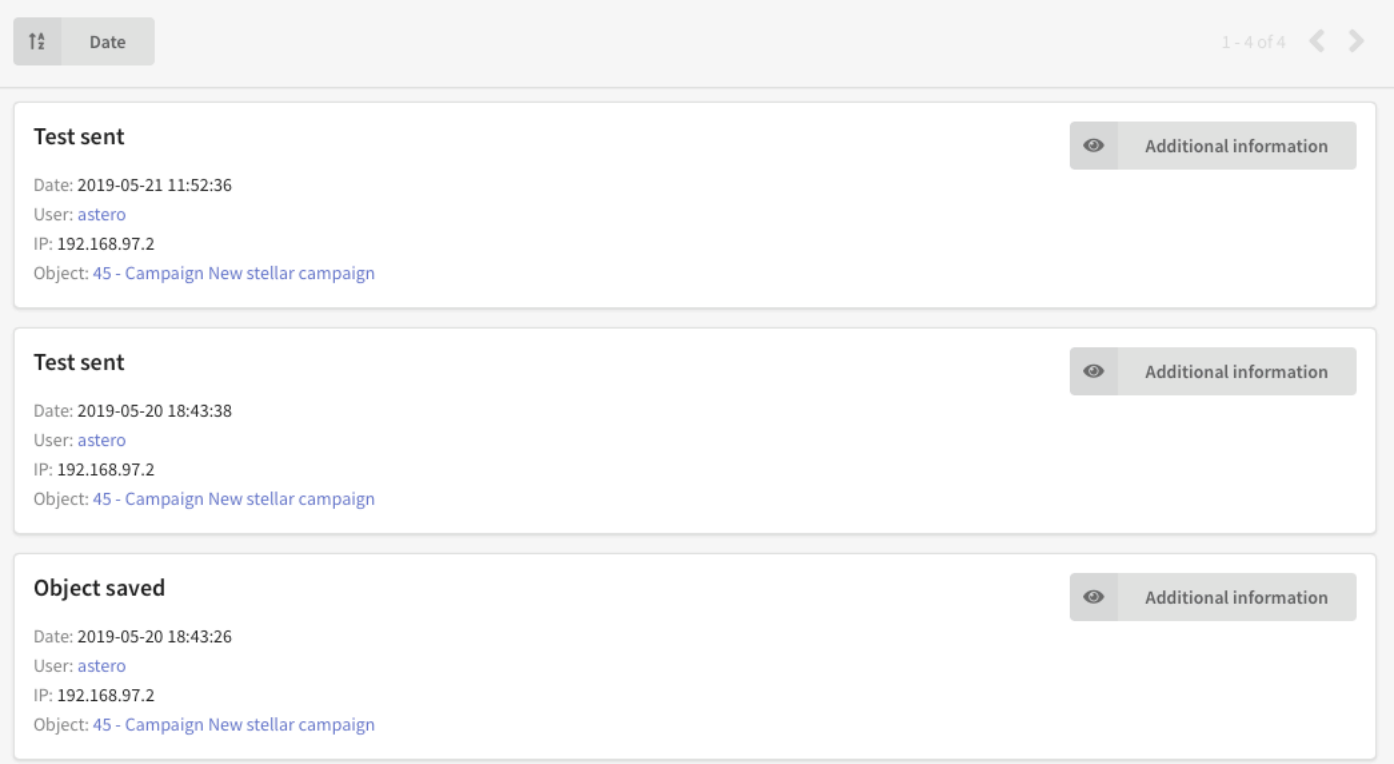


Viewing audit events

Actions that meet the query conditions are displayed as a list of tiles. The title of each tile indicates the type of recorded action. For each action, the date, the user who performed the action, their IP address and the object associated with the action are displayed.

Additional information contains a more detailed description of the event, for example, email addresses to which the test was sent.

Actions in the report can be sorted by date, user, IP address or object.



Campaigns audit actions

The system records the following actions with campaigns (combinations of mailings and scenarios):

Basic campaign operations

- **Campaign creation — creating a new campaign**
- **Campaign update — changing campaign settings**
- **Campaign deletion — deleting a campaign**
- **Campaign activation — launching a campaign**
- **Campaign deactivation — stopping a campaign**
- **Campaign completion — final completion of campaign work**

Local control groups (LCG) management

- **LCG creation — setting up a local control group**
- **LCG update — changing LCG parameters**
- **LCG deletion — deleting a local control group**

Marketing activities management

- **Activity binding — adding a mailing or scenario to a campaign**
- **Activity unbinding — removing a mailing or scenario from a campaign**

Segments management

- **Segment binding — assigning a segment to a campaign**
- **Segment unbinding — removing a segment from a campaign**

Loyalty programs operations

- **Loyalty program creation — creating a new loyalty program**
- **Loyalty program update — changing loyalty program settings**
- **Loyalty program deletion — deleting a loyalty program**
- **Loyalty program activation — launching a loyalty program**
- **Loyalty program deactivation — stopping a loyalty program**

All the above actions are also available for points types and loyalty levels.

Partner operations in loyalty programs:

- **Loyalty program partner creation — creating a new loyalty program partner**
- **Loyalty program partner update — changing loyalty program partner**
- **Loyalty program partner deletion — deleting a loyalty program partner**

Promotion operations:

- **Promotion creation — creating a new promotion**
- **Promotion update — changing promotion settings**
- **Promotion deletion — deleting a promotion**
- **Promotion activation — launching a promotion**
- **Promotion deactivation — stopping a promotion**
- **Promotion completion — final completion of promotion work**

The audit journal also stores information about attaching and detaching loyalty programs to subscriber profiles.

> TIP
>
> **When working with mailings and scenarios within campaigns, the system also records standard actions (deletion, deactivation, completion) with the parent campaign ID specified.**

Searching tags FAQ

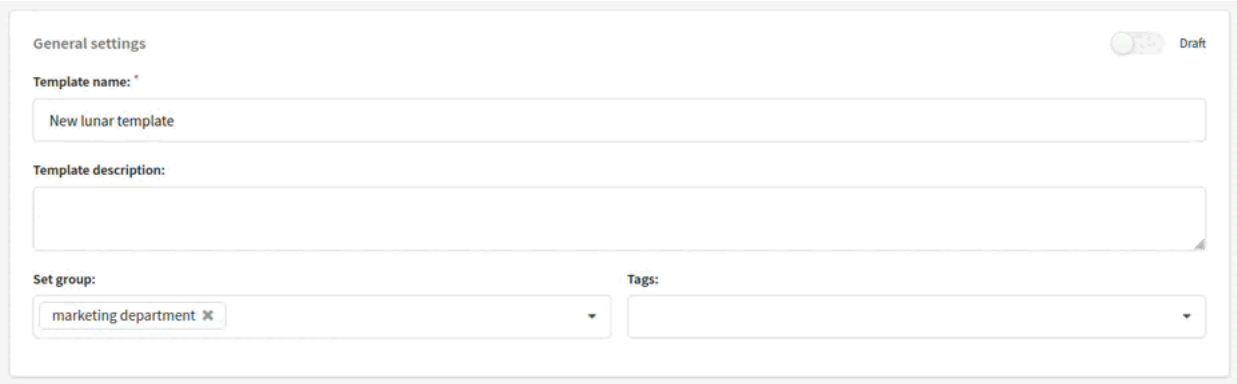
Tag is a label that can be assigned to some platform objects to quickly find these objects later. Tags usually associate objects with a common theme. For example, mailings that are launched as part of one promotion, or templates targeting the same audience.

Which objects can be tagged:

- **databases**
- **resources**
- **segments**
- **templates**
- **scheduled imports**
- **suppression lists**
- **mailings**
- **campaigns**
- **scenarios**
- **forms**
- **products and orders**

How to create a tag

The tag is created directly in the object to which you want to assign this tag. Go to the object editing window, place the cursor in the Tags field and enter the text, then press Add or ↵ Enter.



To unpin a tag from an object, click the cross symbol.

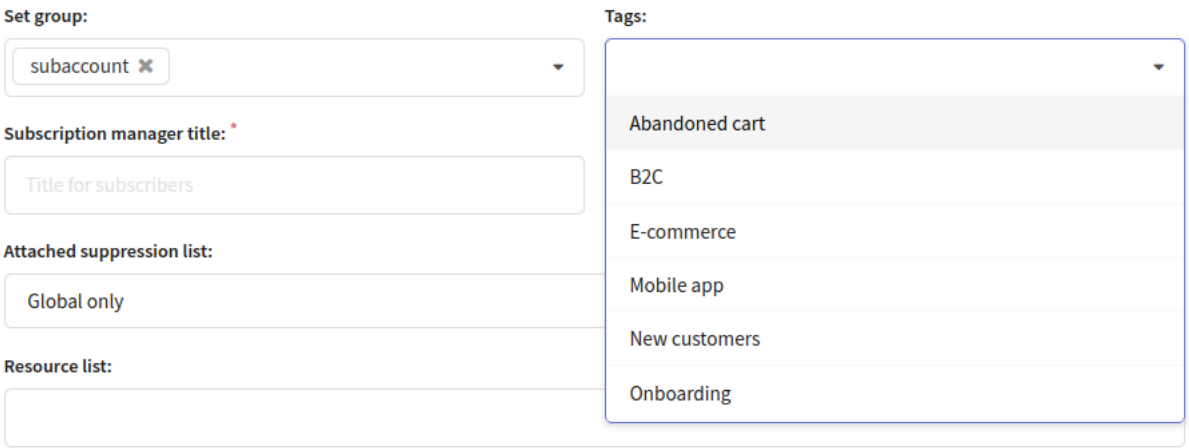
You can assign any number of tags to an object. Tag names are not limited in any way:

- **any number of characters**
- **Cyrillic and Latin**
- **spaces**
- **special symbols**

Tag examples: `new2023, брошенная корзина, abandoned_cart.`

Where I can see my tags

When you save an object with an assigned tag, the tag is also saved. There is no separate window where you can view all tags. However, when editing another object, all previously created tags will be available in the drop-down list:



While a saved tag can be detached from an object, it cannot be completely removed from the platform. All account users see all existing tags.

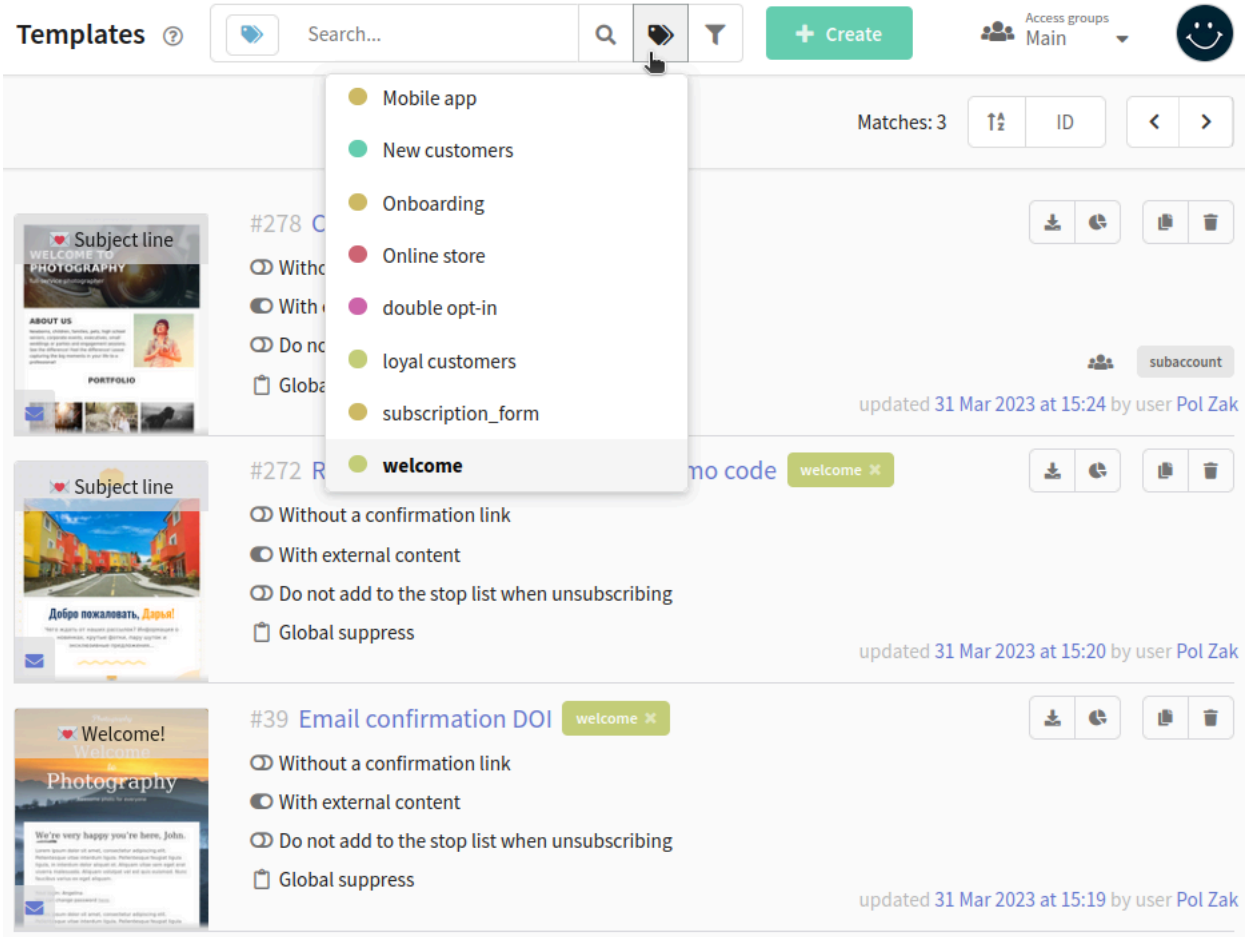
If a tag is not attached to any platform object, it will be automatically deleted.

Where to use tags

1. Search for objects in the general list

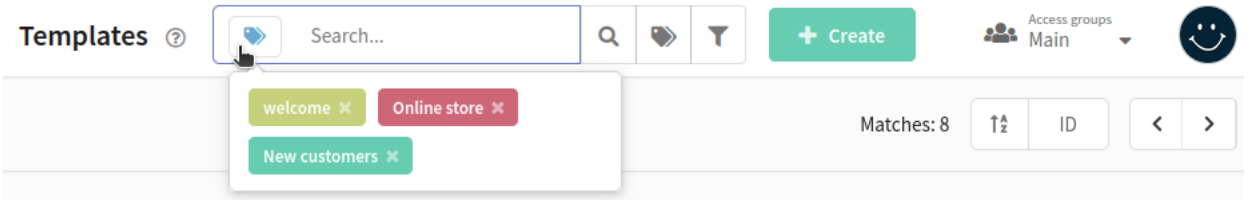
Go to the desired section (for example, Templates) and click the icon

to the right of the search bar. Select one or more tags.



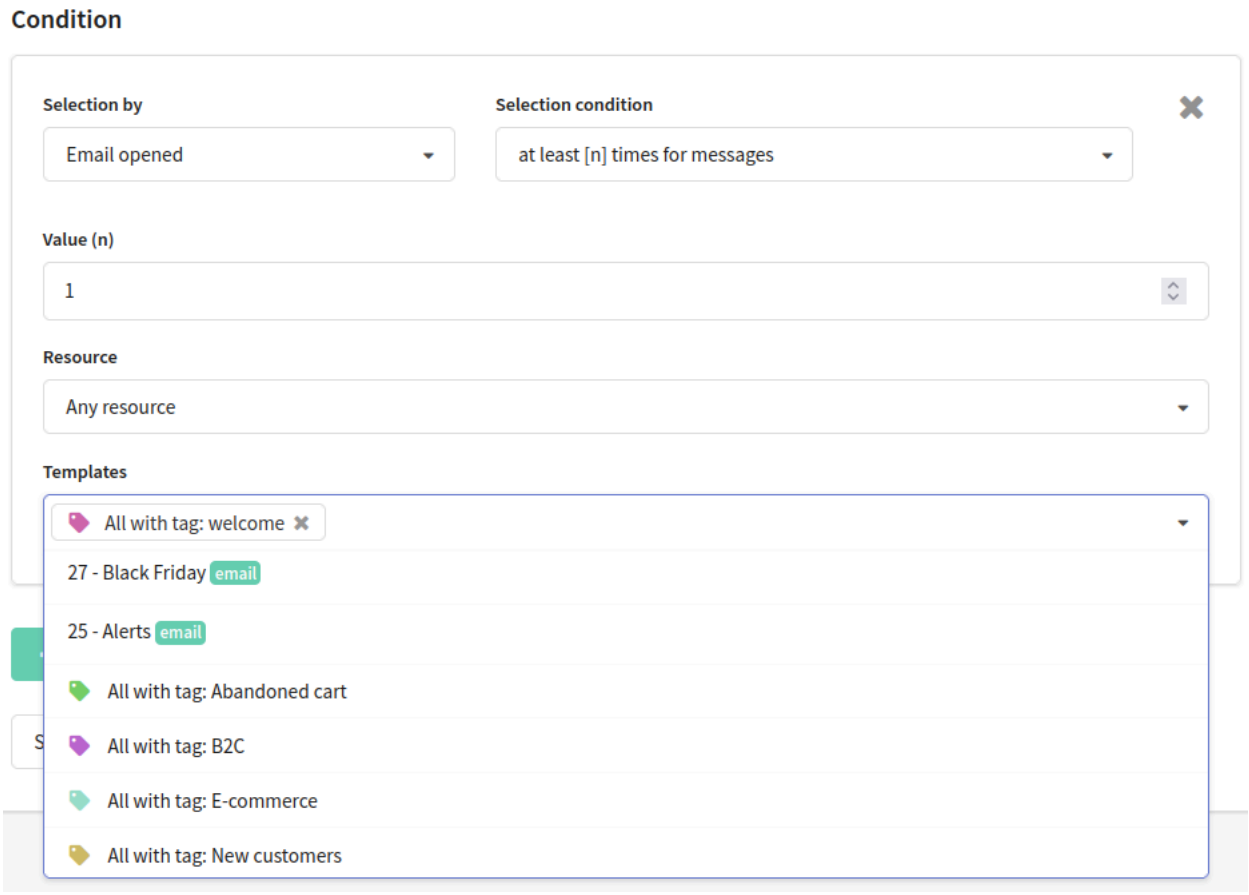
To see what tags have been added to the search, hover over

:



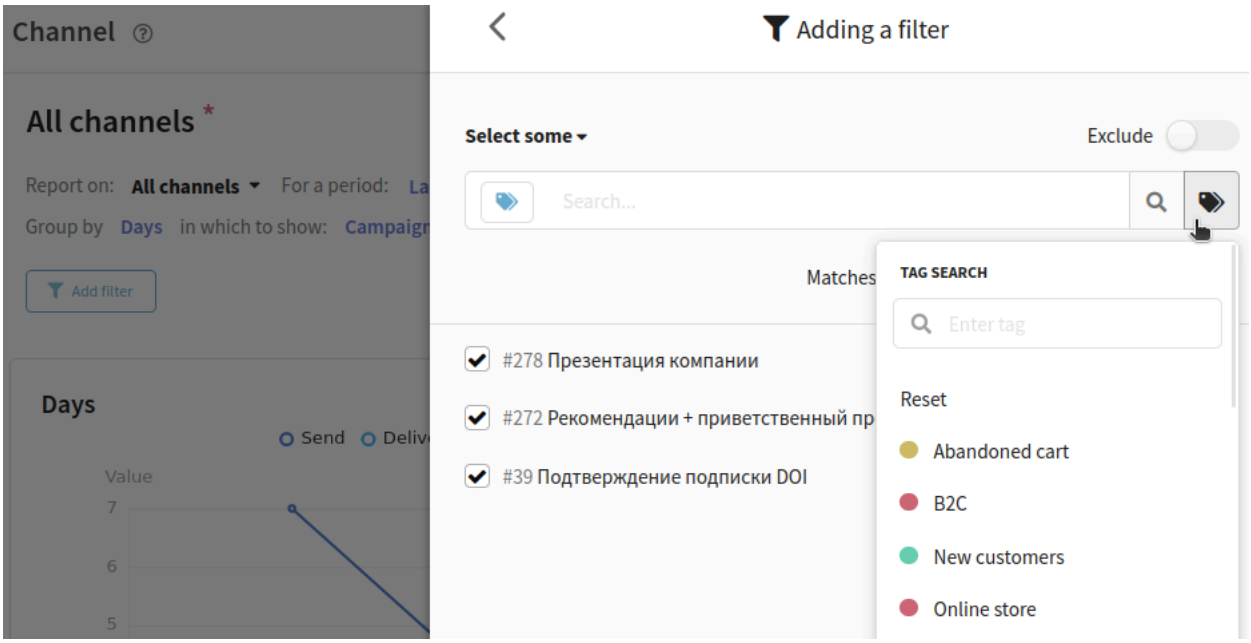
2. Segmentation of profiles by events in mailings or templates (sends, deliveries, opens, clicks)

When you use one of these selection conditions, you can select all mailings or all templates with a specific tag or tags:

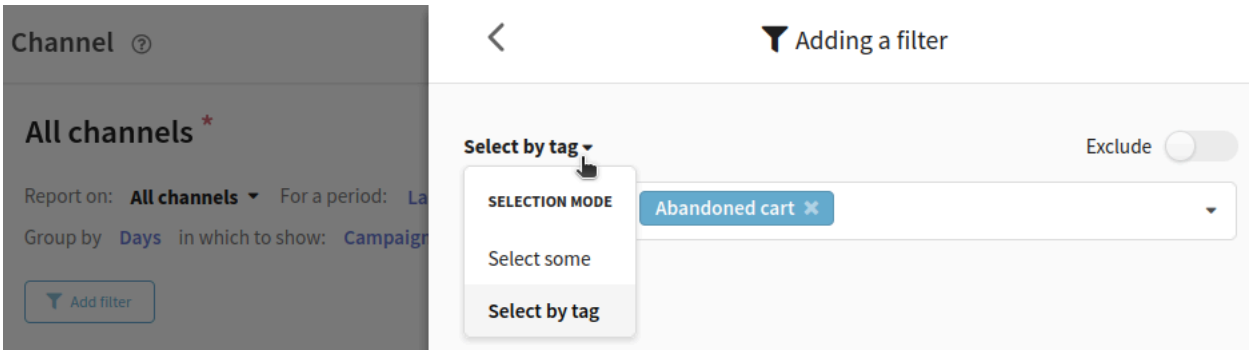


3. Analytical reports

In the [channel report] and in the [traffic report], you can use tags in filters. For example, to quickly find the desired object:



You can also select all objects of the same type that have one or more tags assigned at once. To do this, switch the selection mode: Single selection of campaigns —> Selection of campaigns by tag. Then select tags in the drop-down list and apply the filter:


Channel
All channels *
Report on: All channels
For a period:
Group by Days
in which to show:
Add filter
Adding a filter
Select by tag
Exclude
SELECTION MODE
Select some
Select by tag
Abandoned cart

Attribute settings

What are entity attributes

Attributes are custom user-defined fields that can be added to platform entities. They represent a specific parameter with a set of possible values. Attributes can be used not only to tag specific groups of entities for search, but also as segmentation criteria, variables, or embedded into API requests. Attributes can be assigned to campaigns, mailings and scenarios.

> TIP: Attribute inheritance
>
> **Attributes are inherited hierarchically through the chain: Campaign → Scenario → Mailing. This means that if an attribute value is not set for a mailing, the value from the scenario will be used, and if not set there either — the value from the campaign.**

Creating and configuring attributes

To create and configure attributes, go to Settings — Attribute configuration. This opens the attribute management panel. Create your first attribute by clicking the corresponding button.

A configuration window will appear. First, specify the name and the API name for the attribute. Then select the entities the attribute will be available for — campaigns, mailings or scenarios (at least one type is required).

Next, configure the necessary parameters:

- **Required attribute — the attribute must be filled in for any entity where it is available. Without it, a campaign, mailing or scenario cannot be saved.**
- **Disable attribute editing in active entities — prevents changes to the attribute's value once the entity is activated.**
- **Limit character count — sets the maximum length of the attribute value.**
- **Use a regular expression — validates the attribute value against a regular expression.**

The next step is to define the attribute values. You can also allow users not only to select from predefined values, but to add new ones when editing the entity. Each attribute can have up to 30 values.

> TIP
>
> **Attributes are inherited hierarchically through the chain: Campaign → Scenario → Mailing. This means that if an attribute value is not set for a mailing, the value from the scenario will be used, and if not set there either — the value from the campaign.**

Later, values can be not only deleted, but also archived. Archived values cannot be assigned to new entities but remain available for existing ones and can be used in filters and search.

If needed, set a default value for the attribute. Then save and activate it. After saving, you can deactivate, clone, or delete attributes from the list.

> CAUTION: Displaying changes
>
> **Even saved changes to attribute settings may be displayed incorrectly until the page is reloaded.**
>
> **We recommend refreshing the editing page after each time you save changes.**

Using attributes in the platform

Search and filtering

Attribute values are used to search and filter platform entities.

Profile segmentation

When segmenting by subscriber actions, you can use attribute values to select campaigns, mailings and scenarios for profile filtering.

GCG exclusions

Users can disable global control groups for scenarios, mailings and campaigns with specific attribute values.

LCG exclusions

In local control group (LCG) settings, you can filter profiles by attribute values of campaigns, scenarios and mailings. This allows flexible configuration of profile exclusion rules from the control group.

Variable substitution

You can insert the attribute value into templates and scenarios using the `{attribute_value("attribute_name")}` **variable, where** `"attribute_name"` **is the attribute's API name.**

Usage examples:

- `{attribute_value("test_attribute")}` **— will substitute the value of “test_attribute” attribute**
- `{attribute_value("campaign_type")}` **— will substitute the value of “campaign_type” attribute**
- `{attribute_value("priority")}` **— will substitute the value of “priority” attribute**

API requests

Parameters with the attribute name and value are available in some API requests to the platform. For more information, see this article.

Users, groups and access

In Altcraft, you can restrict the functionality of user accounts as well as tokens used for authorization in the platform API. Access is determined by user or token groups and roles:

- **Groups define the visibility of objects such as databases, templates, campaigns and scenarios, etc.**
- **Roles define the set of available actions with visible objects.**

> CAUTION
>
> **Managing account users and access control is only available to users with master privileges.**

Access groups

Groups in Altcraft account allow you to restrict user access to certain account objects, such as templates, campaigns, databases and segments. A group can be thought of as a container for objects. Thus users will work within their containers and cannot accidentally or intentionally affect objects in other groups.

Groups are convenient when you need to organize the work of several departments within one account with shared statistics, or create conditions for third-party specialists to work, for example - designers for working with templates or website/application developers for debugging integration.

> INFO
>
> **If you need to more strongly separate the areas of responsibility of several company departments, use multiple platform accounts. Accounts can use shared or dedicated resources and components. A common email suppression list is created for all accounts of one platform.**

Main group

The Main group is created by the system and contains all account objects, including objects belonging to other groups. If you delete a custom group, the objects in it are not deleted, but are transferred to the Main group. You can work in the Main group just like in any other, if division into groups is not required.

> CAUTION
>
> **Master access implies full access to the account's Main group regardless of additionally assigned groups and roles. The only exception is access to delete profiles. The role for deleting profiles is assigned separately.**

Create / assign group

Groups are assigned and created dynamically when editing account objects - in the General settings block. To assign an existing Group, select it from the dropdown list in the Set group field. For example, for a Message template.

> CAUTION
>
> **Only Master users can create and set groups.**

To create a new group, enter its name in the same field for any account object and press Enter on the keyboard or Add [group_name] in the dropdown list.

> CAUTION
>
> **Each object created by a user is assigned to the group they were in at that moment.**

Switch / delete group

Groups available to the user are displayed in the menu in the upper right corner of the application, next to the account avatar. You can switch from group to group from any account menu.

> NOTE
>
> **Group deletion is available in the same list - only for master users.**

Access roles

An access role defines what actions with account objects are allowed for a user. To go to role management, select Settings → Roles in the main menu.

The opened list displays existing roles. You can clone or delete a role from the context menu on the right side of the tile. If you want to edit a role, click on its name.

To create a new role, click + Create at the top of the page.

Three parameters are configured for each role:

- **Name, which master users see in role lists.**
- **Groups on which the role restrictions will apply.**
- **Access rules to objects of the selected Groups.**

CAUTION

If the Main group is included in a role, this role will apply to all account objects. Master users are not subject to role and group restrictions. The exception is the ability to delete profiles, which must be activated independently, including for master users.

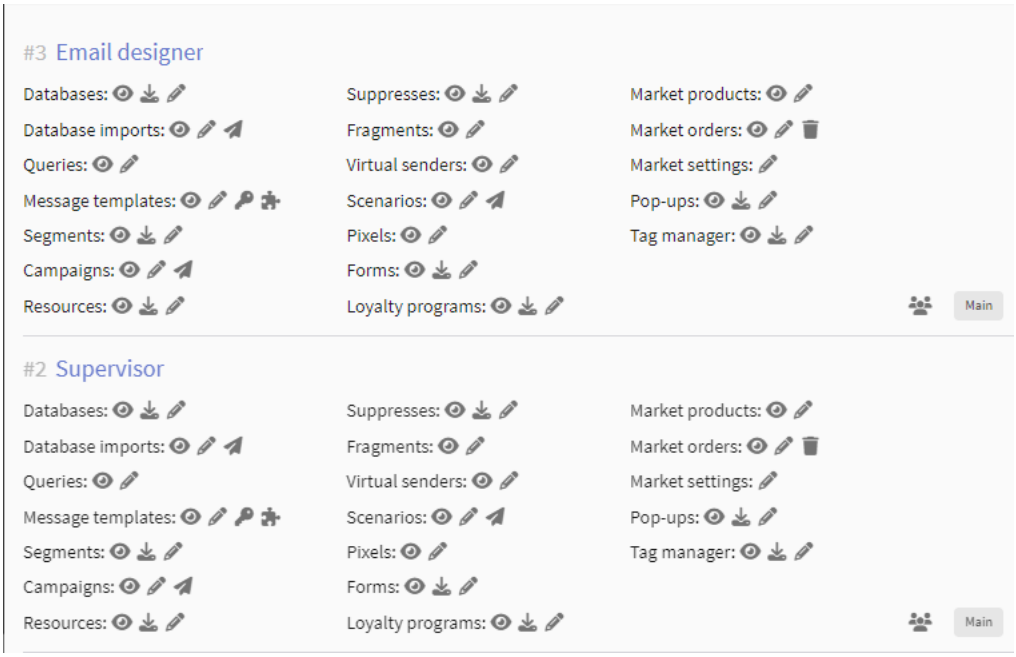
For all account objects, you can allow View, as well as Create and edit. For databases, segments, suppression lists and resources, Read and export is also available - downloading data lists.

NOTE

A role without the Moderation option in the Message access block will only allow saving message templates as drafts. Such drafts will be available in mailings only after activation - by a user with appropriate access rights. Such a role is convenient to set for contractor accounts creating mailing templates.

For Mailings and Scenarios you can prohibit Launch: users with such a role will only be able to prepare mailings, while checking and launching them will be done by a user with sufficient access rights.

A role defines which actions are permitted to a user or an API token within the assigned access groups. To manage roles in the main menu select Settings —> Roles:



The list you will see contains the existing access roles. You can edit, clone or delete a role in a context menu on the right side of the list. To edit a role you can as well click on its name. To create a new role click + Create in page header.

Any role has 3 parameters:

- **Role name. A master user can assign roles to users by their unique names.**
- **Groups where the role permissions are applied.**
- **Permitted actions with account objects.**

CAUTION

If Main group is selected in a role, role permissions will be applied to all account objects. Master-users are not affected by group and role restrictions. The only exception is the ability to delete profiles, it must be activated independently, also for the master user. For Campaigns separate access rights are available:

- **Campaigns - View object — view campaigns and their settings**
- **Campaigns - Create and edit — create and modify campaigns**
- **Campaigns - Launch — activate, deactivate and complete campaigns**

TIP: Permission dependencies

Working with Campaigns requires additional access rights:

- **Campaigns - View object requires view rights for: Mailings, Scenarios, Segments, Databases**
- **Campaigns - Launch requires launch rights for: Mailings and Scenarios**

Roles can restrict user actions with the following platform objects:

- **Resources — tool for managing customer subscriptions to various content through specific communication channels.**
- **Databases — database for storing customer data.**
- **Scheduled database imports — automatic synchronization of Altcraft profile database with external database (e.g., corporate CRM or SQL database).**
- **Relations — used to record interactions between your customers, or customers and company managers.**
- **Queries — SQL queries to external databases. The role regulates user work with template queries and segmentation queries.**
- **Message templates — content you send to subscribers as part of a mailing.**
- **Message fragments — part of a template that can be used in other templates.**
- **Segments — part of a database that includes profiles meeting certain conditions. The role regulates actions with static and dynamic segments.**
- **Suppression lists — list of email addresses, domains and phone numbers excluded from mailing.**
- **Mailings — broadcasts, regular and trigger mailings, multivariate tests, placements.**
- **Campaigns — combination of mailings, scenarios and other activities into a single strategy.**
- **Scenarios — automated chain of conditions and actions that follow when they are met.**
- **Tracking pixels — HTML or JavaScript code placed on a website or in an application, designed to track user actions.**
- **Forms — element on a website page through which the user enters and submits their data to you.**
- **Loyalty programs — module for managing promotions and promo codes.**
- **Virtual senders — set of rules for sending messages for real email and SMS sending modules.**
- **Market — module for working with e-commerce product and order data.**

User accounts

Management of Altcraft account user accounts is available both in the user interface and in the Admin Panel. This article covers the user interface functionality. To go to user account management, select Settings → Users in the main menu. The opened list displays existing user accounts.

Go to changing user settings by clicking on the record name. If you want to delete a user, click on the trash icon on the right.

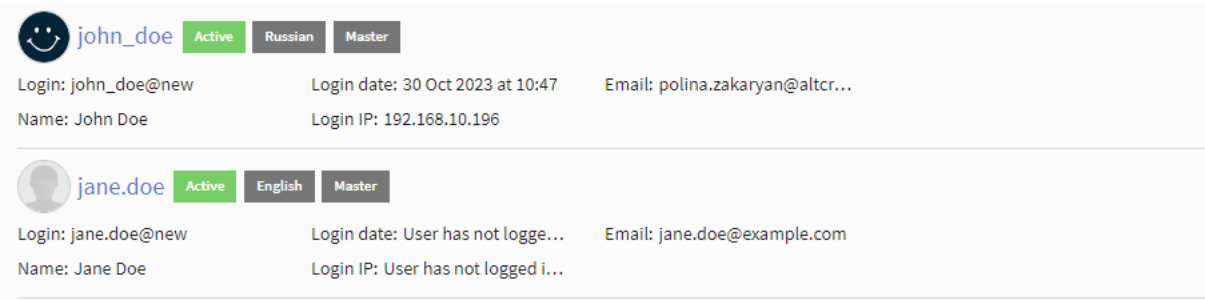
How to create a new user account

Roles can restrict user actions with the following platform objects:

- **Resources is a tool for managing customer subscriptions to various content via specific communication channels.**
- **Databases is a database for storing customer data.**
- **Database imports (scheduled import) is automatic synchronization of the Altcraft profile database with an external database (for example, corporate CRM or SQL database).**
- **Relations are used to record interactions between your clients, or clients and company managers.**
- **Queries are SQL queries to external databases. The role regulates user's actions with templates queries and with segmentation queries.**
- **Message templates is the content that you send to subscribers as part of the campaign.**
- **Fragments are parts of the template that can be used in other templates.**
- **Segments are part of a database that includes profiles meet certain conditions. The role regulates user's actions with static and dynamic segments.**
- **Suppression lists is a list of email addresses, domains and phone numbers excluded from the mailing list.**
- **Campaigns - broadcasts, regular and trigger campaigns, multivariate tests, placement campaigns.**
- **Scenarios is an automated chain of conditions and actions that will follow when conditions are met.**
- **Pixels is HTML or JavaScript code placed on a website or application that is designed to track user activity.**
- **Forms is an element on the page of the website through which the user input and sends you their data.**
- **Loyalty is a module that is used to generate and assign promotional codes to profiles.**
- **Virtual senders is a set of rules for real senders and a powerful tool for fine messaging tuning.**
- **Market is a module for working with data about products and orders of an online store.**

User profiles

Managing Altcraft user profiles is available both in User interface and in Administrative panel. This article contains information about the User interface procedure. To manage user accounts in the main menu select Settings —> Users:



The list you will see contains the existing account users. You can edit or remove a user in a context menu on the right side of the list. To edit a user you can as well click on the record's name. To create a new user click + Create in page header.

General settings

In general settings, specify the following data:

- **Username — account name. Username is used to log into the account, together with the account alias, in the form `login@account`.**
- **First and last name of the user.**
- **Contact email address — address to which access data will be sent.**
- **Contact phone — user's phone number.**
- **Interface language — choose between English and Russian.**

TIP

If platform account access is configured via LDAP protocol, through the organization's directory service, the directory service login and account alias will be used for platform authorization: user@company@account**. The password in this case is also set in the unified directory service.**

Notification settings

Enter the email address to which notifications will be sent to the user. Also select system notifications to be notified about:

- **Limit exhaustion — notification will be sent if the account tariff has limits configured, and the number of profiles or number of sent messages has reached the maximum;**
- **Subscriber complaints — the percentage of complaints about messages has exceeded the allowable value.**
- **Hard bounces — the percentage of hard bounce has exceeded the allowable value.**
- **Soft bounces — the percentage of soft bounce has exceeded the allowable value.**
- **Send daily/weekly reports — a report containing main mailing metrics, as well as data on audience growth and goal achievement, will be sent to the user's email with a certain frequency.**

Security settings

Set a password for the user to log into the account. The password must contain from 8 to 32 characters, including numbers and special characters.

After creating a new user, the password can be automatically transmitted to the owner by checking the Send access email to user option. To make editing this field possible, make any changes to the password. For example, you can add a period and immediately delete it.

If you activate the User must change password on first login field, the user will have to click the "Forget password" button upon login, enter the provided username and create a new password.

NOTE

If platform access is configured only via LDAP, password management in the interface is not used.

Access to account objects

In the Access block, it is configured which actions the user can perform with objects of certain Groups of the account:

- **Master — a user with this setting will have access to any actions in the account's Main group. Accordingly, they will have access to all objects of all groups. At the same time, role restrictions will not apply in any way. Master users have access to the Settings block in the main menu, where they can manage other user accounts: delete users, edit their access, download personal data, as well as change notification and security settings.**
- **Roles — previously created access roles.**
- **Access groups — groups of account objects that the user will be able to interact with.**

For a user to be able to log into the account, their account must be active, and at least one role and at least one group must be set for it.

The chart on the right shows the interaction of user groups and roles:

CAUTION

If a role does not have a group available to the user, the role rules will not work, and the user will not be able to interact with the group's objects.

If a role has a group that is unavailable to the user - it will remain unavailable.

After completing the setup, click the Save or Apply button at the top of the page:

The Deactivate button will make the account inactive. Users with this setting will not be able to log into the platform.

API tokens

API tokens are used for accessing certain platform functions via API. A master user can view, create, edit or delete tokens in Settings —> Tokens:

The opened list displays existing tokens. You can go to token editing by clicking on its name. If you want to delete a token, use the button on the right. To create a new token, click + Create at the top of the page.

For a new token, choose a suitable name or leave the system one. The token itself will become available after saving the record. For a token, as for a user account, it is necessary to set groups of account object visibility and related roles of access.

CAUTION

If a token is given access to the Main group, all account objects will be available to it, including those belonging to other groups.

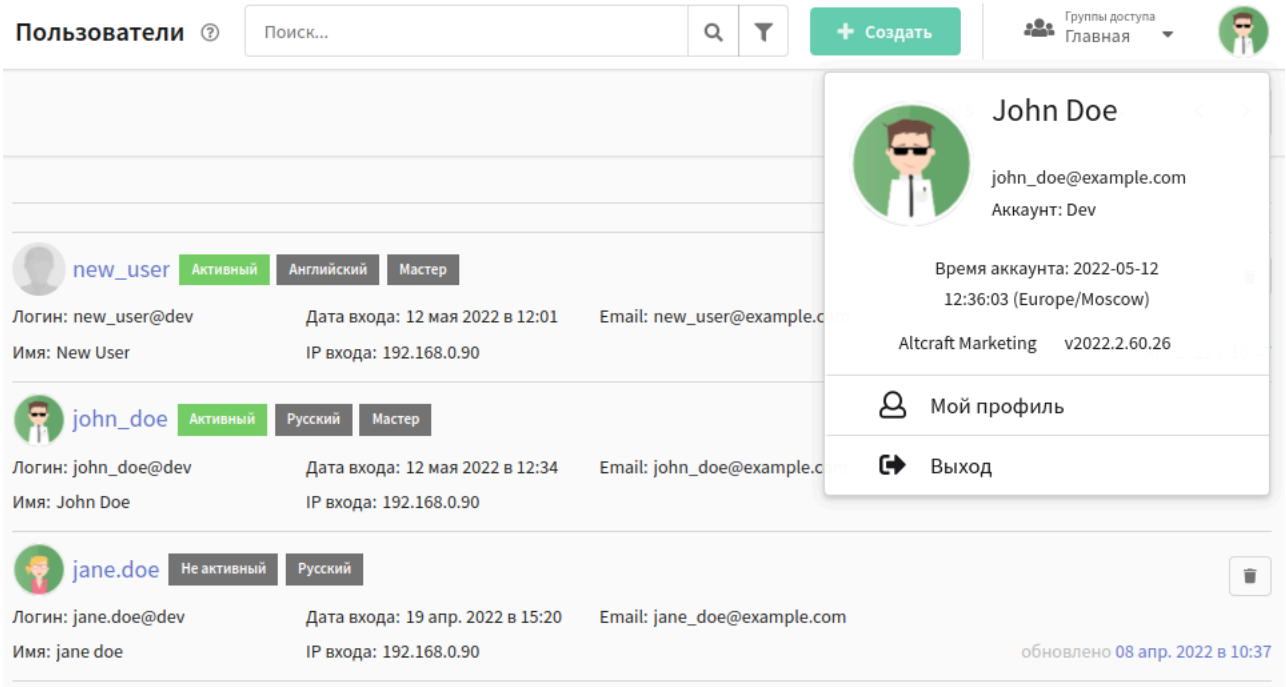
Two-Factor Authentication (2FA)

Description

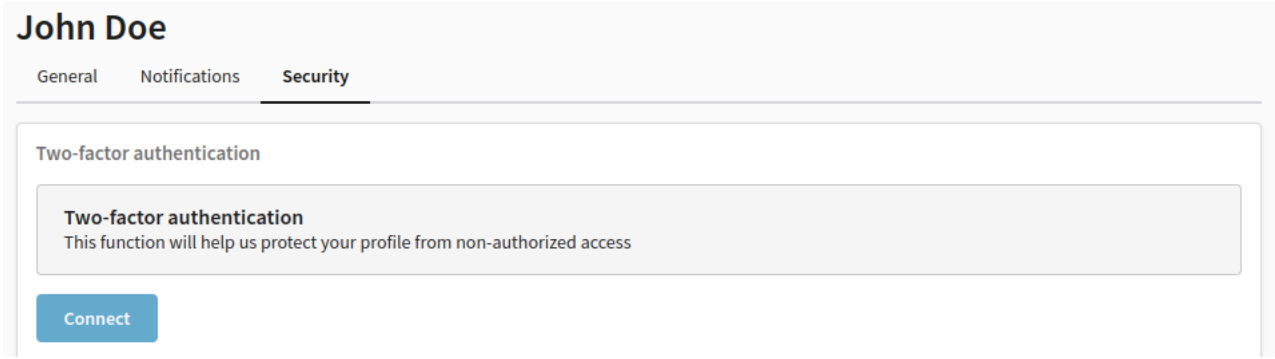
Two-factor authentication (2FA) is a security system that requires not only your username and password but also a special code sent to you. To authenticate, you will need a one-time code generator. For example, Google Authenticator. Once you've enabled 2FA, entering a six-digit code will become a mandatory step every time you log in to the platform.

How to Enable 2FA for an Individual User

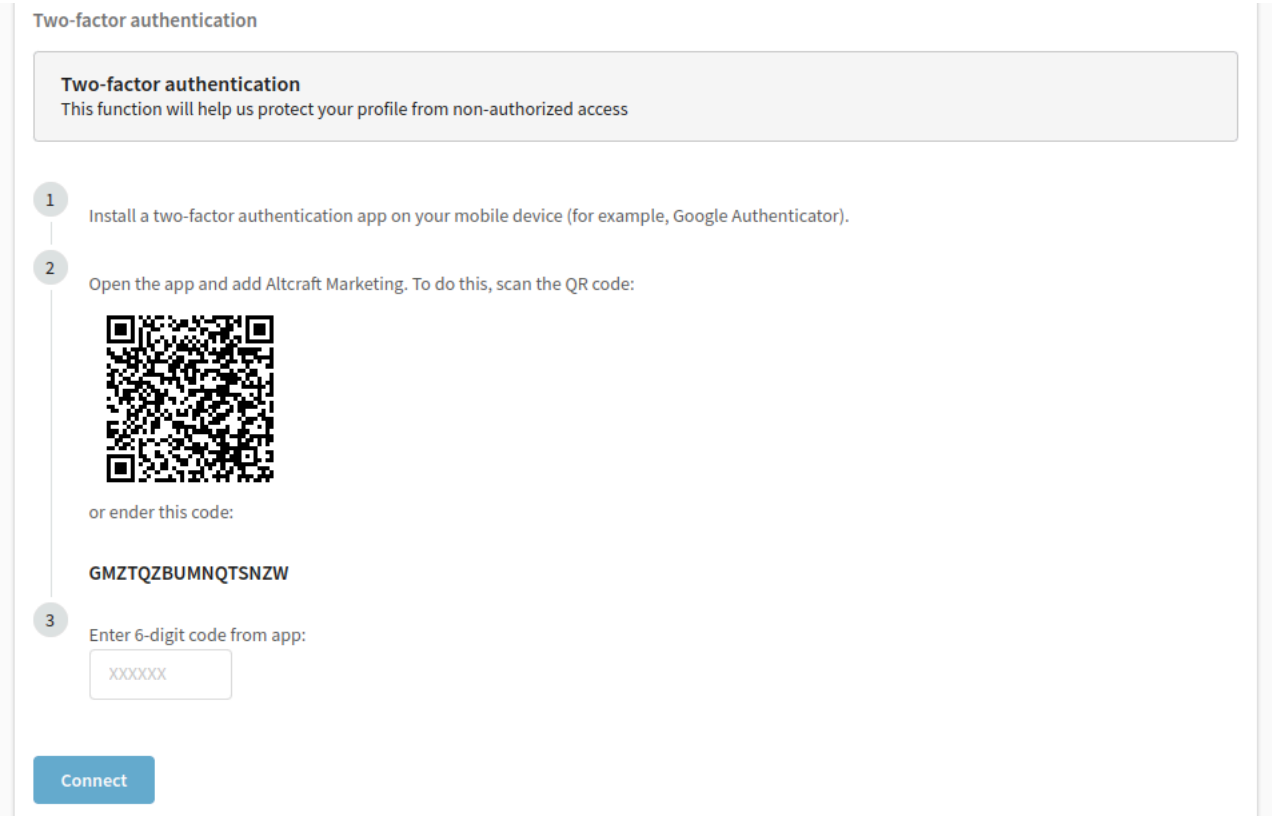
1. Go to your profile settings. To do this, click on the profile picture in the upper right corner. Next —> My profile .



2. Go to Security settings. In the Two-factor authentication block, click "Connect":



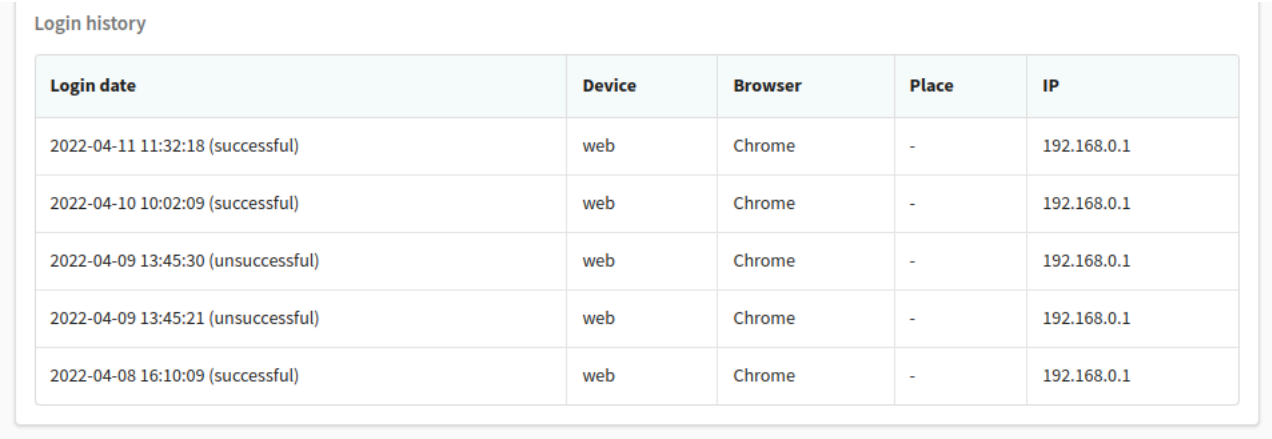
3. The instruction will open. Follow the steps: download the application, add Altcraft Marketing in it and enter the six-digit code sent to you. Then click "Connect".



4. A list of recovery codes will appear on the page. Use them to restore access to your account in case of loss of the device on which the 2FA application is installed. Each recovery code can only be used once. It will be impossible to open this list again, so copy it and store it in an accessible place.

The 2FA setup is complete. Now, every time the user logs in to the platform, they will have to enter not only their username and password but also a six-digit code sent to them in the authentication application.

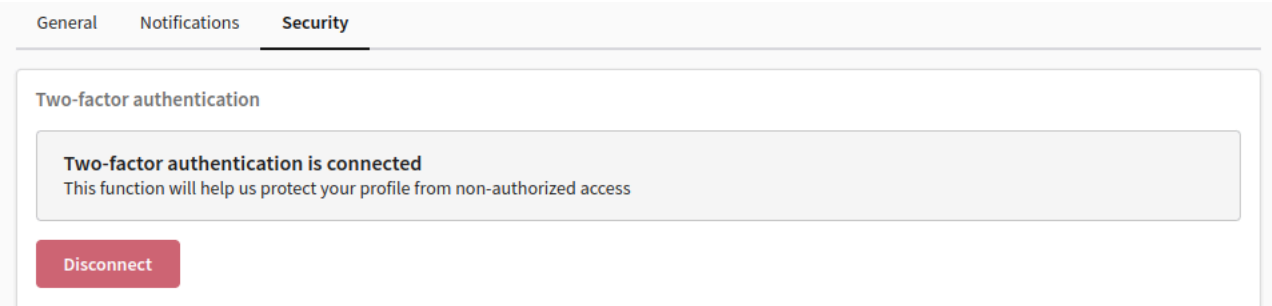
Information about successful and failed login attempts will be displayed in Login history:



Login history

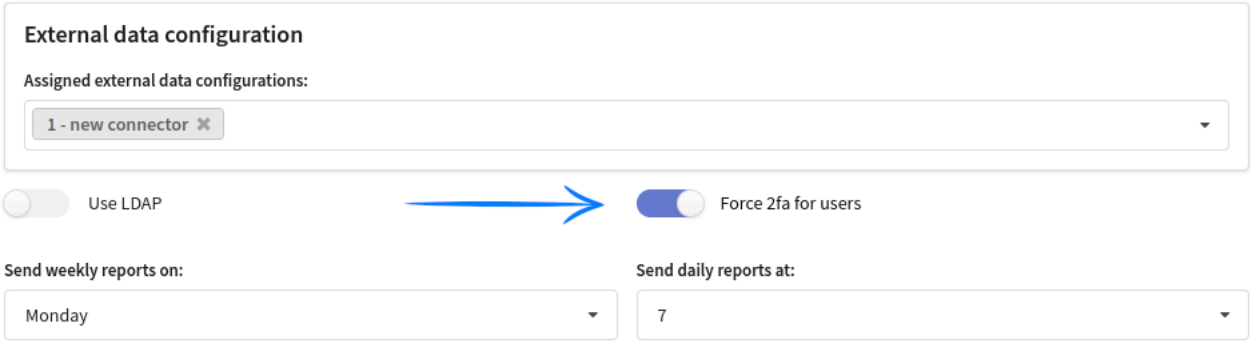
Login date	Device	Browser	Place	IP
2022-04-11 11:32:18 (successful)	web	Chrome	-	192.168.0.1
2022-04-10 10:02:09 (successful)	web	Chrome	-	192.168.0.1
2022-04-09 13:45:30 (unsuccessful)	web	Chrome	-	192.168.0.1
2022-04-09 13:45:21 (unsuccessful)	web	Chrome	-	192.168.0.1
2022-04-08 16:10:09 (successful)	web	Chrome	-	192.168.0.1

If you no longer need two-factor authentication, click on “Disconnect”:

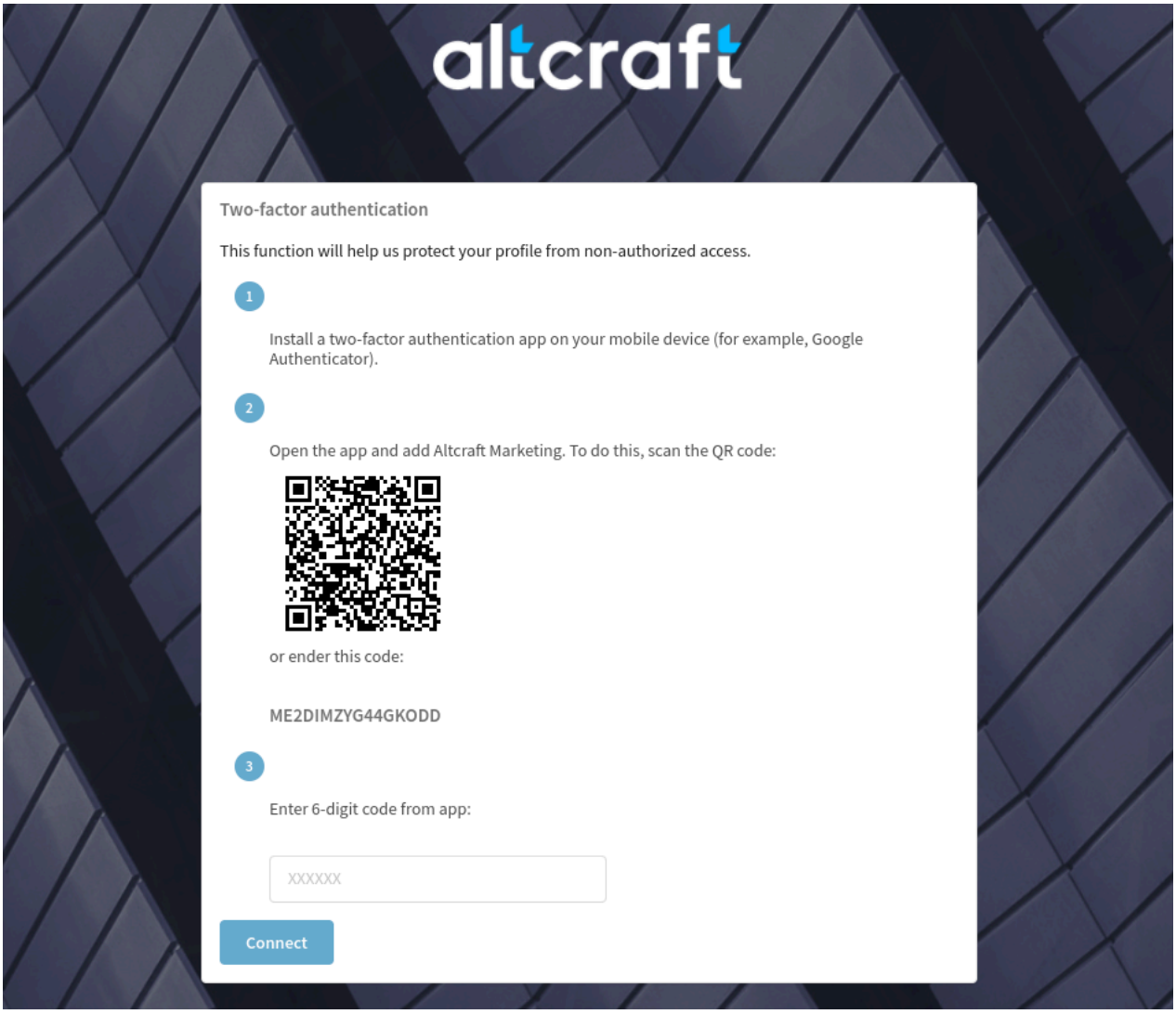


How to Enable 2FA for All Account Users

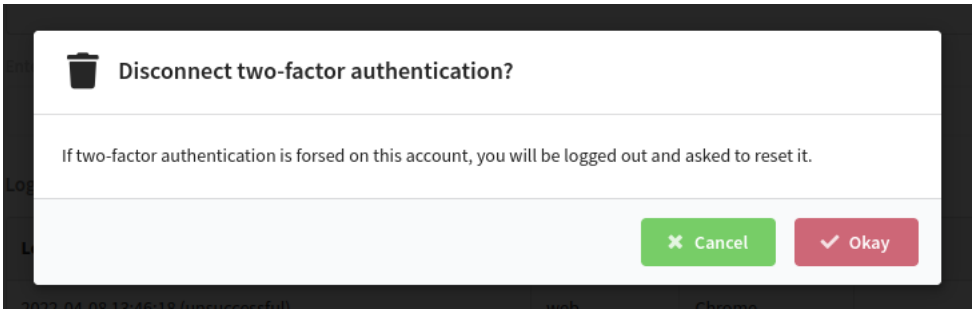
To enable mandatory two-factor authentication for all accounts, the platform administrator must activate the Force 2fa for users in account settings in [the Administration Panel](#):



Once you've activated mandatory 2FA, any user will have to set up two-factor authentication to log in to the platform. The setup window will open when the user enters the correct username and password:



If two-factor authentication is mandatory for all accounts, the user won't be able to disable it. If they click on "Disconnect" a warning window will pop up:



If, despite the warning, the user disables 2FA, he will be logged out from the platform. To log in, he will need to reconfigure 2FA.

Connections

Description

Connections let you authorize in external systems to expand Altcraft Marketing capabilities with the functionality of these systems. You usually need an account in an external system with API access rights to create such a connection.

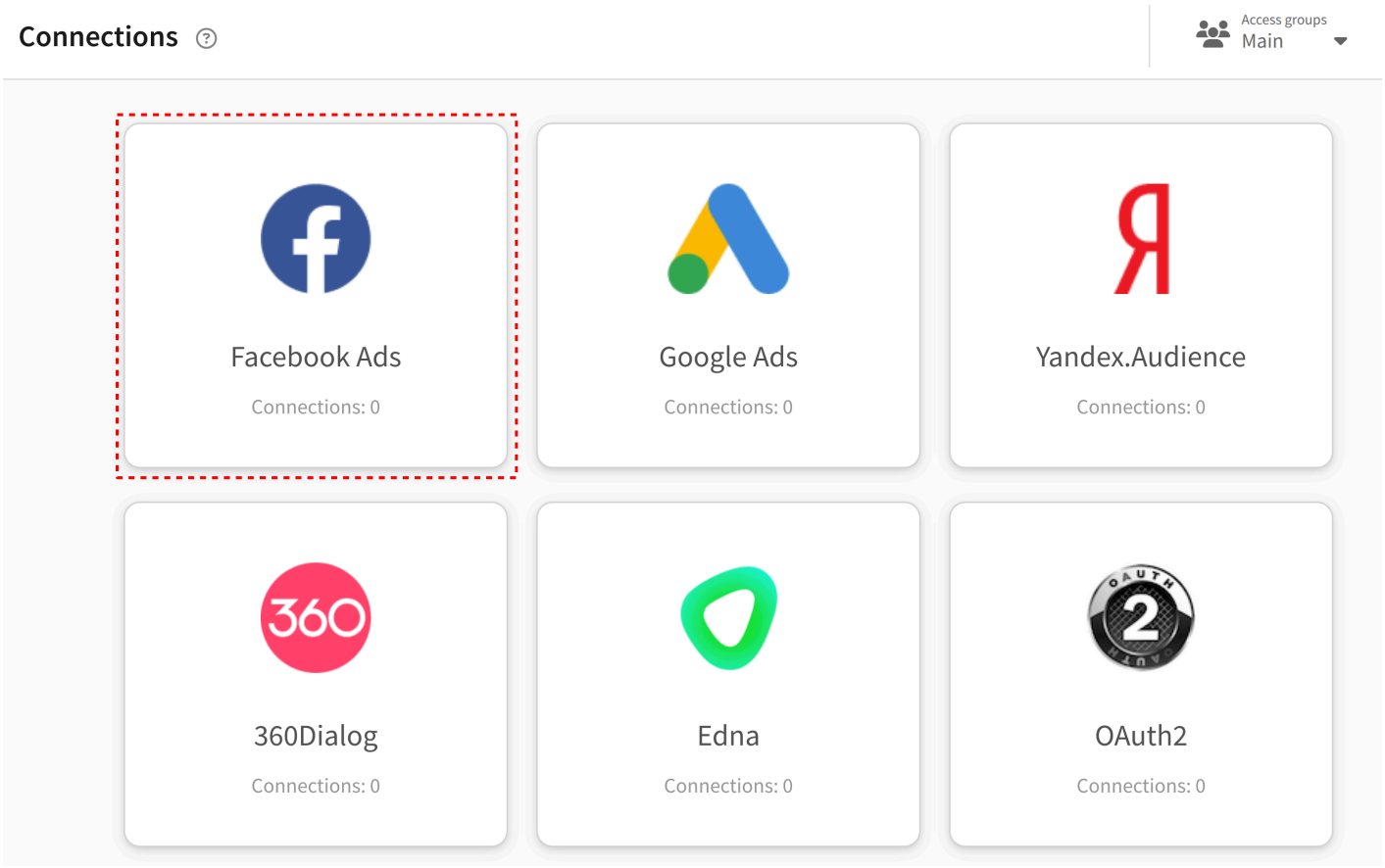
Connection settings

In the main menu click “Settings” —> “Connections”.

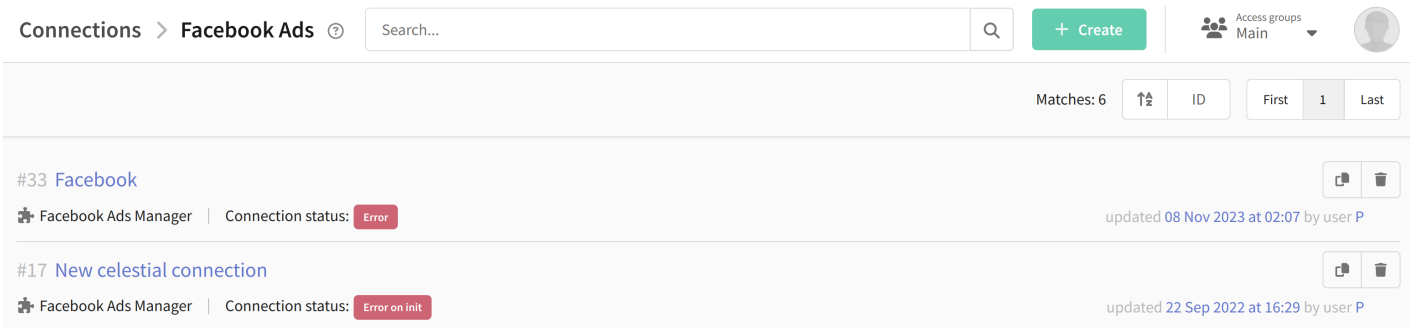
Available connections

Connection to Facebook Ads Manager

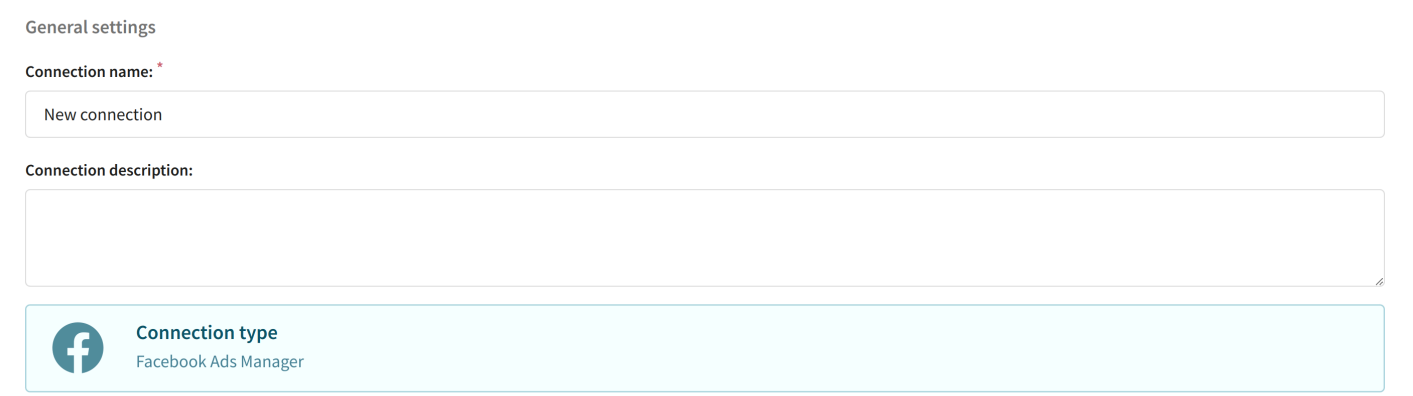
In the main menu, go to Settings —> Connections. Select connection type — *Facebook Ads.



A list of all configured Facebook* connections will be shown. To add a new one, click + Create:



In the general settings enter the name and the description of the new connection and select the connection type.



You need Facebook* application data to connect to Facebook*. The platform will interact with the Facebook* Marketing API via the Facebook* app.

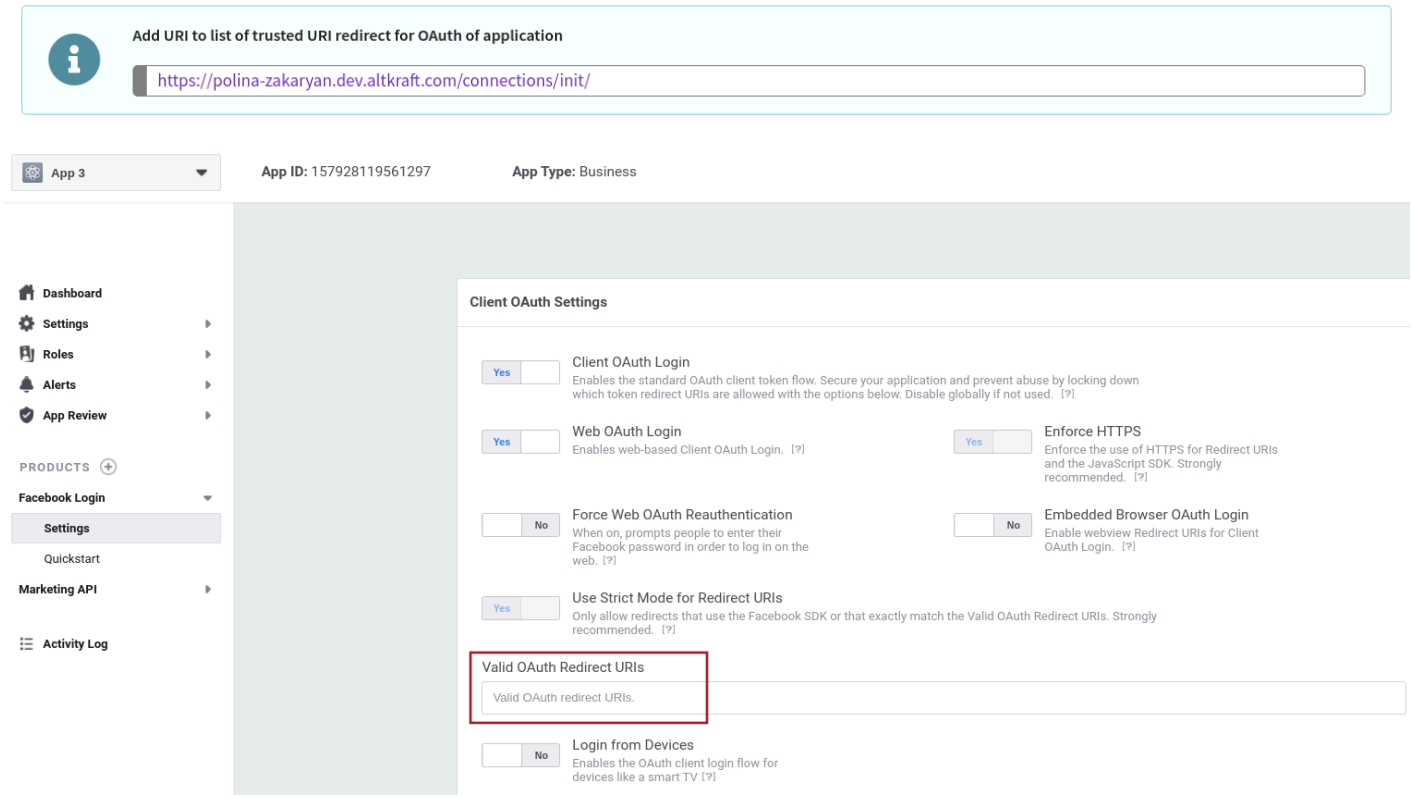
You can create this app on the page https://developers.facebook.com/apps/.

The app you need to create should have the type “Business”. You can read the documentation here: https://developers.facebook.com/docs/development/create-an-app

First of all, find the app panel and go to the Products section. Here you need to add:

- **API Marketing**
- **Facebook* Login**

In the settings of the *Facebook Login product, you need to add URI to list of trusted URI redirect for OAuth of application:**



CAUTION

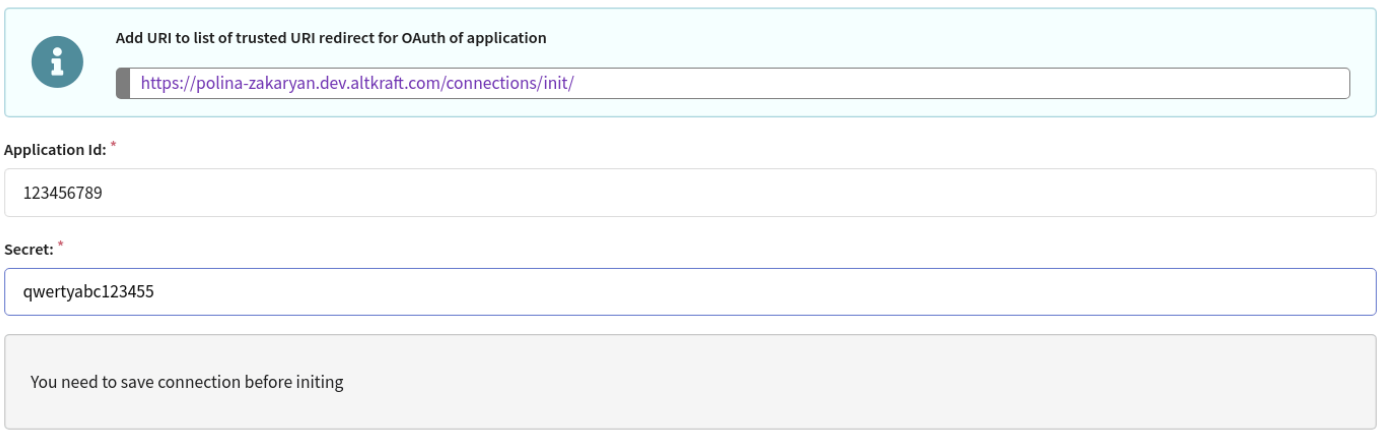
The app must have ads_management rights. In most situations, standard access will be enough (see https://developers.facebook.com/docs/marketing-api/overview/authorization/)

In the Altcraft platform, the configuration consists of the following steps:

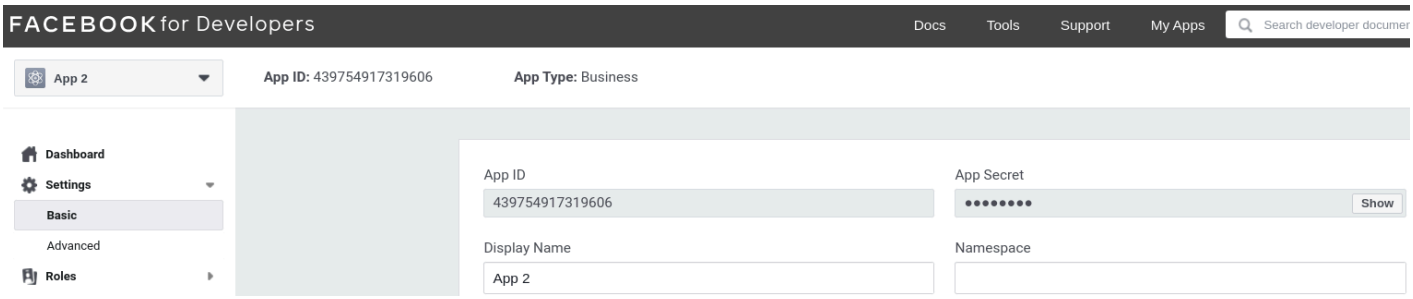
- **Pre-configure Facebook* connection**
- **Connection initialization**
- **Choose an ad account**

Pre-configure Facebook connection

Before initializing the connection, you need to fill in the fields of the following form.

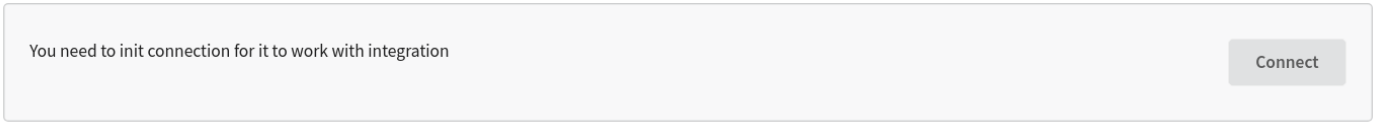


Application ID and Secret can be found on the panel app in the settings:



You need to save the connection and only then continue.

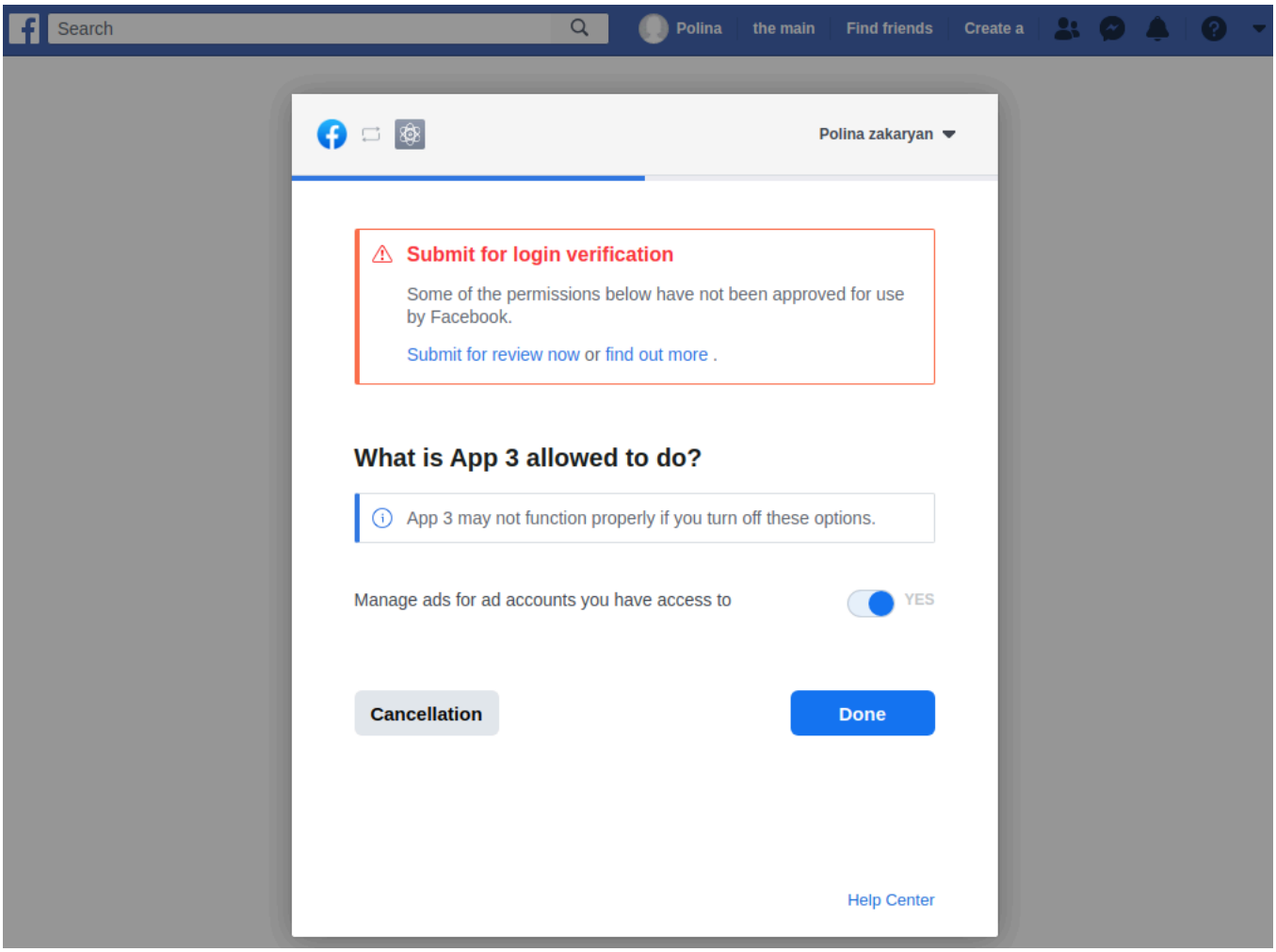
After that, you'll see the Connect button:



Connection initialization

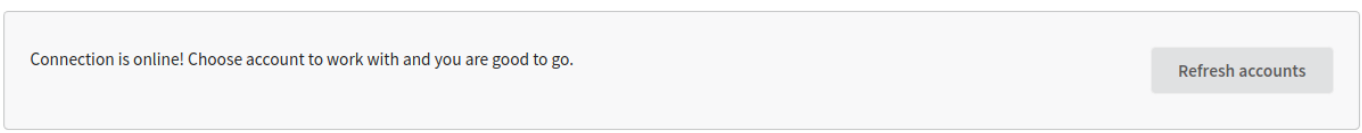
When you click on the Connect button, you will be redirected to Facebook* and then you will get an offer to log in. You need to log in using a profile that has the rights to manage mailings in the target ad account.

After that, the rights to manage ads for your app on behalf of the profile and with profile rights will be requested.



Choose a Facebook ad account

If you've done with the previous steps without errors, then you just have to select an advertising account that you can use to interact with this connection.



When you select an account, you can get some errors. For example, if you haven’t accepted the terms of service yet. In this case, you must accept the terms of service and update your accounts.

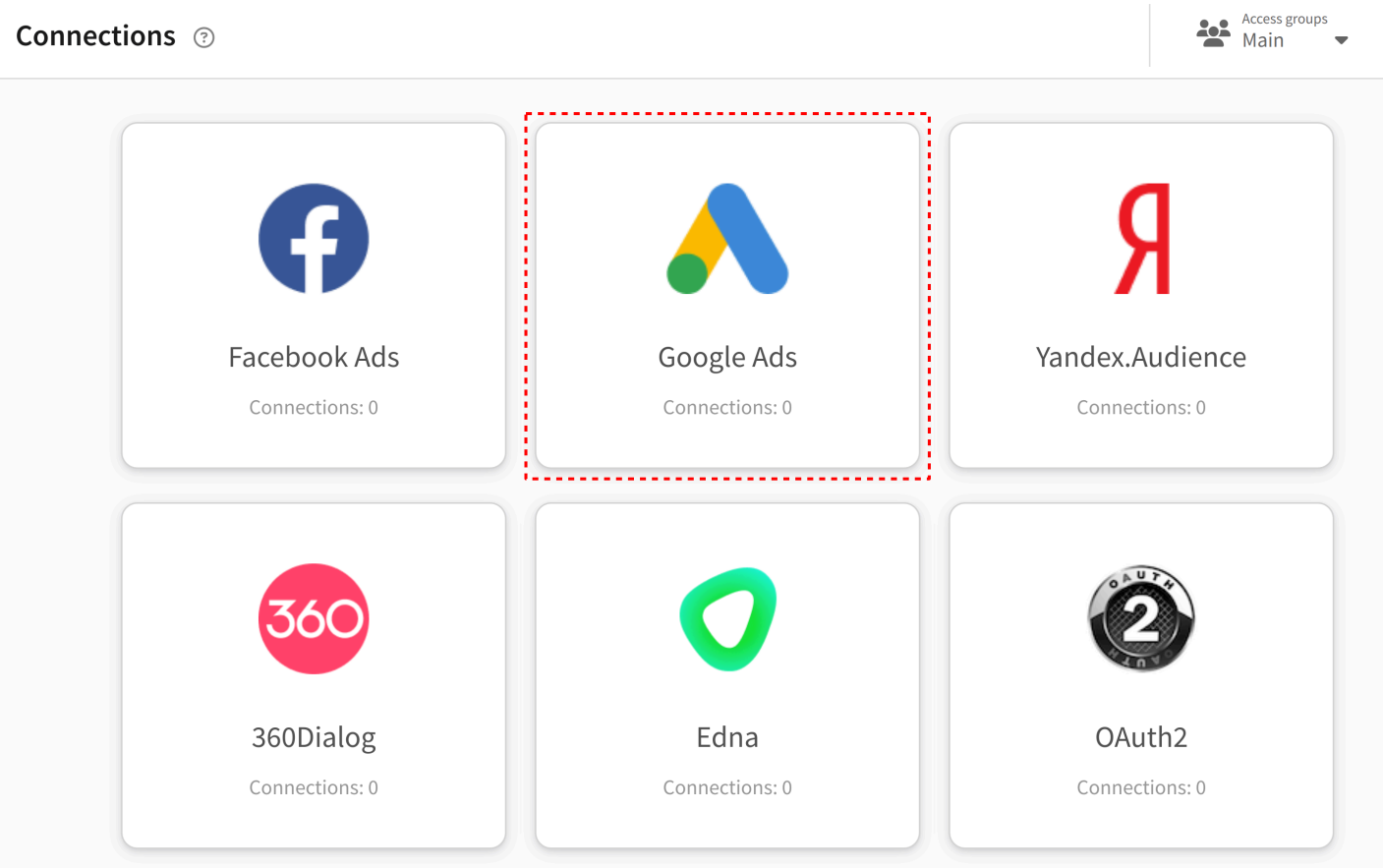
****The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.***

Connection to Google Ads

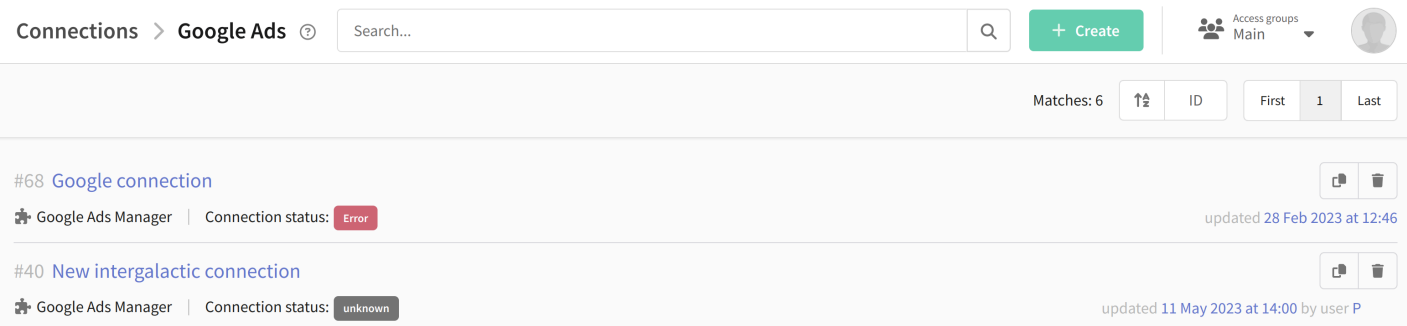
Google Ads is an advertising service by Google for businesses that want to display ads on Google search results and its advertising network (Gmail, YouTube, etc).A connection links your Google Ads account to Altcraft Platform. You can then use this connection to set up your Google Ads integration and sync your Altcraft static segments with your Google Ads audiences.

How to create a connection

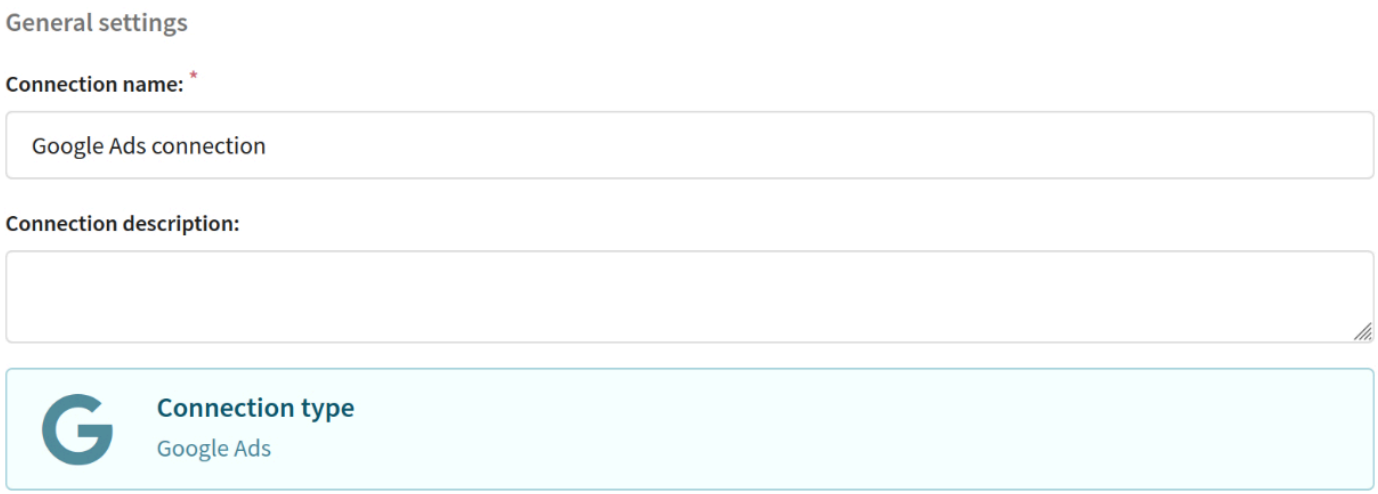
In the main menu, go to Settings —> Connections. Select connection type — Google Ads:



A list of all configured Facebook* connections will be shown. To add a new one, click + Create:



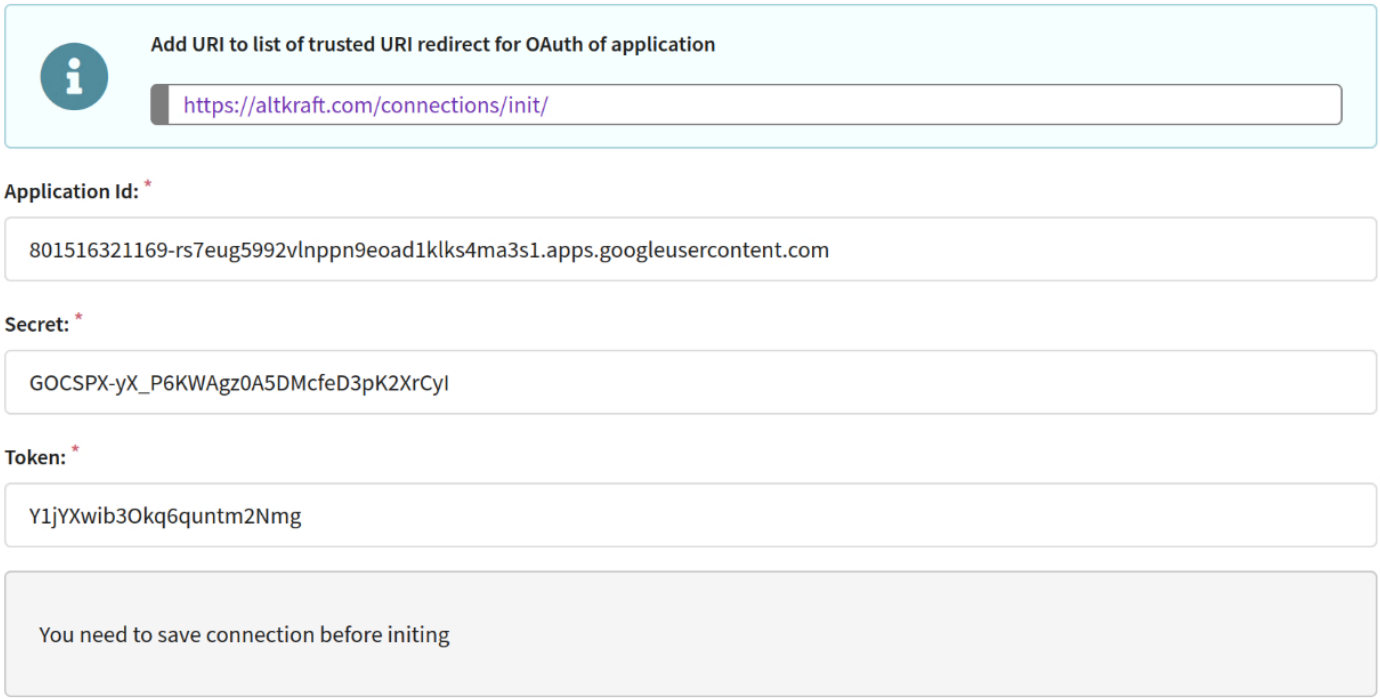
In the general settings, specify the name of the connection and, if necessary, its description:



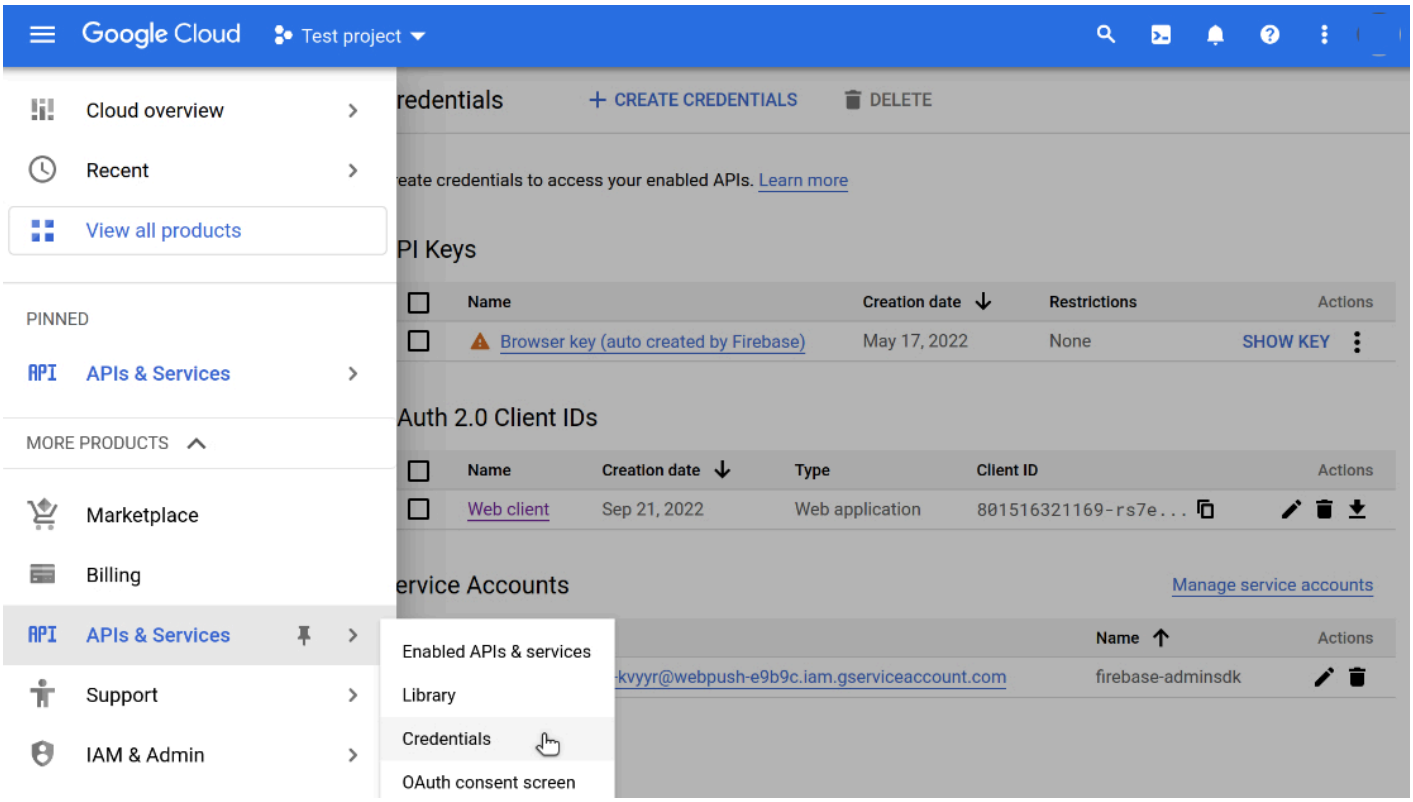
Step 1. Pre-configure Google Ads connection

Connecting to Google requires data from a Google application through which the platform will interact with Google Ads. If you don't have an app yet, create one by following the link: https://console.cloud.google.com/.

Enter application data into the platform:



You can find the App ID and Secret in Google Cloud. To do this, go to the APIs & Services —> Credentials section and click on the required OAuth 2.0 Client ID to view the data:



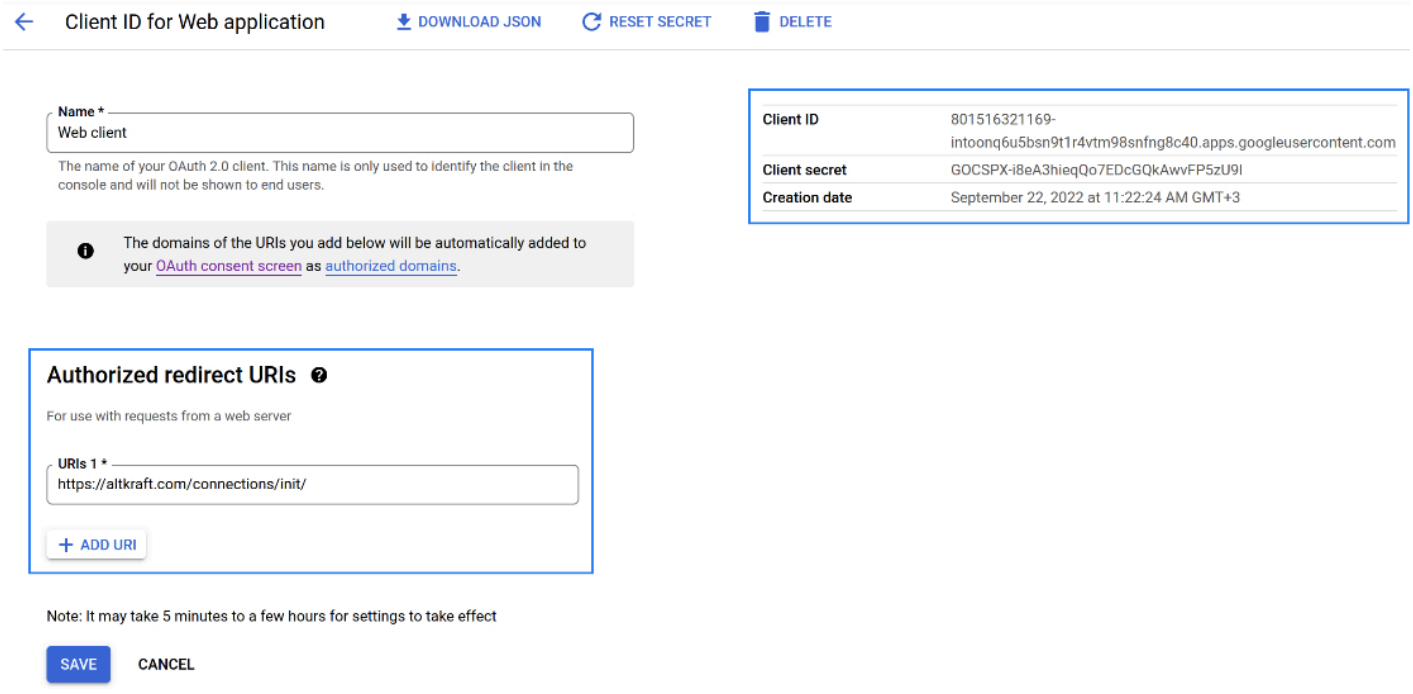
TIP

To create new OAuth 2.0 Client IDs, click the "Create credentials button at the top of the page. Select OAuth 2.0 Client ID and select Web application as the application type.

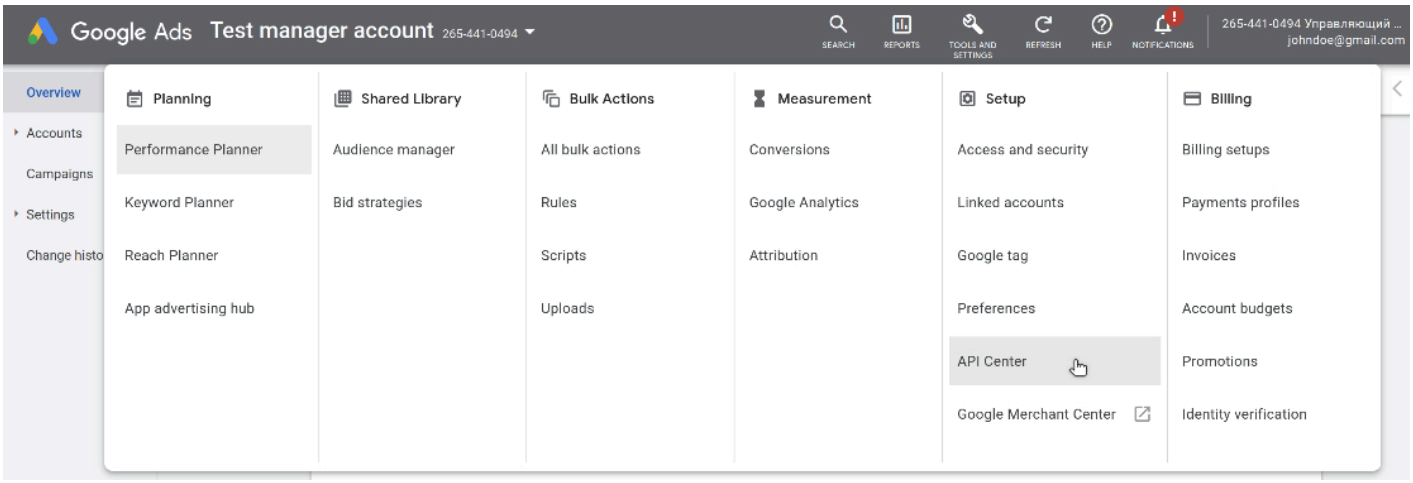
When viewing the created OAuth 2.0 Client ID, in the right block you will find:

- **Client ID — application ID,**
- **Client secret — secret.**

In the Authorized redirect URIs block, add the URI you received in Altcraft Platform when setting up the connection. Don't forget to save your changes.



To get a token, sign in to your Google Ads manager account. Next, go to Tools and Settings —> API Center and copy your Developer token.



TIP

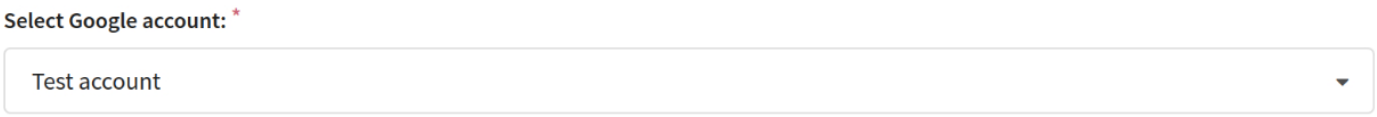
Read more about obtaining a token in the documentation.

Step 2. Initialize Google connection

Once you have entered the Google App data into the platform, save the connection by clicking Apply". After that, the "Connect" button will appear. When you click on it, you will be redirected to Google and asked to log in. Log in to confirm access rights for Altcraft Platform.

Step 3. Choose a Google Ads account

If the previous steps completed without errors, select the ad account you want to interact with through the created connection:



Connection statuses

Status	Description
Ready to work	**The connection has been successfully configured and can be used to upload static segments to Google Ads.**
Connecting	**Connection in progress. A possible reason is that when you set up the connection, you did not log in to Google and did not grant rights.**
Connected	**The connection is not fully configured, additional steps are required. For example, you need to select a Google account.**
Error	**An error occurred while working with the API that requires reconnection. In this case, you need to make sure that the user has access rights.**
Error on init	**An attempt to set up a connection failed.**

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as extremist and banned in Russia.

Connecting to Yandex.Audience™

Connecting to Yandex.Audience is required to configure the corresponding integration.

Step 1. Register a web application in Yandex

`https:///connections/init/`, where `` should be replaced with the URL you use to access the Altcraft platform.	

TIP: Application already exists

If you already have an application created in Yandex.OAuth, simply update its settings according to the pa

Step 2. Create a connection in the Altcraft platform

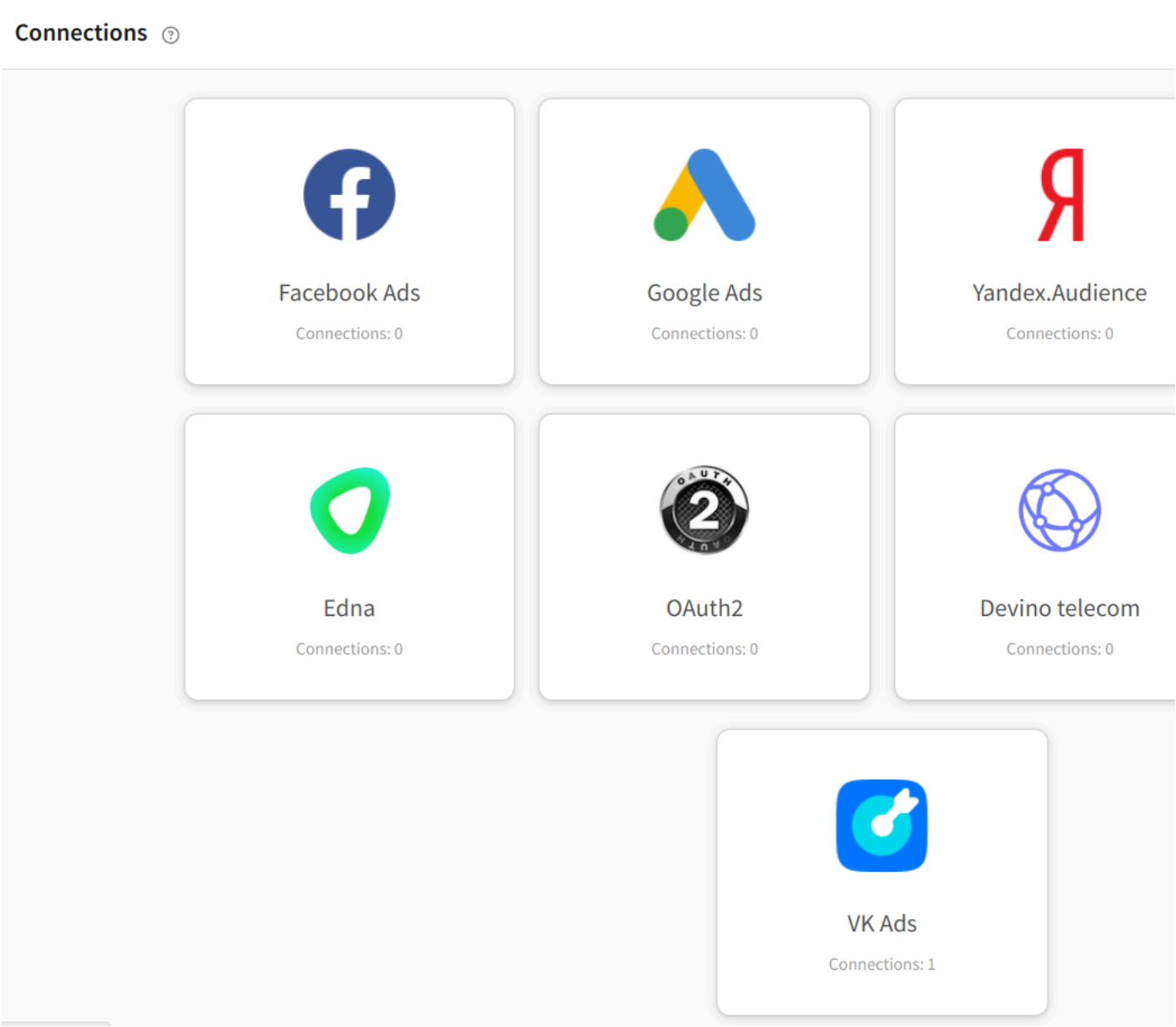
Yandex.Audience integration.

Connection to 360dialog

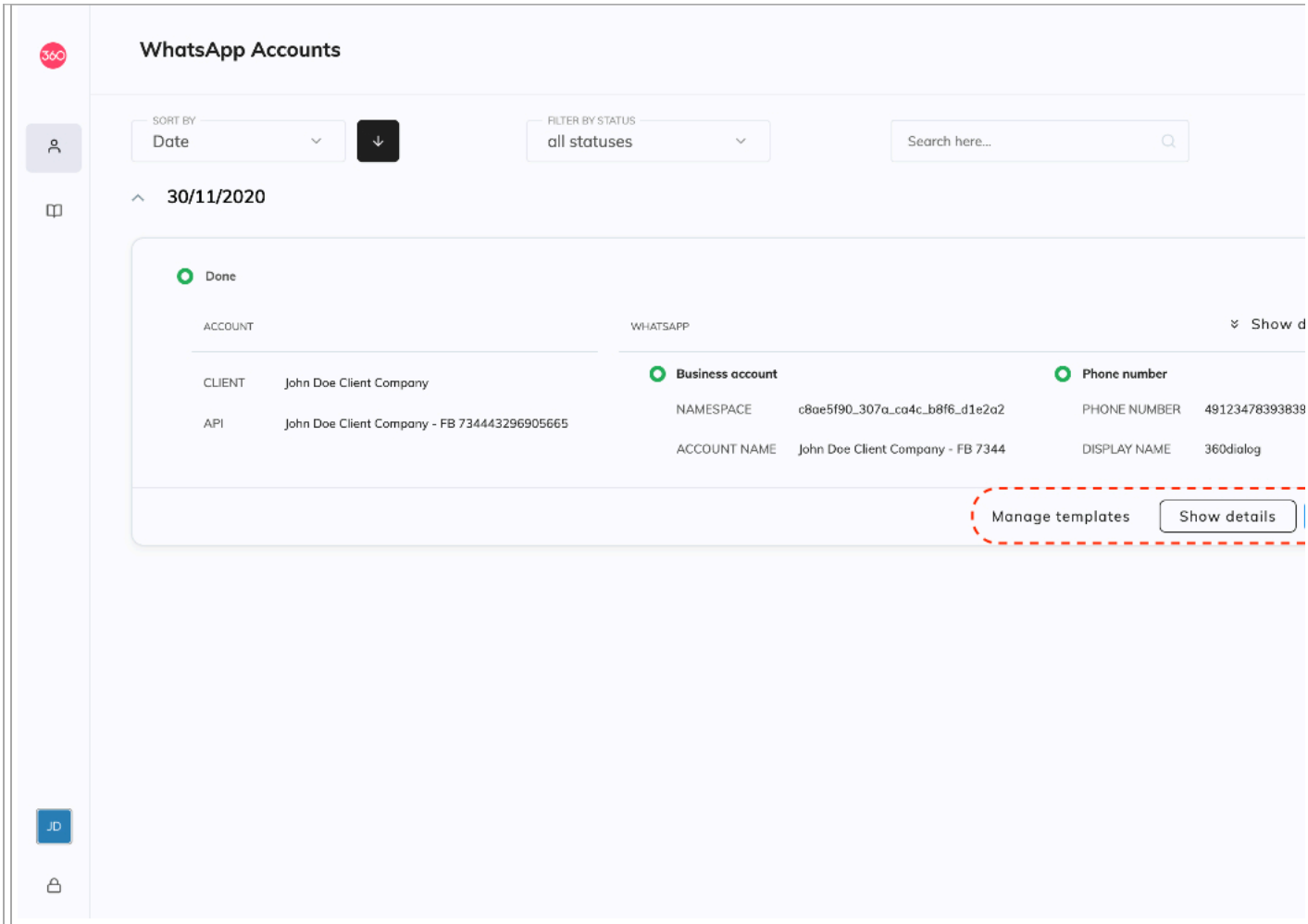
To integrate the platform with WhatsApp*, you need to configure a connection to one of the WhatsApp* B

> **TIP**
> **First, you need to apply for an account registration with 360dialog through this form.**

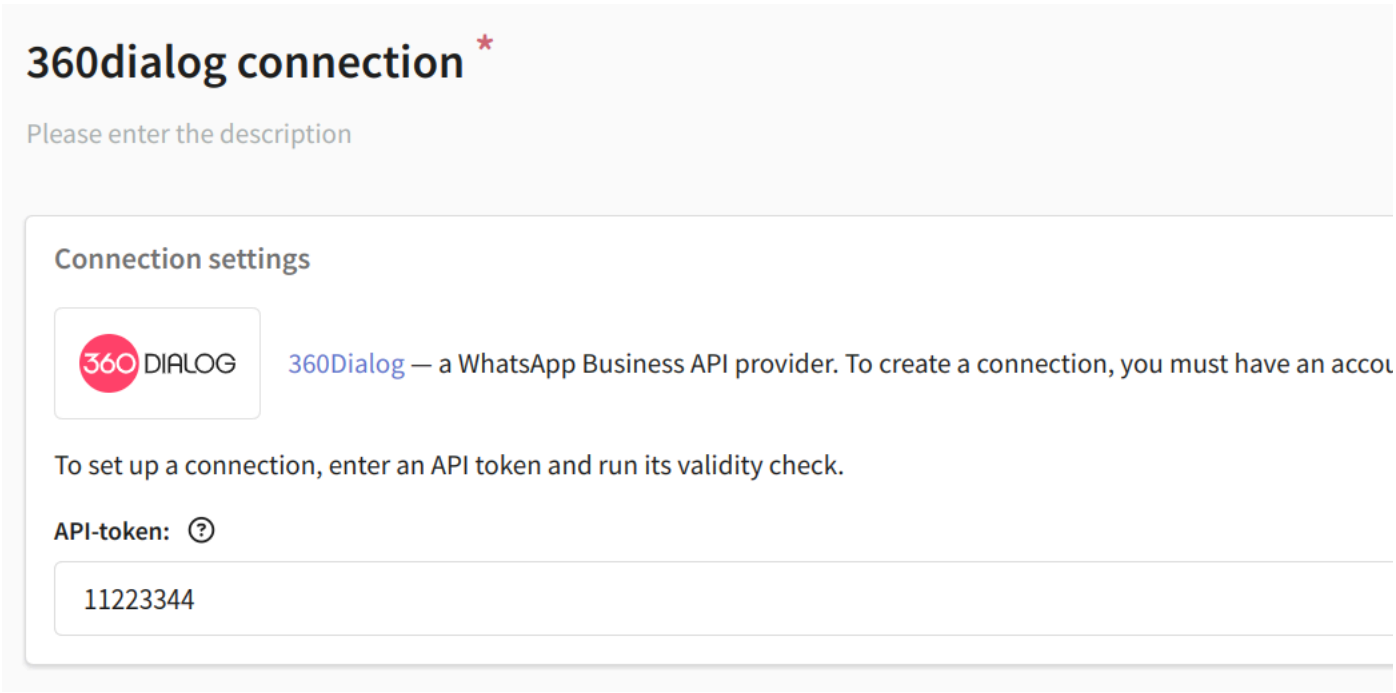
Once you have completed your registration, go to the Altcraft platform.
In the main menu, go to Settings —> Connections. Select connection type — 360dialog. Click the “Create”



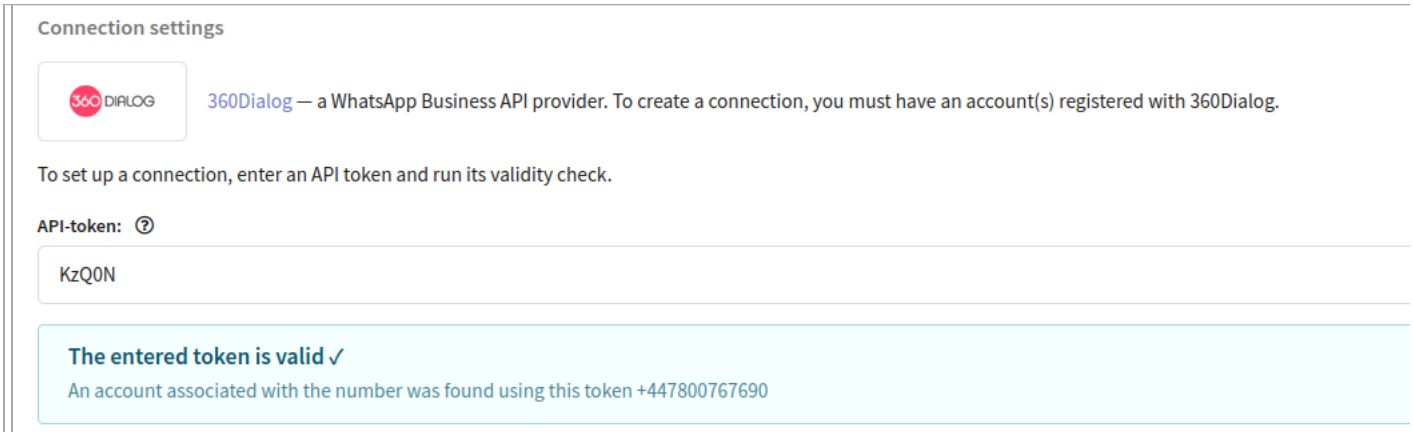
You’ll need an API token from 360dialog. After completing the registration in 360dialog, you will automatic
If you accidentally closed this window, you can generate a new API key. To do this, go to the *WhatsApp A*



Enter the API token generated in the provider's account and click on the Check button to confirm its valid



If the token has been verified, you will see which phone number the connected account is linked to:



Specify a name and description for the connection. Save the connection by clicking on the Save button on

---

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Connection to Edna

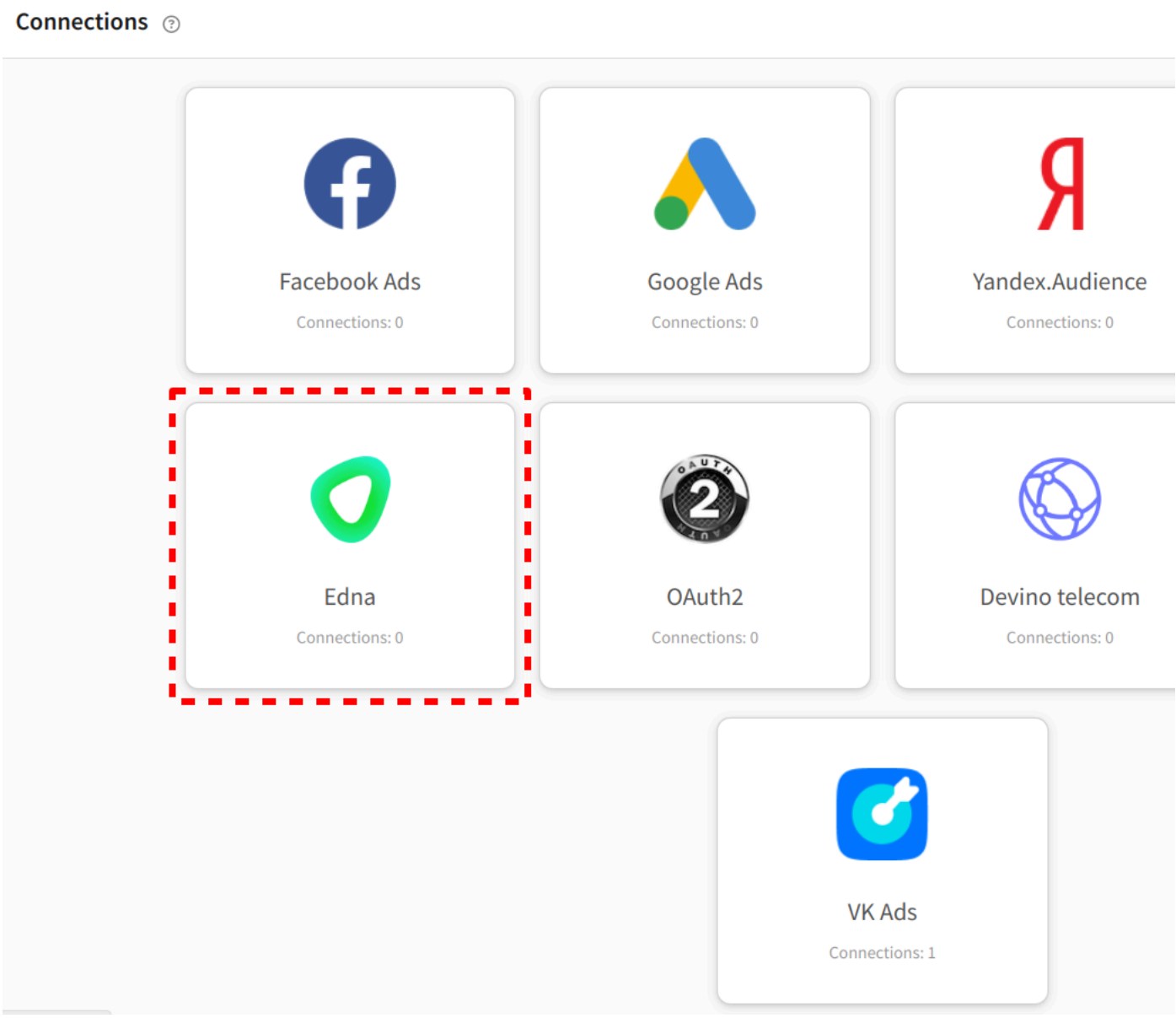
To integrate the platform with WhatsApp* Messenger, you need to configure a connection to one of the W

TIP

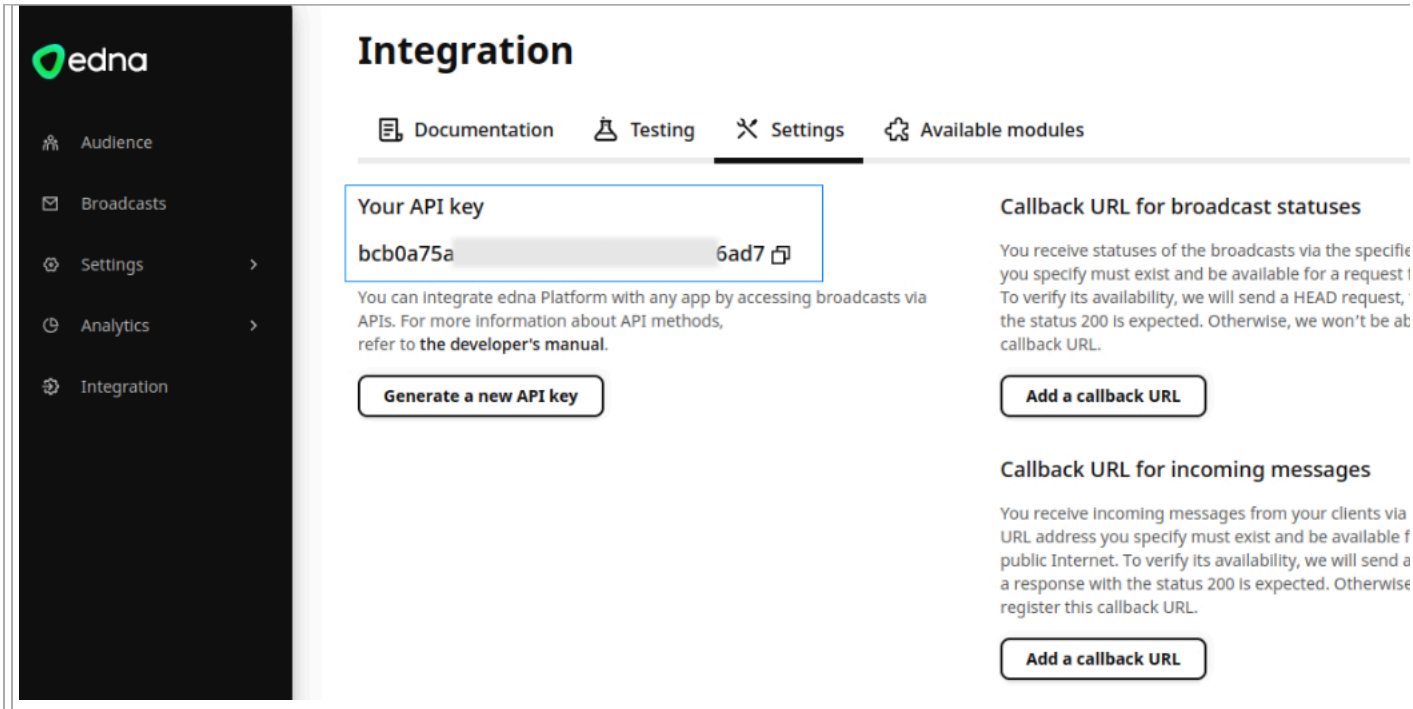
First, you need to apply for an account registration with 360dialog through this form.

Once you have completed your registration, go to the Altcraft platform.

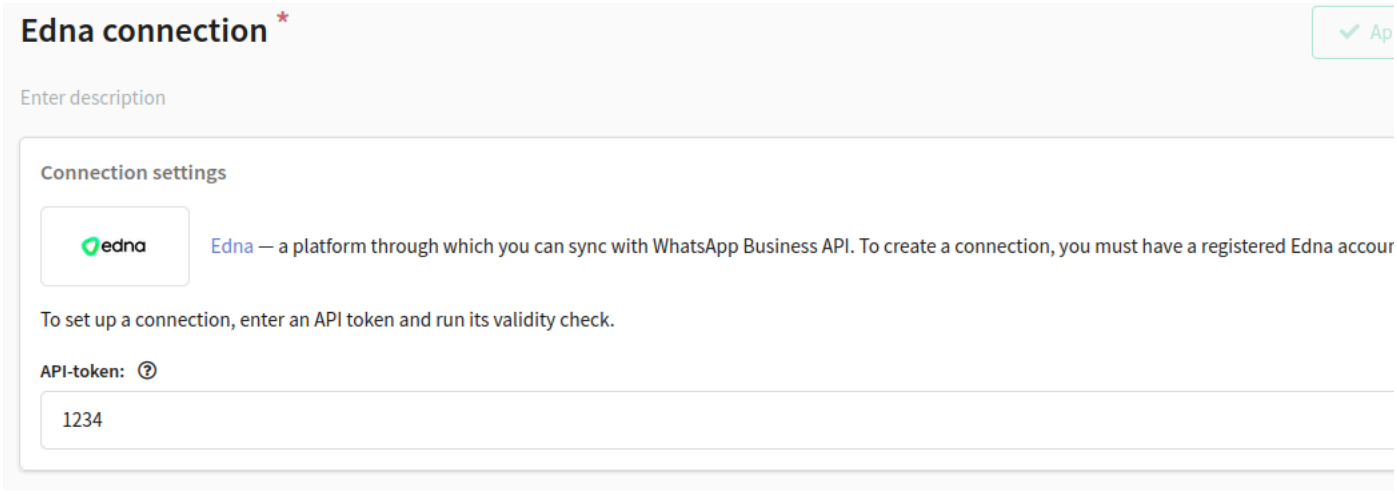
In the main menu, go to Settings —> Connections. Select connection type — Edna. Click the “Create” butt



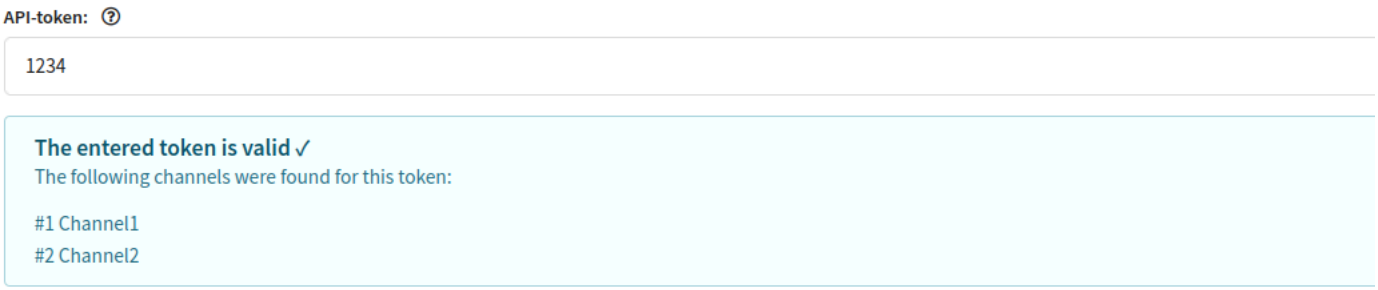
You’ll need an API token from Edna. In your account, go to Integration —> Settings. Copy the API key.



Enter the API token generated in the provider's account and click on the Check button to confirm its valid



If the token has been verified, you will see information about available WhatsApp* channels in your accou



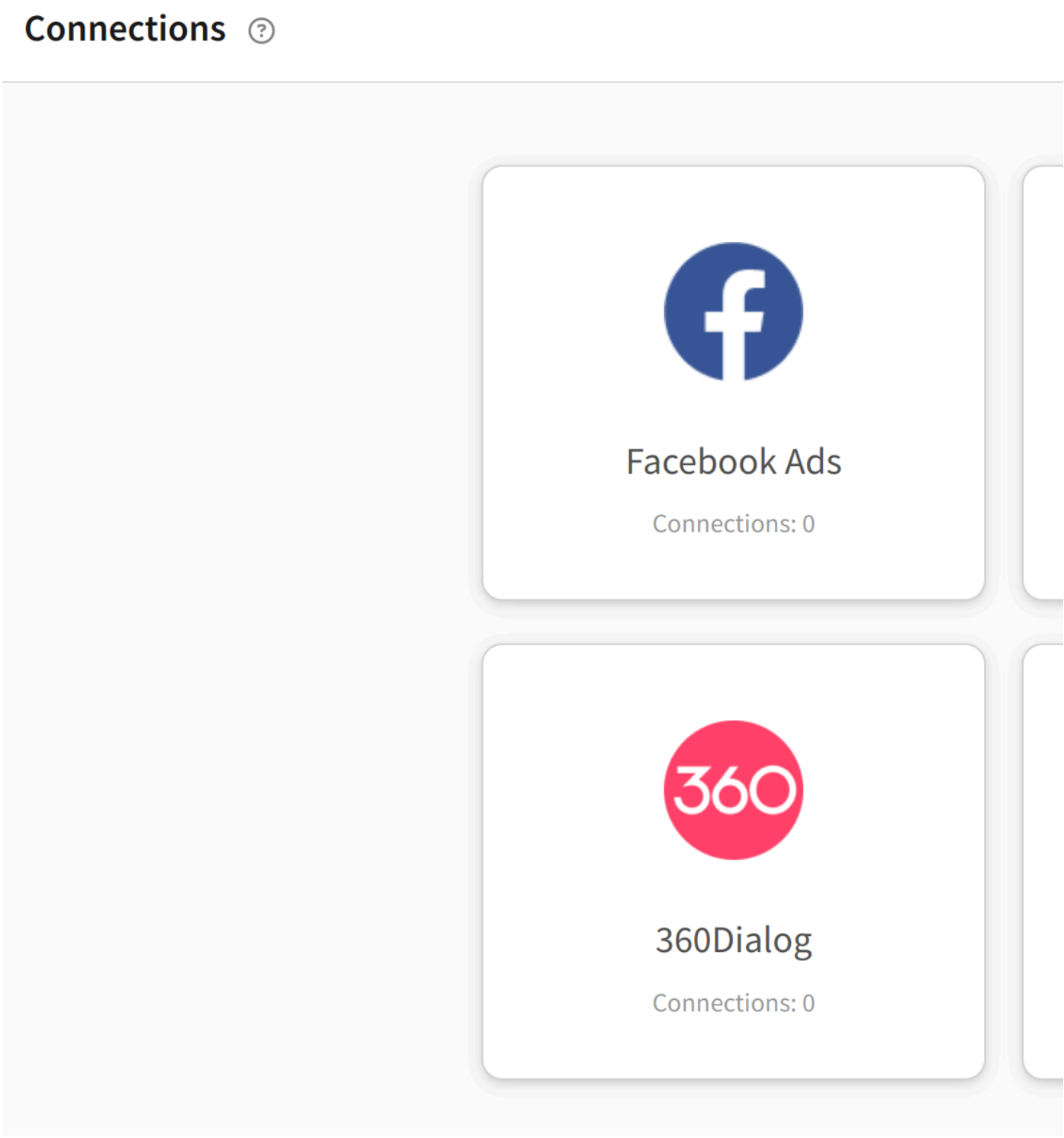
Specify a name and description for the connection. Save the connection by clicking on the Save button on

---

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as

Connection to Devino Telecom

To integrate the platform with Viber*™, you need to configure a connection to one of the providers. This a
To get started, register an account with your provider and then go to Altcraft Platform.
In the main menu, go to Settings —> Connections. Select connection type — Devino Telecom. Click the Cr

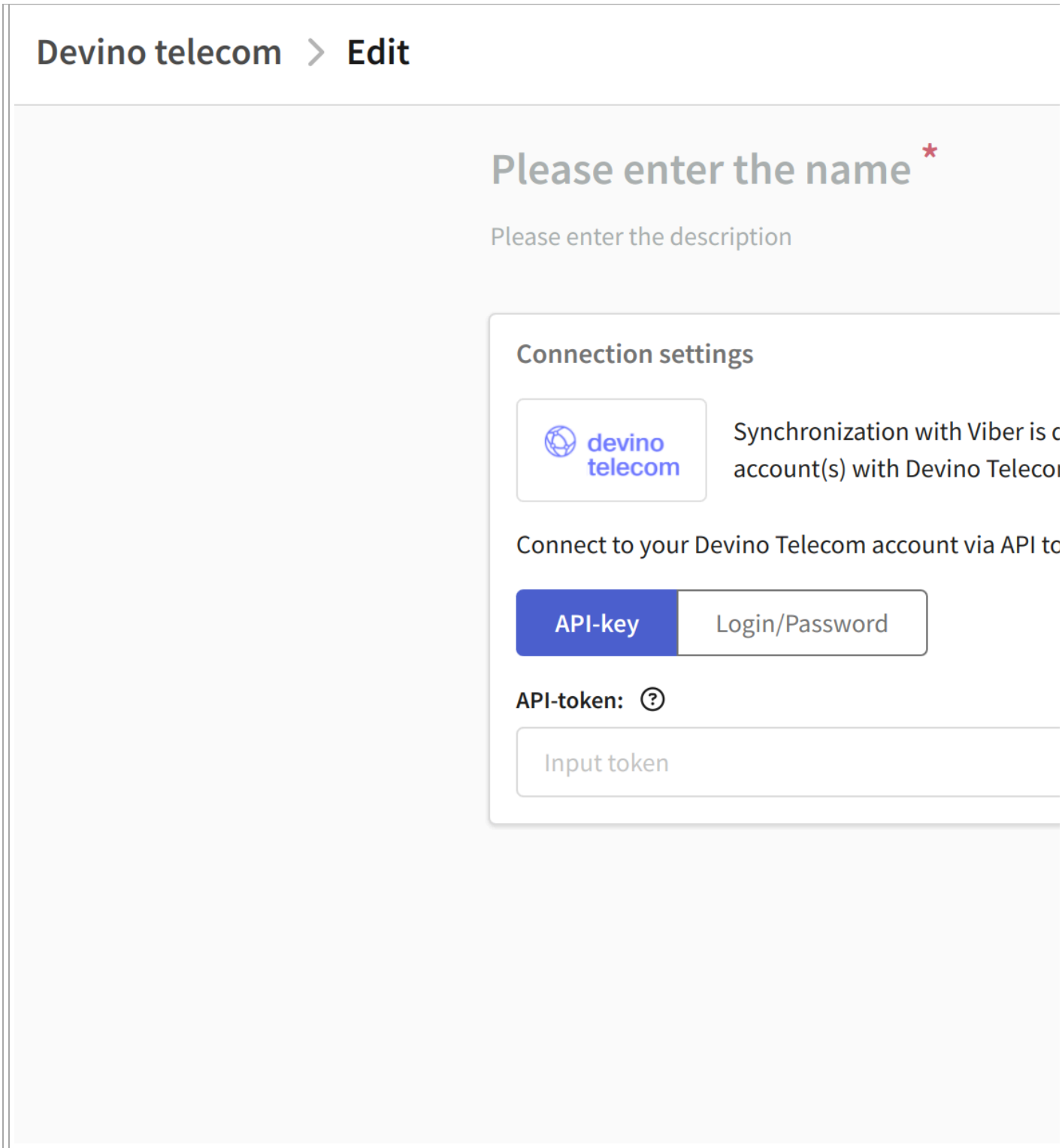


Specify a name and description for the connection. Next, choose the authorization method that suits you b

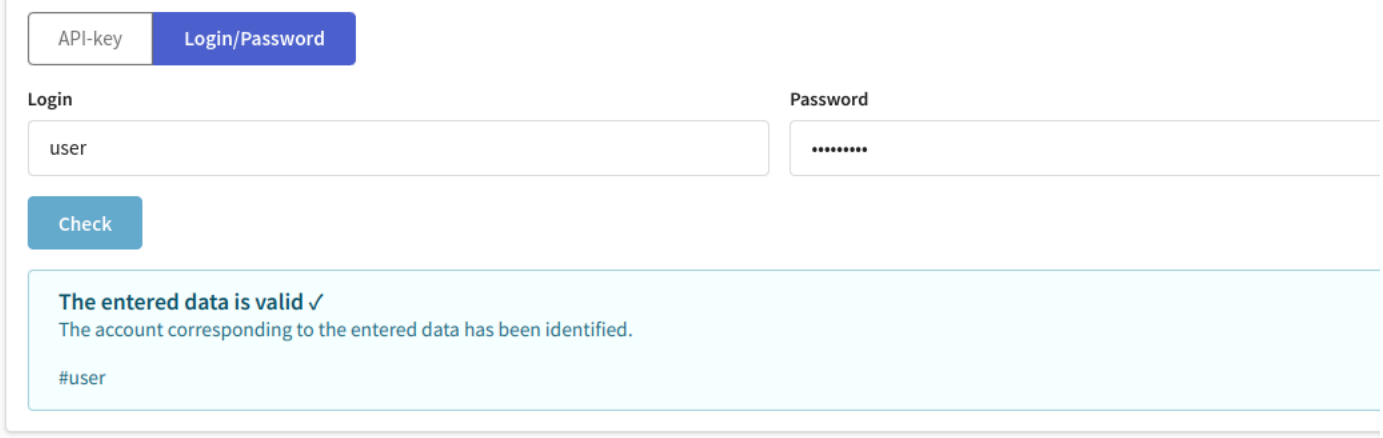
- **API Key — Using a generated API key.**
- **Username and Password — A combination of your Devino.Online account's login and password in Base**

TIP

You will find more information about these authorization methods in the [Devino documentation].



Enter your authorization data. After that, it is necessary to validate the entered information. To initiate the

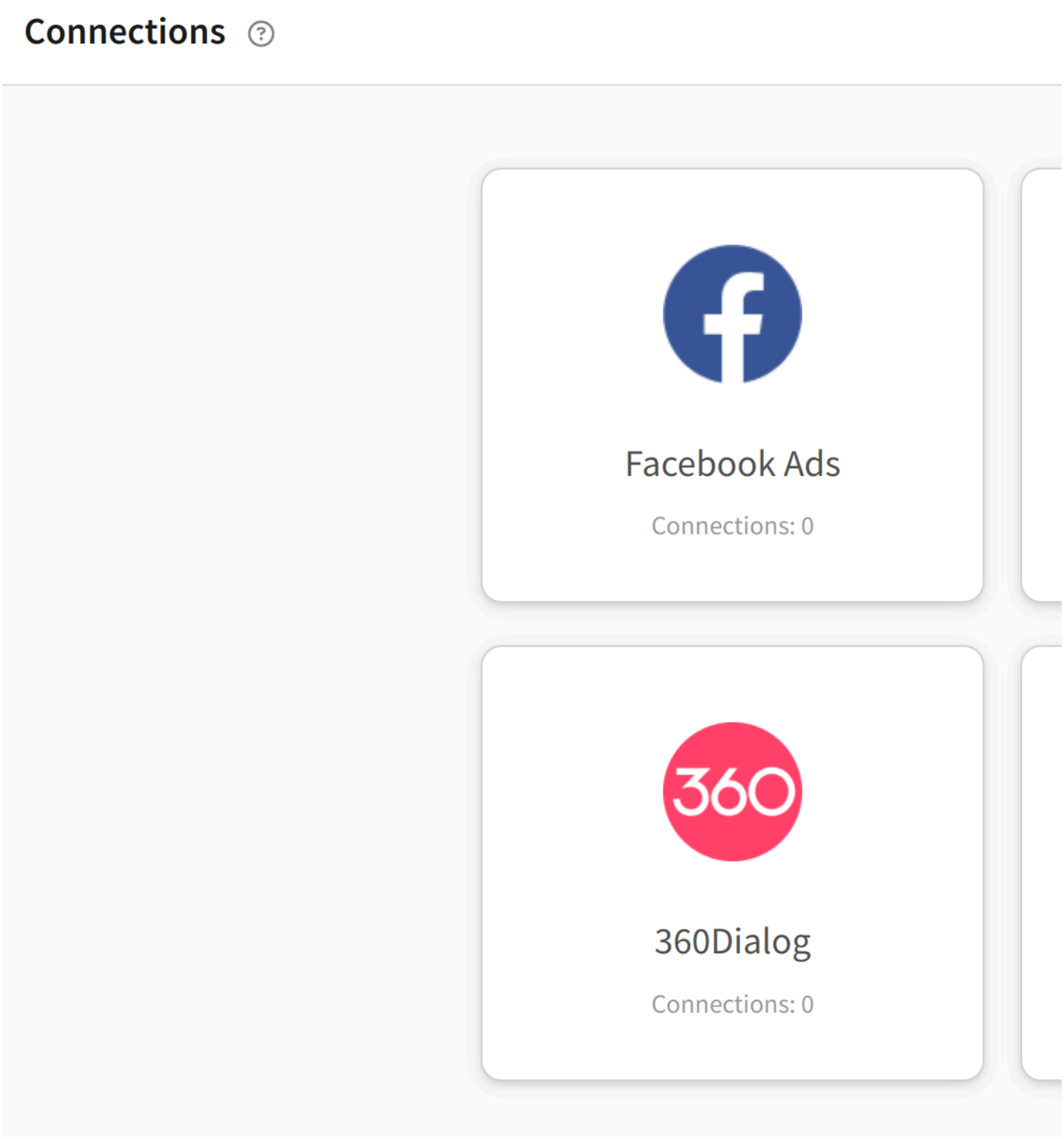


Save your connection using the buttons in the upper right corner. Now you can set up the integration.

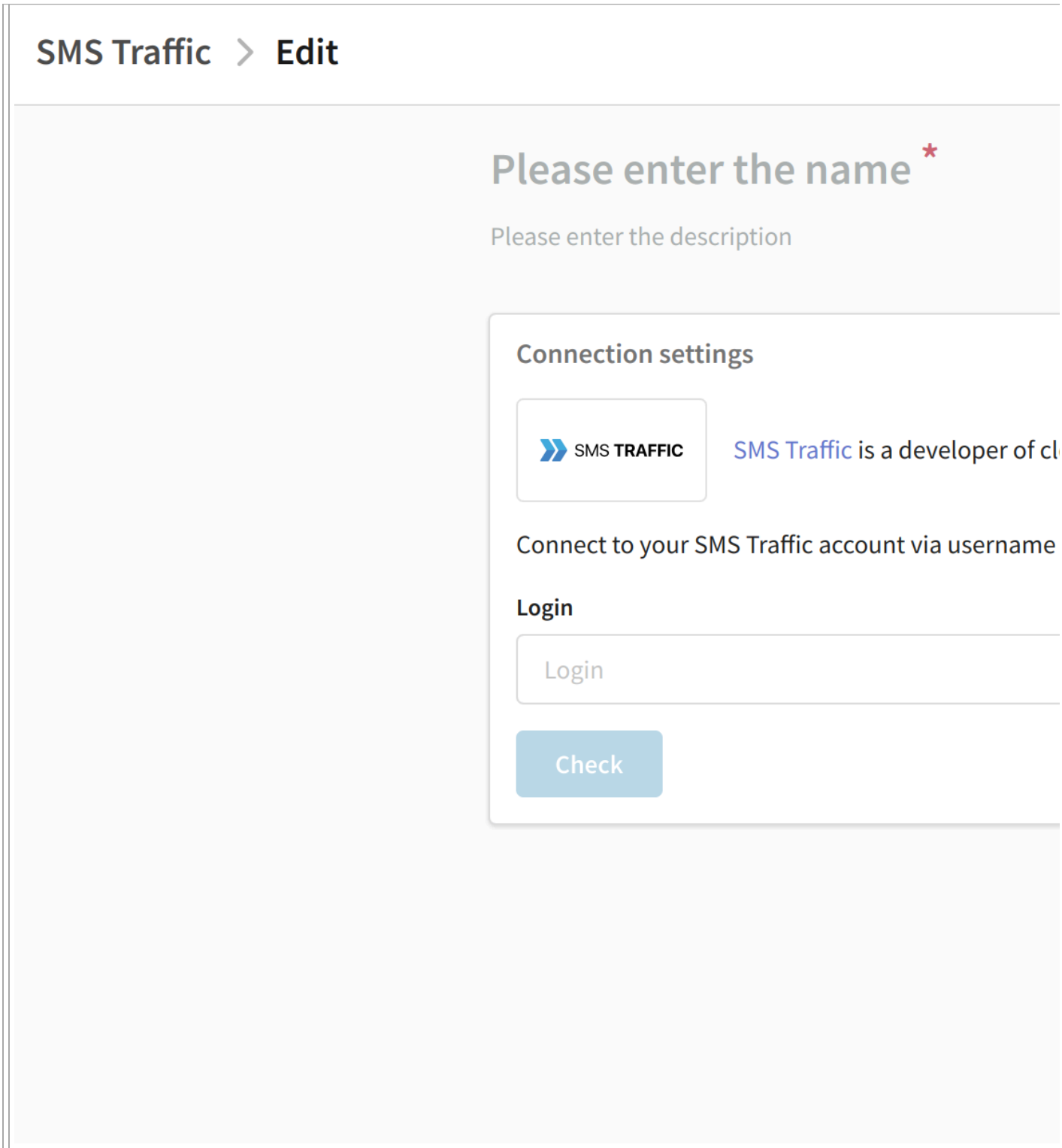
*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Connection to SMSTraffic

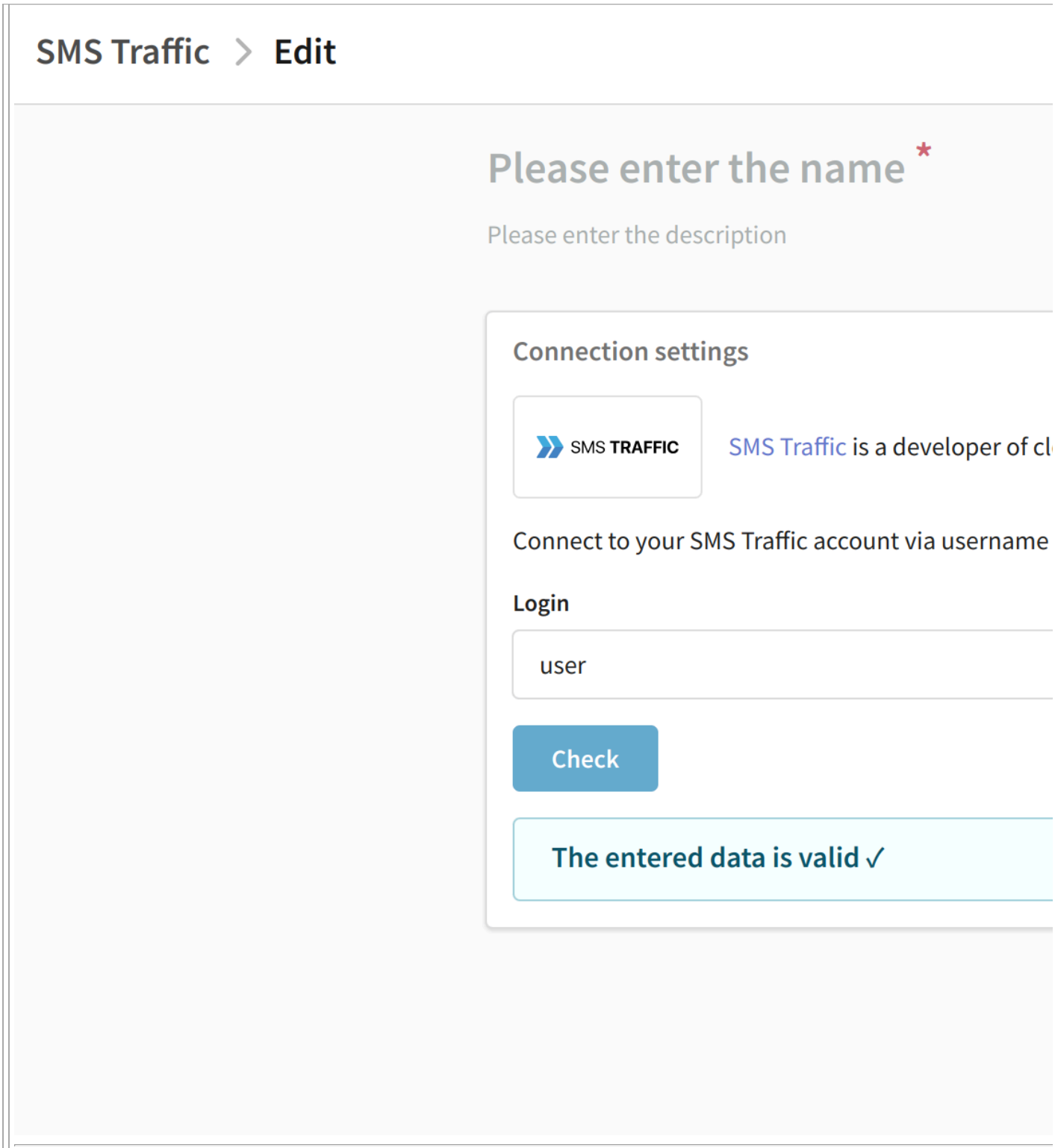
To integrate the platform with Viber*™, you need to configure a connection to one of the providers. This a
To get started, register an account with your provider and then go to Altcraft Platform.
In the main menu, go to Settings —> Connections. Select connection type — SMS Traffic. Click the Create



Specify a name and description for the connection. Next, log in with your password of your SMS Traffic ac



After that, it is necessary to validate the entered information. To initiate the validation process, click the



*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Connection to VK Ads™

INFO

To set up the connection, you need an account in VK Ads™. If you don’t have an account yet, register usi

In the main menu click “Settings” —> “Connections”. Click Create to add a new connection. Choose VK Ac

Connections

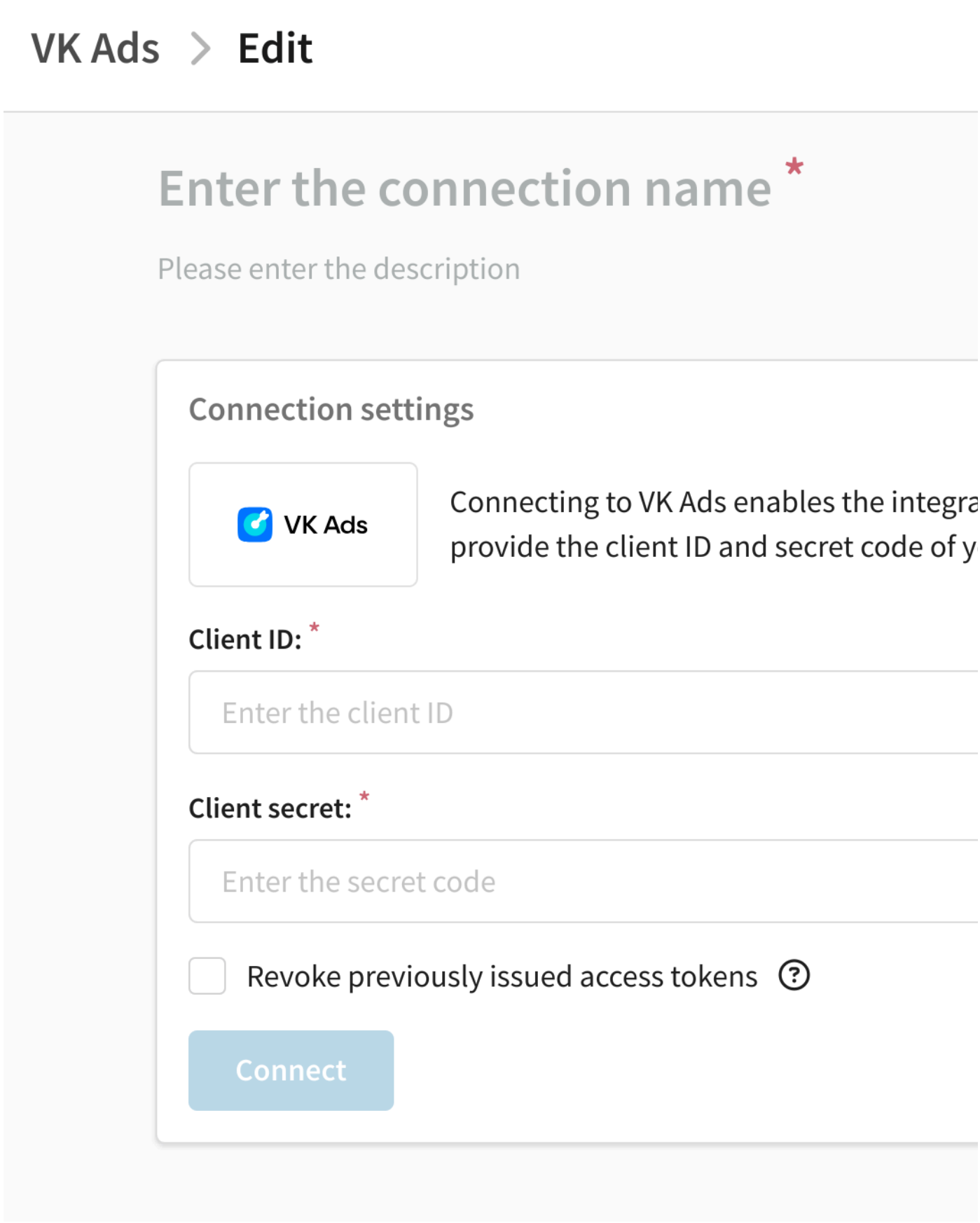
Facebook Ads

Connections: 0

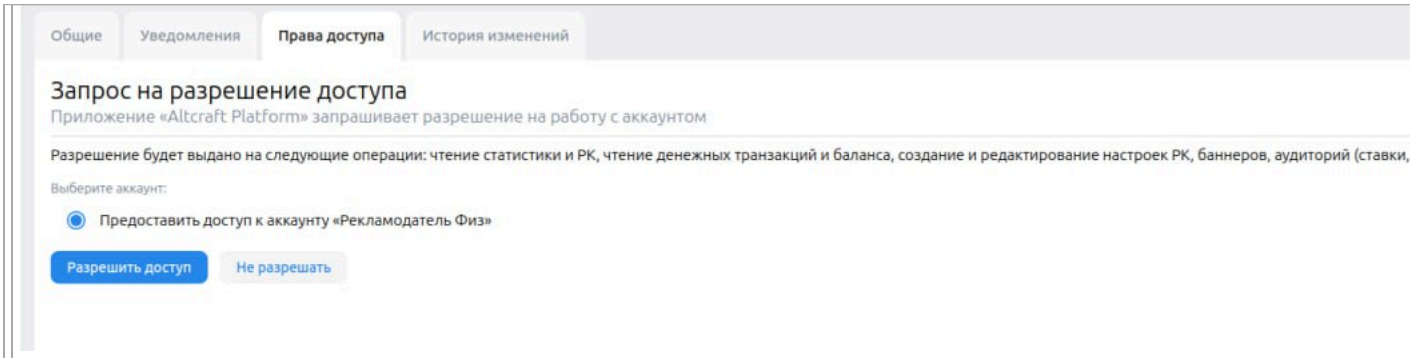
Edna

Connections: 0

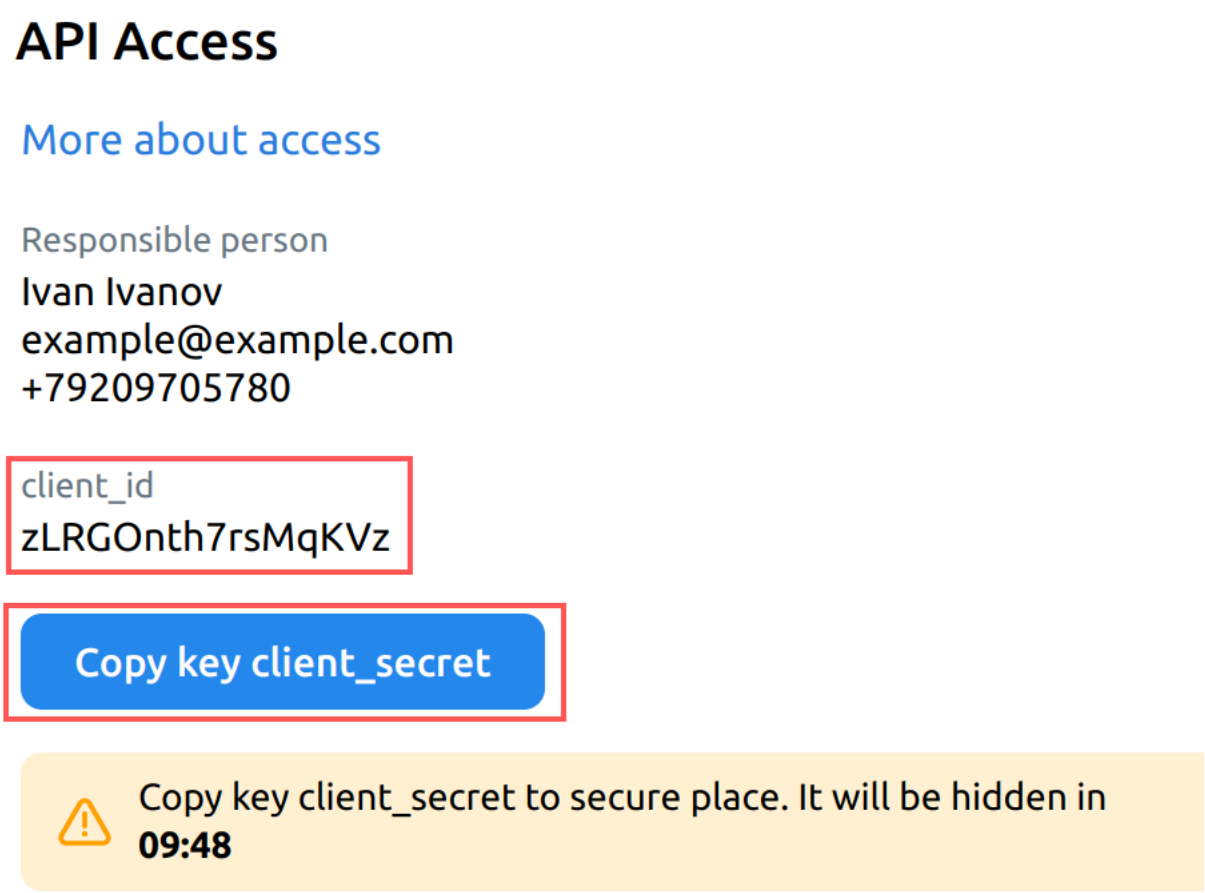
A list of all configured connections to VK will open. To add a new one, click Create. A cretion menu will o



If you use the cloud version of the Altcraft platform, when you try to connect the service, the VK Ads™ se platform.



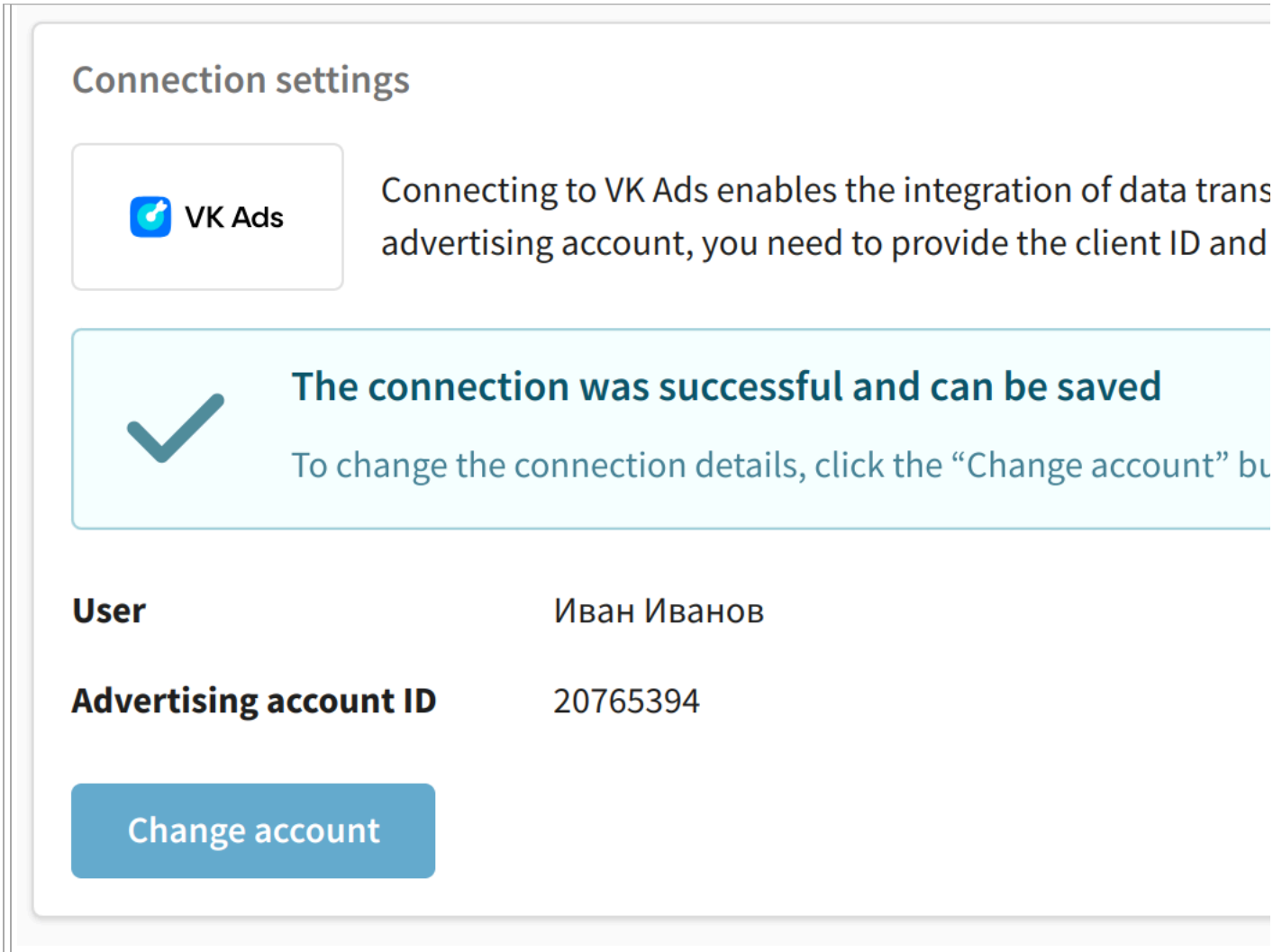
If you use on-premise version, you need to provide a Client ID and Client secret to connect your account. T
Once you have confirmed your contact information, you will have access to your client_id, it is used in the
able to see your Client secret:



When all fields are filled in, click the “Connect” button. If successful, a confirmation will appear on the sc

INFO

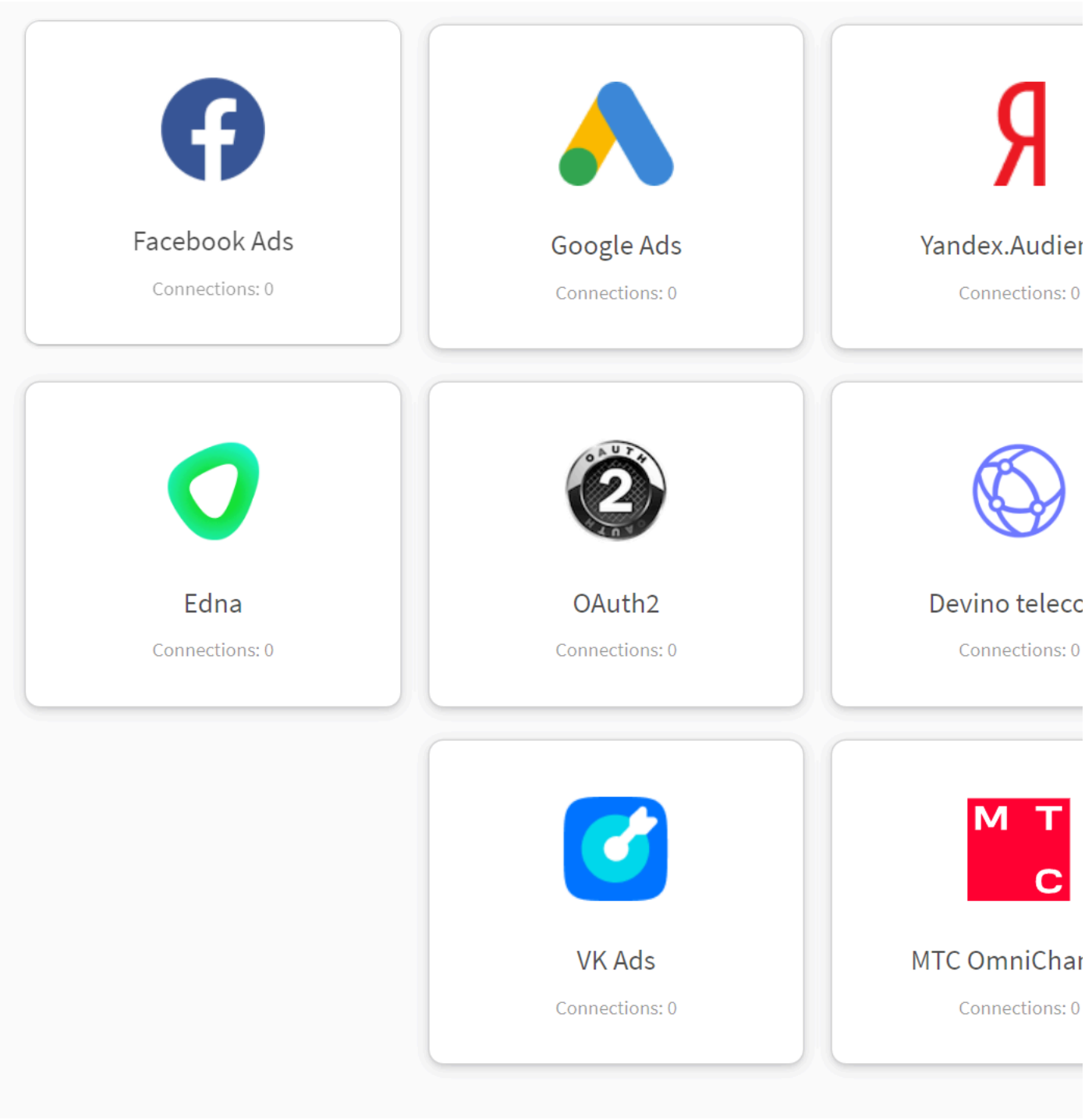
Every time you make a successful attempt to connect to VK Advertising™, the platform creates an acces
activate the “Revoke previously issued access tokens” option when creating a connection.



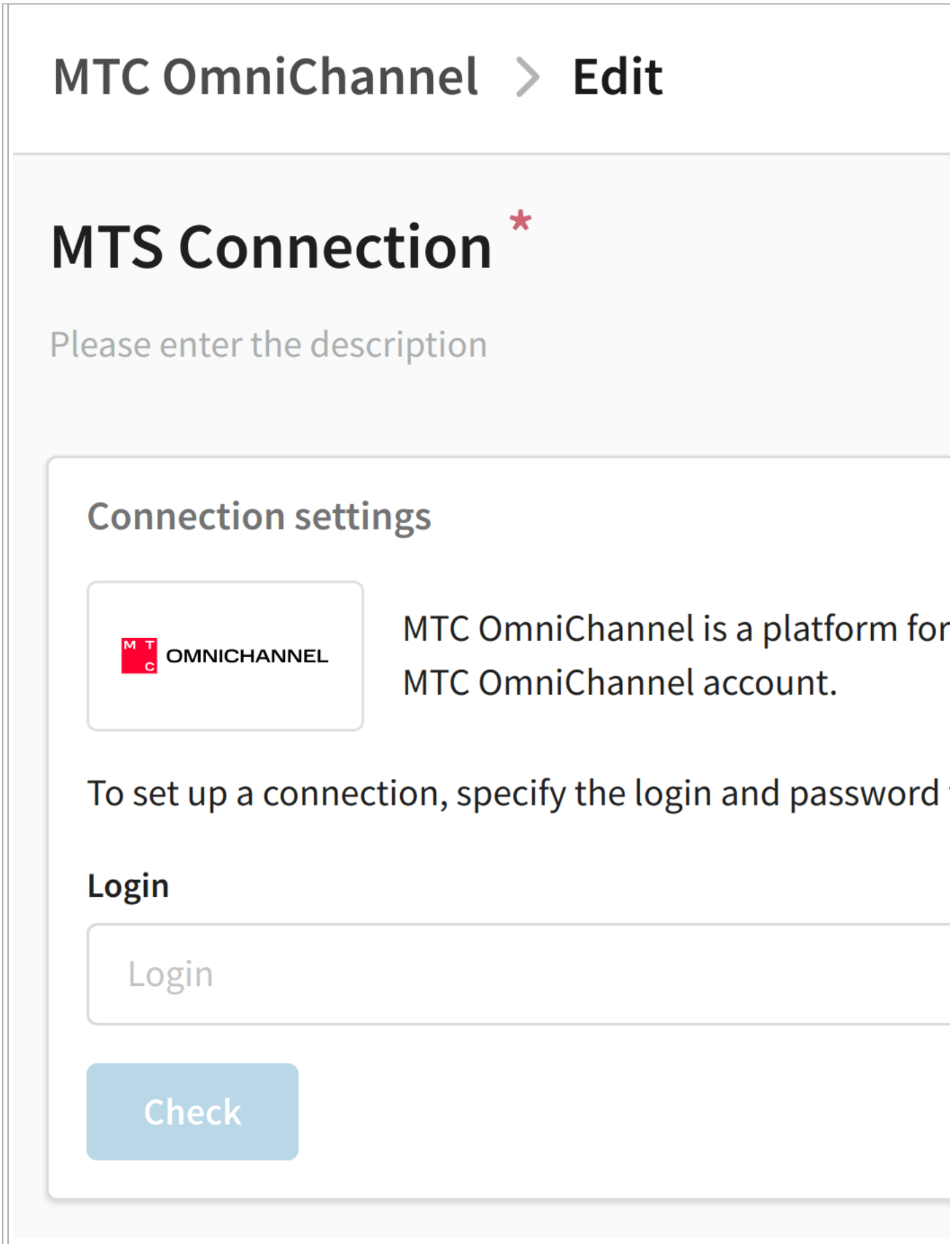
If the connection has been established, click the “Save” button. After saving the connection, you can proc

Connection to MTS OmniChannel

Connection to Viber™ is required to integrate the platform with the messenger. This article describes how
To get started, register an account with your provider and then go to Altcraft Platform.
In the main menu, go to Settings —> Connections. Select connection type — MTS OmniChannel. Click the



Specify a name and description for the connection. Next, log in with your password of your SMS Traffic ac



After that, it is necessary to validate the entered information. To initiate the validation process, click the

MTC OmniChannel > Edit

MTS Connection *

Please enter the description

Connection settings

OMNICHANNEL

MTC OmniChannel is a platform for
MTC OmniChannel account.

To set up a connection, specify the login and password

Login

Check

The entered data is valid ✓

Save the connection by clicking the **Save ** button. After that you can proceed to setup a new integrati

*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

Custom Authentication Connection

Description

Custom Authentication is a connection type that allows you to set custom HTTP headers for API requests,
With this connection type, you can link the Altcraft platform to any external service that requires custom
Where the connection is used:

- **Node “API call”. You can use it to transfer information from the scenario to the connected service or,**
- **Action Hooks. The event capture mechanism allows you to pass information about an event and the a**
- **Import data on a schedule in the Market. Custom Authentication can be used to set up a connection t**

How to create a connection

Go to Settings — Connections and select the connection type Custom Authentication:

Connections

Facebook Ads

Connections: 0

Google Ads

Connections: 0

Edna

Connections: 0

OAuth2

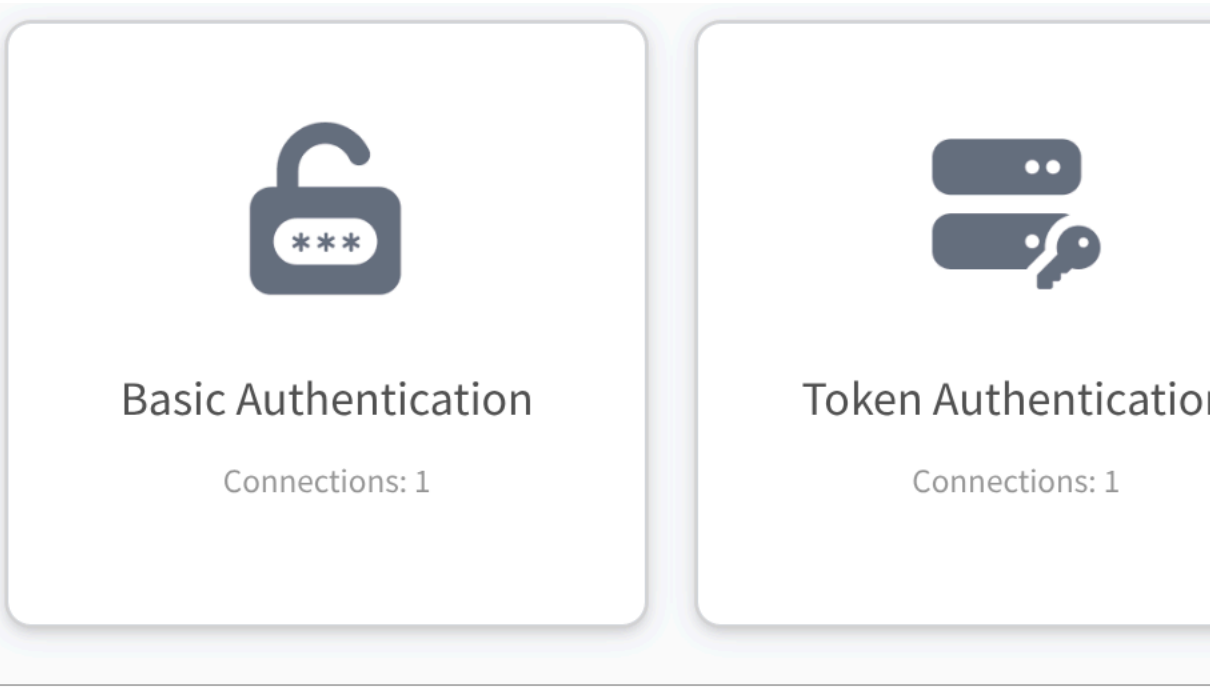
Connections: 0

MTS OmniChannel

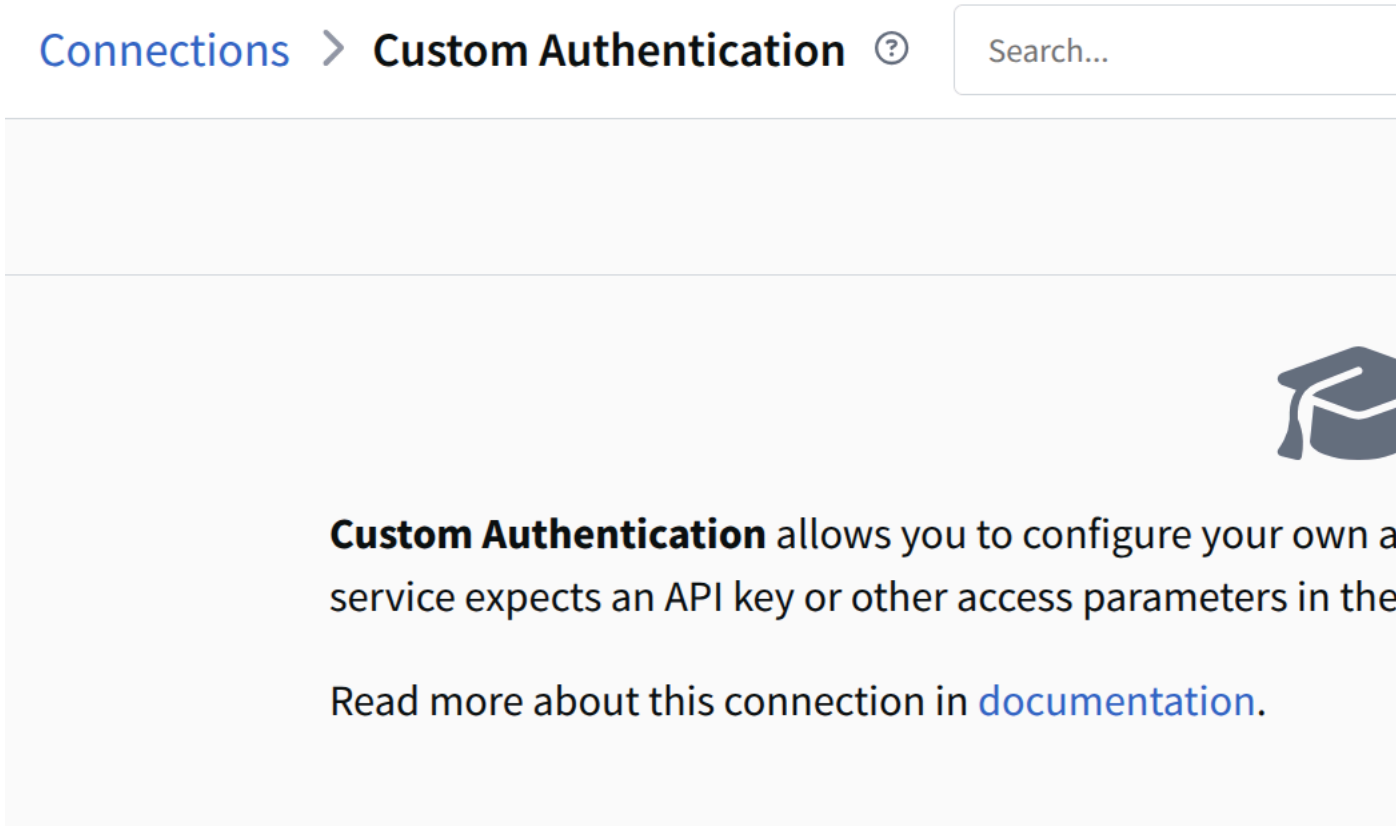
Connections: 0

Rapporto

Connections: 0


***
Basic Authentication
Connections: 1
Token Authentication
Connections: 1

A list of all configured Custom connections will appear. To add a new one, click + Create.



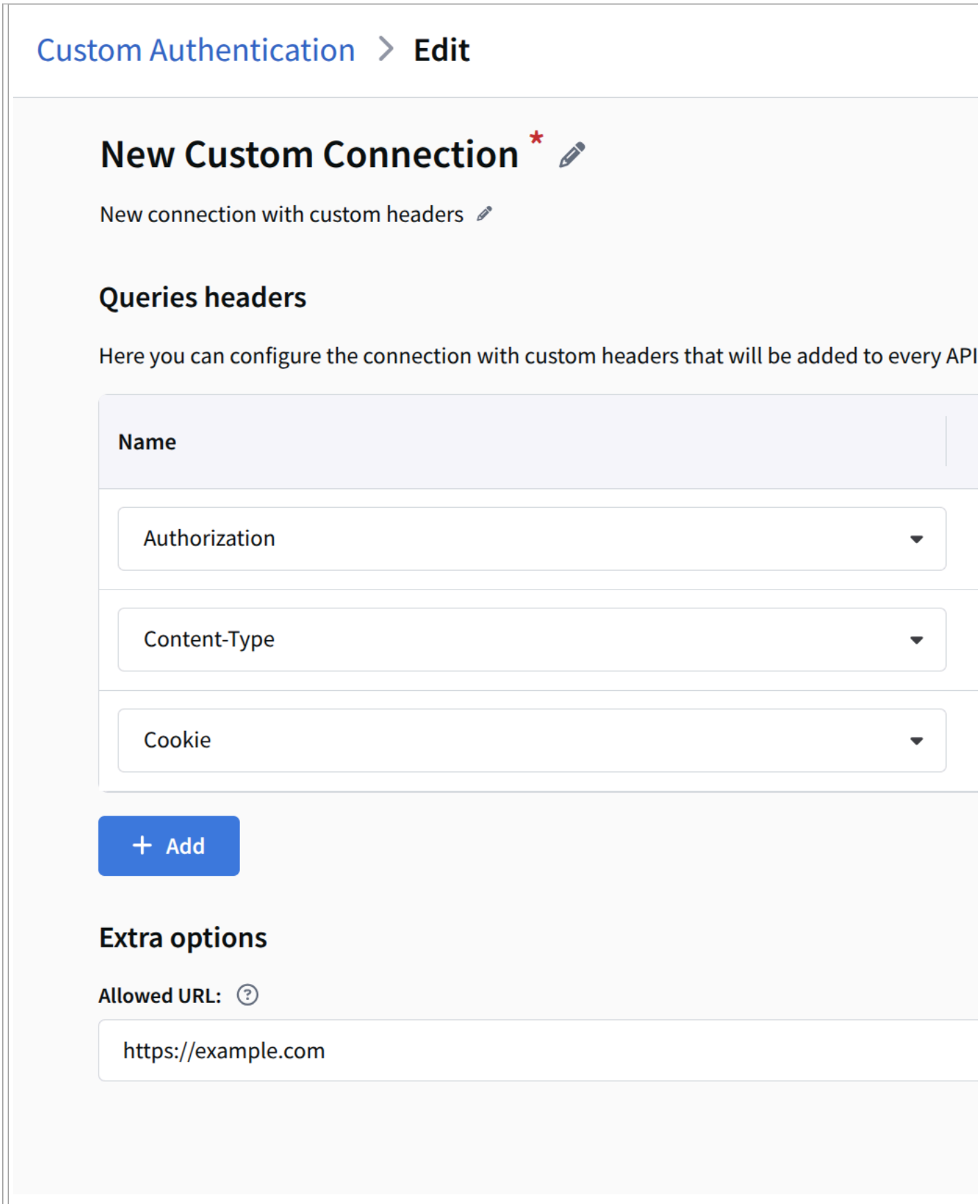
Enter a name for the new connection and, if necessary, add a description. Then add the desired headers a

TIP: Forbidden headers

Some system headers are forbidden to add, as their use may cause issues with the connection. The list

- **“Host”**
- **“Connection”**
- **“Keep-Alive”**
- **“Accept-Encoding”**
- **“Content-Length”**
- **“Transfer-Encoding”**
- **“Trailer”**
- **“Upgrade”**
- **“Date”**
- **“Location”**
- **“Server”**

If you want to provide access only to a specific service, specify its URL in the Allowed URL field:



Save the connection using the buttons in the upper right corner. Connection settings can be changed at a

OAuth2 connection

Description

**OAuth 2.0 is an authorization framework that enables applications — such as Facebook*, GitHub, or a CRM
With this connection type, you can connect Altcraft Platform to any external service that uses OAuth2 as
Where is the connection used:**

- **Node “API call”. You can transfer information from the scenario to the connected service or, on the co**
- **Action Hooks. The event capture mechanism will allow you to transfer data about a platform event, in `custom_data` field. In this case, when the message is opened, the profile’s data can be passed to your C**
- **Import data on a schedule in the Market. To access a protected product feed, authorization via OAuth**

How to create a connection

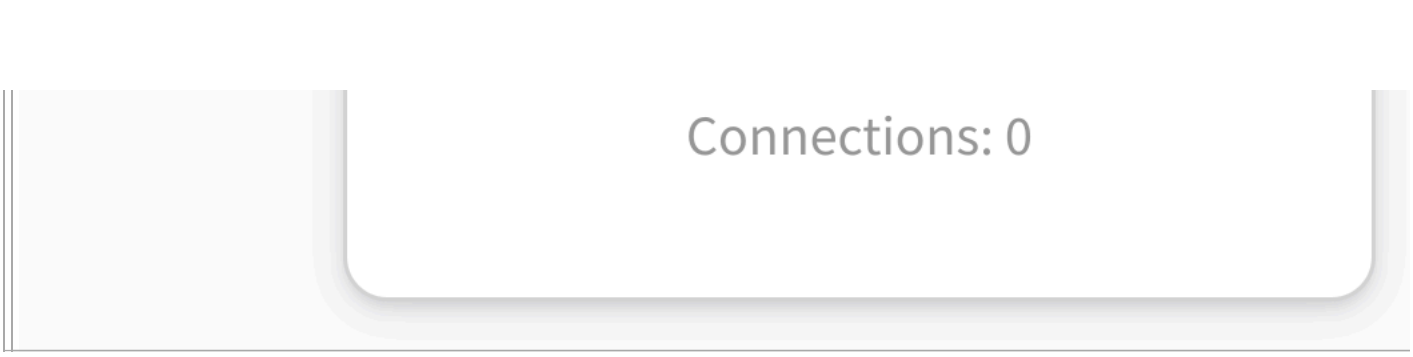
Go to Settings —> Connections and select the OAuth2 connection type:

Connections

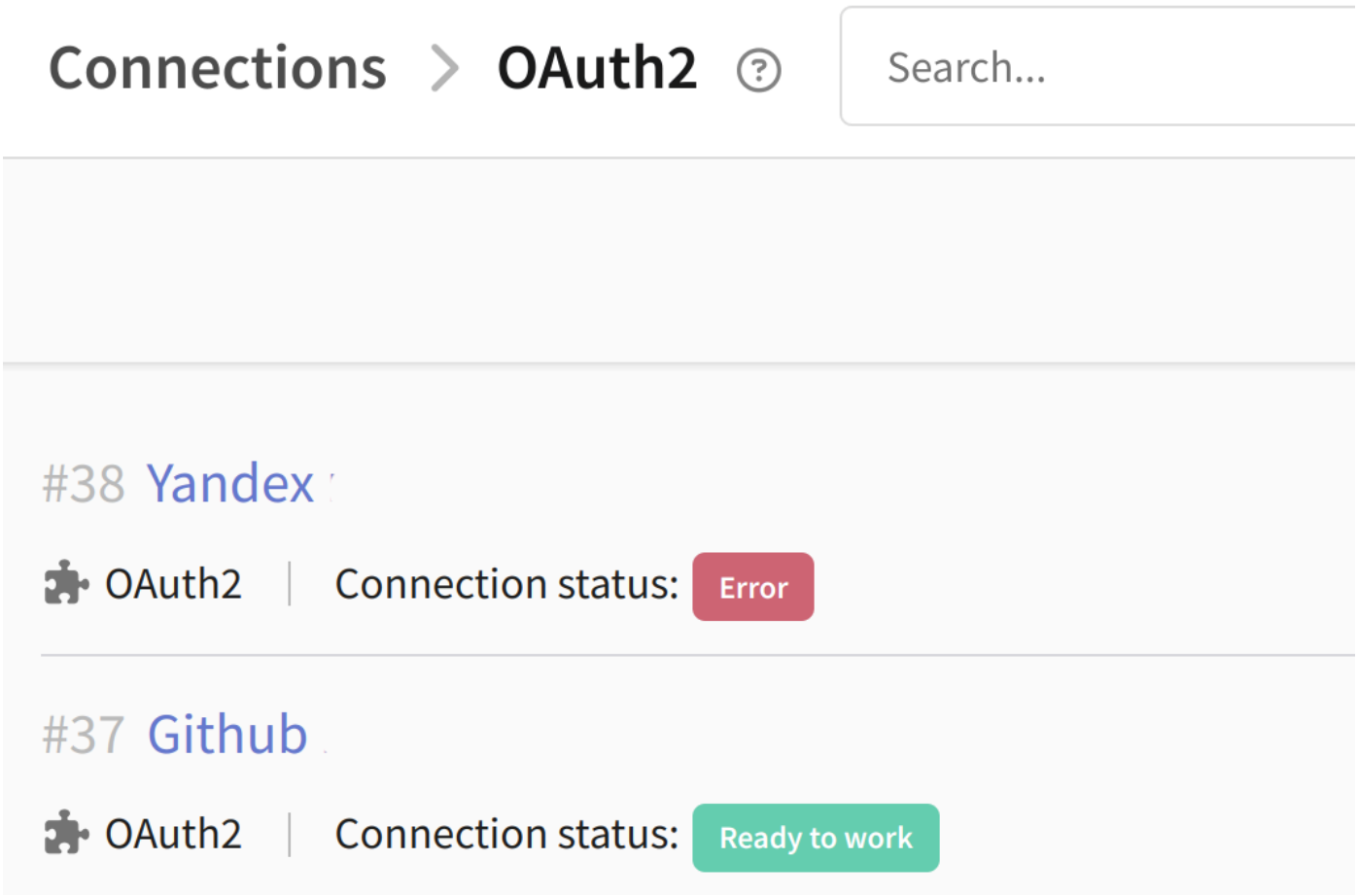
Facebook Ads

Connections: 0

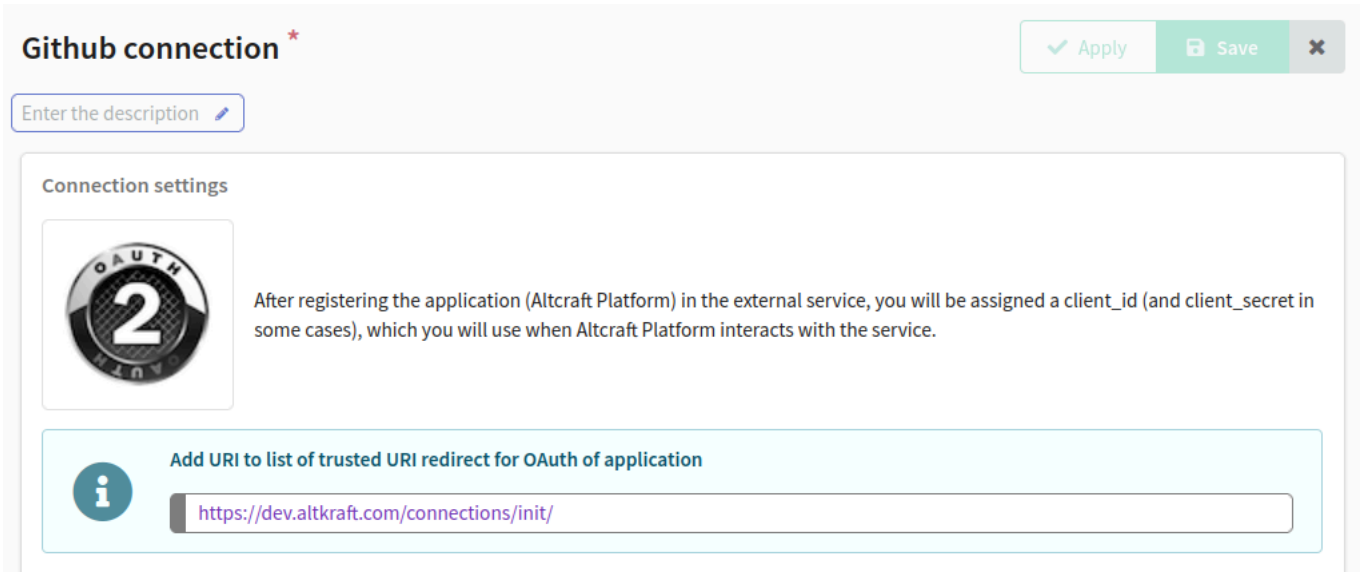
360Dialog


Connections: 0

A list of all configured OAuth2 connections will be shown. To add a new one, click + Create:



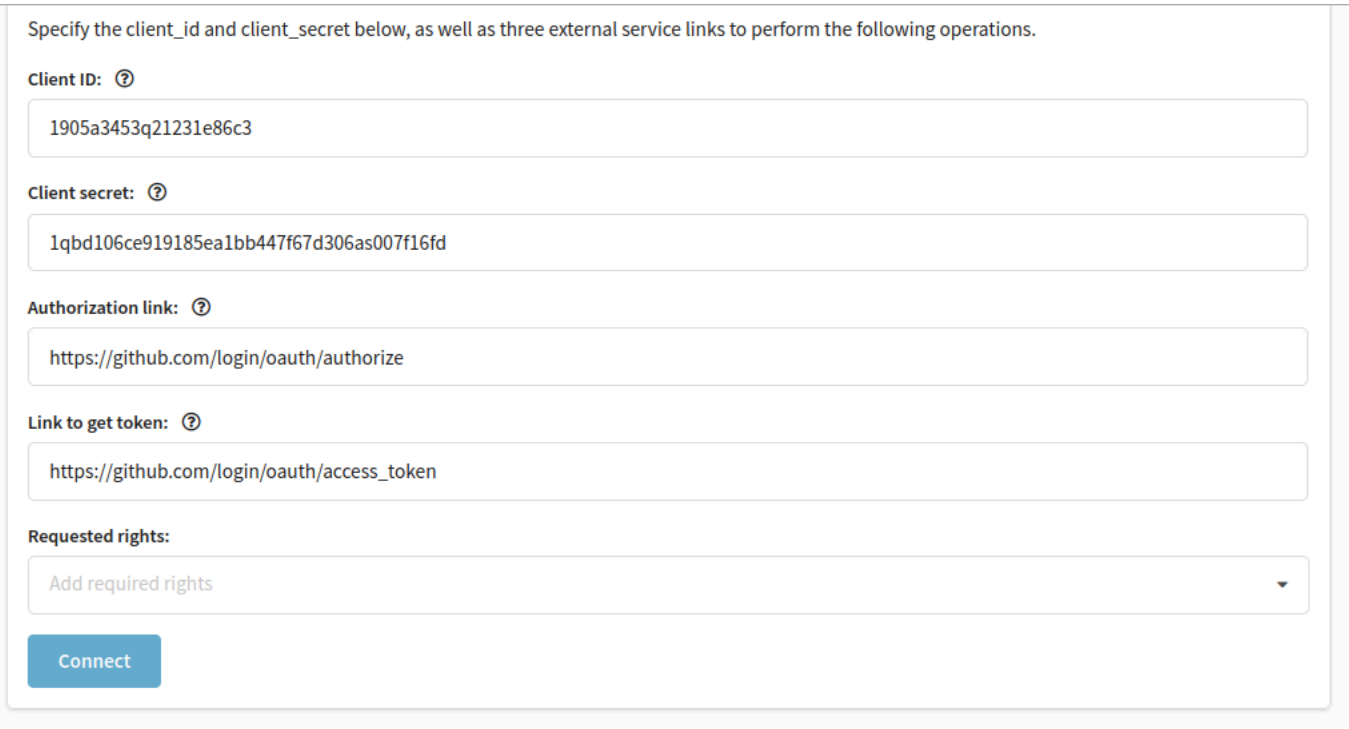
Enter the name of the new connection and add a description if necessary:



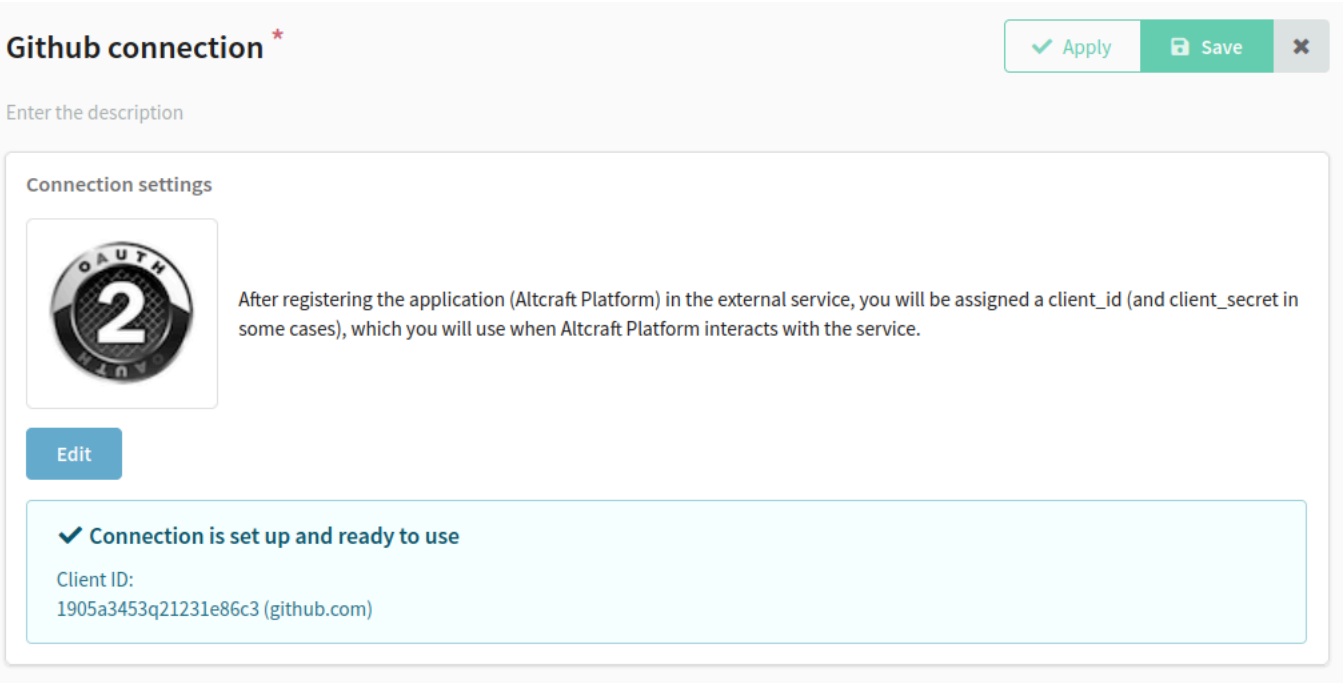
The platform provides a redirection URI — the URL to which the user will be redirected when they give Alt
Next, specify the details of your application that you need so that the platform can access its resources. `

- **Client ID is a public application identifier. Client ID is required for OAuth authorization.**
- **Client secret is the application password, known only to the application and the authentication server**

OAuth operations also require an authorization link and a link to get token. Study the documentation of th
If necessary, you can limit the platform's access to the data by selecting the requested rights:



Click “Connect”. You will be redirected to the connected service, where you need to log in and confirm th
If the connection to the service was completed successfully, save the connection using the buttons in the



Connection settings can be changed at any time. To do this, click “Edit”.

***The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as**

Basic Authentication connection

Description

Basic Authentication is an HTTP authentication protocol that uses a login and password to gain access to
Using this connection type, you can connect the Altcraft platform with any external service that uses Bas
Where is the connection used:

- **Node “API call”. You can transfer information from the scenario to the connected service or, on the co**
- **Action Hooks. The event capture mechanism will allow you to transfer data about a platform event, in `custom_data` field. In this case, when the message is opened, the profile’s data can be passed to your C**
- **Import data on a schedule in the Market. To access a protected product feed, authorization via login a**

How to create a connection

Go to Settings — Connections and select the connection type Basic Authentication:

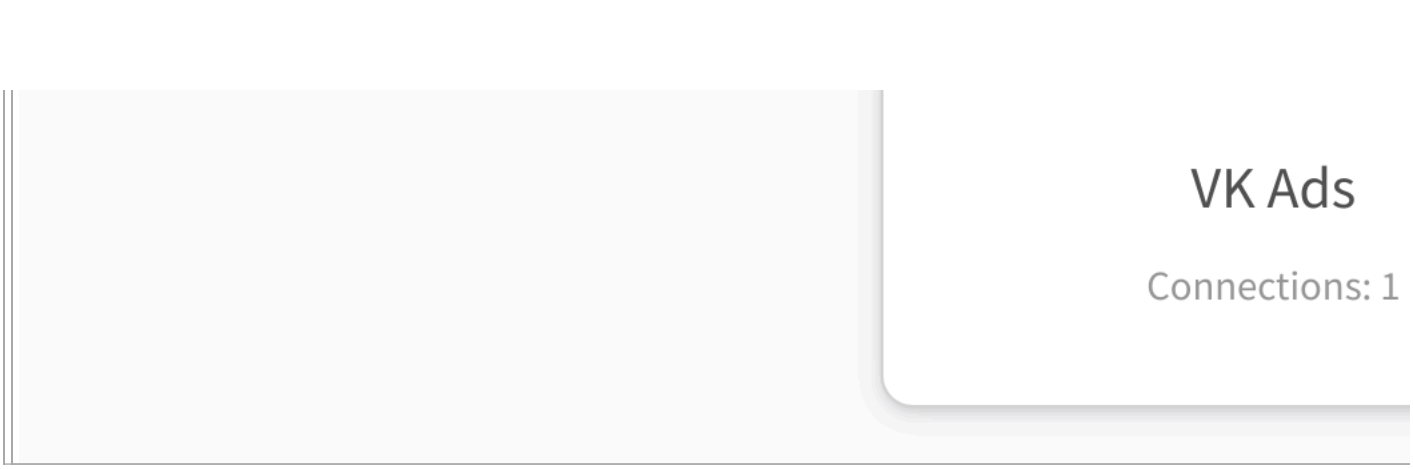
Connections

Facebook Ads

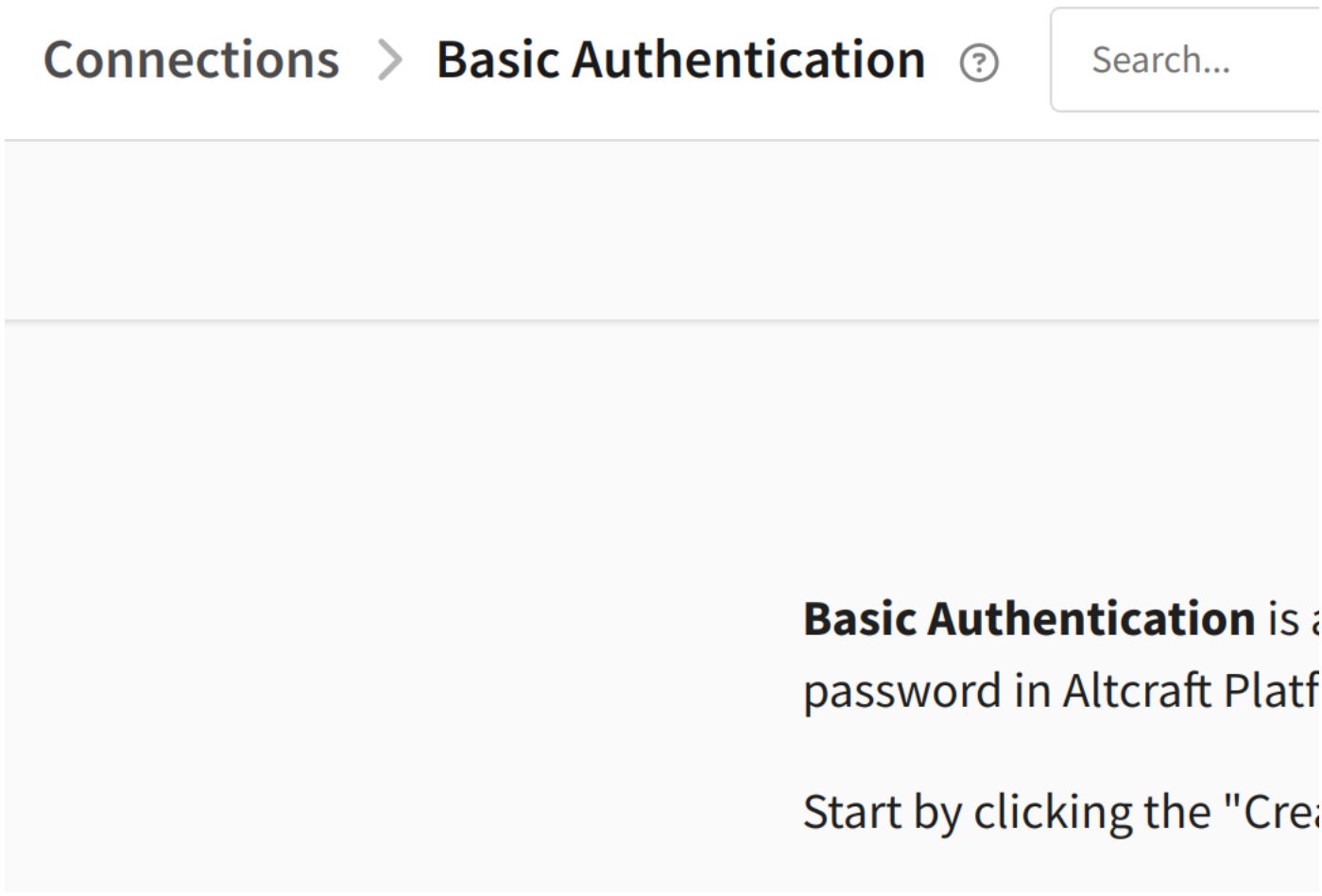
Connections: 0

Edna

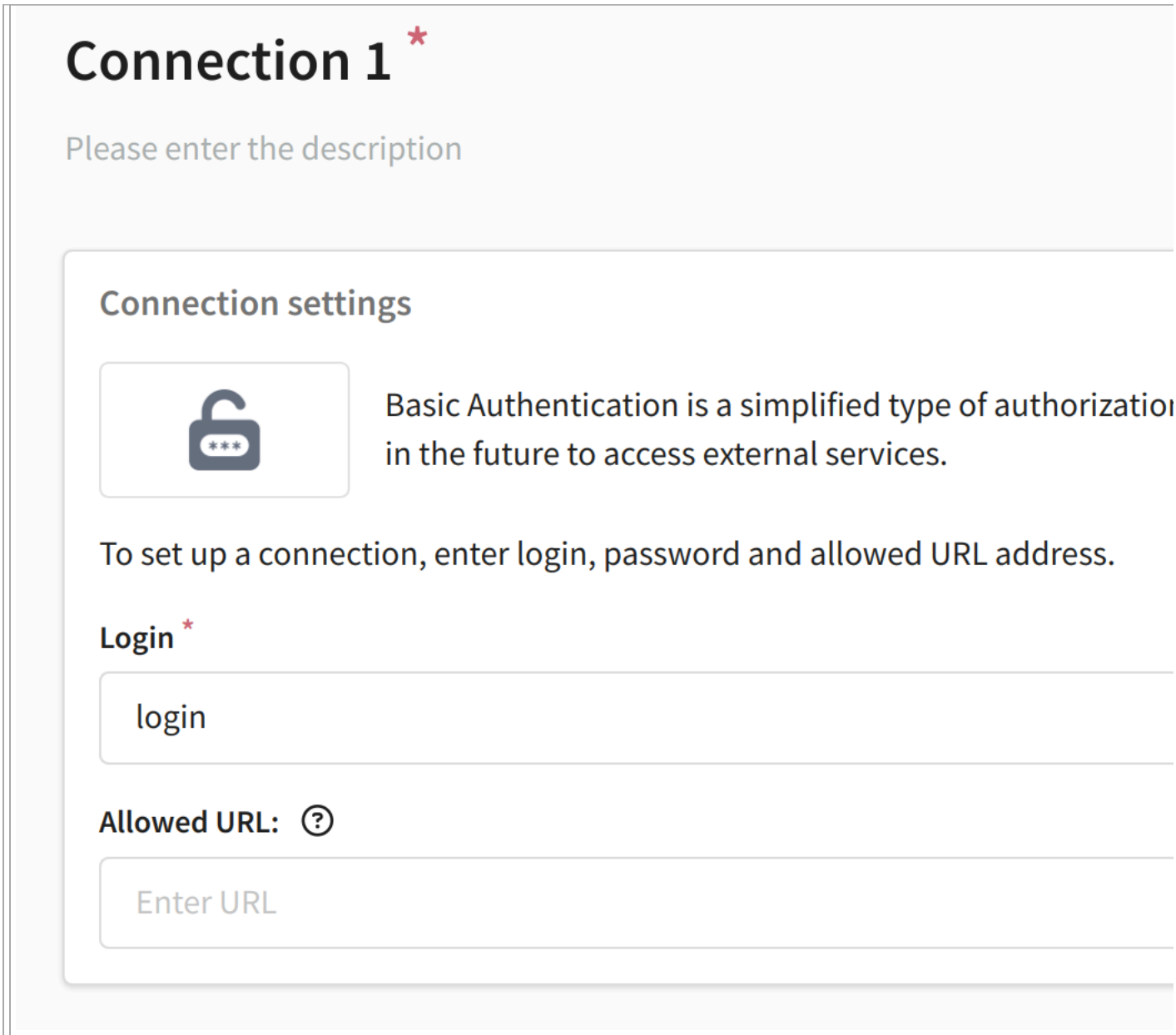
Connections: 0


VK Ads
Connections: 1

A list of all configured Basic connections will open. To add a new one, click + Create.



Enter a name for the new connection and, if necessary, add a description. Enter the username and passw



Save your connection using the buttons in the upper right corner. Connection settings can be changed at a

Token Authentication connection

Description

Token Authentication is an authentication protocol that uses a special Bearer token - a type of authorizati
Using this connection type, you can connect the Altcraft platform with any external service that uses Tok

- **Node “API call”. You can transfer information from the scenario to the connected service or, on the co**
- **Action Hooks. The event capture mechanism will allow you to transfer data about a platform event, in `custom_data` field. In this case, when the message is opened, the profile’s data can be passed to your C**
- **Import data on a schedule in the Market. To access a protected product feed, authorization using toke**

How to create a connection

Go to Settings — Connections and select the connection type Token Authentication:

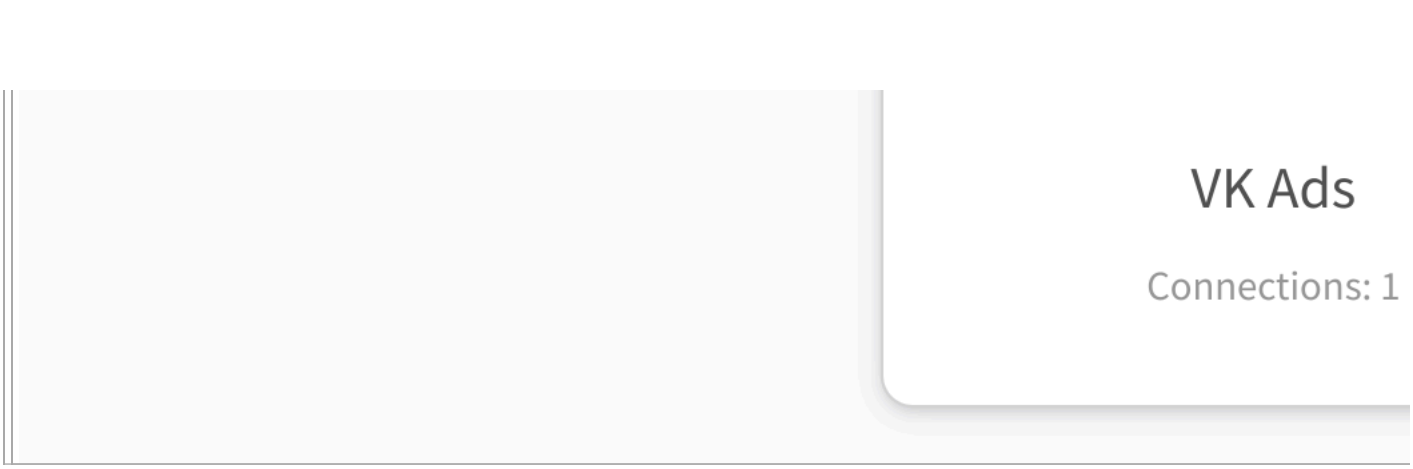
Connections

Facebook Ads

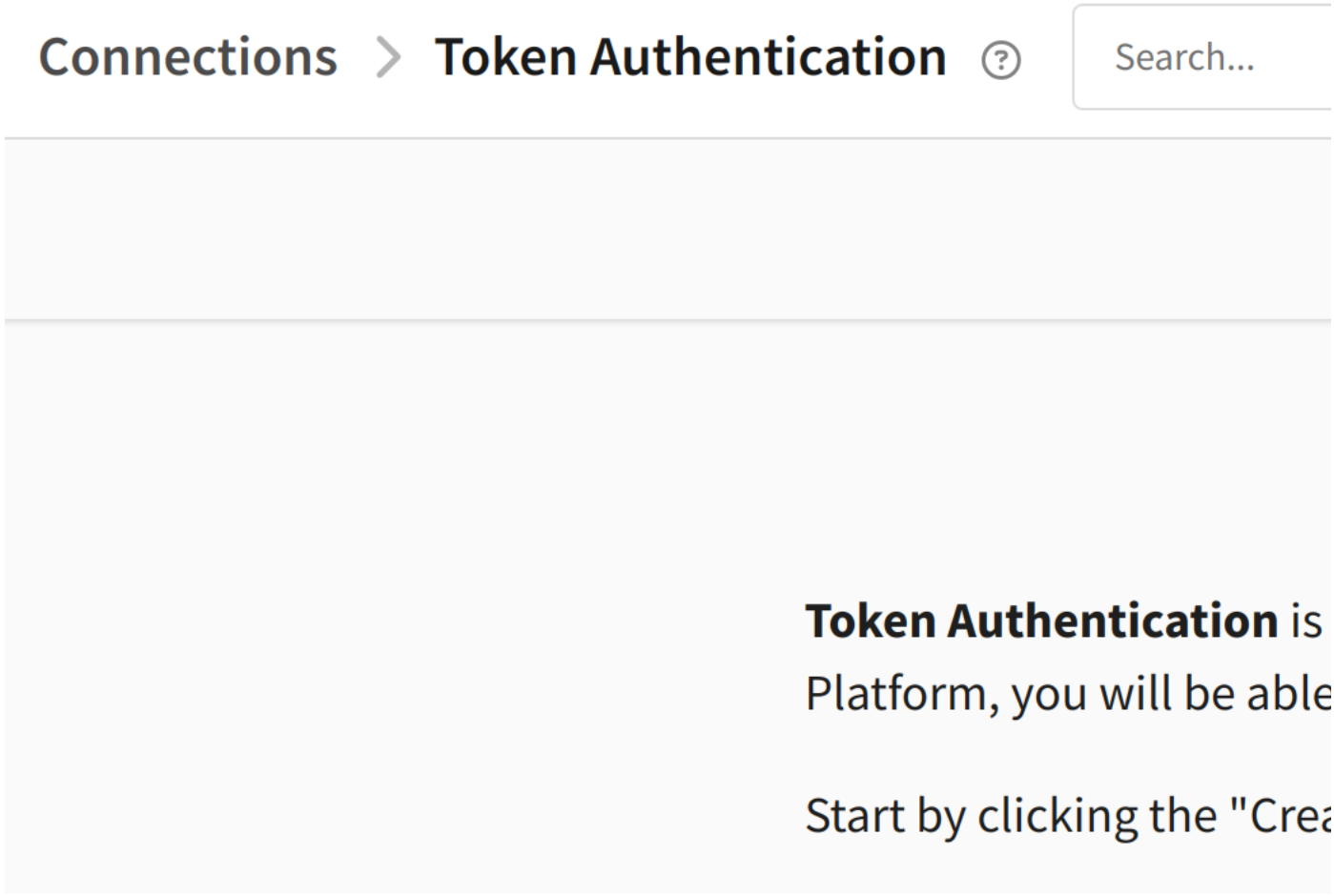
Connections: 0

Edna

Connections: 0


VK Ads
Connections: 1

A list of all configured Token connections will open. To add a new one, click + Create.



Enter a name for the new connection and, if necessary, add a description. Specify Bearer token. Ussually

Connection 1 *

Please enter the description

Connection settings

Token Authentication is a simplified type of authorization
external services.

To set up a connection, enter bearer-token and allowed URL address.

Bearer-token:

Input token

Allowed URL:

Enter URL

Save your connection using the buttons in the upper right corner. Connection settings can be changed at a

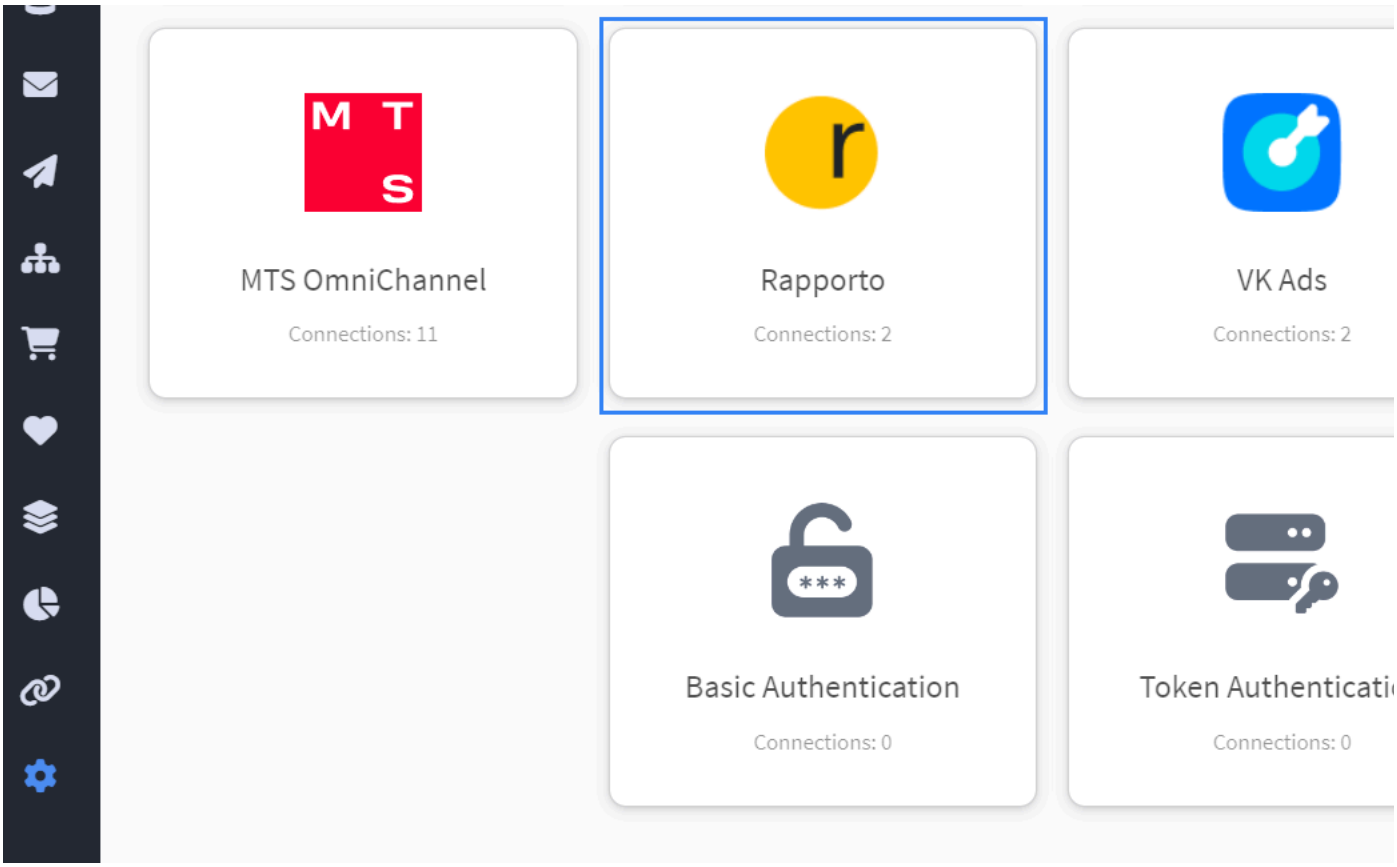
Connection to Rapporto

Connection to Viber*™ is required to integrate the platform with the messenger. This article describes ho

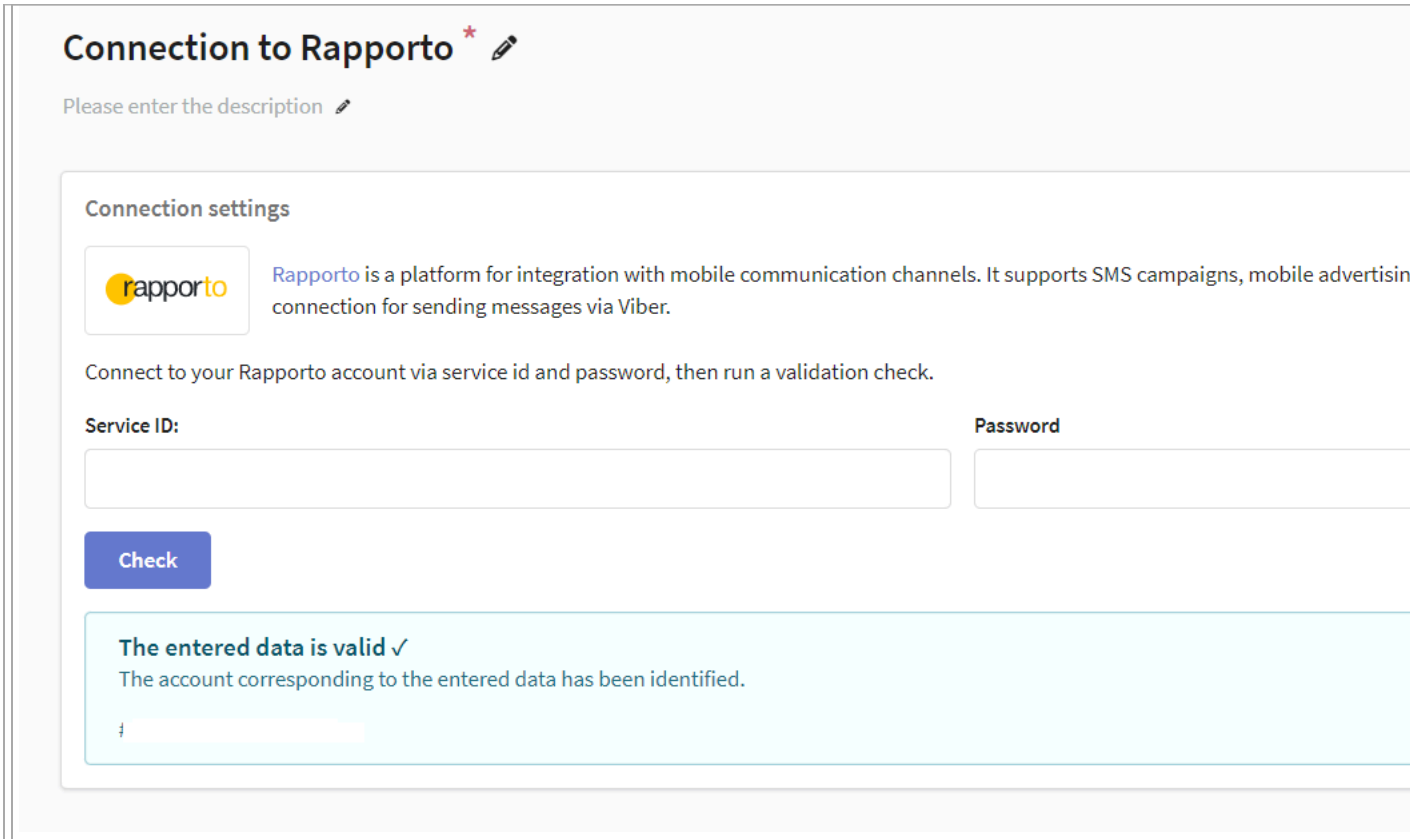
TIP

Before setting up a connection on the platform, read the [Rapporto documentation] in order to set up integ

In the main menu, go to Settings —> Connections. Select connection type — Rapporto:



Click the Create button on the top bar. The connection settings window opens, then enter connection's na
Next you need to pass a validation check. Enter the service ID and password, click "Check. Upon success



Click on the Save" button to save the connection.After that you can proceed to [setup a new integration](#).

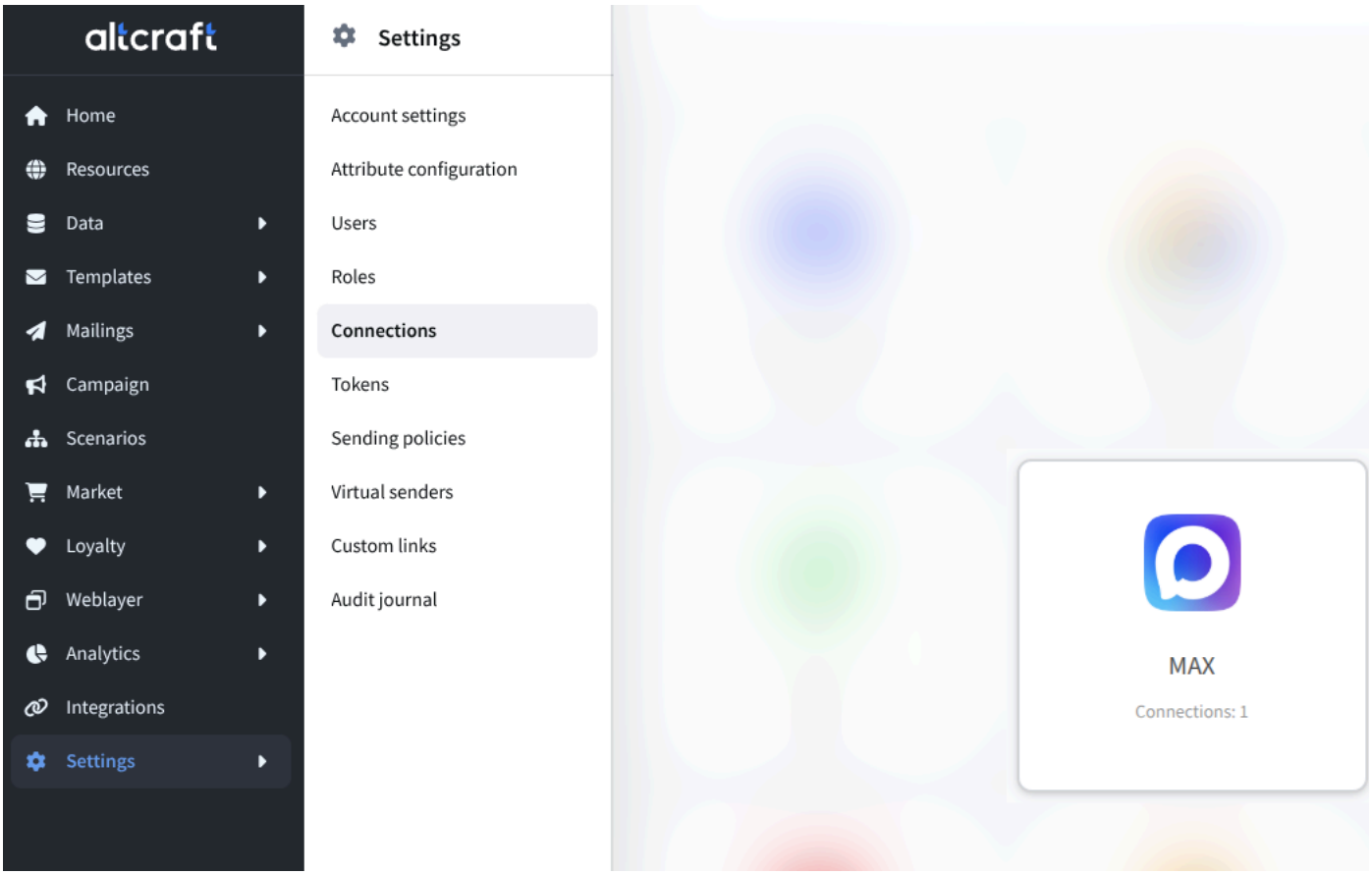
*** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.**

MAX connection

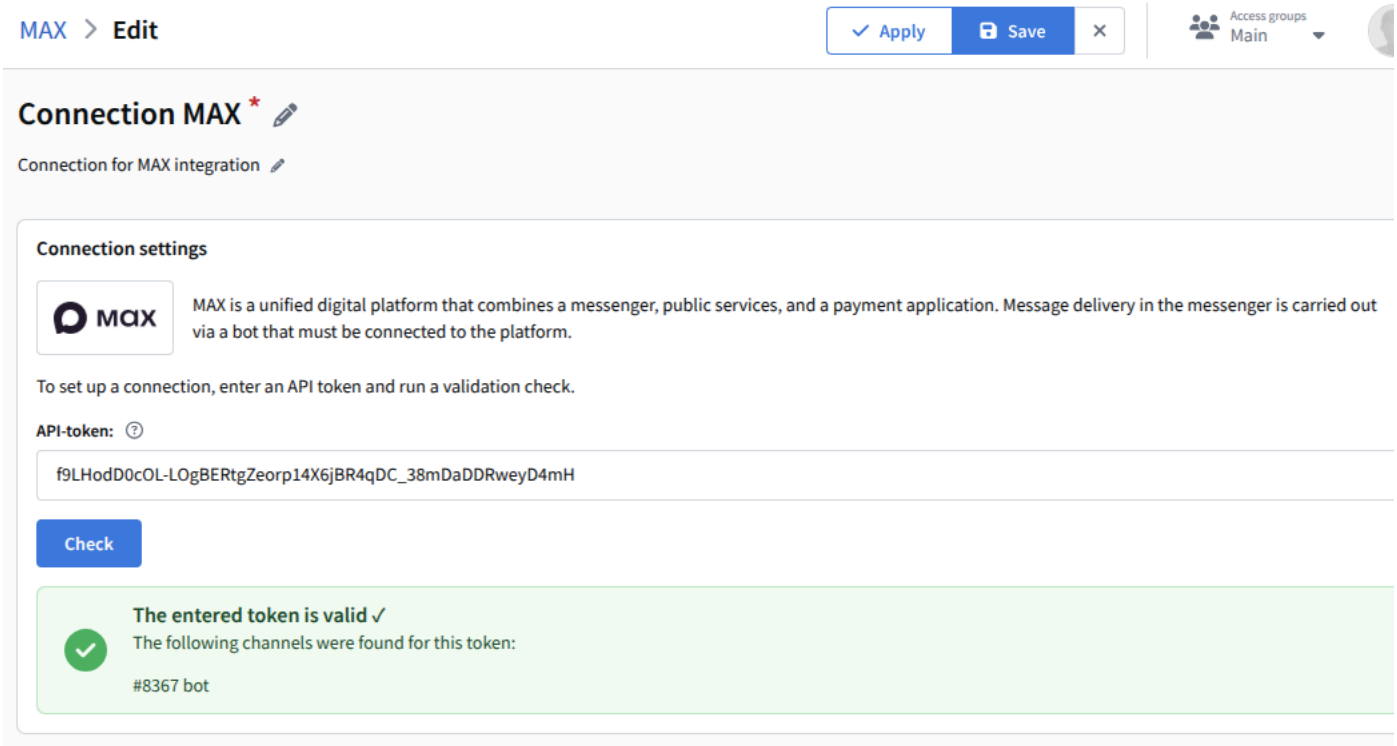
INFO

Before connecting MAX to the platform, you must first create a bot in MAX. Bot creation is available only

A connection to the MAX messenger is required to integrate the platform with MAX for further use as com
To create a connection, you will need your bot's API token in MAX. You can find the token in the Chatbots
Then go to the Altcraft platform. Open Settings — Connections. Select the connection type — MAX:



Click Create on the top panel. Enter the connection name and description.
Then specify the API token obtained in the first step of the guide. Click Check. If the entered data is valid



Save the connection by clicking the Save button. After that, you can proceed to setting up MAX integratio

Connection to Notify

INFO

According to the service policy, new access to the Notify service personal account is currently not provi
Support for current users is provided by MAX Business tech support.

Connection to Notify is used to create an integration and further use the service as a communication cha

TIP

The connection is available to clients who already have direct access to the service, i.e., those who alre

In the main menu, go to Settings —> Connections. Select connection type — Notify:

Connections

Facebook Ads

Connections: 0

Google

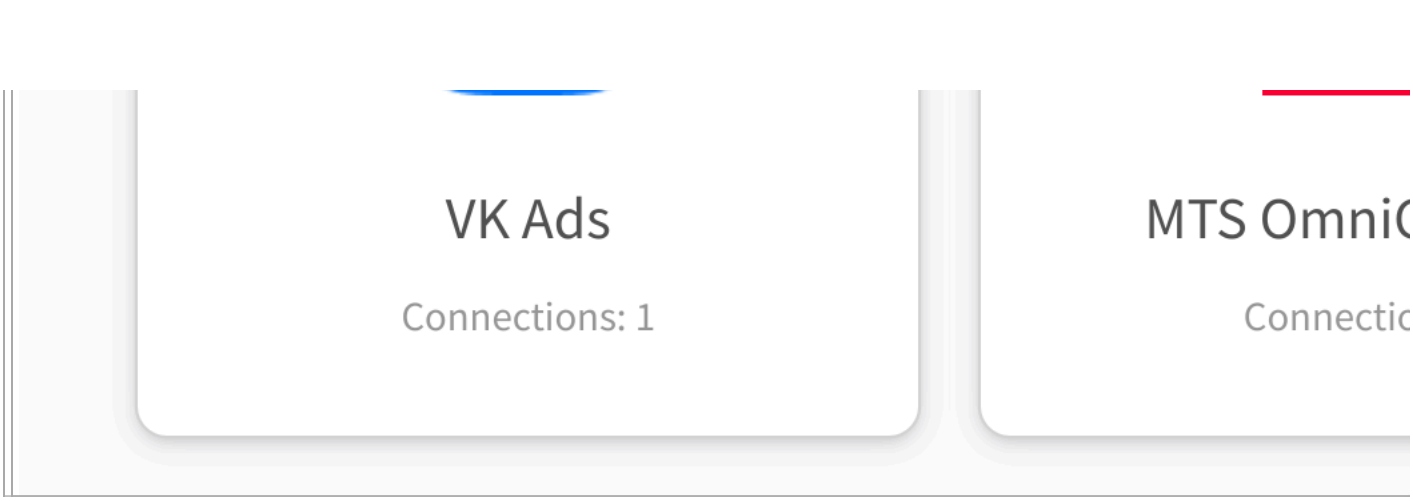
Connectic

Edna

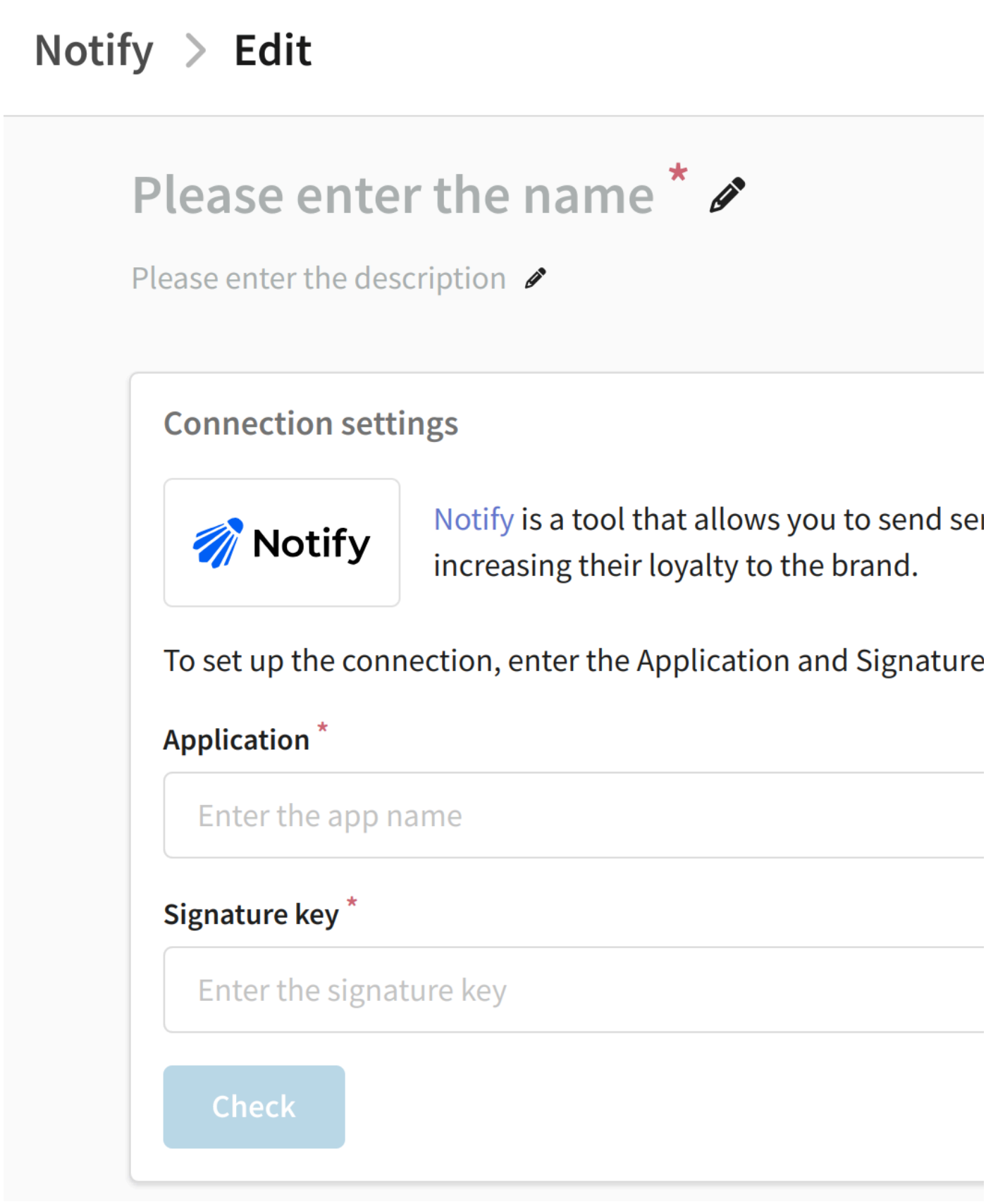
Connections: 0

OAutl

Connectic


VK Ads
Connections: 1
MTS Omni(
Connectic

Click the “Create” button on the top bar. In appearing window enter the name and description of connecti



To create a connection, you will need the Application and Signature key parameters from your Notify acco

Notify > Edit

Notify connection *

Please enter the description

Connection settings

Notify

Notify is a tool that allows you to send se
increasing their loyalty to the brand.

To set up the connection, enter the Application and Signature

Application *

altcraft

Signature key *

Check

The entered data is valid ✓

Notify account identified.

Save the connection by clicking the Save button. You can then move on to Notify integration setup.

API-requests

Most users of the Altcraft platform interact with it through a web interface in a browser. However, you ca
API (Application Programming Interface) is a set of rules that allows programs to communicate with each
As a rule, developers are in charge of writing API requests. However, understanding the principles of API
In this section, you will find guides on how to create basic API requests such as importing a profile, launc

Basic API requests:

Import or update a profile

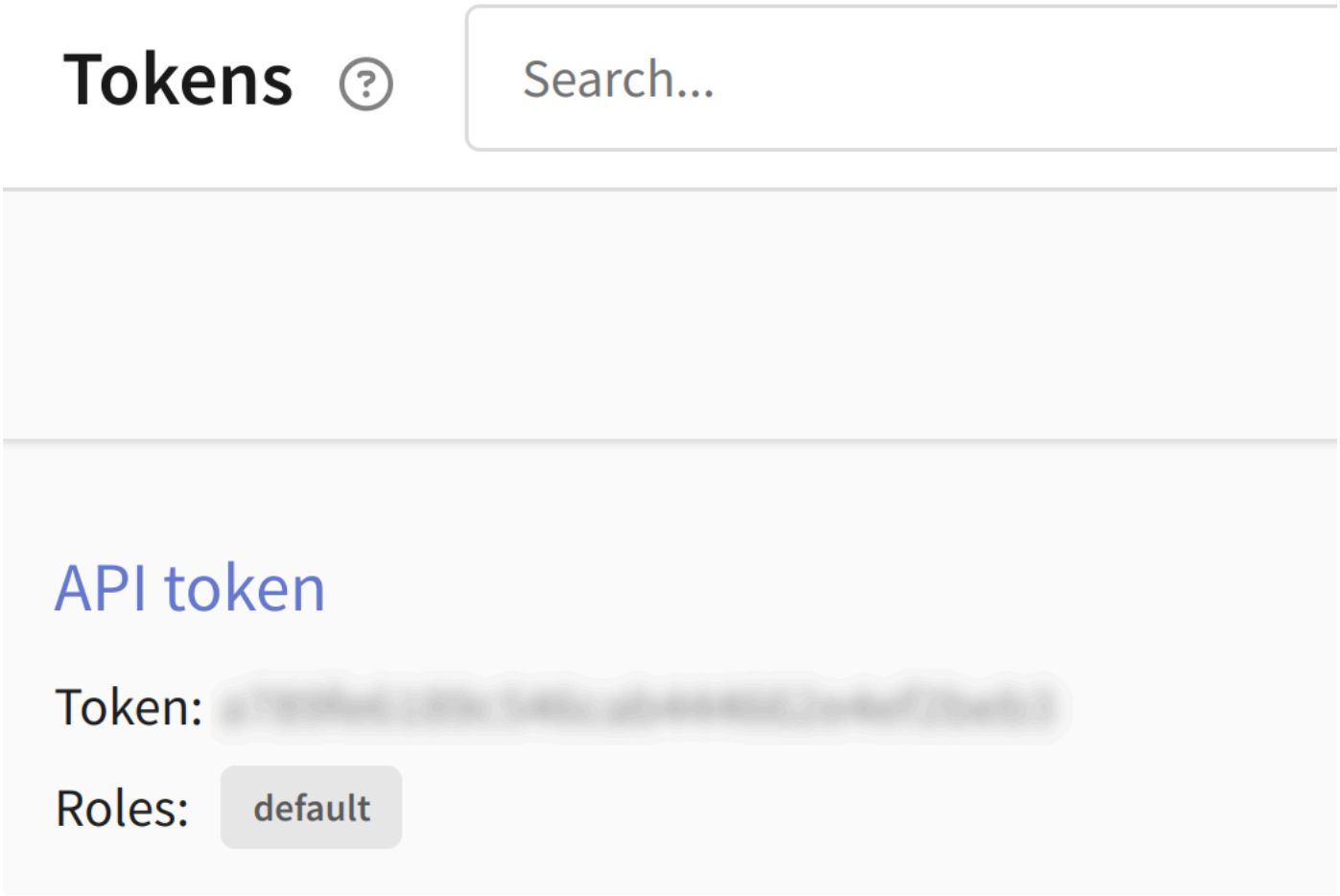
This article will show you how to write and submit API requests to perform the following tasks:

- **Adding a profile to the database**
- **Updating profile data**
- **Subscribing a profile to different channels**

Getting an API token

API token is a key to access the platform via API. It is used in any API request.

You can find the token in Altcraft platform settings, in the Tokens section. If you don't have a token yet, cr



CAUTION

Only master users can create an API token. If you do not have rights to create a token, contact a user wi

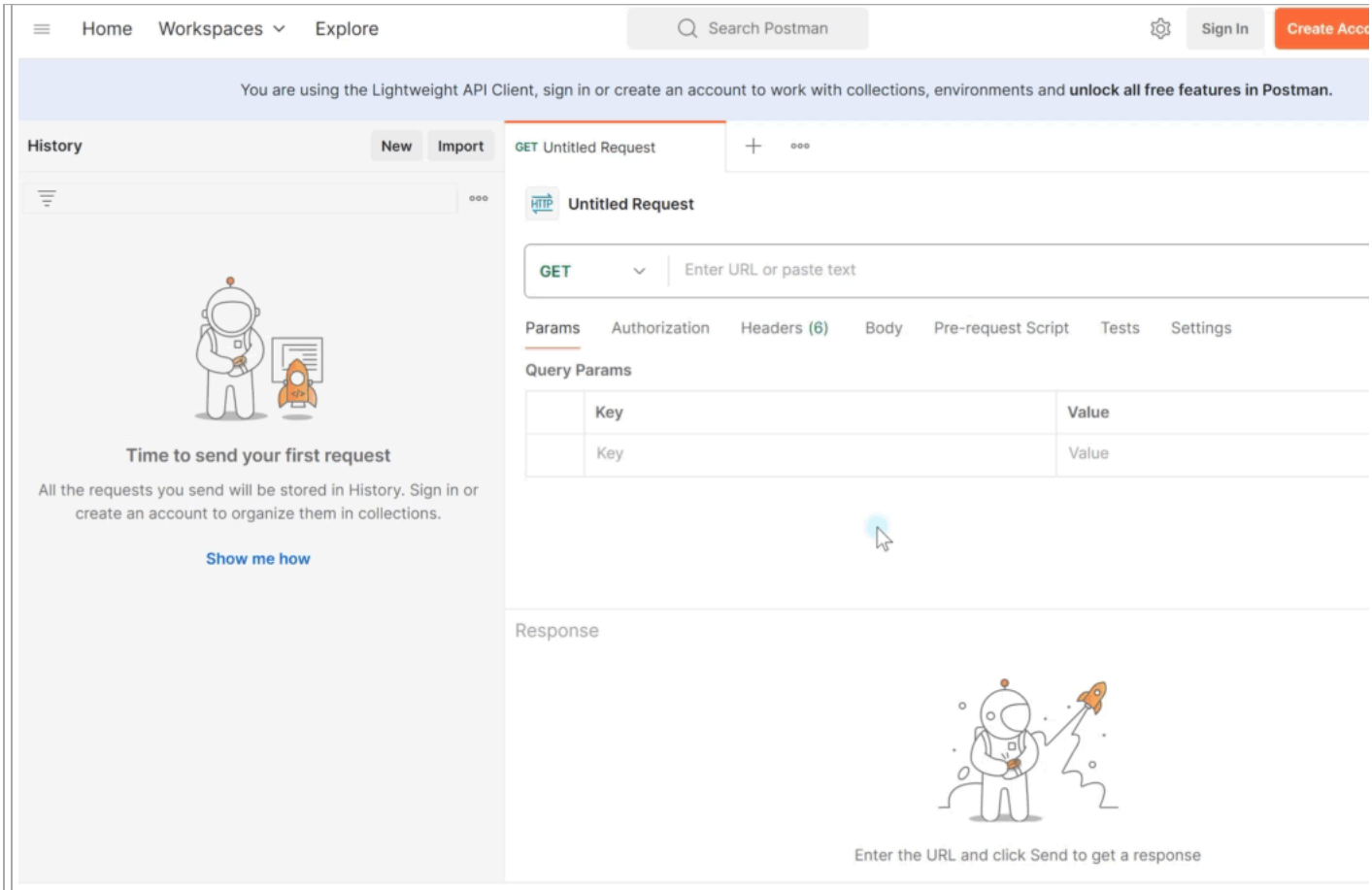
Installing and running Postman

To work with API requests, you need an application that allows you to send and receive them.

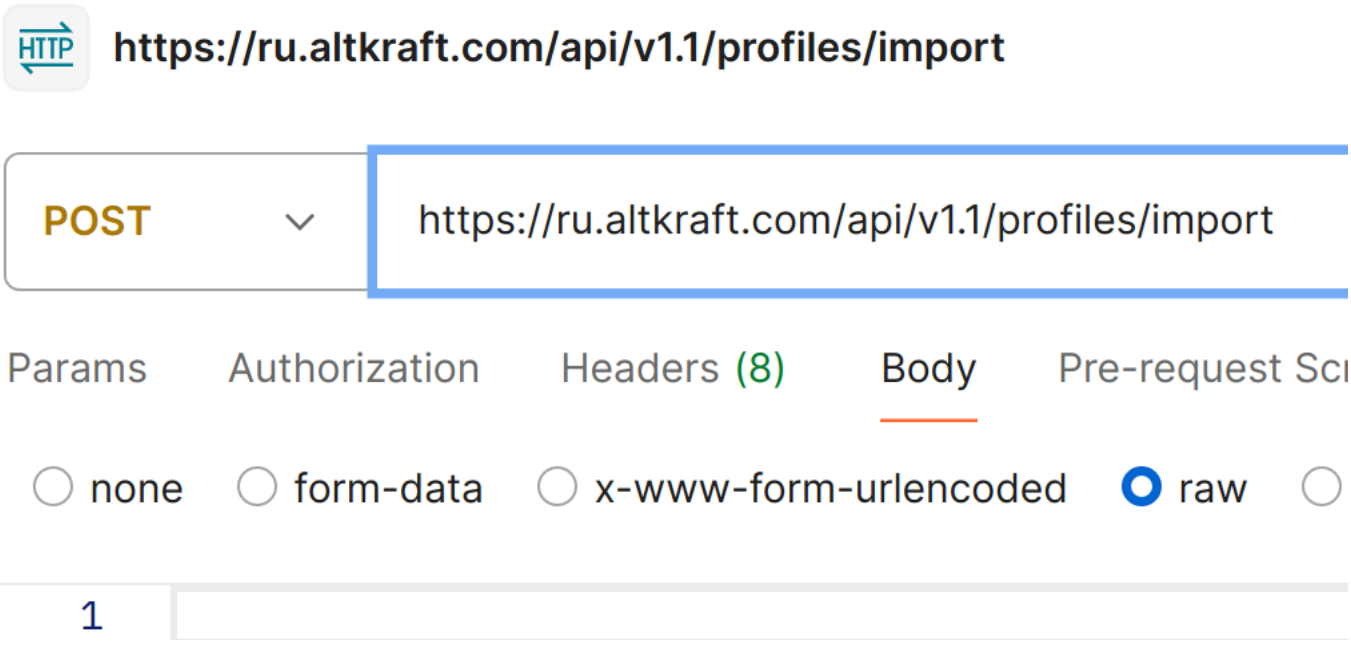
1. Download and install the Postman application or open its web version.

2. Select the POST method for your request.

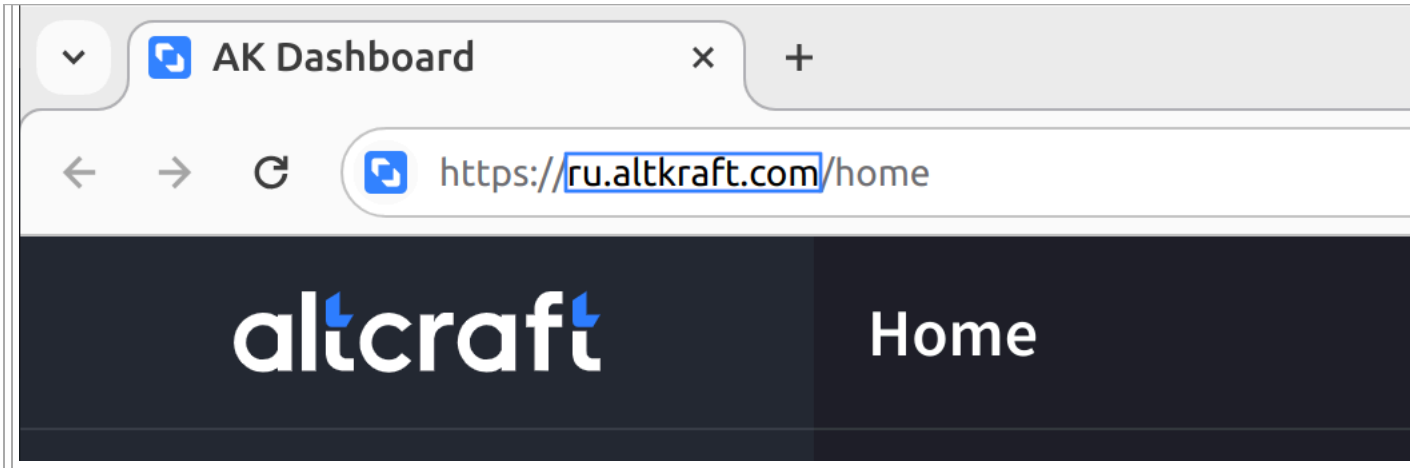
3. Click the Body tab and select raw mode. Make sure the JSON format is selected.



Request URL



To send a request to the server you need to enter its URL. It is individual for each request. For example, fo
`https://example.com/api/v1.1/profiles/import`.
Instead of `example.com` **you need to specify the address where the platform is located. You can get it direct**



Creating a request

Adding a new profile

Let's move on to writing the query. Suppose we want to manually add a customer profile with e-mail `i.petr`

```
{
    "token": "abcdefghklmnopqrstzxc",
    "db_id": 5,
    "matching": "email",
    "email": "i.petrov@example.com",
    "data": {
       "email": "i.petrov@example.com"
    }
}
```

- ****token ** - API token. You can read more about where to find it at the beginning of the article.**
- ****db_id ** - number of the database to which the profile is added. This number is indicated next to the**
- ****data ** - the object that contains the profile data. In the example above, we only pass the client's er**
- ****matching ** and **email ** - parameters by which the query will determine whether this profile exis new profile. The "matching" parameter can take different values. However, it is not mandatory and if i**

More details about the `matching` parameter

The `matching` parameter defines by which field the profile will be searched in the database. Several search
The main thing is that the field by which you launch a search in the database is unique for the profile, i.e.
If you want to find a profile by its email, then `matching` will look like this:

```
{
    "matching": "email",
    "email": "example@example.com"
}
```

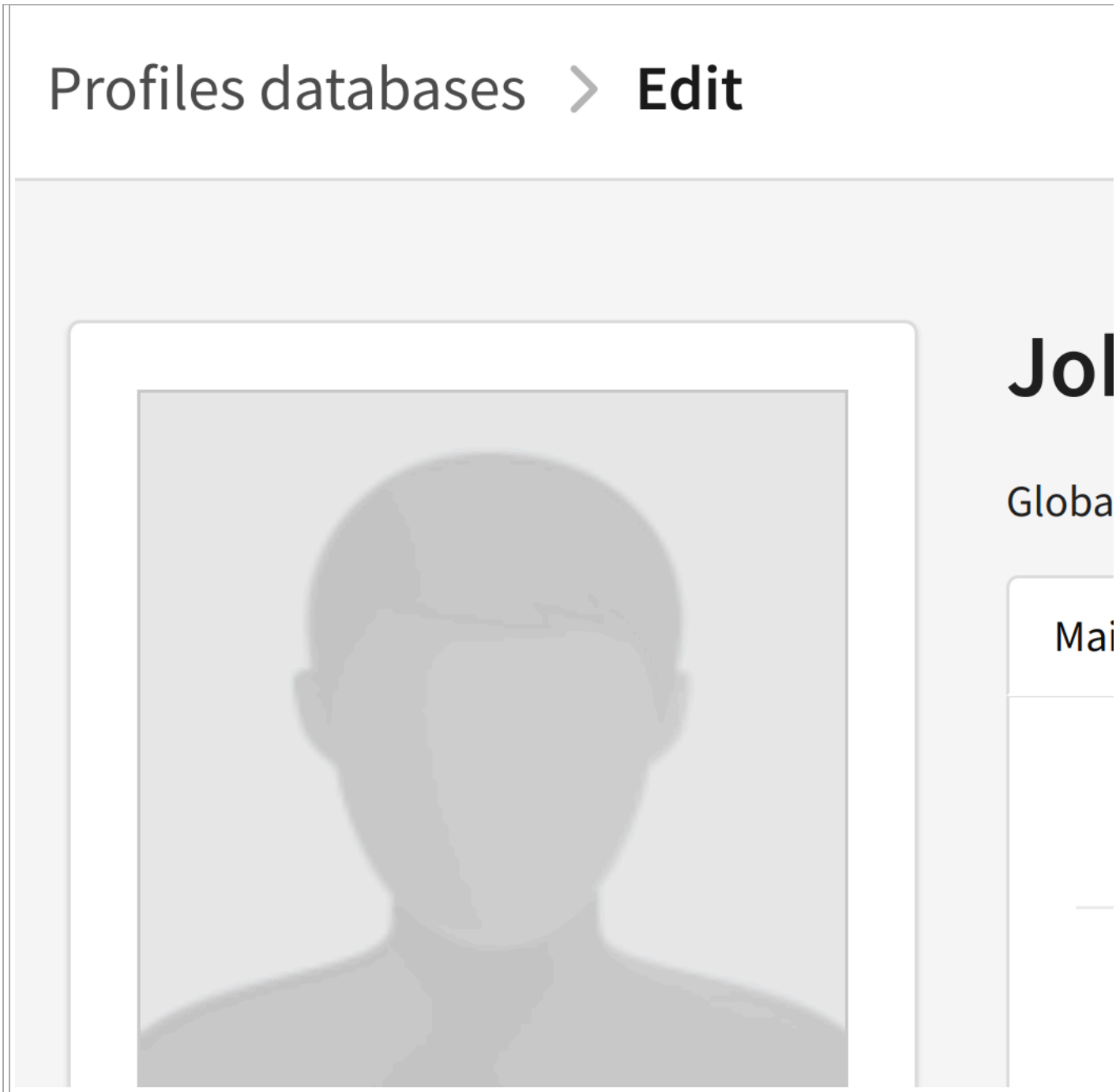
Search for a profile by phone::

```
{
    "matching": "phone",
    "phone": "+79000000000"
}
```

Search for a profile by ID (only when updating a profile):

```
{
    "matching": "profile_id",
    "profile_id": "012452gf62fcd71fjh8ec0"
}
```

The profile ID can be found here:



If you use a custom field to search for a profile, then your query will look like this:

```
{
    "matching": "custom",
    "field_name": "CRM_ID",
    "field_value": "12345"
}
```

where `CRM_ID` is the name of the field, and `12345` is the value of this field for the profile.

Updating profile data

Let's assume that we have new data about the client: his name is Ivan Petrov, born 15.01.1990, lives in Mo

```
{
     "token": "abcdefghklmnopqrstzxc",
     "db_id": 5,
     "matching": "email",
     "email": "i.petrov@example.com",
     "data": {
        "email": "i.petrov@example.com",
        "_fname": "Ivan",
        "_lname": "Petrov",
        "_sex": "male",
        "_bdate": "1990-01-15T12:00:00Z",
        "_city": "Moscow",
        "phones": ["+79000000000"]
    }
}
```

We have extended the `"data"` parameter with the following data:

- **_fname - first name**
- **_lname - last name**
- **_bdate — date of birth in the format “1990-01-15T12:00:00Z”**
- **_sex - gender, “male” for men, “female” for women**
- **_city - city**
- **phones — phone number.**

Adding subscriptions to a profile

We have updated the profile and now we need to subscribe it to email, sms and push channels in resource

```
{
    "token": "abcdefghklmnopqrstzxc",
    "db_id": 5,
    "matching": "email",
    "email": "i.petrov@example.com",
    "data": {
      "email": "i.petrov@example.com",
      "_fname": "Ivan",
      "_lname": "Petrov",
      "_sex": "male",
      "_bdate": "1990-01-15T12:00:00Z",
      "_city": "Moscow",
      "phones": ["+79000000000"],
      "subscriptions": [
            {
                "channel": "email",
                "email": "i.petrov@example.com",
                "resource_id": 1
            },
            {
                "channel": "sms",
                "phone": "+79000000000",
                "resource_id": 1
            },
            {
                "channel": "push",
                "provider": "Firefox",
                "resource_id": 1,
                "subscription_id":"abcdefghijklmnqrstuvwxyz"
            }
        ]
    }
}
```

For each communication channel, the following parameters must be specified:

- **channel — type of channel, for example, email, sms, push**
- **resource_id is the resource ID that you must configure before. Also, depending on the channel you are to specify the provider (`provider`) and the push token (`subscription_id`).**

> **TIP**
>
> **Pay attention to the syntax of your request. Any extra character, be it a parenthesis, comma or period, c**

Sending a request

To send a request, click Send in Postman. If the request is successful, the following message will appear:

Pretty Raw Preview Visualize JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "profile_id": "66267f05ecc3993fba680df7"
}
```

If something went wrong, the error code and its description will be indicated in the reply message:

Pretty Raw Preview Visualize JSON

```
{
    "error": 400,
    "error_text": "Invalid request"
}
```

If you are unable to resolve the error yourself, please forward the error code and description to Altcraft Su

TIP

This article describes only some of the possible parameters for this request. The remaining parameters

Trigger mailing launch

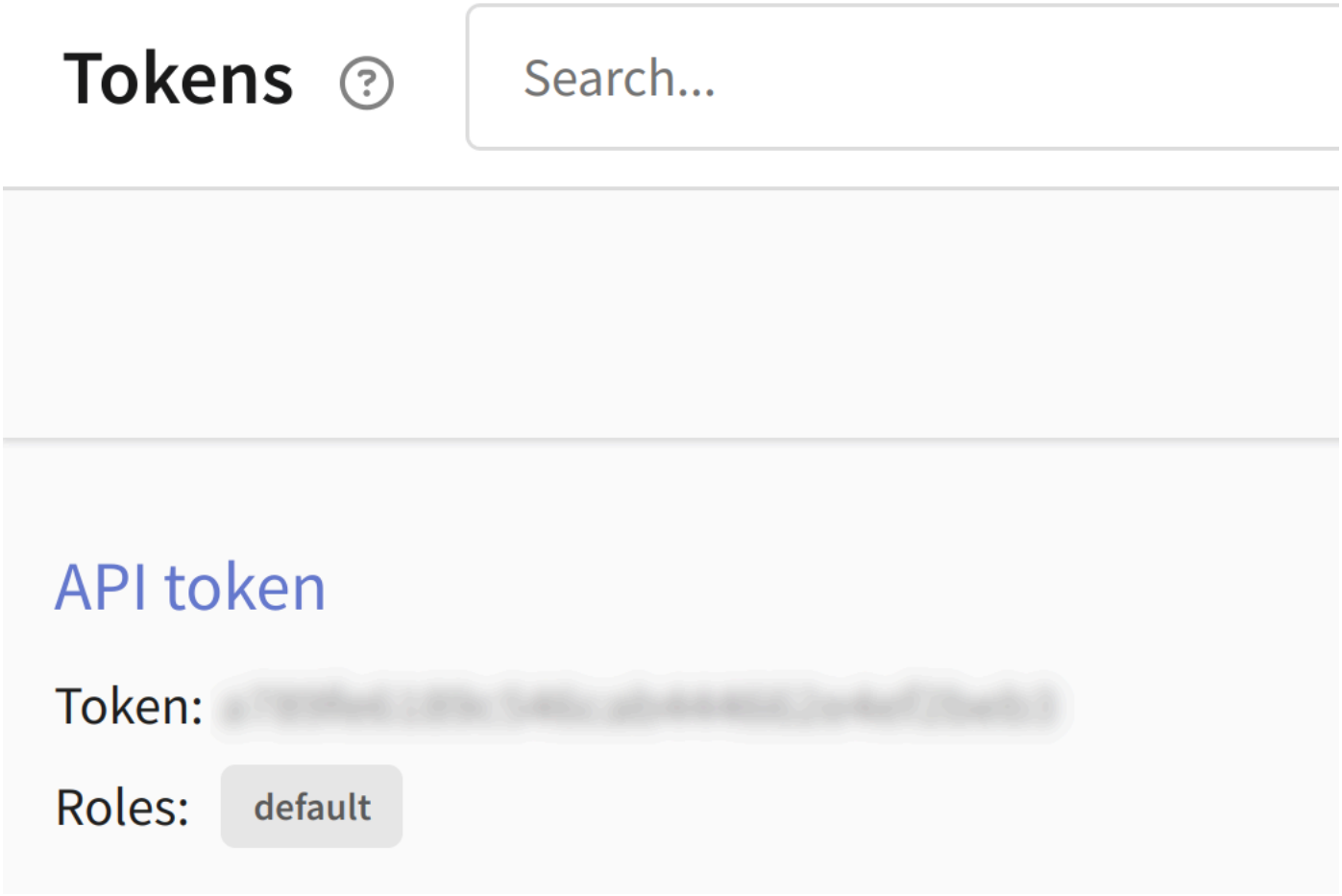
In this article, we will show you how to run a trigger mailing through the API. Let's consider two variants o

- **Sending a message to a single customer**
- **Launching a mailing for multiple subscribers**

> TIP
>
> **Only active trigger mailings set to the "When using a special API call" event can be run via the API. In thi**

Getting an API token

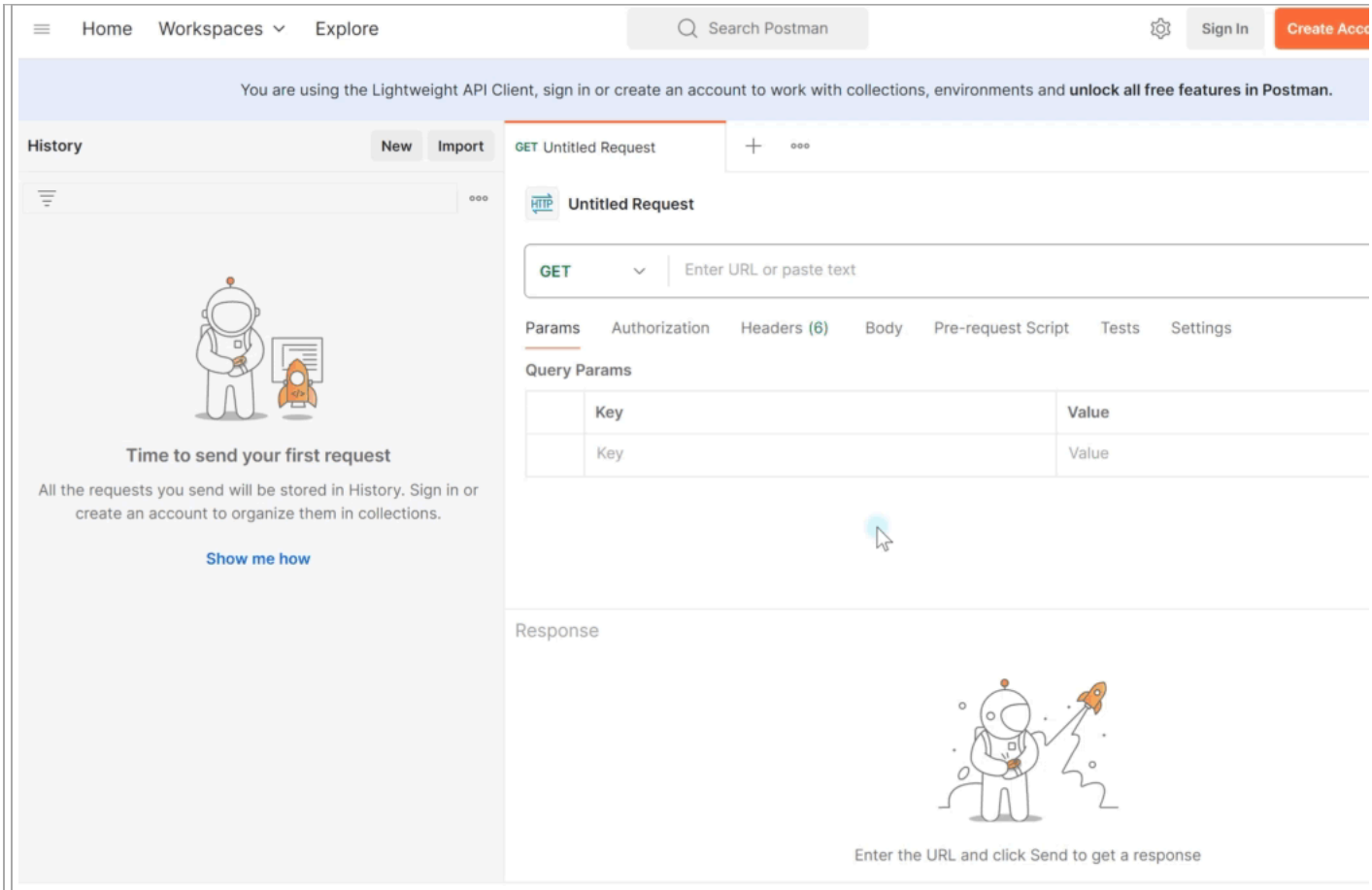
API token is a key to access the platform via API. It is used in any API request.
You can find the token in Altcraft platform settings, in the Tokens section. If you don't have a token yet, cr



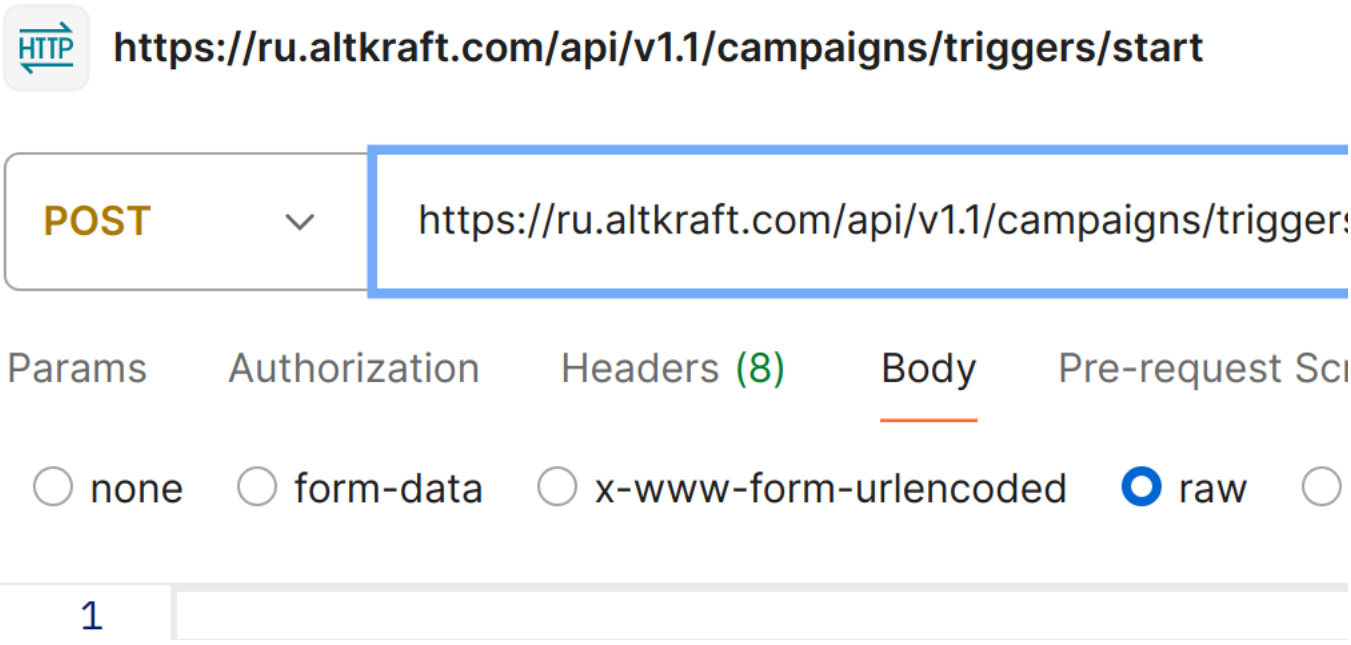
> CAUTION
>
> **Only master users can create an API token. If you do not have rights to create a token, contact a user wi**

Installing and running Postman

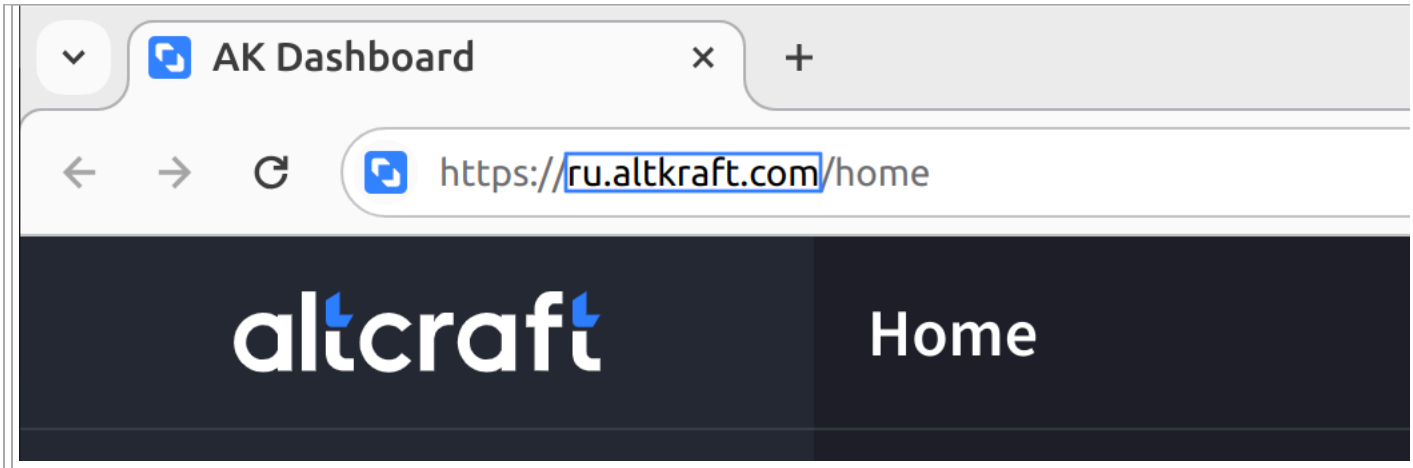
To work with API requests, you need an application that allows you to send and receive them.
1. Download and install the Postman application or open its web version.
2. Select the POST method for your request.
3. Click the Body tab and select raw mode. Make sure the JSON format is selected.



Request URL



To send a request to the server you need to enter its URL. It is individual for each request. For example, fo
`https://example.com/api/v1.1/campaigns/triggers/start`.
Instead of `example.com` **you need to specify the address where the platform is located. You can get it direct**



Creating a request

Launching a mailing for one user

Let's move on to writing the query. Suppose there is a subscriber with the email address `zaycev@example.com`

```
{
  "token": "abcdefghklmnopqrstzxc",
  "id": 1,
  "matching": "email",
  "email": "zaycev@example.com"
}
```

- **token — API token. More details about where you can find it are provided at the beginning of the artic**
- **id is the number of the trigger mailing. It is shown next to the mailing name to the right of the # sign.**
- **matching and email - parameters by which the query will find the required profile in the database. In t**

More details about the `matching` parameter

The `matching` parameter defines by which field the profile will be searched in the database. Several search
The main thing is that the field by which you launch a search in the database is unique for the profile, i.e.
If you want to find a profile by its email, then `matching` will look like this:

```
{
     "matching": "email",
     "email": "example@example.com"
}
```

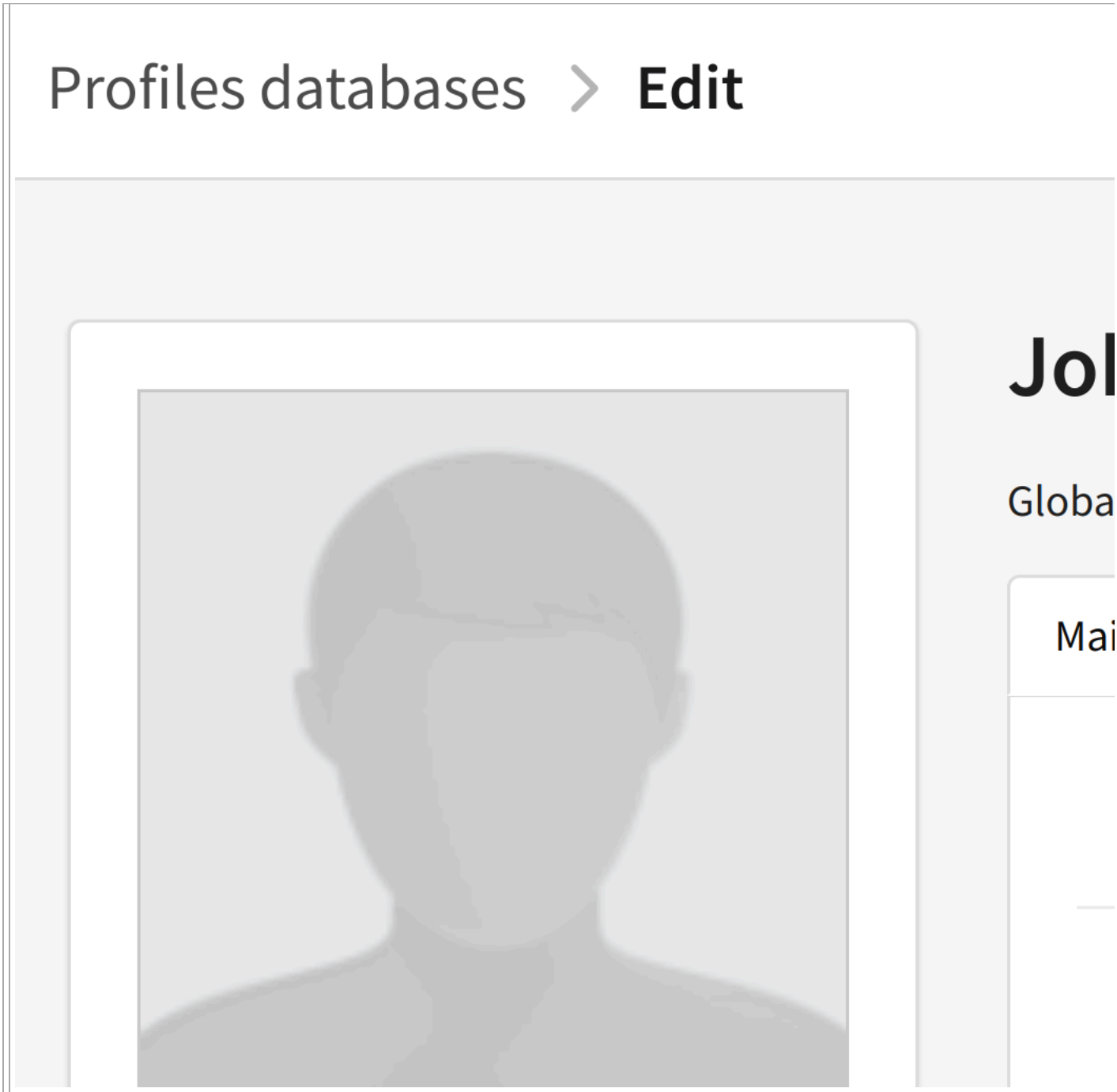
Search for a profile by phone:

```
{
     "matching": "phone",
     "phone": "+79000000000"
}
```

Search for a profile by ID (only when updating a profile):

```
{
     "matching": "profile_id",
     "profile_id": "012452gf62fcd71fjh8ec0"
}
```

The profile ID can be found here:



If you use an additional field to search for a profile in the database, your query will look like this:

```
{
     "matching": "custom",
     "field_name": "CRM_ID",
     "field_value": "12345"
}
```

where `CRM_ID` is the name of the field, and `12345` is the value of this field for the profile.

Launching a mailing for subscriber with custom field

Let's say we want to send a message with a new product catalog to the person who liked a VKontakte po

```
{
  "token": "abcdefghklmnopqrstzxc",
  "id": 5,
  "matching": "custom",
  "field_name": "vk_id",
  "field_value": "123456"
}
```

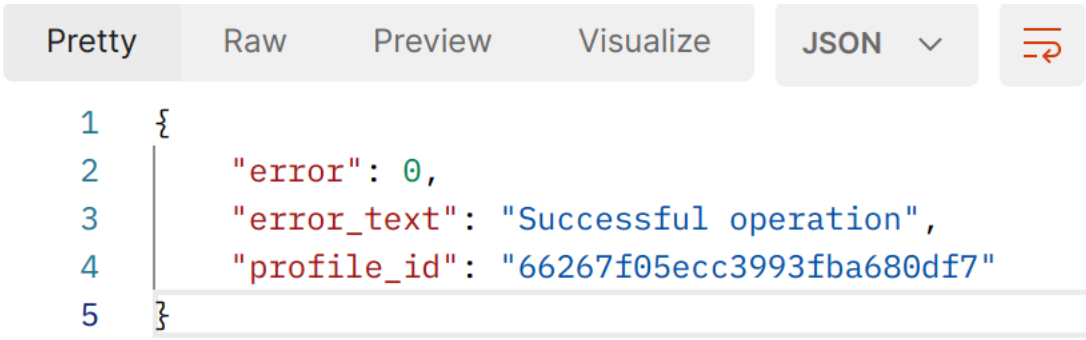
The search for profiles will search for an additional field, since we specified `custom` in `matching`. The name are interested in, so the message will be sent only to him.

TIP

Pay attention to the syntax of your request. Any extra character, be it a parenthesis, comma or period, ca

Sending a request

To send a request, click Send. If the request is successful, the following message will appear:



If something went wrong, the response message will indicate the code of error and its description:

If you are unable to resolve the error yourself, please forward the code of error and its description to Altc

TIP

This article describes only some of the possible parameters for this request. The remaining parameters

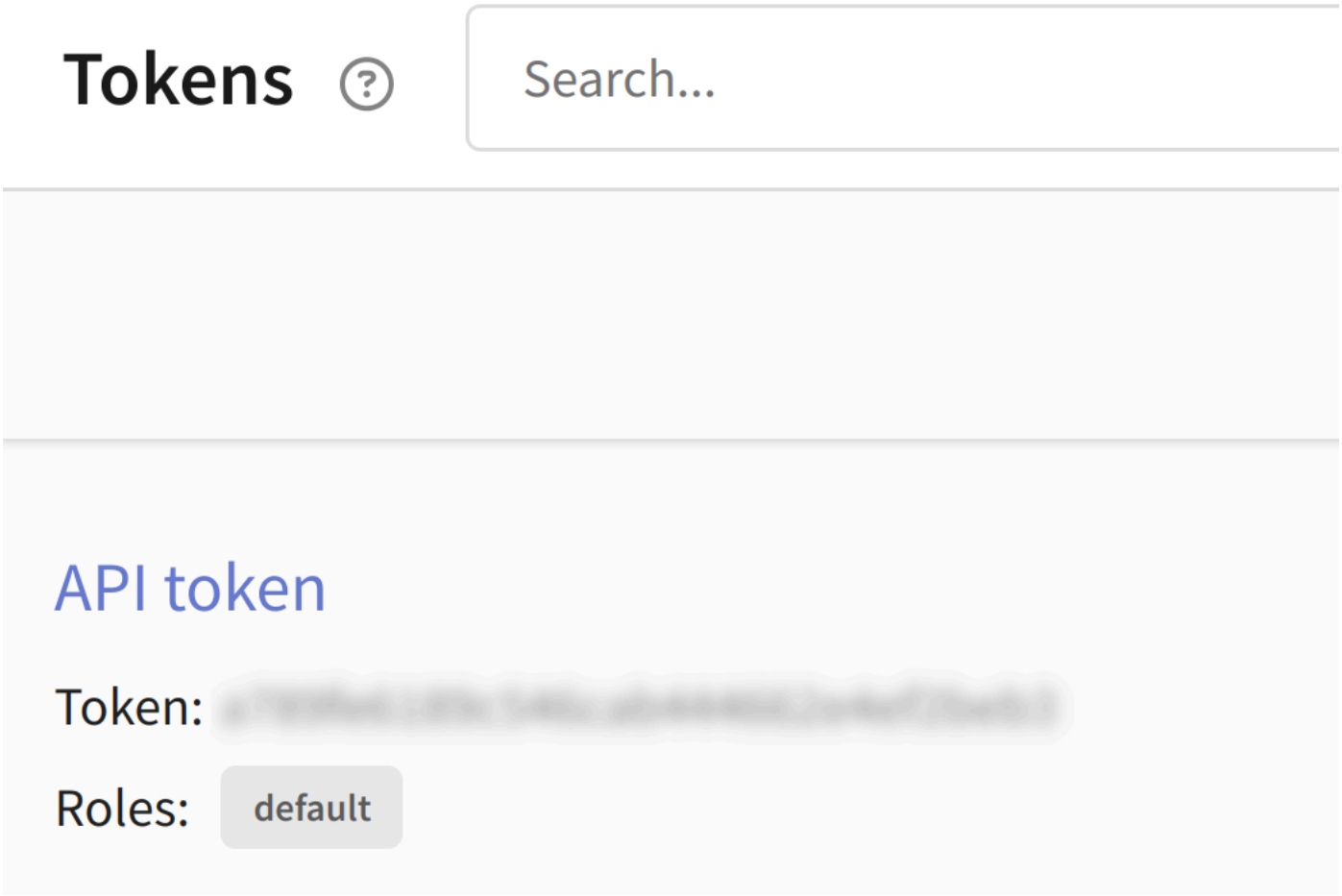
Engage profile in scenario

In this article we will explain how to use an API request to send to the script a profile that already exists i

> TIP
> **In order to run a profile in a scenario, the scenario must be created and activated. In this article you can**
> **Note that the profiles in this run are included in the script via the Default **node.**

Getting an API token

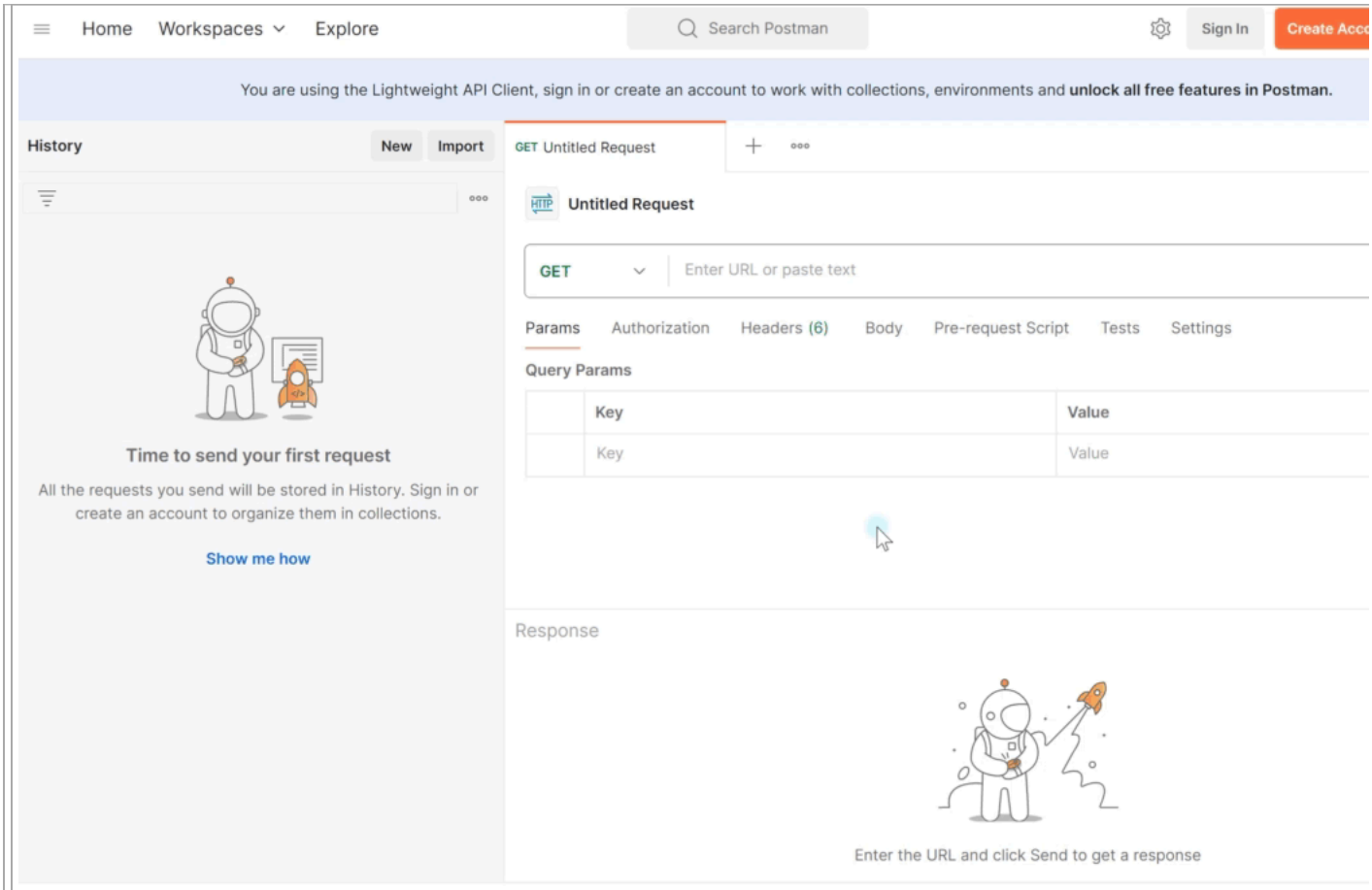
API token is a key to access the platform via API. It is used in any API request.
You can find the token in Altcraft platform settings, in the "Tokens" section. If you don't have a token yet,



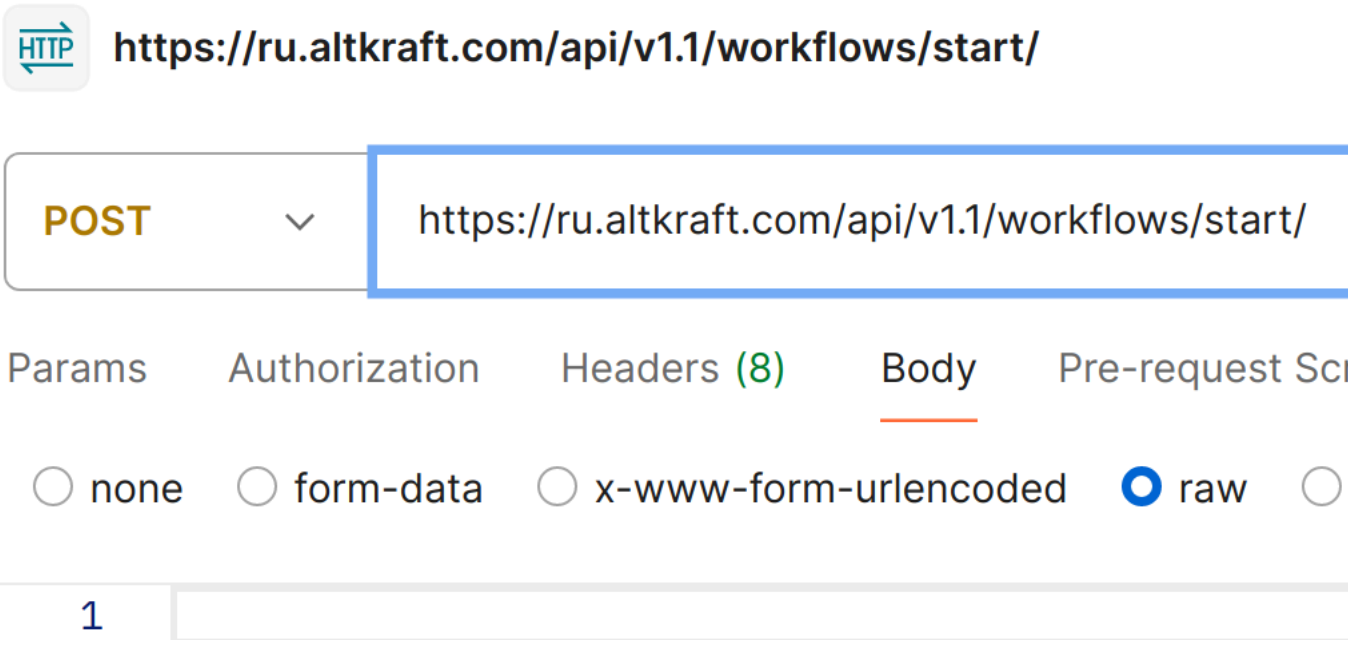
> CAUTION
> **Only master users can create an API token. If you do not have rights to create a token, contact a user wi**

Installing and running Postman

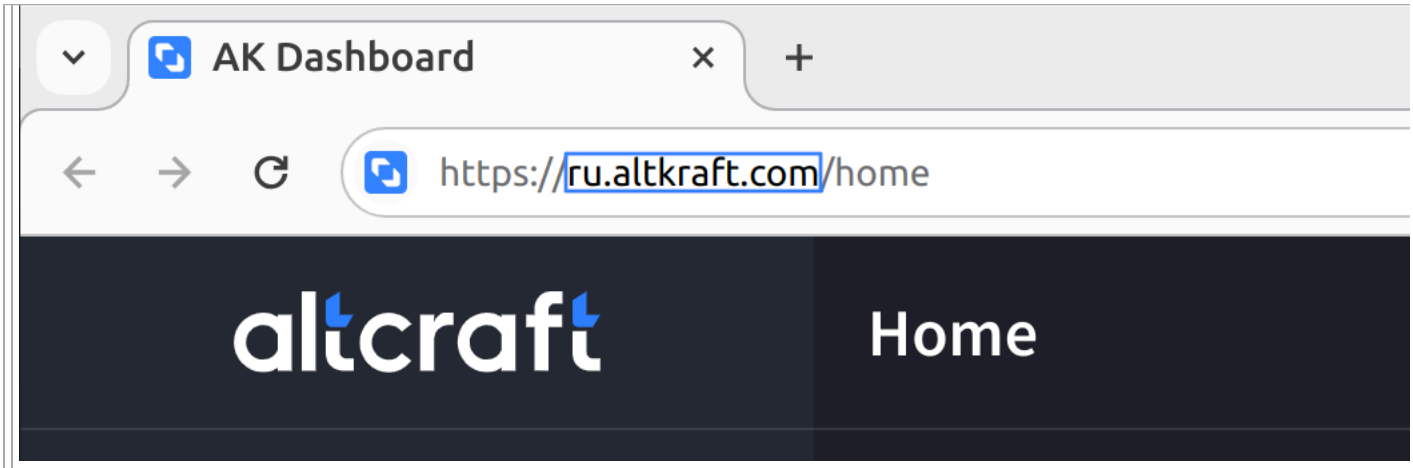
To work with API requests, you need an application that allows you to send and receive them.
1. Download and install the Postman application or open its web version.
2. Select the POST method for your request.
3. Click the Body tab and select raw mode. Make sure the JSON format is selected.



Request URL



To send a request to the server you need to enter its URL. It is individual for each request. For example, fo
`https://example.com/api/v1.1/workflows/start/`.
Instead of `example.com` **you need to specify the address where the platform is located. You can get it direct**



Creating a request

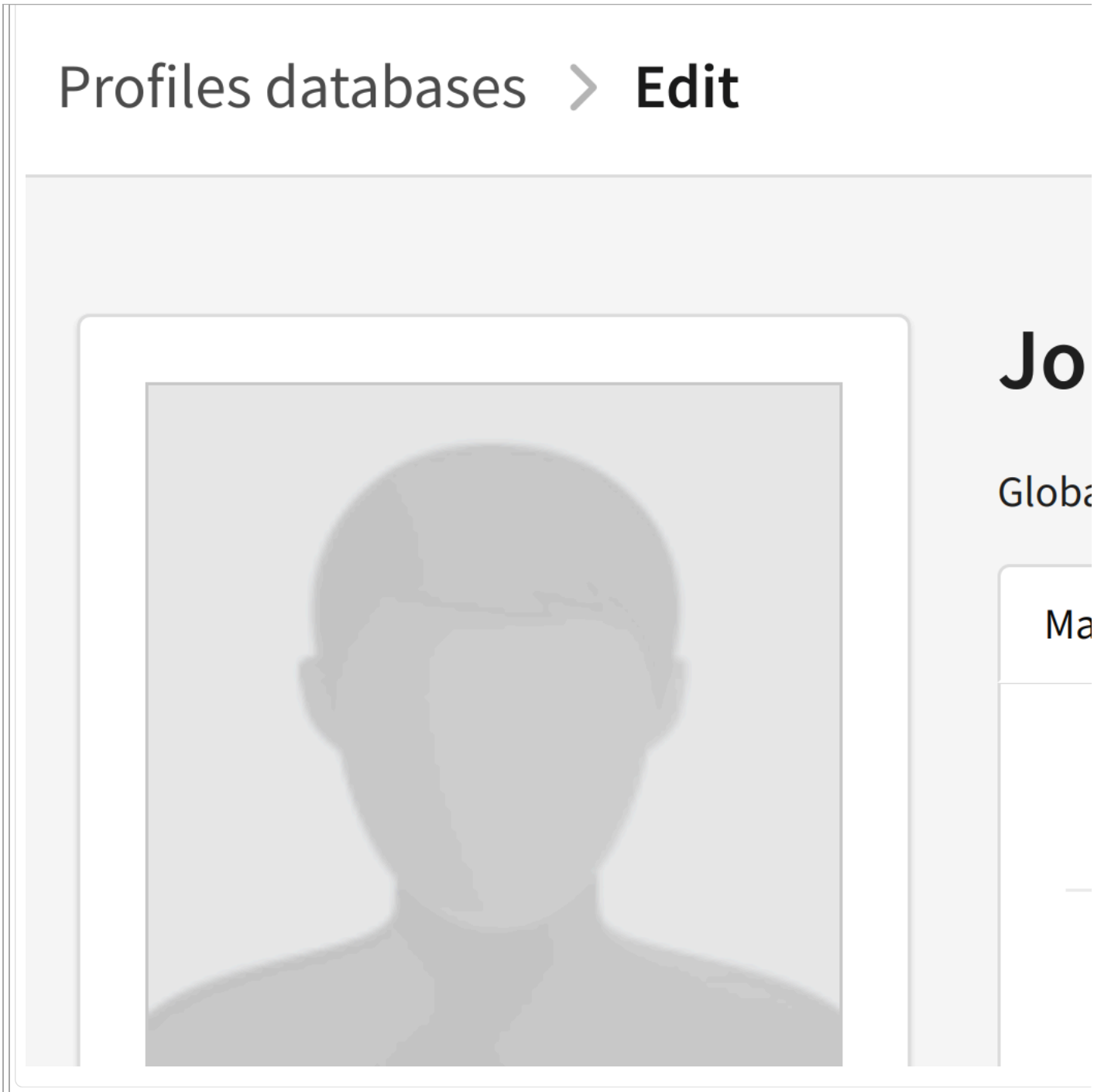
Let’s move on to writing the query. Suppose we want to send a subscriber with ID `66acb337240d5213037986d8` t

```
{
  "token": "abcdefghklmnopqrstzxc",
  "id": 3,
  "matching": "profile_id",
  "profile_id": "66acb337240d5213037986d8"
}
```

- **token — API token. More details about where you can find it are provided at the beginning of the artic**
- **id — scenario number. It is listed with the # sign next to the scenario name. The `id` of a scenario is a**
- **matching and profile_id - parameters by which the query will find the required profile in the database.**

INFO

The profile ID can be found here:



More details about the `matching` parameter

The `matching` parameter defines by which field the profile will be searched in the database. Several search

The main thing is that the field by which you launch a search in the database is unique for the profile, i.e.

If you want to find a profile by its email, then `matching` will look like this:

```
{
    "matching": "email",
    "email": "example@example.com"
}
```

Search for a profile by phone:

```
{
    "matching": "phone",
    "phone": "+79000000000"
}
```

Search for a profile by ID (only when updating a profile):

```
{
    "matching": "profile_id",
    "profile_id": "012452gf62fcd71fjh8ec0"
}
```

If you use an additional field to search for a profile in the database, your query will look like this:

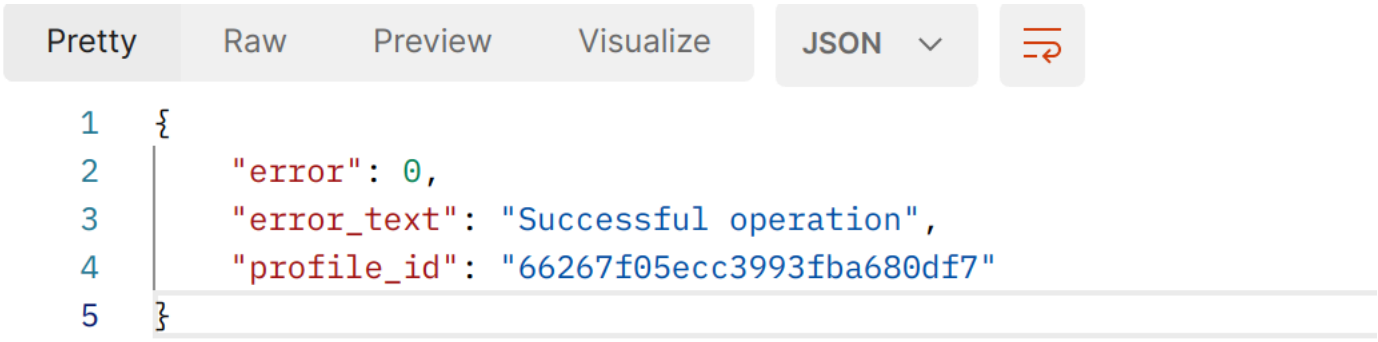
```
{
    "matching": "custom",
    "field_name": "CRM_ID",
    "field_value": "12345"
}
```

where `CRM_ID` is the name of the field, and `12345` is the value of this field for the profile.

> **TIP**
>
> **Pay attention to the syntax of your request. Any extra character, be it a parenthesis, comma or period, c**

Sending a request

To send a request, click Send. If the request is successful, the following message will appear:



If something went wrong, the response message will indicate the number and description of the error:

If you are unable to resolve the error yourself, please forward the error number and description to Altcraft

> **TIP**
>
> **This article describes only some of the possible parameters for this request. The remaining parameters**

Changelog

INFO: **Platform update policy**

Updates are distributed to users based on their requests. First of all, they become available to those wh
After receiving a new build, it is recommended to perform the update within 14 calendar days. Updates i
Fixes and updates are only released for the current version of the platform. Outdated versions are not su

TIP

The platform version number indicates the release date and scope of the changes. For example, version

v2026.2.77

v2026.2.77.189 — IDigital Direct API integration and improvements

IDigital Direct API SMS provider integration

This update adds support for a new SMS provider — IDigital Direct API.
When configuring a sender for this provider, you need to specify a token for API authorization.

Supported:

- **Request format — JSON array (batch sending up to 1000 messages)**
- **Callback statuses:** `delivered`, `undelivered`, `unsent`
- **Retry logic for HTTP codes 429, 500, 502, 503 (exponential backoff)**

If needed, the sending URL can be overridden via the `SMS_IDIGITAL_CUSTOM_SENDING_URL` **configuration paramete**

Improved

- **Changed the sorting library, which accelerated the recalculation of static segments.**
- **Improved error messages when calculating a global control group with stratification.**
- **Optimized the PostgreSQL database process, simplifying compliance with information security require**
- **Added the ability to automatically clean up old files in the** `download_taskreport` **directory.**
- **Added localized messages for new error types in the mailing log.**
- **The scenario editor performance has been improved.**
- **Improved interface — the sender queue view button is now only displayed for Email channels.**
- **Improved interface — the list of product segments now correctly shows the user who last updated the**

Other

- **Fixed the passing of** `custom_data` **fields to event capture when launching a scenario via the** `workflows/sta`
- **In the mailing testing section, a test profile is now created only if there is at least one subscription.**
- **When deactivating a global control group, the ability to remove profiles from the group has been adde**
- **Fixed the execution of the** `point_trc_migrate` **points transaction migration for ClickHouse in replicated r**
- **Importing templates via archive no longer results in** `http` **being substituted into image links, reducing**
- **In scenarios, when configuring a "Scheduled segment", the selected launch month is now correctly sa**
- **Fixed error display when no profiles match the stratification conditions in the global control groups in**
- **Fixed the default value substitution for numeric fields using a predefined list of values in custom char**
- **Fixed an issue where profiles with a large number of additional fields were not added to the loyalty pr**
- **In the "Update static segment" scenario node, only segments from the current profile database are no**
- **In analytics reports, deleted loyalty program group levels no longer appear as empty lines — their stat**
- **Fixed an issue where profiles were not captured into a scenario when switching the trigger condition**
- **When creating indexes in the admin panel, the** `_xxh` **field can now be manually selected from the gene**
- **When changing the schedule type in broadcast mailings, incompatible options are no longer saved.**
- **In segments, user fields are no longer replaced by system fields when creating selection conditions.**
- **Fixed the display of** `procmarket` **and** `webmarket` **process versions.**
- **Added new conditions to segments for selecting profiles based on exclusion from the global control g**
- **Fixed the issue with selecting resources and segments in segment conditions for users with limited p**
- **Fixed an issue where only numeric values were written to a field of type "List" when importing profile**
- **Fixed profile capture into a scenario when activating a pixel with the "contains" or "does not contain"**
- **Improved the event capture interface for custom channels: eliminated the ability to select an event th**
- **Fixed the issue with campaign activation when one of the MongoDB replica members is unavailable.**
- **Attribute fields with editing restrictions in active objects are now correctly disabled after activation.**
- **Fixed the "for the last X days" condition in segments: value 1 now correctly selects events only for th**
- **Fixed slow processing of the** `hook_events` **queue when capturing events with message content.**
- **Fixed an issue where synchronization of static segments with Facebook Ads Manager did not work wh**
- **When sending a broadcast mailing by profile time zone, if no matching profiles are found, the message**
- **Fixed the display of field headers in segment conditions — the headers defined in the segmentation q**

References

v2026.2.77: PL-18253 PL-19527 PL-20943 PL-20994 PL-21310 PL-21505 PL-21515 PL-21579 PL-21610 PL-21 21909 PL-21922 PL-21929 PL-21931 PL-21934 PL-21937 PL-21949 PL-21967 PL-21970 PL-21988 PL-22007 P 21953 PL-21713 PL-21847 PL-20466 PL-19945

v2026.1.76

v2026.1.76 — MAX integration, Loyalty Programs & Segments optimization

Migration

> **DANGER**
> **Migration is required for the platform to function correctly.**

To perform the migration, follow these steps:

1. **Replace the platform files with the files from the new build.**
2. **Set the `CH_READ_ACTIONS: true` parameter in the configuration file on ALL hosts.**
3. **Run `./akd upgrade` and confirm the migration.**
4. **Set the `CH_READ_ACTIONS: false` parameter in the configuration file on ALL hosts.**
5. **Fully restart the platform using the `./akd restart` command.**

The migration is resource-intensive and requires updating all profiles across all account databases. Mong

---

Additionally, in this update, all active mailings that do not use a resource will stop working.
To ensure mailings continue to work correctly, create a new or select an existing resource and assign it i
A list of such mailings can be obtained using the command `./aktool mailings upgrade-blocker-list` or `./aktool m`

MAX messenger integration

This update introduces support for a new communication channel — integration with the MAX messenger.

- **MAX Bot — for sending personalized messages to users**
- **MAX Group — for sending messages to group chats**

You can create a message template for the MAX messenger using the built-in template editor. A template

- **Text.**
- **Image.**
- **Video.**
- **Audio.**
- **File.**

You can also immediately add interaction buttons for recipients:

- **Text button — displays a button with text.**
- **Link button — opens a specified link.**
- **Callback button — returns a specified text or command to the bot after clicking.**
- **Open application — opens a mini-application.**
- **Request location — requests the user's location.**
- **Request contact data — requests the user's contact details and phone number.**
- **Button group — a combination of buttons of several types.**

The channel can be used in mailings, scenarios, and campaigns, and analytics is also available:

Segmentation Optimization

Processing of exists / does not exist conditions for channel events (send, open, click, etc.) has been optim

- **up to 90x faster for simple conditions.**
- **up to 8x faster for complex conditions with multiple filters.**

The algorithm for recalculating static segments based on dynamic conditions has also been optimized, re
To use the accelerated segmentation mechanism, complete the migration steps described at the beginnin

Other

- **When importing into the stop list, a report with rejection reasons is now displayed (similar to databas**
- **Added the ability to update object attributes (for example, completed Mailings).**
- **Fixed an issue where Mailings inside a campaign were not sent when using dynamic segments.**
- **Fixed an issue with `procloyalty` startup on dedicated servers.**
- **Product segments can now be used in order filters.**
- **Added logging for Web Push notifications in Safari.**
- **Fixed an issue where links did not work in preview for Telegram bot templates and Custom channels.**
- **Improved loyalty program analytics interface.**
- **Fixed an issue where it was impossible to add a Mailing via a Custom channel after cloning a Scenario**
- **Optimized performance of loyalty program APIs.**
- **Fixed event duplication in ClickHouse. Added deduplication and reversal mechanisms for duplicate ev**

- Fixed an issue with profile entry into a Scenario when using scheduled segments with the “in dynamic
- Fixed an issue where profiles returned to a previous node if the “Condition” node was unavailable.
- Fixed an issue where static segments with conditions were not recognized as updatable.
- Profiles now correctly exit the “Wait for event” node in Scenarios.
- Fixed incorrect display of Scenario IDs in condition nodes.
- Attributes can now be selected in segment or RTG setup even if no value is assigned.
- Deactivated and hierarchically assigned attributes are now available for viewing in entity objects.
- Fixed issues related to form creation and configuration.
- Fixed an issue where content in the event capture storage was saved when it should not have been.
- Fixed an issue when creating a transaction with points accrual and a 5-year expiration.
- Added message lifetime (TTL) configuration. You can now define message validity in Smart Mailing se
- Segmentation by Custom channel events now works correctly.
- Fixed issues with parallel execution of identical tasks on different cluster hosts.
- Fixed an issue where scheduled segments did not start in Scenarios linked to Campaigns.
- Unsubscribe events via API are now correctly captured in event tracking.
- Link shortening in Telegram communications works correctly again.
- Reduced memory consumption of certain platform processes.

References

v2026.1.76.0: PL-21468 PL-16462 PL-21464 PL-21432 PL-21456 PL-21443 PL-20569 PL-21441 PL-21439 PL-2 21357 PL-20988 PL-21193 PL-21294 PL-21295 PL-17213 PL-21269 PL-21209 PL-21230 PL-21223 PL-21215 P

v2026.1.76.951 — Fixes and improvements

> CAUTION: Important
>
> **All platform processes now write logs exclusively in JSON format. If you use external log collection syst**

> INFO
>
> **This update changes the event registration logic in the `eventer` pipe for custom channels.**
> - **The `event_type` field is no longer used for system events (`read`, `open`, `click`, etc.). If the pipeline conta**
> - **The exception is the `stop`, `moderate_pass`, and `moderate_error` events, for which `event_type` remains requ**

Improved

- Base Docker image has been updated to Ubuntu 24.04.
- Optimized the performance of the webcontrol process when calculating dynamic segments.
- Optimized the operation of API, segments, mailings, and scenarios — read requests are now distribute
- Logs from all processes are now correctly written in JSON format.
- Fixed an issue where a disabled loyalty module continued to create queues.
- Fixed an issue where support chat history was lost after page reload.
- Added performance monitoring metrics for mailings.
- Added API handlers `loyalty/register_member_batch` and `loyalty/register_member_batch_task` for bulk profile imp

Other

- Fixed profile looping in the “Condition” node when no filters were present.
- Improved loading speed of the static and updatable segment editing page.
- Fixed an error with date substitution via variables in segmentation queries to external databases.
- Fixed an issue where synchronization of updatable segments with SQL databases was not working co
- Fixed recalculation of segments with segmentation queries containing “array” type parameters.
- Fixed an issue where segment search was not working when importing conditions.
- Calculations with “OR” conditions inside nested segments now work correctly.
- Fixed an issue where custom channel mailings were not sent when the template contained required fi
- Links in the preview of custom channel templates now open in a new window.
- Added link validation for buttons in Telegram templates — a template with an invalid link can no longe
- Fixed JSON validation when saving templates — templates with dynamic JSON containing variable su
- Variables in custom channel mailing fields are now correctly substituted.
- Fixed the resource filter in profile subscriptions — the filter no longer resets when deleting a subscrip
- Fixed an issue where a profile with a custom channel subscription could not be saved.
- Fixed an error in the “Condition” node when checking the total purchase amount in the Market module
- Conditions for selecting updatable product segments are now available in scenarios.
- Fixed column display in analytics.

- **Users without “Master” permissions can now save promo codes.**
- **Fixed matching by `profile_id` in scheduled imports.**
- **Fixed list filtering by attribute with the “Not set” value.**
- **Fixed minor UI issues related to the block editor and browser navigation.**

References

v2026.1.76.951: PL-21645 PL-21633 PL-21630 PL-21613 PL-21595 PL-21599 PL-21615 PL-21606 PL-21597 P
21184 PL-20897 PL-20634 PL-20940 PL-19867 PL-15846 PL-21484 PL-21530 PL-21528 PL-21522 PL-21496 P
21277 PL-21316 PL-21270 PL-21138 PL-21121 PL-19911 PL-20235 PL-17714 PL-21016 PL-20713 PL-20983 P

---

v2026.1.76.1578 — TTL for messages

Added

TTL for sent messages

Message time to live (TTL) is the period during which the system tries to send a message from the queue.
TTL should be used for campaigns that are only relevant for a limited time. For example, when a message
To configure TTL, enable the Limit message time to live (TTL) parameter in the campaign Smart settings a
After the specified time expires, the system stops trying to deliver the unsent message and registers a no

TIP: Details of using TTL for messages

- **The TTL value must be within the limits set by the `MTA_MIN_TTL` and `MTA_MAX_TTL` configuration paramet**
- **If TTL is set in several places, the value with the highest priority is applied: API parameter `expiration`**
- **TTL is calculated separately for each message from the moment it is placed in the queue. Therefore,**
- **After TTL expires, the system stops delivery attempts, even if the channel can still technically delive**
- **TTL does not work for channels that do not return message sending statuses: Telegram, MAX, and cu**

Improved

- **Added the “Segment participation” filter for product and SKU segments.**
- **Removed the limit of 30 values for a single attribute.**
- **Improved UI/UX components.**
- **Added deduplication for web events and mSDK events.**

Other

- **Connections to the MTA server no longer accumulate in the cache or cause performance issues.**
- **Fixed an error where the delete button for an active user was available in that user’s settings.**
- **Improved stability of the `procworkflow`, `api`, `webcontrol`, and `procloyalty` processes.**
- **Fixed some errors that occurred when calculating the GCG.**
- **Variables are now correctly substituted into URLs inside the “API call” scenario node.**
- **Fixed an error where the Play Mobile SMS gateway did not return campaign statuses.**
- **The “at least [n] times within a period” condition in segments now works correctly again.**
- **Fixed errors in the handler for updating profile fields.**

References

v2026.1.76.1578: PL-21638 PL-21815 PL-21755 PL-21800 PL-21803 PL-21784 PL-21772 PL-21765 PL-21754
21389 PL-10664 PL-21329 PL-21290 PL-21293 PL-21273 PL-18968 PL-21050 PL-20603 PL-20766 PL-20948 P

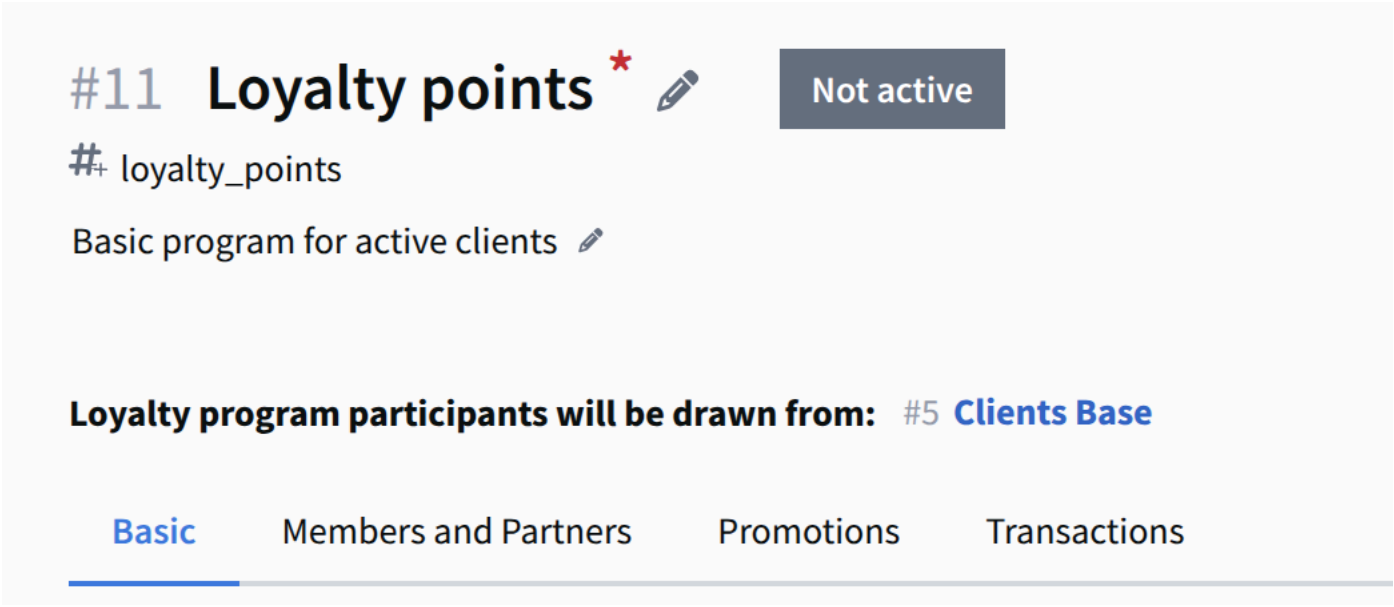
v2025.4.75

v2025.4.75.0 – Loyalty Programs, New Segmentation, UI Update

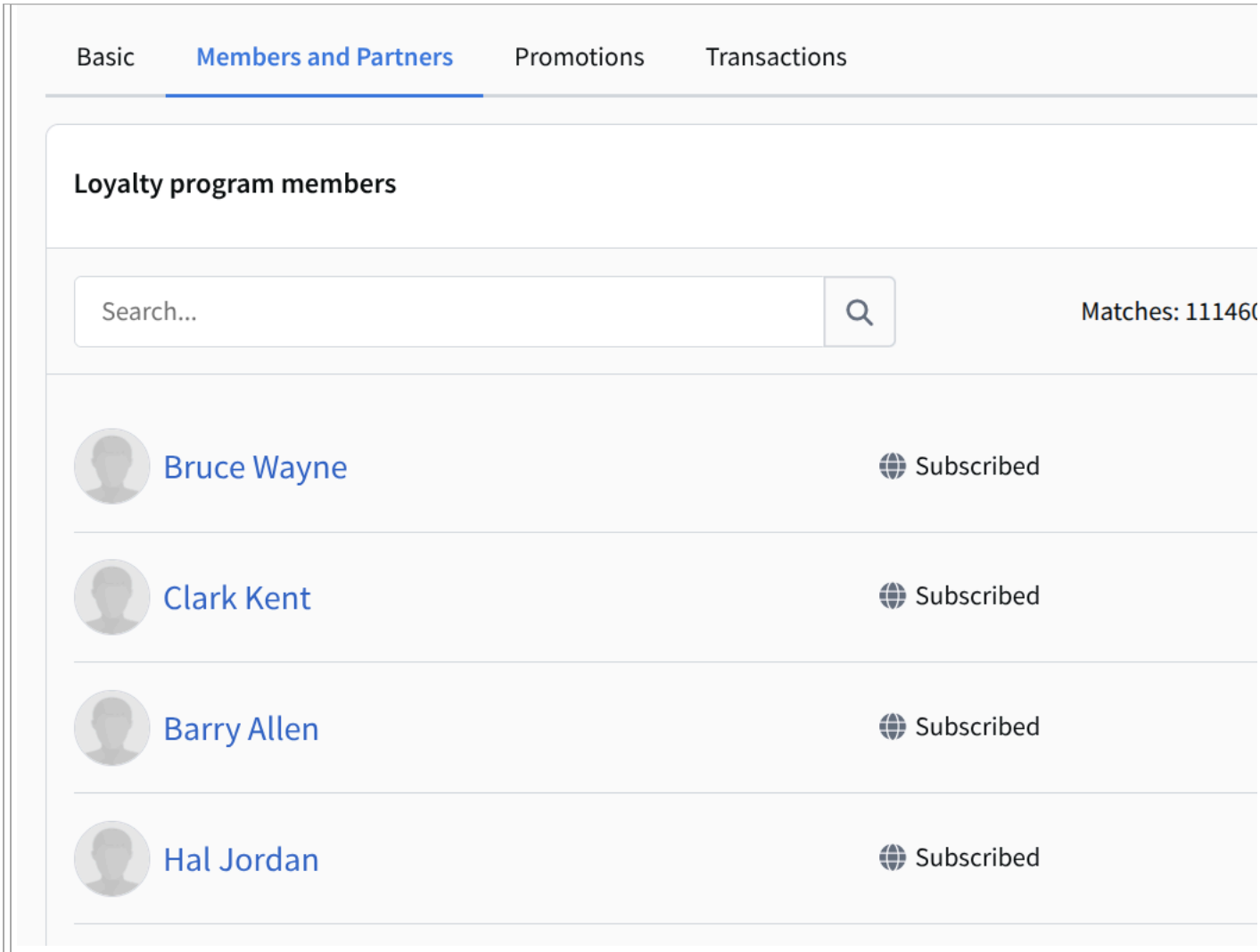
> CAUTION
>
> **To ensure correct platform operation, a migration is required. To do this:**
>
> 1. **Replace the platform files with files from the new build;**
> 2. **Run `./akd upgrade` and confirm the migration;**
> 3. **Fully restart the platform using `./akd restart`.**
>
> **You also need to add the following required objects to the `main.json` configuration file:**
>
> - `SYSTEM_REDIS` **— main Redis storage configuration**
> - `LOYALTY_CACHE` **— Redis storage configuration used for loyalty program operations**
> - `PROCWORKFLOW_DISTRIBUTED_CACHE_DB` **— parameters of the distributed cache storage used by the** `procworkflo`
> - `CAMP_DUPLICATESDB` **— Redis configuration for the duplicates database**
>
> **More details about these configuration parameters are available in the administrator documentation.**

Loyalty Programs

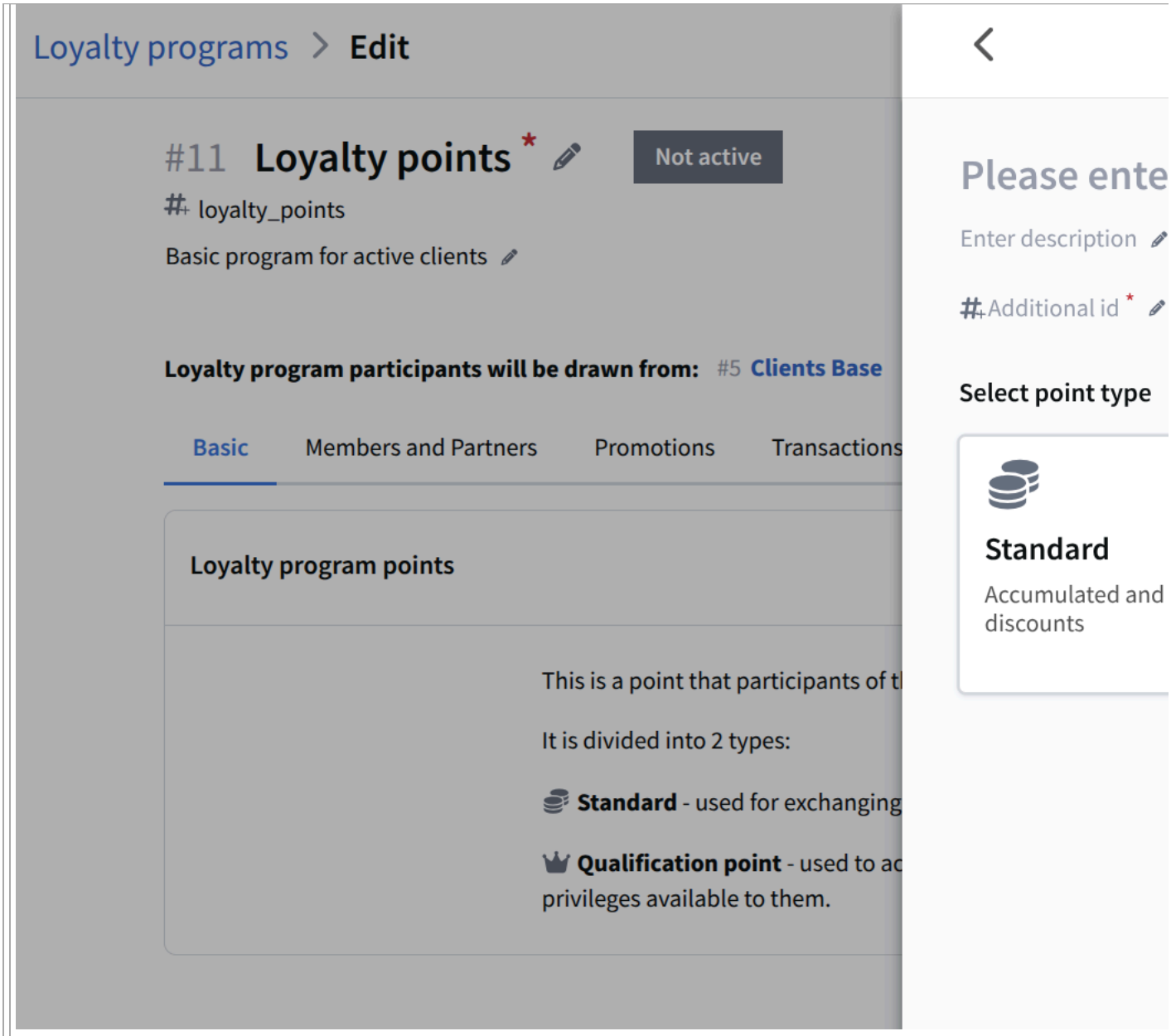
In this update, the Loyalty Programs module has been added to the Altcraft platform. A loyalty program is services.



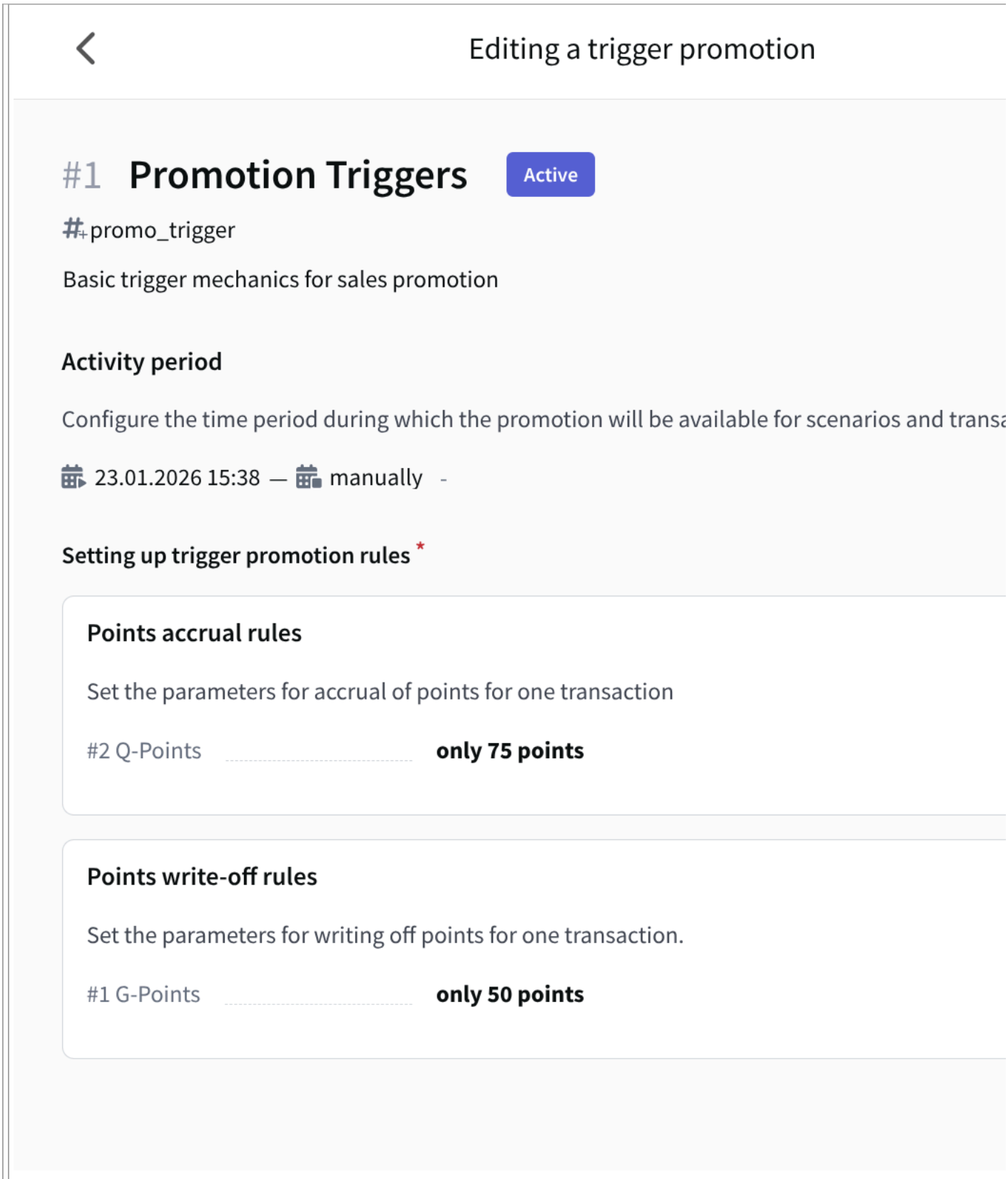
All profiles in the database or a specific segment can participate in a loyalty program. Participants can al



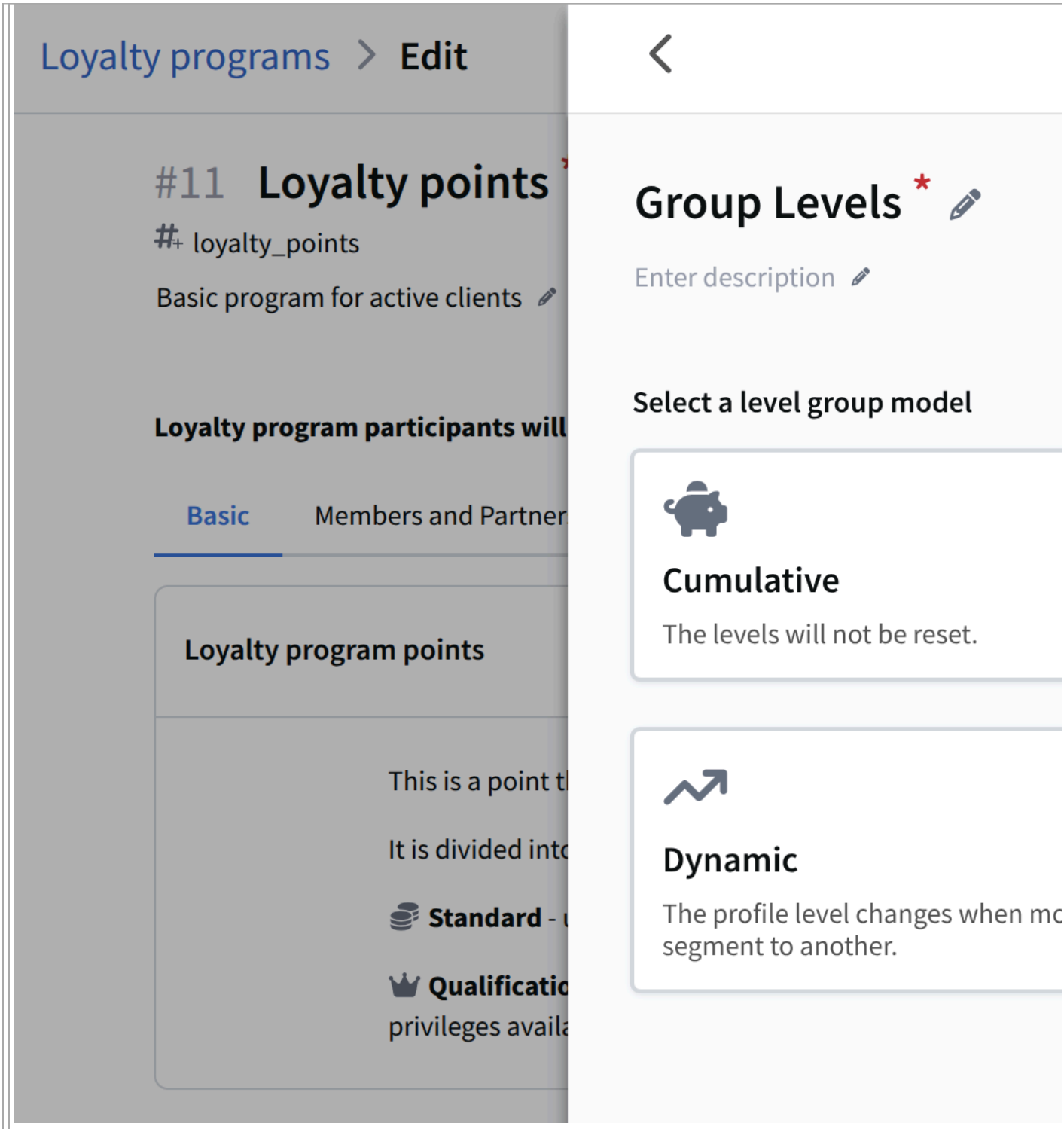
Loyalty program participants can earn points. Points are divided into standard and qualification points. St



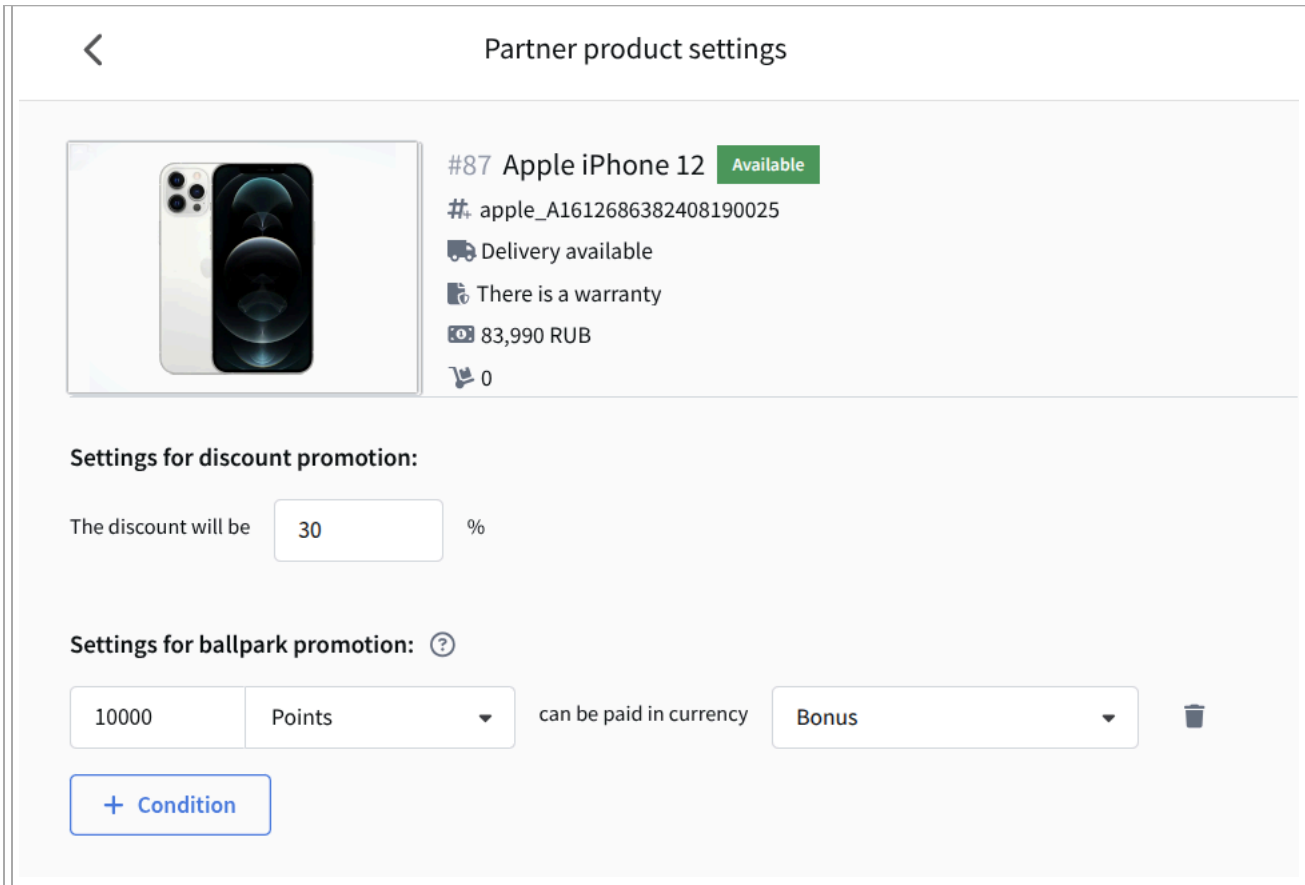
Point accrual and redemption are managed using trigger-based promotional campaigns:



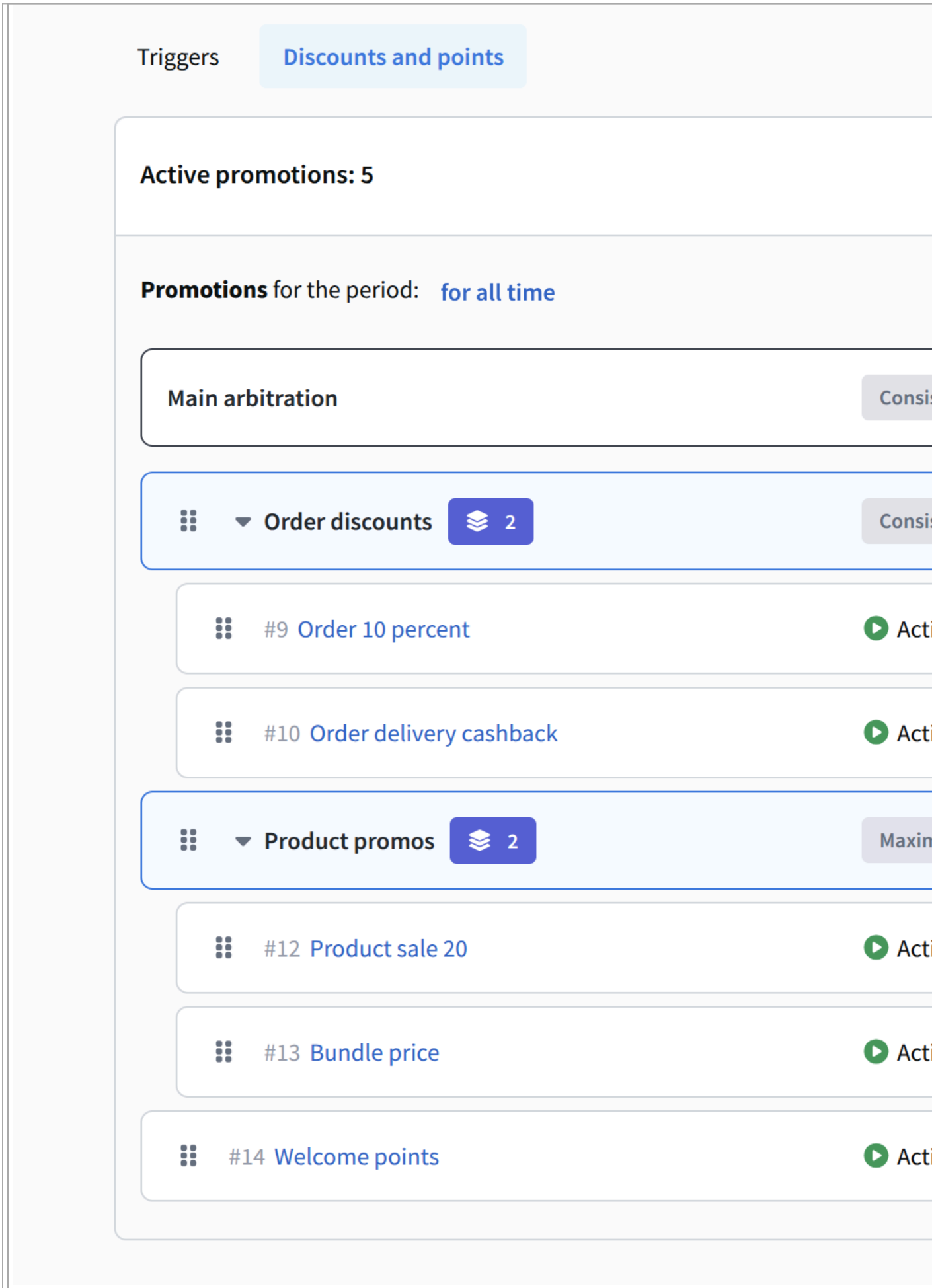
For loyalty levels, groups are configured with their own rules for status review and point reset. Different r



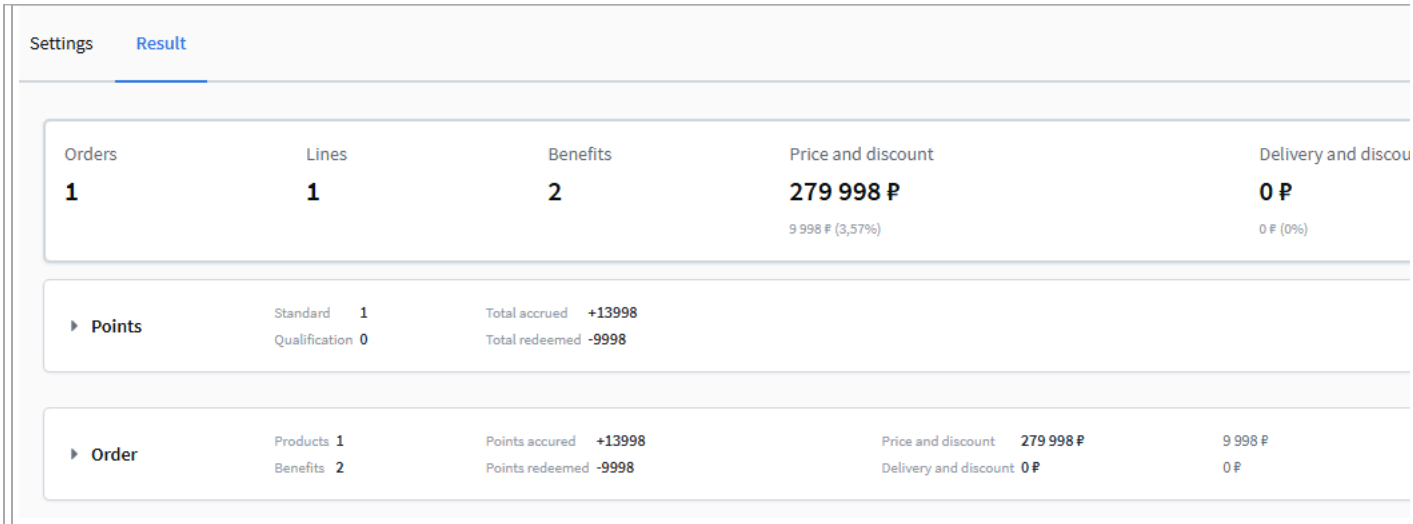
You can configure partners within a loyalty program. Special offers are available for partner products from



Discount and points-based promotional campaigns handle rewards for ordering products from the Market.



After configuring points and promotional campaigns, you can verify that everything works correctly using



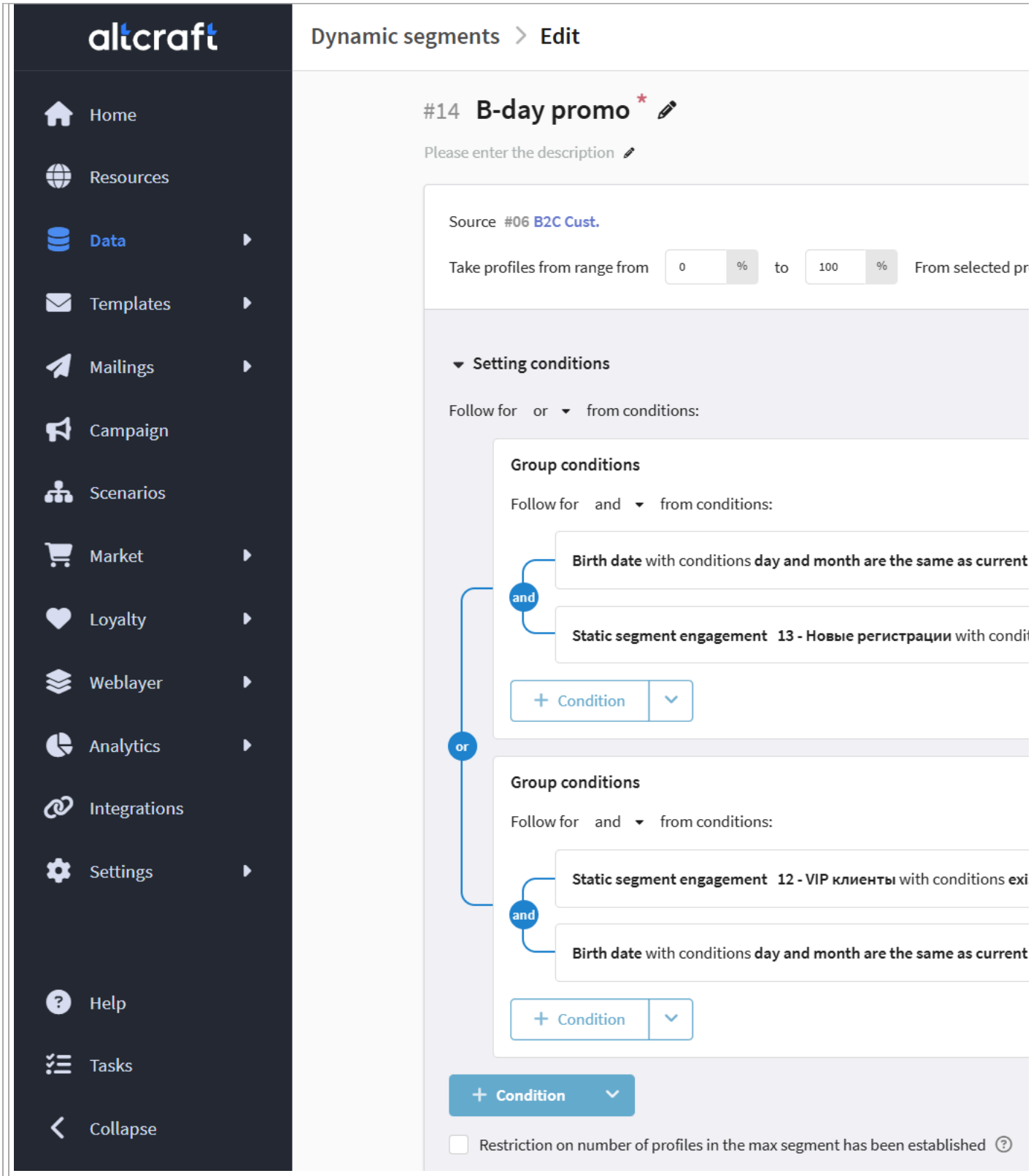
Testing makes it clear how and in what order benefits are applied for loyalty program participants.
Learn more about creating and configuring loyalty programs here.

Updated Segmentation

Major changes have been introduced to the profile segmentation mechanism. The key update is a redesig

- **Static segment — an entity for storing profiles, filled manually or via API. The segment is not recalcul**
- **Dynamic segment — essentially a set of conditions rather than a list of profiles. The segment is recal**
- **Updatable segment — cannot be filled manually or via API. Profiles are added based on defined condit**
- **Quick segments have been removed. Use a dynamic segment instead and click Calculate when config**

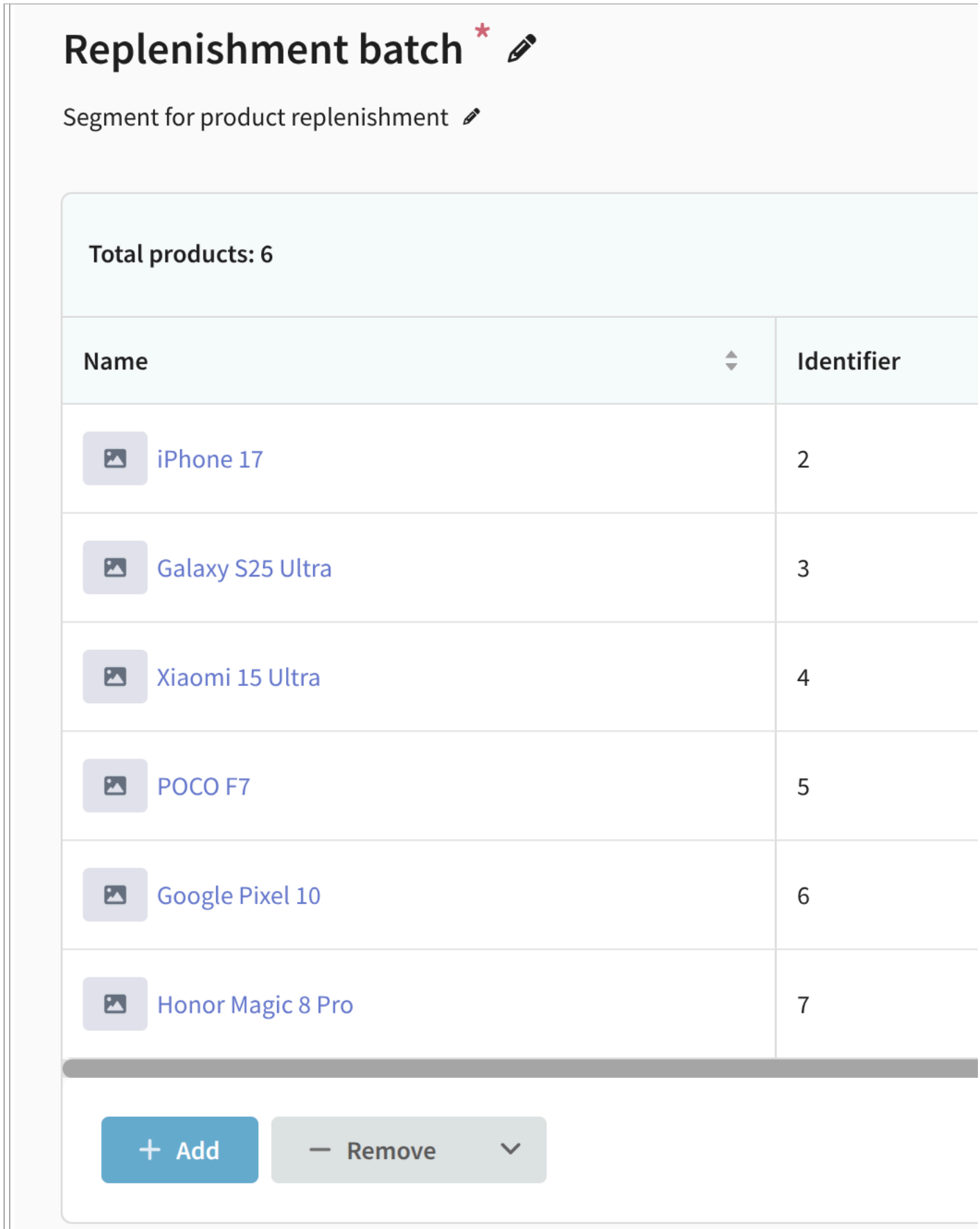
The segment configuration menu UI has also been significantly redesigned:



The interface is now more modern and intuitive. More details about segmentation are available in [this sec]

Product Segments

Within the Market, you can now group products into product segments. These are sets of products and th



There are two types of segments: static (grouped manually or via API using selection conditions) and upd
Instructions for configuring product segments are available in [this article] of the Market documentation.

UI Updates

Along with the update, the interface has been improved:

- **Updated the platform's color palette;**
- **Changed the appearance of some components, such as the date range picker;**
- **Improved the display of auxiliary side menus (for example, stratification in control groups);**

- **Tags and attributes in lists now look more consistent with the overall design;**
- **Fixed several table display issues.**

Color Palette

Before: After:

Date Ranges

Before: After:

List Display

Before: After:

These small improvements enhance usability, make the interface more cohesive and modern, and reduce

Keycloak Integration

This update adds support for integrating the platform with Keycloak. Administrators can configure access
Connector creation and access configuration are performed via the Admin Panel.

Other

- **Changes in virtual sender settings are now saved correctly again.**
- **Fixed an issue where the `CLICKHOUSE_SYSTEM.CONN_TIMEOUT` parameter affected the overall host connection t**
- **Control groups no longer cause errors after being recreated.**
- **Fixed an issue where a required custom channel field could become optional.**
- **Email addresses are now correctly converted to lowercase when creating LDAP users.**
- **Fixed an issue where the regular expression for arrays only validated the first array item and failed JS**
- **Telegram template previews in scenarios now display correctly.**
- **Improved display of errors and warnings across the platform.**
- **Duplicate emails warning about profile import limits are no longer generated.**
- **Fixed an issue where some scenarios could not be saved.**
- **Profile import speed has been optimized.**
- **Deleted custom events no longer appear in the event capture list and no longer trigger events.**

References

v2025.4.75.0: PL-20429 PL-20573 PL-20313 PL-20551 PL-20498 PL-20474 PL-20485 PL-20095 PL-19769 PL-
19912 PL-19289 PL-19935 PL-19938 PL-19762 PL-20032 PL-20060 PL-19785 PL-19941 PL-18104 PL-18357 P

---

v2025.4.75.1005 – Loyalty fixes, minor improvements

Improved

- **Added the ability to wait until a specific date in the “Event Waiting” node.**
- **Added validation to ensure a template is selected when launching a mailing, as well as a restriction o**

Other

This release includes fixes in the loyalty program module, as well as a number of other issues.

- **Creating a mailing by a non-master user no longer causes errors.**
- **Fixed an issue where some scenarios were not attached to campaigns.**
- **The restriction on repeated form submission works correctly again.**
- **Added validation and inheritance of configuration parameters related to Redis storage.**
- **Fixed issues in collecting and displaying loyalty program analytics.**
- **Events in the campaign log are now displayed correctly.**
- **Improved stability of the `webcontrol` process.**
- **Added filtering by statuses in campaign lists.**
- **Fixed an issue that occurred when attempting to save a campaign.**
- **The platform now displays correct error information in case of connection issues with Yandex.Audien**
- **Navigation to a template from the preview in the template list now works correctly.**
- **Fixed an issue where the profile ID was not passed when exporting form data.**
- **The time in the “Pause” scenario node now updates correctly.**
- **Fixed an issue that prevented entering a value less than 100% in virtual sender rules.**
- **Filtering by groups in the loyalty program list works correctly again.**
- **Fixed issues that occurred when opening an email template.**

- Deactivating a mailing during sending now works correctly.
- Fixed several issues in the visual editor.
- The preview of the AMP version of an email template now opens and displays correctly.
- Fixed an issue where the provider was not specified in complaint events.
- The date in event capture and API is now passed correctly.
- Fixed an issue where the preview in custom channels was always displayed as an HTML message.
- The `subscription_fields` variable is now correctly passed in test sendings.
- Fast segment calculation with a query to an external database now works correctly.
- Added validation to prevent linking to a non-existent group in the platform.

References

v2025.4.75.1005: PL-20912 PL-20916 PL-20899 PL-20826 PL-20881 PL-20624 PL-20847 PL-20791 PL-20794 PL-20632 PL-20704 PL-20552 PL-20702 PL-20635 PL-20683 PL-20693 PL-20684 PL-20190 PL-20618 PL-19500 PL-20233 PL-20148 PL-20625 PL-20606 PL-20314 PL-19211 PL-20523 PL-20572 PL-20534 PL-19602 PL-20500 PL-18741 PL-18918 PL-16688 PL-16687

---

v2025.4.75.2408 – React Native mSDK, product segment improvements

> DANGER: Migration
>
> **If you have already upgraded to version 75, you need to run a migration to ensure correct operation of st**
>
> 1. **Stop the `webcontrol`, `api`, and `proctask` processes.**
> 2. **Run the migration using the command `./aktool segments run-separation-migration --config=path-to-configurati`**
> 3. **Start the stopped processes.**

Altcraft mSDK for React Native

Support for the cross-platform framework React Native has been added to the Altcraft mobile SDK. This m
You can learn more about integrating and using Altcraft mSDK for React Native here. The NPM package is

Improved

- Added a filter by participation in segments for product segments, as well as the ability to import cond
- When creating and editing a template or template fragment, the interface now displays a list of availa
- API requests that transmit mailing data now include information about the associated campaign.
- Added the ability to update profile fields in the mobile SDK.
- Segment audience updates are now blocked if the segment is used as the main segment of a campaig
- Added a notification about the need to complete a campaign before deleting it.
- Added a Redis storage connection check at platform startup.

Other

- Attributes are now correctly displayed in the mailing template preview.
- JSON from the Object and Array of objects fields in the API request node is now substituted correctly
- Fixed Viber sending issues via SMSTraffic caused by UTC time in the `start_date` parameter.
- Improved collection and display of loyalty program reports.
- The event type filter in the profile card works correctly again.
- Emails about form submissions no longer contain invalid special characters.
- Fixed several issues related to attributes.
- The limit in the `http_request` pipe of custom channels now works correctly.
- Fixed issues when cloning custom channel templates.
- The “exists” condition for string fields in fast segments works correctly again.
- Improved stability of the `webadmin`, `proctask`, and `api` processes.
- Increased the processing speed of the `trk_pixel_def` queue.
- Links in push templates in the beta editor now work correctly.
- Fixed issues with connecting to the platform via LDAP.
- Fixed an issue where template previews were not saved in FileDB.
- Processing of a custom field with the name of a system field now works correctly.
- Custom channels can be used in scenarios again.
- Fixed an issue where the audit log export contained a time not matching the account time zone.
- The `utm_medium` UTM parameter is now updated in the profile card when registering a goal via API.
- Fixed an issue where hard bounces from Telegram bot mailings were not registered.
- Sending policies are now applied during массовой trigger-based sending of profiles to a scenario.
- The character limit is no longer applied when using dynamic variables with long logical conditions.

- **Fixed an issue where the `/profiles/list` API method did not return all profiles from the database.**
- **Fixed issues with displaying the External object value in custom channel templates.**

References

**v2025.4.75.2408: PL-20956 PL-21090 PL-21117 PL-21116 PL-21118 PL-21115 PL-19263 PL-21087 PL-21089
20780 PL-18407 PL-17935 PL-19747 PL-20682 PL-20719 PL-20652 PL-20312 PL-18757 PL-18575 PL-18663 P
20862 PL-20541 PL-20764 PL-20738 PL-20553 PL-19277 PL-20428 PL-19278 PL-20201 PL-17754 PL-18981 P
20965 PL-20891 PL-20954 PL-20958 PL-20951 PL-20908 PL-20955 PL-20927 PL-20935 PL-20922 PL-20297 P
20539 PL-20688 PL-20709 PL-20685 PL-18299 PL-20712 PL-20623 PL-20408 PL-20164 PL-20677 PL-20160 P**

---

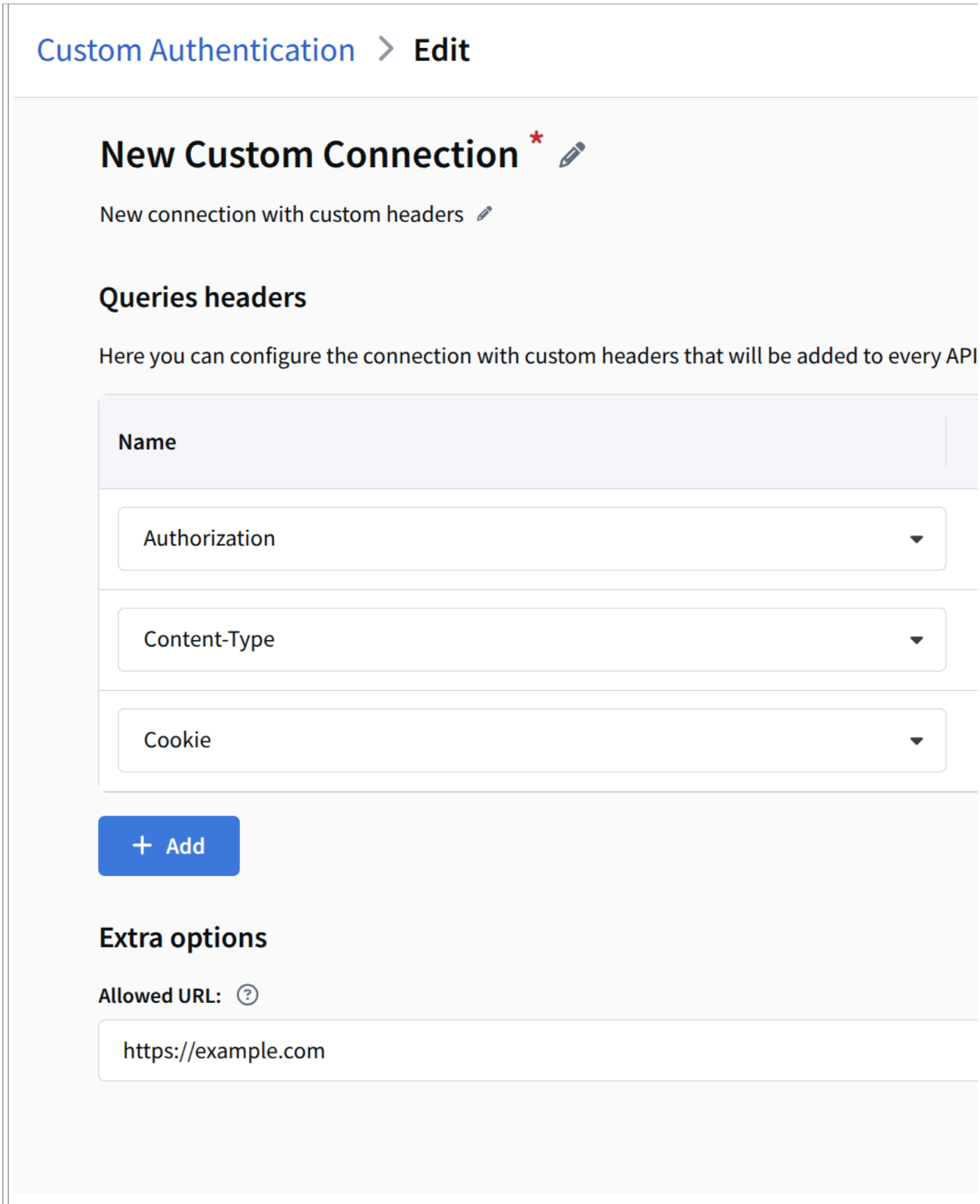
v2025.4.75.2774 – Custom Authentication, block editor improvements

> **CAUTION: Sending without a resource will be disabled**
>
> **In upcoming updates, all active mailings that do not use a resource will stop working.**
>
> **To avoid issues, create a new resource or use an existing one and add it in the mailing settings.**
>
> **You can retrieve the list of such mailings using the `./aktool campaigns upgrade-blocker-list` or `./aktool campaign`**

Added

**A new type of connector for external connections has been introduced — Custom Authentication. This co
and supports validation and error handling.**

With this connection type, you can integrate the Altcraft platform with any external service that requires



Learn more about configuring this connection type here.

Improved

The email template block editor has been improved:

- **ability to insert links and emoji into tables added;**
- **save and export UI updated;**
- **proxying for empty images fixed;**
- **visual and UI improvements applied to several interface elements.**

Read more about the block editor in this article.

Other

- The Altcraft team continues fixing issues in the Loyalty module and improving its stability.
- Fixed an issue where the `data/segment` directory was not cleared automatically.
- PostgreSQL connection passwords containing special characters now work correctly.
- Fixed an issue that prevented creating transactions with a long expiration period for points.
- Editing of mailings, scenarios, and campaigns is now restricted if a required attribute is assigned witl
- Improved platform stability when executing multiple trigger API calls.
- Fixed issues with analytics and report display in the platform.
- Values in templates for `external object` fields are now saved correctly.
- Fixed several issues related to attributes and their value processing.
- Campaigns now deactivate correctly even if the deactivation moment coincides with the audience ge

References

v2025.4.75.2774: PL-20989 PL-21186 PL-20844 PL-21107 PL-21228 PL-21163 PL-19018 PL-19865 PL-21208
21064 PL-20938 PL-21000 PL-20071 PL-20932 PL-20066 PL-20973 PL-21015 PL-20972 PL-20824 PL-20828 F

v2025.4.74

v2025.4.74.0 – Campaigns

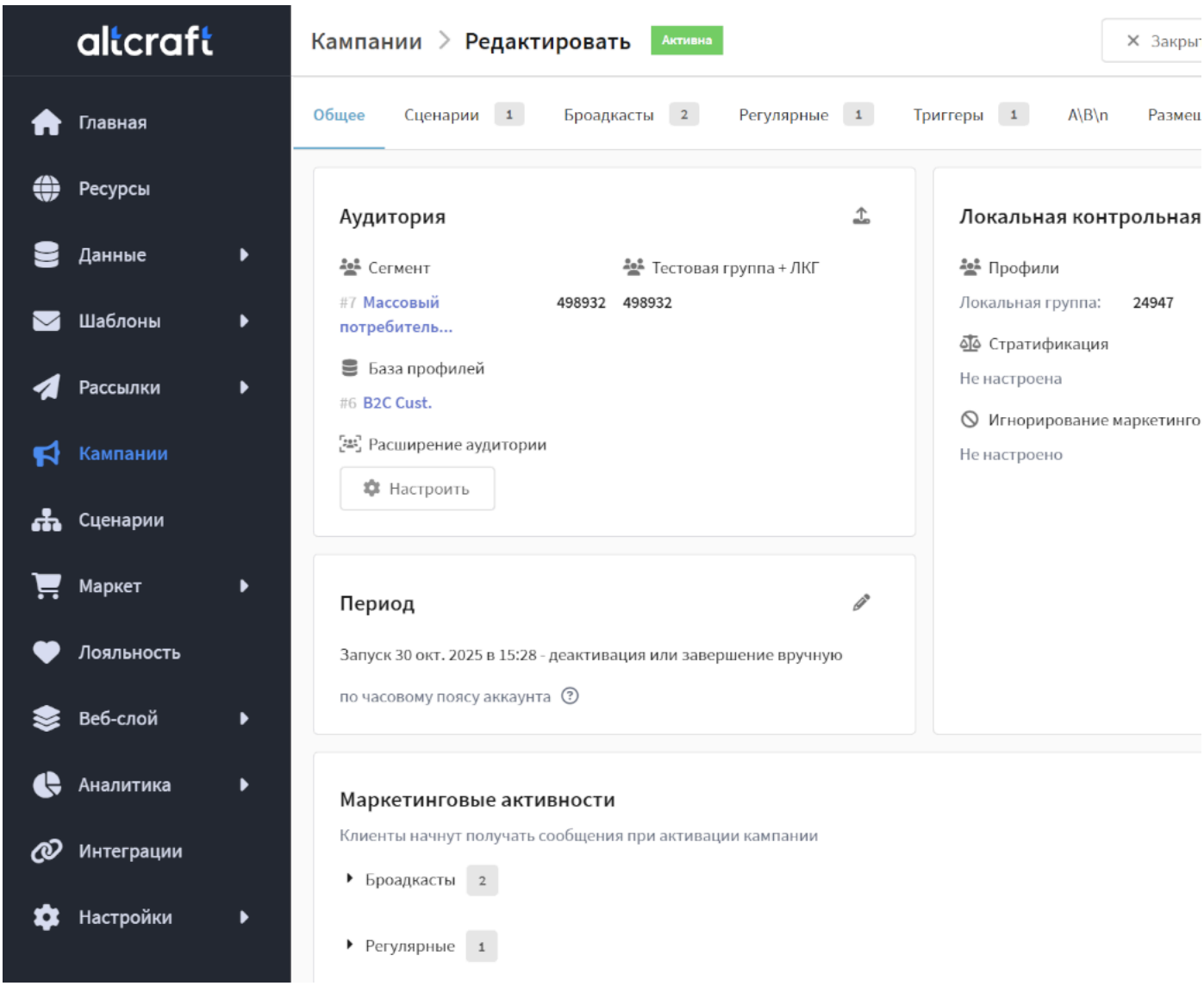
CAUTION

To ensure correct platform operation, a migration is required. Follow these steps:

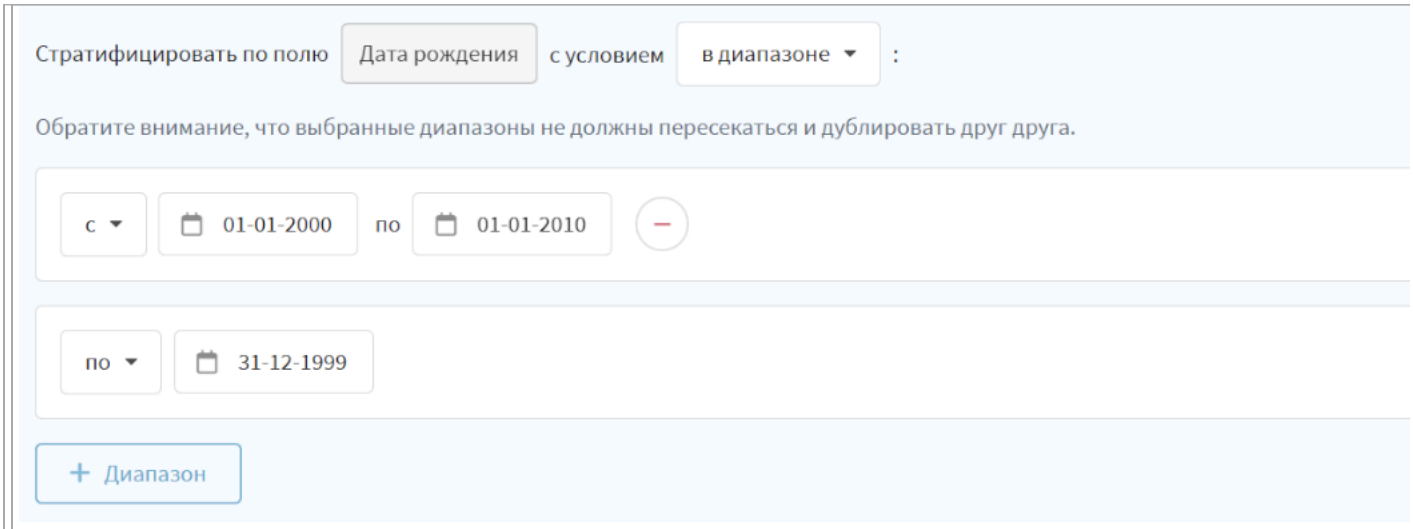
1. **Replace the platform files with files from the new build;**
2. **Run `./akd upgrade` and confirm the migration;**
3. **Fully restart the platform using `./akd restart`.**

Campaigns: a comprehensive marketing tool

Previously, all types of marketing messages in the platform were called Campaigns. In the new platform v
Within a campaign, you can configure audiences, segment databases, manage user profiles, control group



Campaigns also support local control groups. A local control group allocates part of the audience that wil
into strata based on a selected attribute. Stratification creates a control group closely matching the targe



Analytics for all activities within a campaign are available in a unified report with multiple metrics:

New campaigns centralize management of different activities, track communication effectiveness, and su

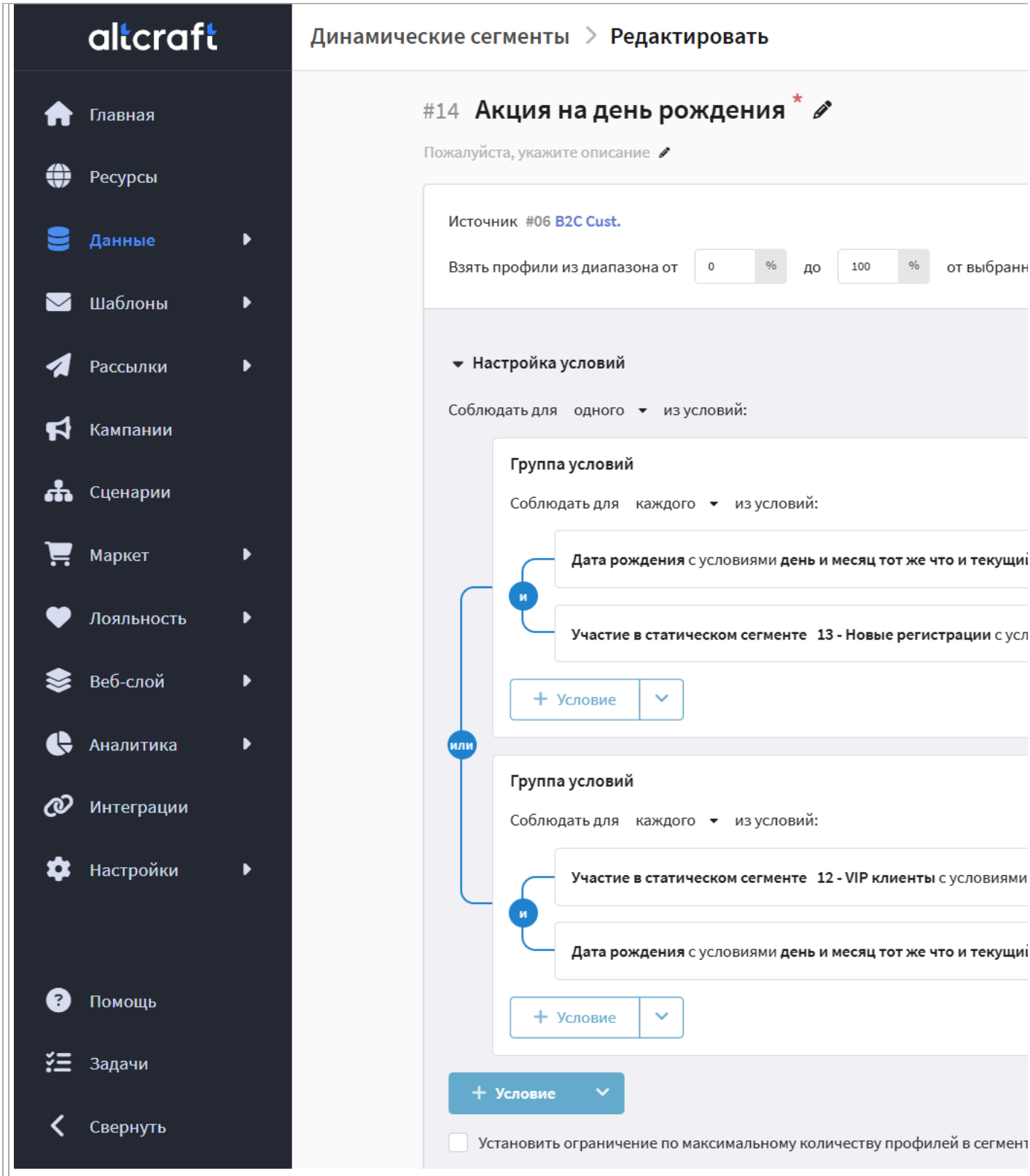
Improved

Updated segmentation

Major changes were introduced to the profile segmentation mechanism. The key update is a redesign of s

- **Static segment — an entity for storing profiles filled manually or via API. It is not recalculated on use**
- **Dynamic segment — a set of conditions rather than a stored profile list. It is calculated before each u**
- **Updatable segment — cannot be filled manually or via API. It is populated based on defined conditions**
- **Quick segments were removed. Dynamic segments replace them; use the “Calculate” button when co**

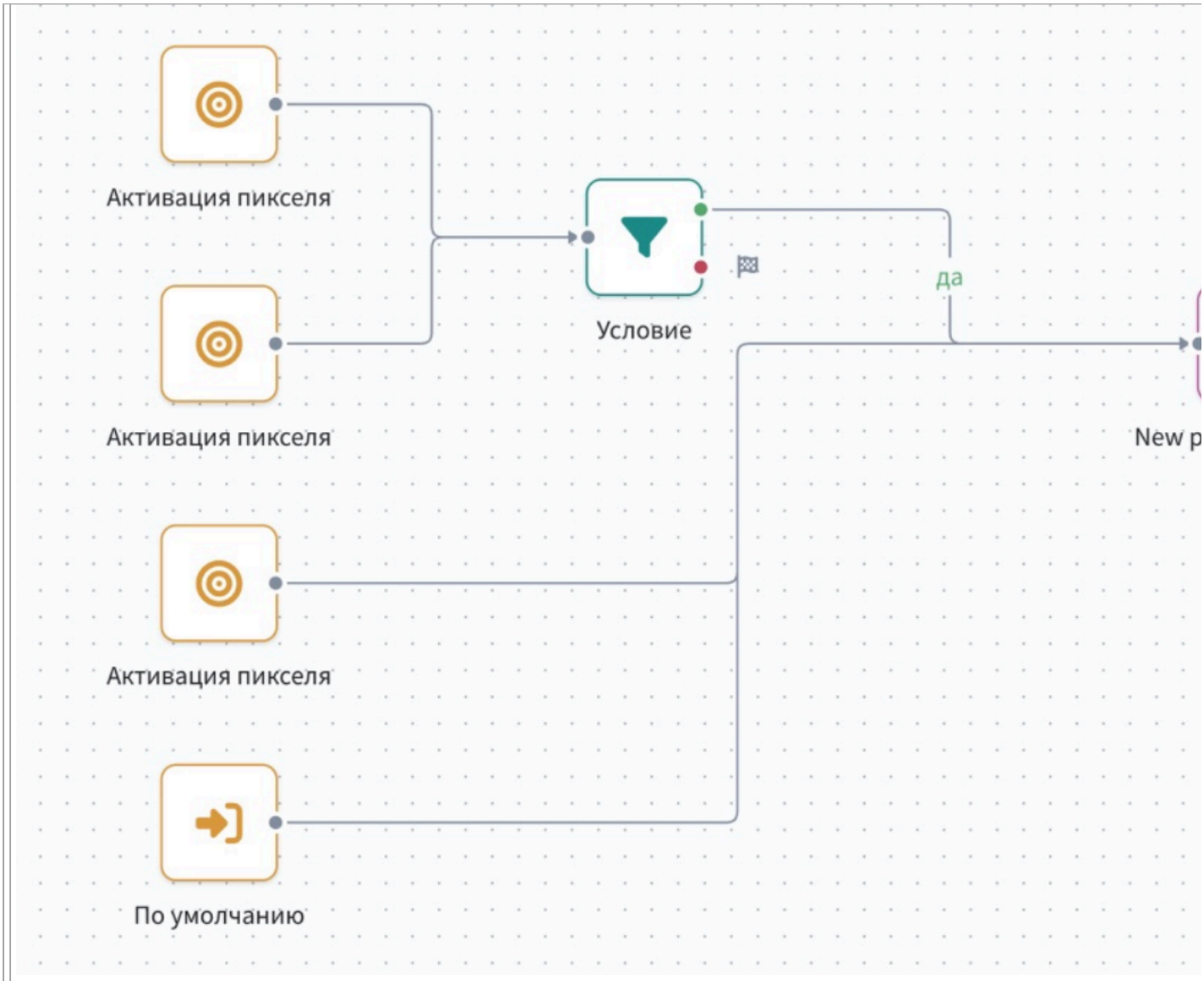
The segment configuration UI was also significantly redesigned:



The interface is now more modern and intuitive. More details on segmentation are available [in this sectio](#)

Combined profile capture in scenarios

Previously, a scenario supported only one entry trigger per type. Now multiple triggers of the same type c



The maximum number of start nodes is limited to 10. This value is configurable via `WORKFLOW_MAX_START_NODES_`

Other

- **Several UI/UX components were improved.**
- **Fixed issues in some Market APIs.**
- **Improved stability of `proctask` and `procpixel` processes.**
- **Fixed issues with campaigns using custom channels.**
- **Message delivery in the Intellin SMS provider integration is now correctly registered.**
- **Fixed cases where segments failed to calculate or calculated extremely slowly.**
- **Goals with SMID from a custom channel are registered correctly again.**
- **When starting the platform with the `--config` flag, the custom configuration file path is now passed to**

References

v2025.4.74.0: PL-16253 PL-20320 PL-20303 PL-20304 PL-19139 PL-20225 PL-20226 PL-20194 PL-19904 PL-19633 PL-19328 PL-18624 PL-19600 PL-19567 PL-1785 PL-19053 PL-17465 PL-19441 PL-17851 PL-19266 PL

v2025.3.73

CAUTION

Starting from v2025.3.74, the following API changes will take effect:

- **The old API version 1.0 will be disabled. Please switch to the current version in advance.**
- **The API token can no longer be sent in GET parameters or in the JSON body. Use the header instead**
- **The `format` parameter will no longer be supported. Set the desired format using the Accept header (f**

We recommend making the necessary changes in advance to avoid API issues.

Also, campaigns without a linked resource will no longer run. Please make sure that all your campaigns l

v2025.3.73.1 – Attributes, GCG stratification, Scenario improvements

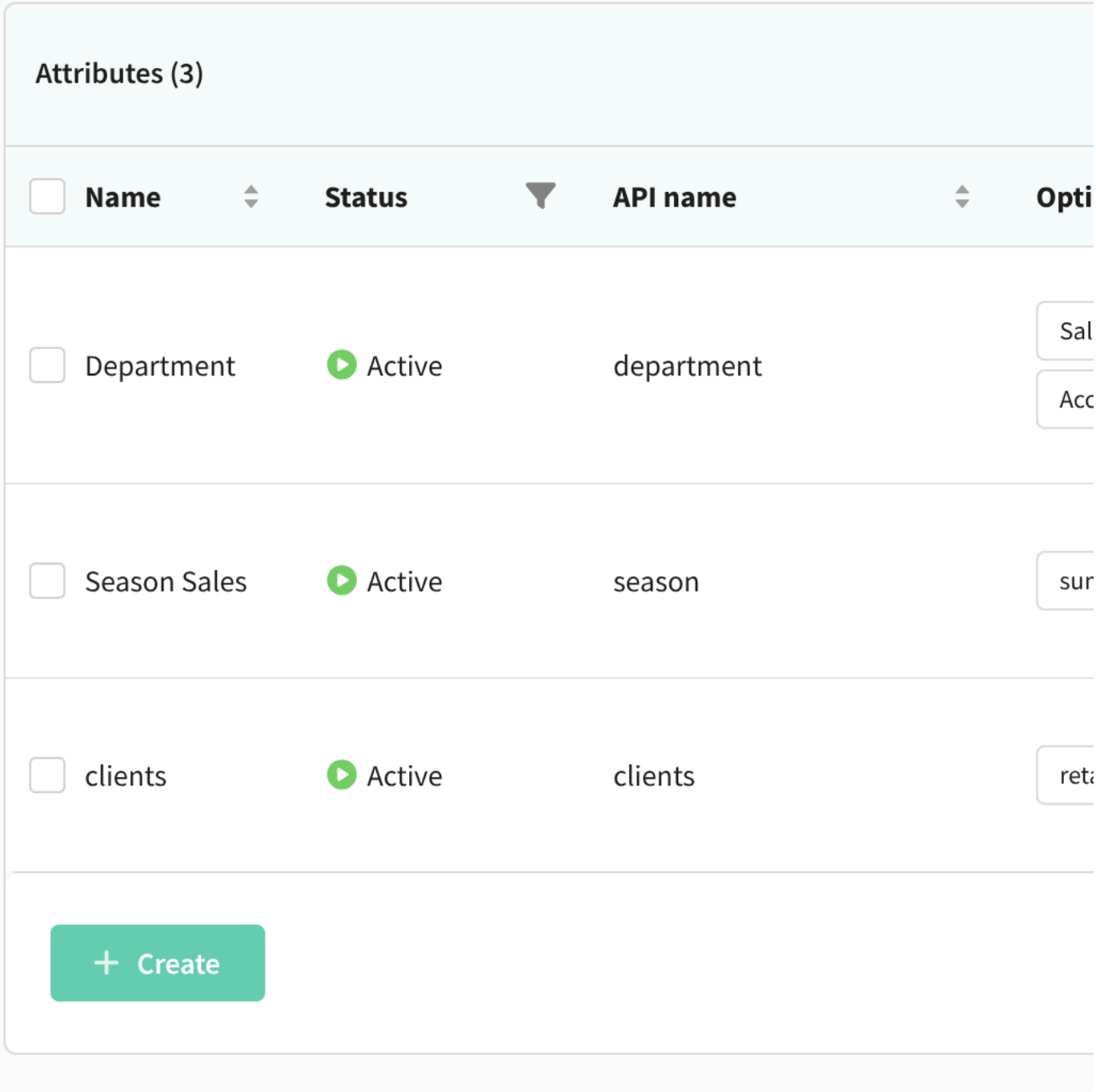
Added

Attributes

Attributes have been added for platform entities. Attributes are special custom fields added to platform e

With attributes you can:

- **search and organize objects;**
- **select campaigns and scenarios for segmentation;**
- **insert values into content and API requests.**



Learn more about attributes in [this section](#).

Stratification in GCG

Now you can configure stratification for the global control group. Stratification is a method of randomly s whole population.

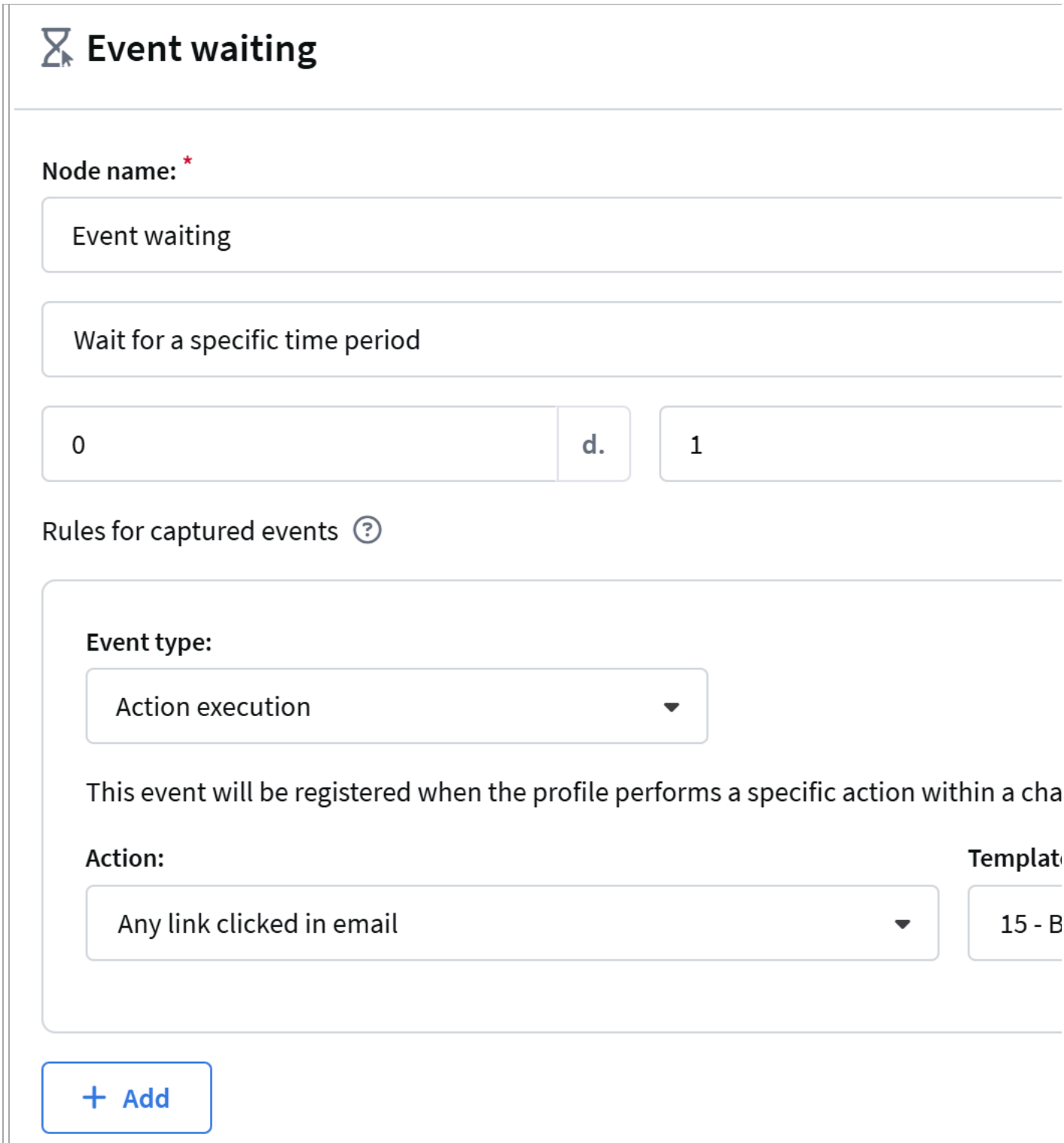
Stratification

Stratification is a method of random selection of profiles with preliminary divisi
(strata) according to specified characteristics. The audience is divided into grou
composition of the whole aggregate.

From each group, a portion of profiles is randomly selected proportional to the size of that group. This hel documentation.

New Scenario element “Event Waiting”

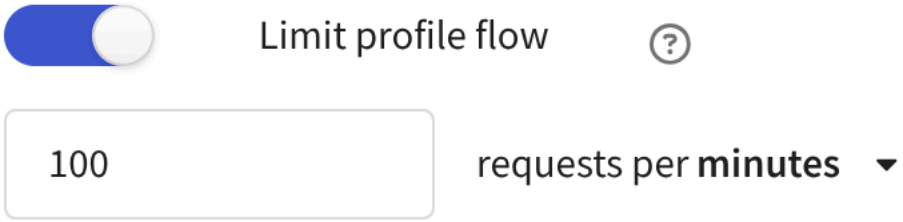
A new scenario element has been added — the Event Waiting operator. It stops the profile flow until a spe within the specified time, the profile follows another branch.



With this element, you can configure reactivation chains or, for example, an abandoned cart scenario. Rea

Profile limiter in the "API Call" element

For the "API Call" scenario element, it is now possible to limit the number of profiles passing through this more in the corresponding section of the article "Scenario elements".



Improved

- **Optimized the performance of `proctask` and `procintegras` processes.**
- **Updated the CSS inliner in message templates and campaigns.**

Other

- **Various UI/UX issues in the platform have been fixed.**
- **Retry counters in senders now correctly display the queue.**
- **Fixed errors that occurred when trying to create and save a multivariate test.**
- **Display and filtering of senders in the Admin Panel now works correctly.**
- **Fixed an error that prevented searching for profiles in a specific scenario element. JSON content link**
- **SQL export to segments works correctly again.**
- **Fixed errors related to generating and displaying template previews.**
- **Added the ability to save a connector to an external database with SSL connection without specifying**
- **Analytics for goals with slashes in the name is displayed correctly again.**

References

v2025.3.73.1: PL-18644 PL-18219 PL-18646 PL-18645 PL-18715 PL-18218 PL-18220 PL-18217 PL-18452 PL-18537 PL-18072 PL-18864 PL-18861 PL-18875 PL-18825 PL-12690 PL-18240 PL-16060 PL-18944 PL-18878 P

---

v2025.3.73.1140 – Altcraft Mobile SDK v2

Added

Altcraft mobile SDK v2

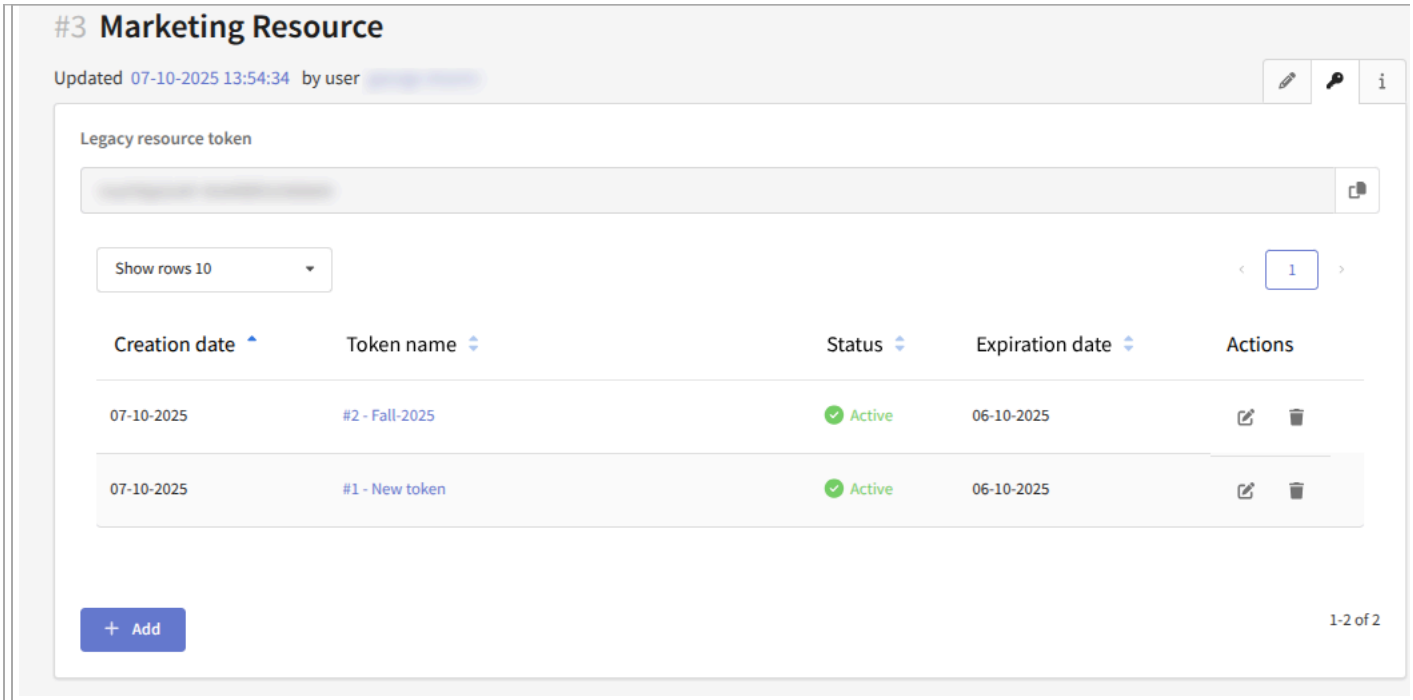
We have improved our mobile SDK for handling push notifications in applications. Key updates include:

- **Support for multiple authorization types: JWT authorization and Altcraft role token authorization;**
- **Ability to display push notifications with context configured in the platform;**
- **Automatic token refresh when it changes;**
- **Automatic delivery and open event tracking;**
- **New functions for working with anonymous and registered users, as well as with multiple users on a s**

Learn more in the developer documentation section. Altcraft mSDK is available on GitHub.

Other

- **Added variables for using pixel event data in Scenarios and various types of content. More about thes**
- **Added a new tab for managing resource tokens. On this tab, you can create a token for a resource, se**



- **Added API handlers for deleting products and SKUs from the Market.**
- **Added the option to create an integration with Yandex.Appmetrica without creating a pixel. This can l**
- **Added validation to ensure that dynamic JSON provided by the user contains an object (arrays are not**
- **Added the ability to reference the current day in selection conditions and segmentation.**
- **Added a Redis connection check during platform startup.**

Improved

- **Message history in Jivo chat is now linked to the user's login — this helps support staff resolve custo**
- **Platform process stability has been improved.**
- **Various UI/UX component improvements and fixes.**
- **The short link generator no longer creates links ending with a dot.**
- **The `procintegras` process logs now include profile-related information and push delivery details for Fire**

Other

- **Default value for the `FIREBASE_PUSH_BATCH_WORKERS` parameter is now 200.**
- **Fixed an issue where a dot or comma could be entered into an integer campaign field.**
- **SMS sending status handling in the SMS Traffic integration now works correctly again.**
- **Fixed an issue that caused segments with the "Participation in segment" condition to be calculated m**
- **Reserved tasks are now correctly set to the `registered` status.**
- **Fixed an issue where a boolean profile field did not update after a form submission.**
- **Attributes now correctly substitute values when used in Custom channels.**
- **Fixed analytics issues in NPS forms.**
- **The integration with EDNA now works properly again.**
- **Fixed errors related to data migration during platform updates.**
- **Rustore push event registration now works correctly.**
- **The `formatdate` function in the "API Call" scenario element now correctly processes profile fields.**
- **Fixed an issue where web version collections were deleted too early.**
- **The subscription manager now correctly displays recipients of messages.**
- **When updating a template via API, the modification date and time are now updated correctly.**
- **Occupied IP addresses are no longer displayed when creating a sender.**
- **Fixed an issue that prevented sending a profile to a scenario containing an attribute ignored by the CI**
- **Fixed several frequent issues in the visual email editor:**
  - **multiple font display now works correctly;**
  - **Transition, Perspective, and Transform functions are functioning properly again;**
  - **background setup and saving issues have been resolved;**
  - **images in style blocks now load correctly.**
- **Link redirection in emails now works correctly again.**
- **Default values for parameters `SEGSTATIC_PROFILE_CHANGES_BATCH_SIZE` and `API_GET_PROFILES_BATCH_SIZE` have bee**
- **Fixed an issue where the `templates/list` API request did not return the `next_from_id` field.**
- **Password change via the `aktool` utility now functions correctly.**

- **Fixed an issue where the stratification limit from configuration parameters was not displayed in the p**

References

v2025.3.73.1140: PL-19333 PL-19309 PL-19326 PL-18399 PL-19176 PL-19273 PL-18437 PL-17717 PL-18369 19218 PL-18836 PL-15418 PL-19233 PL-19204 PL-17933 PL-19214 PL-19223 PL-18414 PL-19118 PL-19210 F 17809 PL-17768 PL-18308 PL-13780 PL-18366 PL-19119 PL-18333 PL-19114 PL-19115 PL-18155 PL-18471 F 19262 PL-16464 PL-18837 PL-19296 PL-19228 PL-19277 PL-18994 PL-19238 PL-18363 PL-19105 PL-18343 F

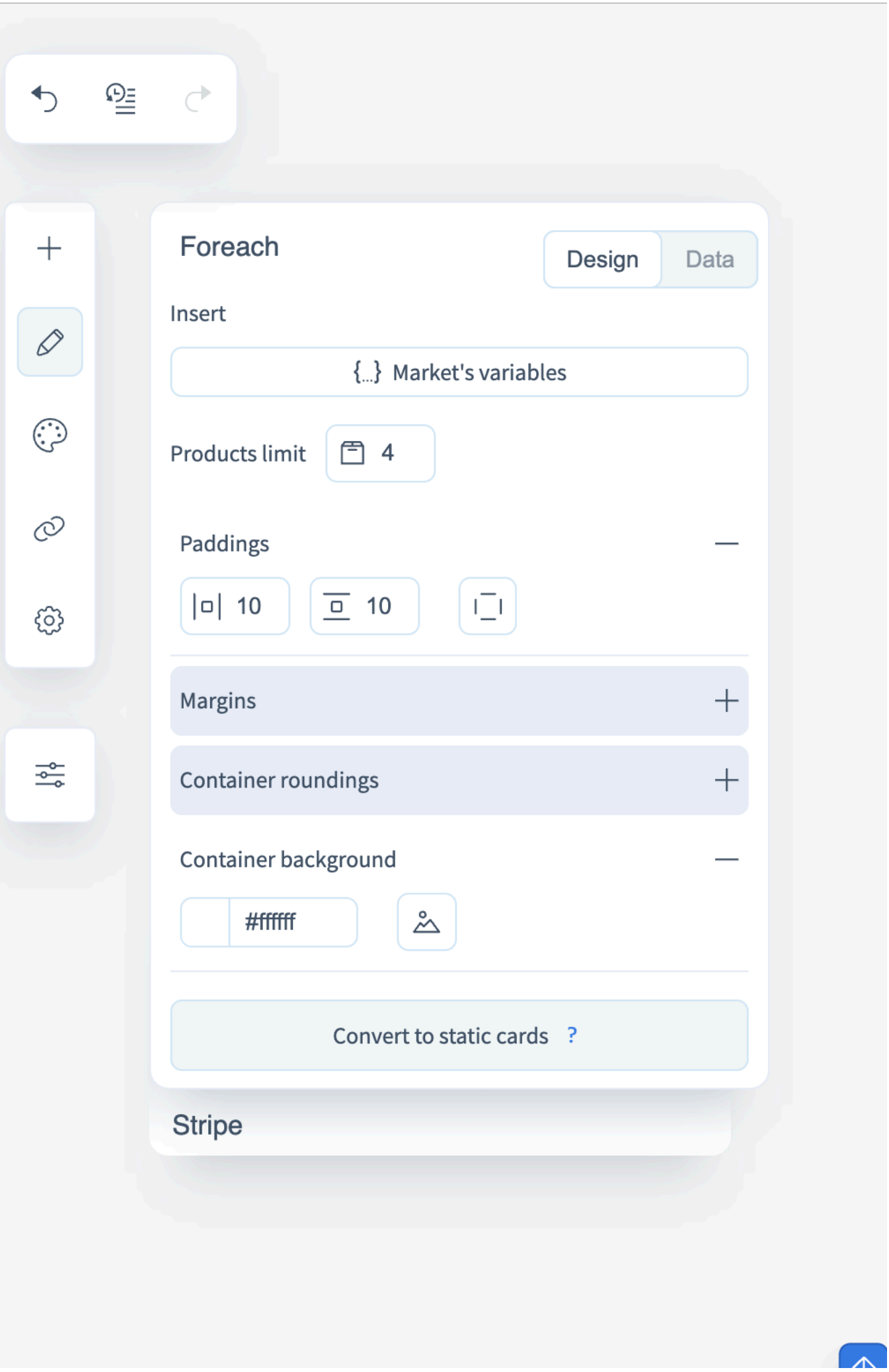
---

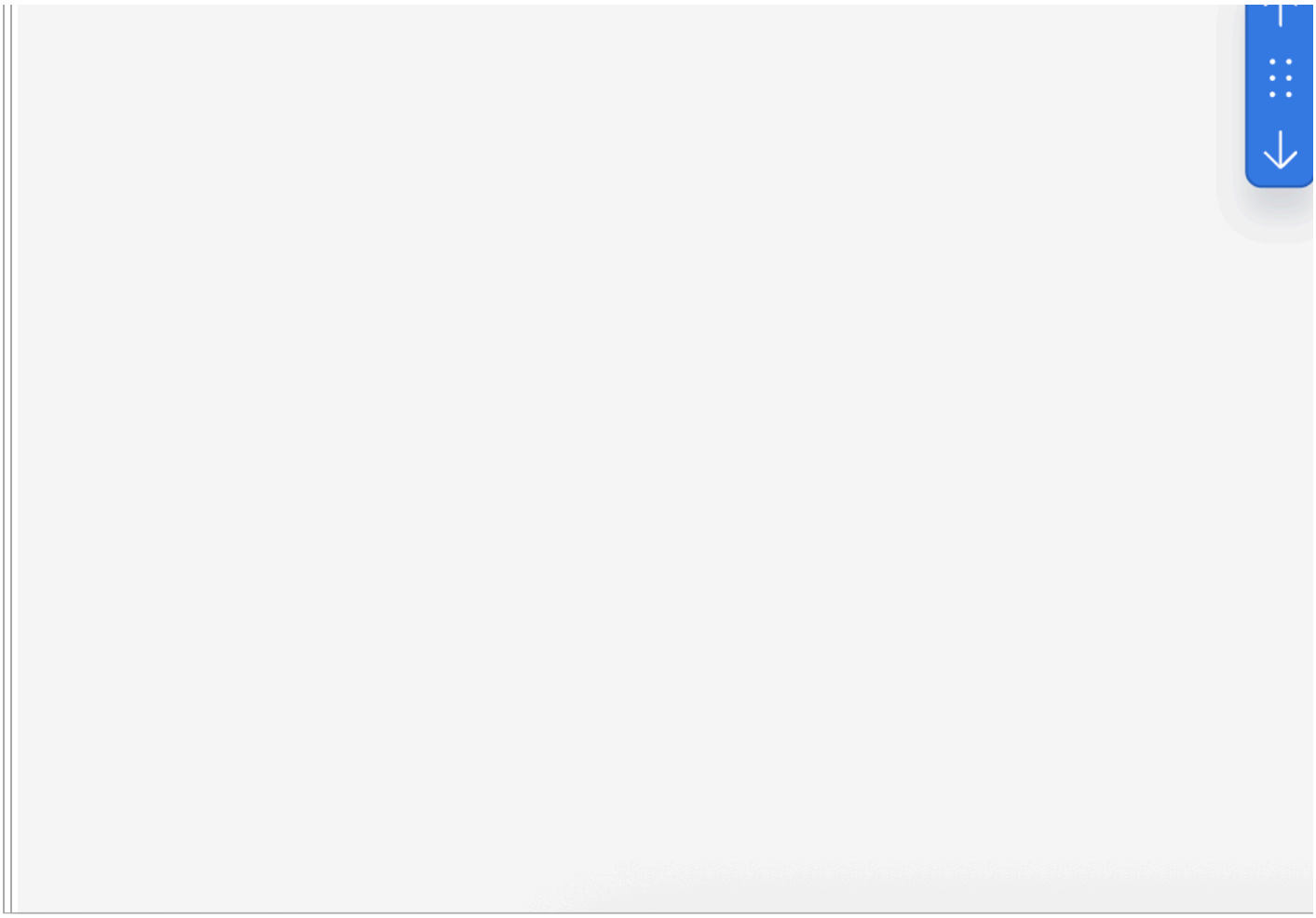
v2025.3.73.1942 – Updated email block editor, Faster Imports and Campaigns

Added

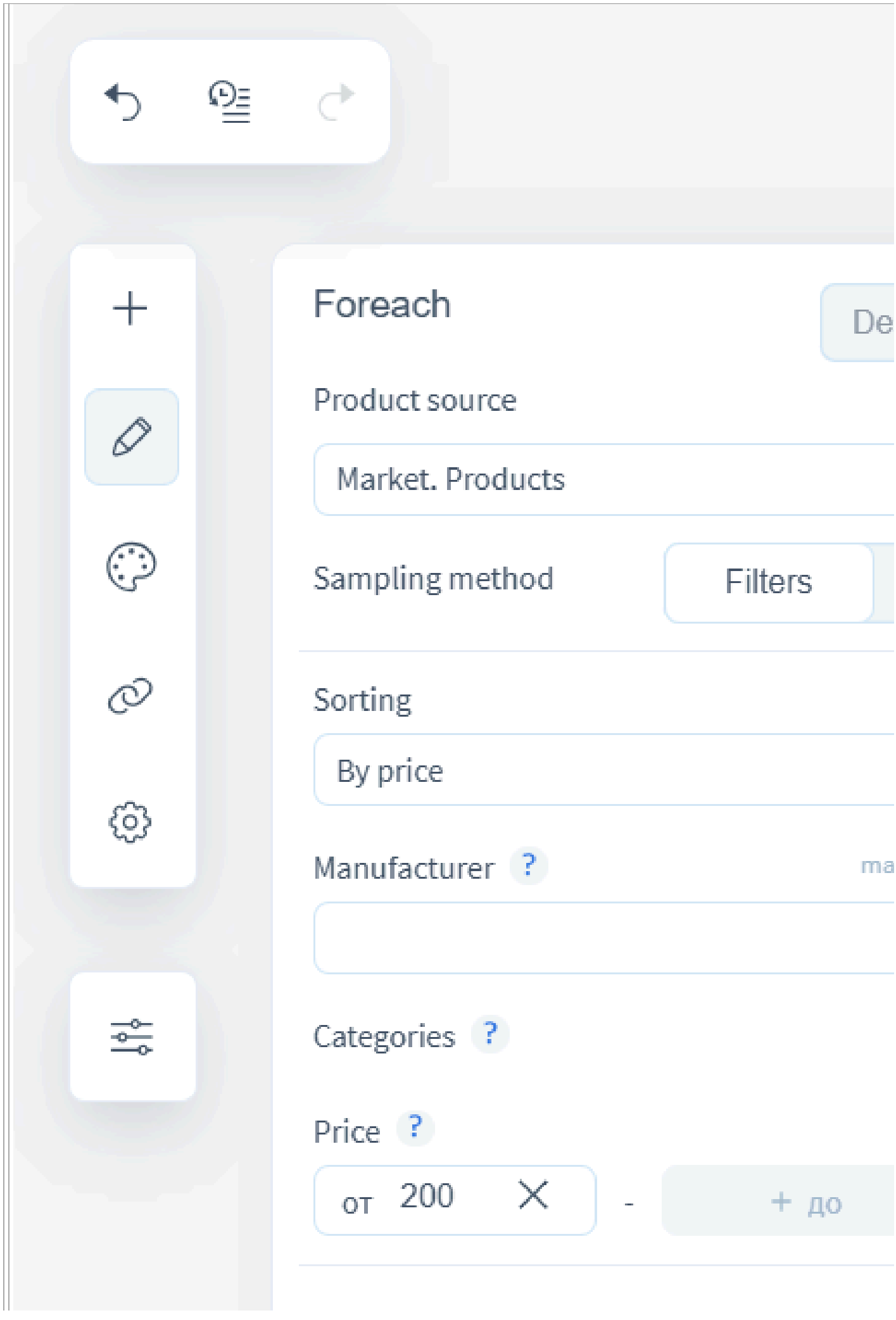
Updated Block Email Builder

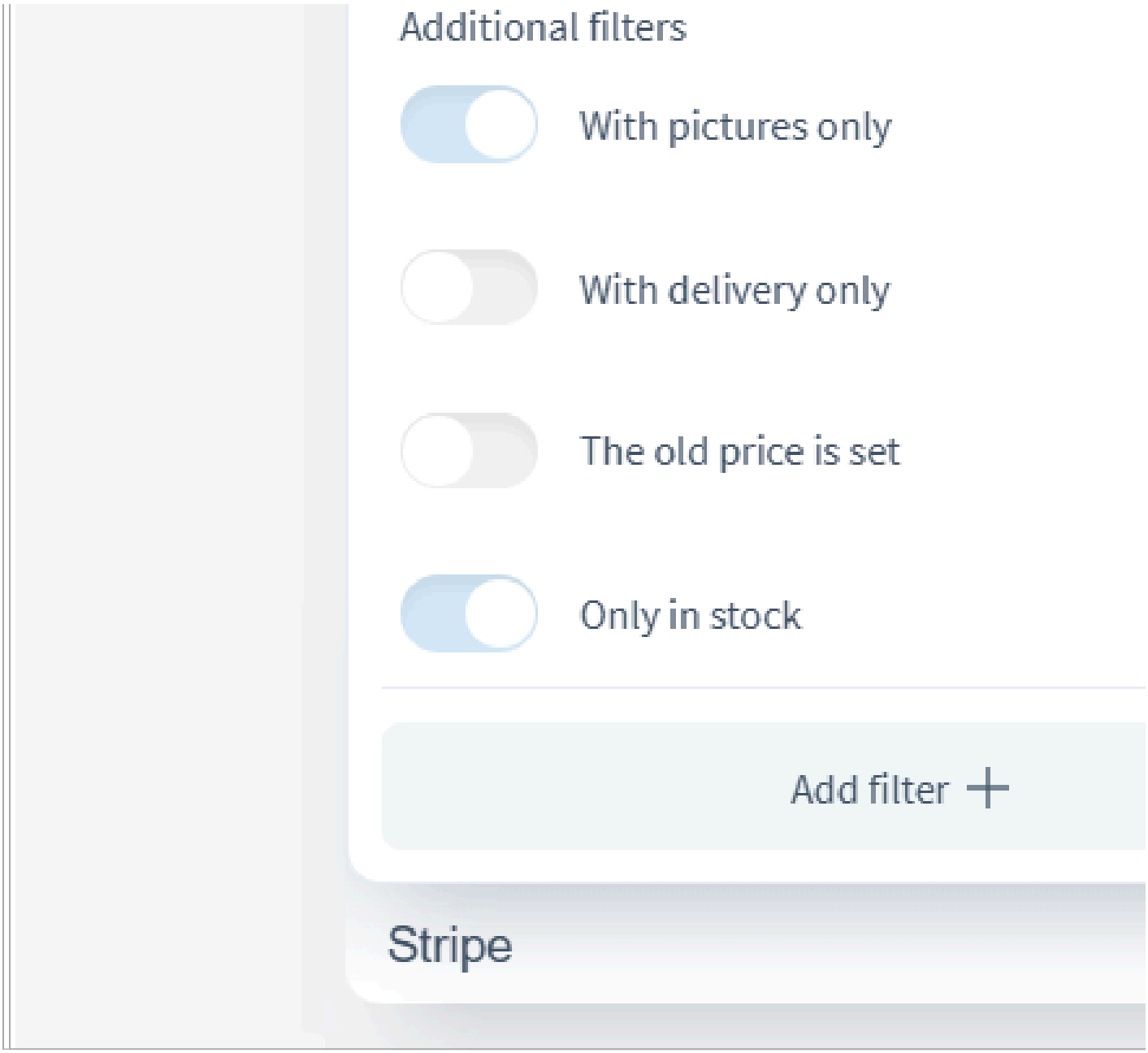
The Block Email Builder is now more functional and user-friendly. It includes a new “Products” element th


Foreach
Design
Data
Insert
{...} Market's variables
Products limit
4
Paddings
10
10
Margins
Container roundings
Container background
#ffffff
Convert to static cards ?
Stripe



You can customize the design of product cards and apply filters directly in the editor using an intuitive int


Foreach
De
Product source
Market. Products
Sampling method
Filters
Sorting
By price
Manufacturer
?
ma
Categories
?
Price
?
от 200
-
+ до


Additional filters
With pictures only
With delivery only
The old price is set
Only in stock
Add filter
Stripe

The block editor gallery also now lets you edit images — crop, resize, rotate, and flip them directly in the

Other

- **The SDK now supports registering custom user events.**
- **The method *`/api/v1.1/templates/get`* is extended: parameters are added to filter templates by channels**
- **If a campaign runs inside a scenario, the methods *`/api/v1.1/campaigns/triggers/get`* and *`/api/v1.1/camp`***
- **Custom subscription fields can now be displayed in templates using the `subscription_field` function.**
- **A `len` function was added to compare the length of an array with a given number. For example, to inc**

Improved

- **When exporting collected form data, the profile ID is now included in the report, making it possible to**
- **Improved the audit log event recording process: events are now processed through a separate `ak_audi`**
- **Significantly optimized the process of checking and updating sending policies. Previously, during cam execution speed up to 10 times.**
- **Optimized the profile import process through batch data processing. In some cases, this can double t**

Other

- **In the “Dependent objects” section of segments, all related objects are now shown correctly, includin**
- **API content is now displayed correctly when viewing a profile through a scenario on wait nodes.**
- **Global control group calculation now works correctly for large profile databases.**
- **Some security vulnerabilities in the platform have been fixed.**
- **Fixed an issue where after merging profiles, a goal was saved in history but the profile did not appear**
- **Sending policy now works correctly, and profiles that do not match campaign conditions are no longe**
- **When using SQL queries in templates, the template now renders correctly even if the profile is missin**
- **Fixed goal registration via API for custom channels using `send_message_id`.**
- **Spaces at the beginning and end of goal names are now automatically trimmed.**

References

v2025.3.73.1942: PL-19442, PL-19540, PL-19064, PL-17853, PL-19472, PL-19467, PL-19453, PL-18128, PL-1 17817, PL-18222, PL-19195, PL-18533, PL-19098, PL-18954, PL-19055, PL-18796, PL-18542, PL-18529, PL-1

---

v2025.3.73.2276 – Platform Improvements, Fixed Event Context

This update fixes frequent issues and improves the platform’s functionality and stability.

Improved

Event Context

Fixed errors in the handling of variables related to the event context:

- **The `event` object now contains the number of events specified in the `WORKFLOW_EVENT_CONTEXT_LENGTH` paran**
- **The event iterator within a template (such as `{for $index $event = event.all}`) now works correctly again**

Learn more about these variables and how to use them [in this article](#).

Trigger Re-fire Blocking

The pixel option Block trigger re-firing now limits specific goals of a pixel rather than the entire pixel. For

Other

- **Fixed a logging issue related to the incorrect behavior of the `LOG_ROTATE_MAX_FILES` parameter. It now de**
- **Added configuration parameter `PROC_TASK_RESERVED_TASK_IDLE_TIME_LIMIT_MIN`. It defines the maximum idle ti**
- **Static segments are no longer recalculated automatically when sent to a Scenario — only profiles tha**
- **Scenarios using SKU additional field filters now work correctly again.**
- **Special characters now display properly in push messages sent via AppMetrica.**
- **Fixed an issue where link addresses were truncated in web versions of emails.**
- **Export of emails created in the [Visual Editor](#) now works correctly again.**
- **Fixed an issue where some push templates failed to insert Promo Code variables.**
- **The URL decoder in the Admin Panel can now process tracking links from Custom channel messages.**
- **Fixed a message delivery issue that occurred after changing Custom channel settings. Templates in s**
- **Fixed an authorization issue that occurred when using the `PASSWORDS_MAX_AGE` parameter.**

- **Message templates containing the `{resource_token}` variable now render correctly.**
- **Optimized workflow statistics logging, reducing RabbitMQ load.**
- **Improved process stability.**
- **Fixed several UI/UX issues.**
- **Added Redis connection check at platform startup.**
- **Connection to the platform via LDAP (both SSL and TLS) now works correctly again.**

References

v2025.3.73.2276: PL-19691 PL-18213 PL-19277 PL-11085 PL-19553 PL-19022 PL-19574 PL-16848 PL-19242 18362 PL-18568 PL-19155 PL-11262

v2025.3.73.2565 – Action Hook for Past Events

Added

Historical Events Capture

You can now transfer past events using the historical action hooks. When configuring the capture, use the Learn more about historical action hooks here.

Improved

- **The platform now registers a hard bounce event and updates the subscription status when sending a**
- **The block editor has been updated: template rendering is more stable, the default email template has**
- **A new version of the 360Dialog integration for WhatsApp messaging is now available.**
- **Interface improvements have been made for tables and counters.**
- **Events related to global control groups have been added to event capture.**

Other

- **Validation for wait time and pixel value on the pixel code page has been added.**
- **Stability of the `procsmsev` and `proctask` processes has been improved.**
- **An issue occurring when connecting a database without tables has been fixed.**
- **Markdown support for custom channel previews has been added.**
- **A misleading modal window during scenario activation has been fixed.**
- **The `/cookiesaver/readiness` handler works correctly again.**
- **Issues with the SMS Traffic sender integration have been resolved.**
- **A user with an expired password is now required to change it upon the next login.**
- **A parameter can once again be inserted and saved in the "Click-through link (SSL only)" field when ed**
- **Logging of web push deliveries to Safari has been improved.**
- **The `format` function no longer strips the minus sign from negative fractional numbers beginning with z**

References

v2025.3.73.2460: PL-19898 PL-19216 PL-19876 PL-19680 PL-18358 PL-19638 PL-19681 PL-19862 PL-19573 19592 PL-19594 PL-19394 PL-19469 PL-19334 PL-18736 PL-17537 PL-16721

---

v2025.3.73.2888 – Tag Manager variables in HTML tags

Added

Using Tag Manager variables in HTML tags

Previously, variables could only be used in Tag Manager triggers. Now they can be inserted directly into a

Other

- **The pixel activation trigger now checks date ranges correctly.**
- **NPS survey statistics no longer reset after updating the form.**
- **Scenarios that use external table queries now run without errors.**
- **Multi-step forms with profile field substitutions can be submitted successfully.**
- **Fixed an issue where apicontent was incorrectly handled during bulk profile imports into a scenario.**
- **Numeric values from apicontent are now inserted into API node URLs in the correct format.**
- **Profile history export now includes goal data, including pixel_data.**
- **Fixed an issue that prevented connecting to a Yandex Cloud database using a CA certificate.**
- **Optimized the performance of proctask and procintegras processes.**

- **Some UI/UX issues in the interface have been fixed.**

References

v2025.3.73.2888: PL-16683, PL-17392, PL-18422, PL-18751, PL-18847, PL-18862, PL-18876, PL-19523, PL-1
20320

v2025.2.72

v2025.2.72.0 – Redis Storage, Process Optimization

Added

Redis Storage

For the correct functioning of the platform, it is now required to use Redis-compatible storage. Such stora

- **enables a gradual transition away from file-based persistent storage;**
- **increases the platform's fault tolerance and manageability;**
- **ensures more reliable and faster interaction between services.**

We recommend using Kvrocks. The platform installation documentation includes a guide for connecting K
Using Redis is allowed but only with AOF data persistence mode or RDB + AOF mode.

Support for RabbitMQ Quorum Queues and Unified Virtual Host

> **CAUTION**
>
> **Before installing the update, you need to create a new virtual host (by default, altcraft) and ensure that t**

> **INFO**
>
> **If there are messages left in old virtual hosts, you can use the Shovel plugin to transfer them. Below is a**

```
rabbitmqctl set_parameter shovel trk_read '{
  "src-protocol": "amqp091",
  "src-uri": "amqp://altcraft:abcdefghijklmnopqrstuvwxyzABCDEF@localhost:5672",
  "src-queue": "trk_read",
  "dest-protocol": "amqp091",
  "dest-uri": "amqp://altcraft:abcdefghijklmnopqrstuvwxyzABCDEF@localhost:5672/altcraft",
  "dest-queue": "trk_read"
}'
```

Other Changes

- **Users can now create a new report based on an existing one.**
- **Added support for configuring a handler pool for the RMQ Event Sender.**
- **The user session duration is now configured via the `PROCAUTH_SESSION_TIME_SEC` parameter, and the ability**

Improved

- **Platform performance has been improved.**
- **Stability of the `procintegras` process has been improved.**
- **Task processing in the `proctask` process has been optimized.**

Other

- **Platform performance improved.**
- **Fixed several UI/UX issues.**
- **Increased stability of the `procintegras` process.**
- **Fixed authorization-related errors.**
- **Optimized task processing logic in `proctask`.**
- **Fixed incorrect display of analytics metrics.**
- **Confirmation link validation in templates works properly again.**
- **Automation Scenario elements now operate more reliably — previously occurring errors have been re**

References

v2025.2.72.0: PL-18318 PL-18343 PL-18519 PL-18367 PL-18349 PL-18323 PL-18192 PL-18321 PL-18322 PL-18295 PL-18255 PL-17490 PL-17886 PL-18134 PL-17904 PL-16949 PL-16946 PL-17729 PL-17590 PL-16965 P 17091 PL-11284 PL-16885 PL-17030 PL-17029 PL-16969 PL-17013 PL-6658 PL-16541 PL-16356 PL-16702 PL

---

v2025.2.72.707 – Custom channel improvements, link shorter optimization

> **DANGER**
>
> **Pay attention to the important changes in the link shortener behavior:**
>
> - **The content of the web version of the message is now transmitted without modifications. This mean**

- **The link format has changed from `^(/s/[A-Za-z0-9_-]{5,20})$` to `^(/s[A-Za-z0-9_\-!~\.]{10,12})$`.**

Added

- **Added dynamic display of process status when starting and stopping the platform using `akd`.**
- **The number of profiles entered and exited is now displayed above scenario elements.**
- **The variable insertion menu now includes scenario name and scenario ID, as well as element ID and e**
- **Users can now override form templates using `custom` files.**
- **Added a launch log for the trigger scenario element "Scheduled Capture".**
- **The profile card now displays information about scenario errors if the profile was in the GKG.**
- **API requests for the scenario list and template info now support `updated_user` and `update_time` paramete**

Improved

Custom Channel Pipeline Improvements

The operation of custom channel pipelines has been significantly improved. The following changes were n

- **Fixed errors occurring during pipeline execution.**
- **Added access to object names.**
- **Pipelines can now accept event dates in the EventObject.**
- **Added a unique message identifier in the pipeline, matching the ID in ClickHouse.**

Other Improvements

- **It is now possible to override the Firebase URL for each platform account in the configuration file.**
- **The search and filtering interface across the platform has been updated.**
- **Refactored link shortener operation. Duplicate link IDs were eliminated, and the maximum link length**
- **The `maxAmountOfChanges` constant was moved from `segstat_recalc` to a separate configuration parameter –**
- **Optimized the Best Send Time ML service. Related configuration parameters were also updated.**
- **Improved platform performance.**
- **Improved stability of `procworkflow`, `webcontrol`, and `proctask` processes.**

Other

- **Static segments created via API are now recalculated correctly.**
- **Fixed incorrect calculation of segments with identical file names.**
- **The last update date is now correctly displayed in profile databases.**
- **The subscription capture trigger in scenarios now considers the selected channel and resource.**
- **Fixed a JavaScript error on the campaign type selection page when creating a campaign from the cal**
- **The session in the web interface no longer expires too quickly.**
- **The `adddate` function now works correctly in the "API Call" scenario element.**
- **The JSON filter now supports fields with underscores in their names.**
- **Push templates imported via API can now be opened in the beta editor.**
- **The `not_strict` and `subscription_filter` parameters in scenarios now work correctly.**
- **AKMTA now correctly processes DSN when using the `From` field address in `MAIL FROM`.**
- **Fixed 500 error when requesting the web version via API for profiles in the stop list.**
- **Fixed an error that occurred when placing a pixel via the Tag Manager.**
- **Fixed parsing of status responses from the SMSTraffic sender.**
- **Phone numbers in .xls files are now parsed correctly.**
- **The `send_message_id` value in custom JSON for Push now matches the `send_message_id` from the event cap**

References

v2025.2.72.707: PL-18764 PL-18548 PL-18710 PL-18763 PL-18671 PL-18659 PL-17043 PL-16495 PL-17891 P
17894 PL-17772 PL-18132 PL-18672 PL-18703 PL-18712 PL-18669 PL-18619 PL-18613 PL-18648 PL-18546 P
18660 PL-18658 PL-18447 PL-18494 PL-18632 PL-15330 PL-18592 PL-17661 PL-17120 PL-18534 PL-18446 P

---

v2025.2.72.1475 – password recovery and billing info

CAUTION

As part of this update, the structure of stored data has been changed, so for the platform to work correc

`./aktool migrate add-updatefeed --config=/opt/MarketingPlatform/config/main.json`

Do not run the migration multiple times! The migration is performed only once. If any errors occur, pleas

Added

- **Metrics of profile processing time have been added for each campaign stage.**
- **A numeric postfix has been added to the log file upon rotation.**
- **For content, variables `{segid}` and `{segname}` have been added — ID and name of the segment.**
- **New parameter `PROCAUTH_WELCOME_LINK_TTL_SEC` added — lifetime of the registration completion and passw**
- **Additional logs have been added to the Reply processing in AKMTA.**
- **New indexes have been added in MongoDB for the most frequent queries to improve platform stability.**
- **Integration with SMS provider 7tech has been added.**
- **Logs for Notify requests have been added to obtain integration status.**

Improved

- **Profile sorting removed when filling a static segment.**
- **Extended information in the freeze and account deactivation notification.**
- **Password change page updated, along with the success notification email.**
- **Improved stability of the `proctask` process.**
- **Additional Market product fields are now imported with correct data types.**

Other

- **Fixed several UI/UX issues in the interface.**
- **The `--faster` flag of the `./akd restart` command works correctly again.**
- **Fixed a bug where conditions with array references were not displayed in the “JSON filter” element.**
- **Images in Viber templates are once again correctly inserted through the gallery.**
- **Eliminated false triggers related to profile relations.**
- **Campaign schedule start time is now calculated correctly, even when using a time zone different from**
- **Fixed errors occurring when recalculating profile segments.**
- **Scenarios are now saved correctly when using the “Capture profiles on subscription add/change” node**
- **Fixed JS error when opening the “Segment by schedule” node.**
- **AKMTA now saves emails received from the postmaster mailbox.**
- **Added `X-Letter-Id` header for Sendsay sender.**
- **Profile card now displays unsubscribe events for the Viber™ channel again.**
- **The `uppercasefirst` function now works correctly with profile fields.**
- **Fixed errors related to sending emails after form submissions.**
- **Telegram template preview works correctly again.**
- **Fixed a bug where incorrect data was displayed in GCG.**
- **Fixed issues related to user roles and access rights.**

References

v2025.2.72.1475: PL-18764 PL-18548 PL-18710 PL-18763 PL-18671 PL-18659 PL-17043 PL-16495 PL-17891 17894 PL-17772 PL-18132 PL-18672 PL-18703 PL-18712 PL-18669 PL-18619 PL-18613 PL-18648 PL-18546 P 18660 PL-18658 PL-18447 PL-18494 PL-18632 PL-15330 PL-18592 PL-17661 PL-17120 PL-18534 PL-18446 P

---

v2025.2.72.1664 – interface improvements

> **TIP**
>
> **As part of the update, the structure of stored data has been changed, so to ensure the platform works c**
> **./aktool migrate add-updatefeed --config=/opt/MarketingPlatform/config/main.json**
> **If you have already performed the migration, do not run it again! Migration is performed only once. If you**

Improved

Interface fixes

Several UI/UX issues have been resolved:

- **Localization has been added instead of system names in certain places**
- **Improved display of some warnings and error messages**
- **Added a documentation link for Basic Authentication and Token Authentication connections**
- **Updated fonts and styles of some texts in the platform**
- **Fixed overlapping of certain Scenario elements**

Other

- **WhatsApp sending with a “PHONE” button in EDNA now works correctly**

- **Fixed an issue where templates with images could not be imported from an archive**
- **Field values are now saved correctly when creating promo codes**
- **Optimized performance of the `procintegras` process**
- **Fixed issues with sender queue counters**
- **Improved stability of the “Change field value” node**
- **Fixed several issues related to template previews**
- **Added the ability to save a connector to an external DB with SSL connection without specifying certif**
- **Fixed an error when trying to view analytics for goals containing slashes in their name**
- **Viewing API content through a Scenario element now works correctly**
- **Fixed `empty query` error in SQL segmentation requests**

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated as a
****Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.***

v2025.1.71

v2025.1.71.0 - Global control groups, administrative panel rework, NPS testing

> **CAUTION**
>
> **For the platform to work correctly after the upgrade you need to perform a migration, to do this:**
>
> **1. Before running `procevent`, perform the data migration for ClickHouse:**
>
> ```
> /opt/MarketingPlatform/ak/bin/aktool profile_history add_gcg_id_column --config /opt/MarketingPlatform/config/main.json
> /opt/MarketingPlatform/ak/bin/aktool history_storage add_gcg_id_column --config /opt/MarketingPlatform/config/main.json
> /opt/MarketingPlatform/ak/bin/aktool history_storage add_profile_marks --config /opt/MarketingPlatform/config/main.json
> ```
>
> **The platform can only be started after the ClickHouse migrations are completed.**
>
> **2. After completing the ClickHouse migrations, run `aktool` from the new build to migrate MongoDB:**
>
> ```
> ./aktool --config config/main.json profiles migrate-gcg-flags
> ```
>
> **Note that the MongoDB migration may take from one to several hours. The global control groups feature**
>
> **3. Start `procevent`.**

Major changes

Global control groups

The platform now includes a feature to select global control groups. Global control groups are parts of th
main part of the clients and the control group. In case the groups' metrics differ insignificantly, the marke

Global control groups

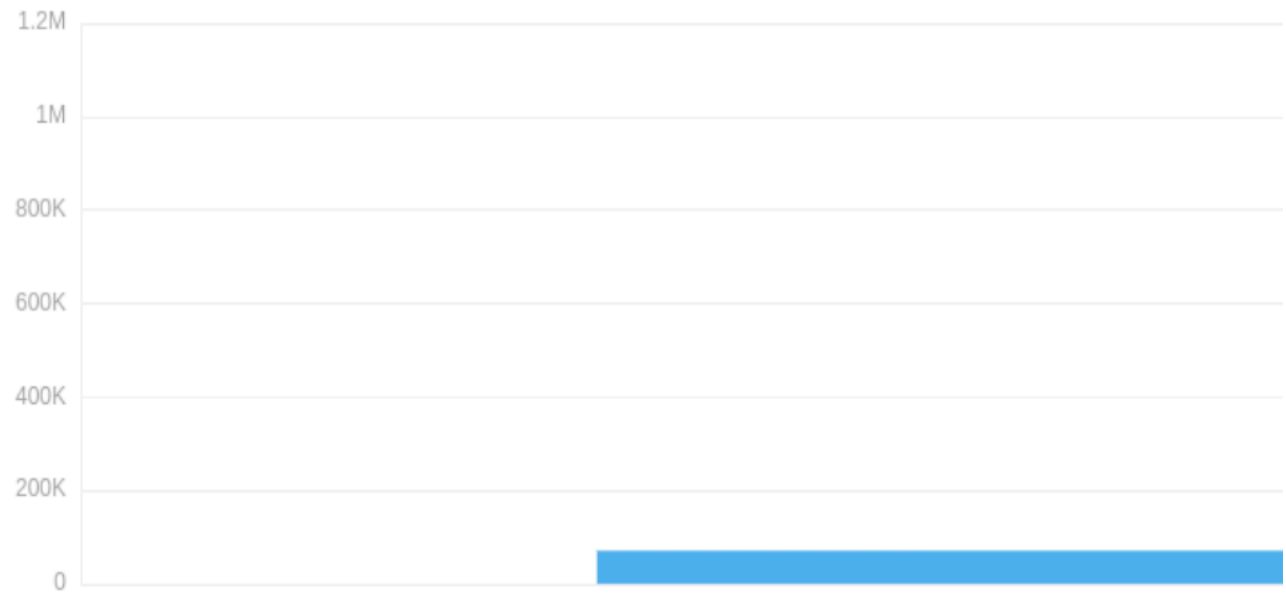
Marketing promotion *

Report on database #36 **GCG** by: **goals** for the period: **This month**

Group by **Months** in which to show:

Add filter

Sum value



You can read more about global control groups here.

NPS testing in Forms

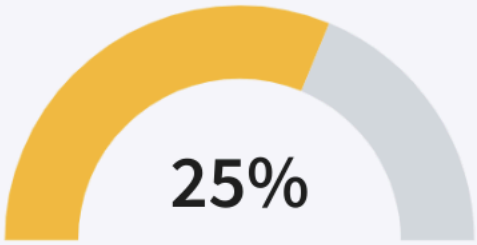
You can now add a NPS survey to web forms created in the Altcraft platform. These surveys help you calc allows you to analyze the strengths and weaknesses of a product and adjust your marketing strategy:

#4 NPSform

Results analysis | Statistics of survey

General Form Results

General Net Promoter Score



General group dis

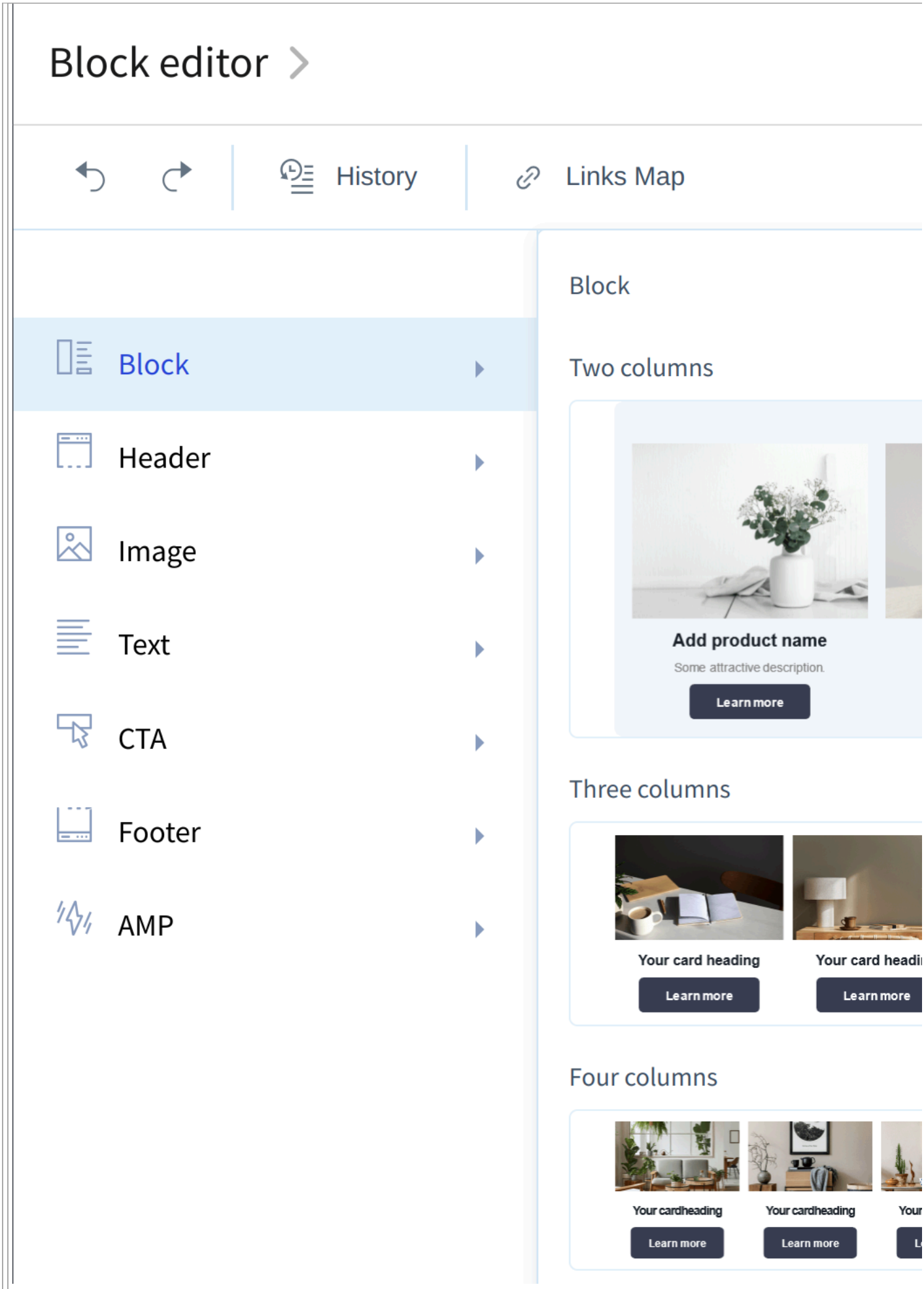
General NPS trend

-100%
2024-11-05
2024-11-06
2024-11-07

To learn how to set up NPS surveys in the platform, read the article or use this video guide.

Block email editor

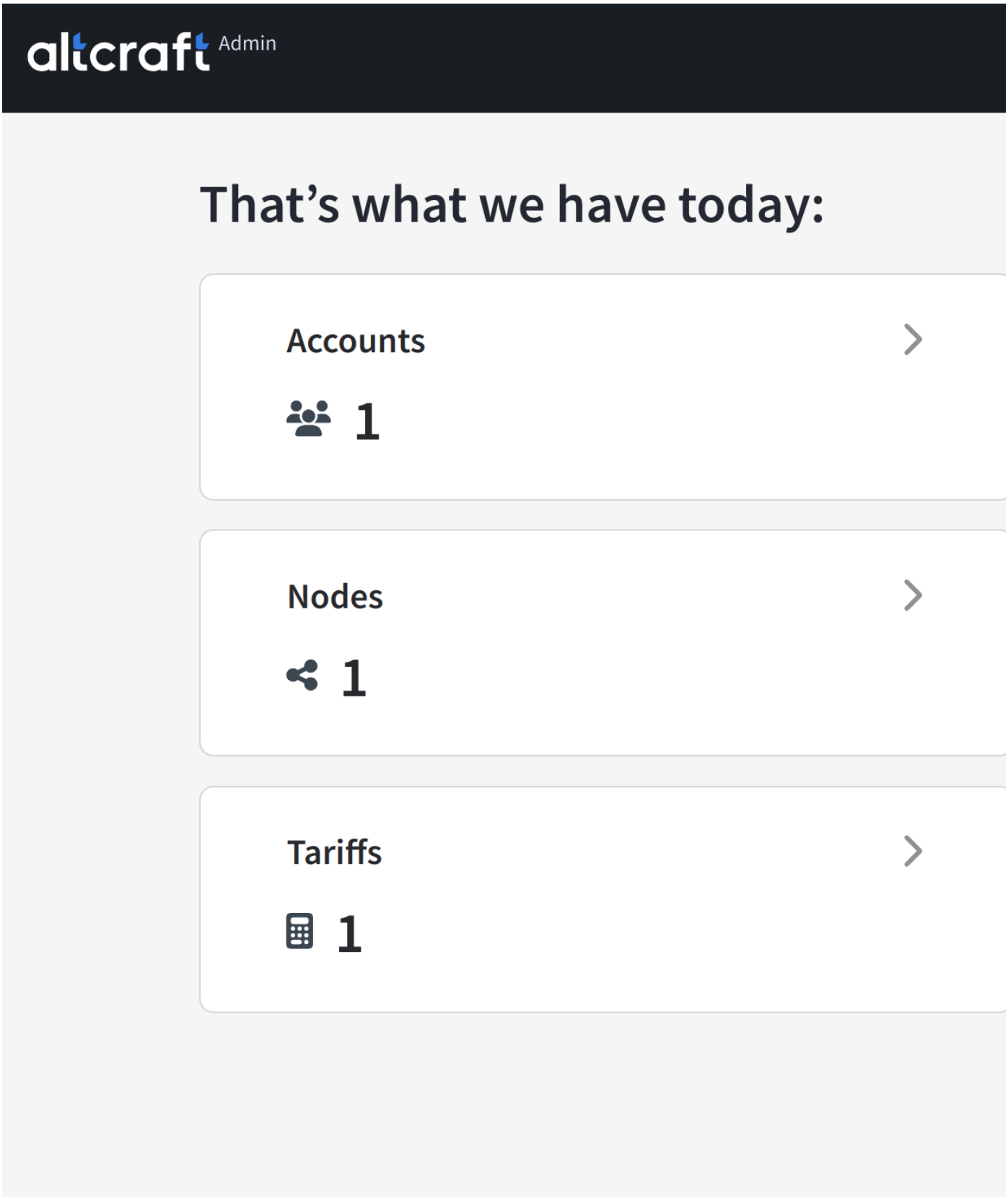
The block editor Pixcraft has been added to the email template editor. It allows you to quickly and easily



You can read more about block editor here.

Redesigned administrative panel

The administrative panel has been significantly redesigned: the interface has been changed, new function



WARNING

Starting with version 72, the platform will require the use of Redis-compatible storage, we suggest using

Integration with the SMS provider Stream Telecom

Added an integration with the Stream Telecom SMS provider. To set up the integration, configure the corr

Minor changes

- **Mass operations in the platform (activation, deactivation and deletion) for such elements as scripts, t**
- **Added support for Request-ID and Token header passing for API 1.1. requests, also improved request**

- The connection interface has been updated.
- Added a warning when attempting to delete profiles that have recently participated in campaigns.
- Updated the scheduling settings interface for some platform sections.
- Clicking the “Save” button no longer closes the editor or redirects the user to the section menu.

Fixes

- Fixed a bug where the status of machine learning settings was not saved in the platform.
- Now the user cannot delete entities from the cart if the account is frozen.
- Fixed a bug that caused clicks in SMS and push messages to register unnecessary delivery events.
- Fixed errors in Tag Manager variables.
- Scenarios configured to capture when a pixel is activated now work correctly.
- Made some changes to the Rich Push editor.
- Fixed a bug where while importing products with custom fields, an attempt was made to re-create the
- Screening a link with parameters inside a pixel now works correctly.
- Improved stability of the `procworkflow`, `procintegras`, `proctask`, `proctrigger`, `procevent` and `akmtad` processe
- Fixed an issue where a Viber campaign with SMS Traffic sent a message to only one subscriber.
- Information about a profile’s participation in a scenario is now updated correctly.
- Resolved an issue where login credentials were not displayed in the welcome email.
- Messages in RabbitMQ queues are now processed correctly again.
- Added validation for file extensions when creating a segment based on the “In file column” condition.
- Special characters in file names no longer cause errors when querying the data table.
- The SMS template is now can be saved even if the “SMS Limit” field is empty.
- Resolved issues with test sending for Telegram channels.
- Form data is no longer duplicated when transferred to the platform.
- Fixed an issue where clicking links in push notifications did not work in Mozilla Firefox.
- Telegram campaigns now use less memory when launched.
- Improved stability of the Analytics module: fixed various issues related to report generation and data
- The `$dateNowFunc` function in JSON requests now works correctly again.
- Fixed an issue where tags were not added to multiple objects.
- Template preview images are now exported correctly.
- Fixed errors in the `market/sku/import` and `profiles/segstatic/list` API requests.
- Added validation for deleting external databases.
- Deep Links no longer disappear when saving a template in the push message editor.
- Tags in scenarios are now displayed correctly.
- Fixed an issue where a campaign would not open when navigating from a report or profile history.
- Profiles frozen in a scenario are no longer deleted when the scenario is modified.
- Fixed a bug where AKMTA was sending empty messages.
- Day of the week waiting condition works correctly in scripts again.
- Improved validation of email addresses in the list of recipients of a hidden copy of an email.
- Push message receive and open events are logged correctly.
- Optimized the process of generating and sending Push messages.
- Improved logging for certain processes.
- Interface improvements: fixed display issues for some elements and resolved errors and typos in platf

References

v2025.1.71.0: PL-17608 PL-17599 PL-17574 PL-17572 PL-17570 PL-17564 PL-17558 PL-17556 PL-17549 PL-17374 PL-17373 PL-17372 PL-17369 PL-17327 PL-17322 PL-17305 PL-17303 PL-17287 PL-17285 PL-17283 P 16953 PL-16930 PL-16907 PL-16854 PL-16701 PL-16700 PL-16679 PL-16641 PL-16640 PL-16607 PL-16600 P 16386 PL-16358 PL-16350 PL-16348 PL-16344 PL-16328 PL-16296 PL-16247 PL-16201 PL-16144 PL-16069 P 15245 PL-15021 PL-14968 PL-14230 PL-13369 PL-12949 PL-12732 PL-12297 PL-17562 PL-17634 PL-17579 P

---

v2025.1.71.255 - Mandatory use of the resource in campaigns

> CAUTION
>
> **Starting with the next update, sending campaigns without an assigned resource will not be possible!**
> **If you have been using campaigns without a resource, please assign a resource to them.**

Major changes

Mandatory use of the resource in campaigns

From the next update, all campaigns in the platform will use a resource. Therefore, it is now impossible to

> In this campaign, sending without specifying a resource is used. Saving of such campaigns is disabled. You can find more information about
> subscriptions in the documentation.

Minor changes

- **Improved logging of Reply processing in AKMTA.**
- **Added configuration parameter responsible for the maximum size of unloaded data from a SQL table c**

Fixes

- **Fixed display problems with some interface elements.**
- **Improved operation of scenario elements “Capture profiles on pixel activation” and “Profile update”, a**
- **Fixed errors occurring when registering Viber events in SMS Traffic integration.**
- **Viber integration with Rapporto now works correctly on all accounts.**
- **Optimized `procevent` queue parsing process.**
- **Fixed a bug where deleting profiles was available to users without proper permissions.**
- **Profile counter in a segment now works correctly when sending via Telegram bot.**
- **Fixed an issue where data received from push messages was not fully transferred to the platform.**
- **Improved stability of `procworkflow` and `proctask` processes.**

References

v2025.1.71.255: PL-17644 PL-17561 PL-17618 PL-17648 PL-17643 PL-17640 PL-16953 PL-17552 PL-17557 P

---

v2025.1.71.507 - Restructuring the policy counter repository

> CAUTION
>
> **You will need to migrate for the platform to work correctly after the upgrade. Before launching the new v**
>
> **1. Stop the required platform services:**

```
./akd stop {proctrigger,procworkflow,proctask}
```

> **2. Run `aktool` from a new build with the command:**

```
./aktool migrate policystore --config /opt/MarketingPlatform/config/main.json
```

> **3. Restart the platform services:**

```
./akd restart
```

> **Help for the migrator:**

```
Migrate policystore db to accounts dbs collections policy_contact, policy_profile

Usage:
  aktool migrate policystore [flags]
  aktool migrate policystore [command]

Available Commands:
  ensure_policystore_indexes Ensures policystore migrate indexes created

Flags:
      --account int          account id
      --batch_interval int   batch interval in seconds (default 1)
      --batch_size int       batch size (default 10000)
      --batch_workers int    number of batch workers (default 8)
  -h, --help                 help for policystore
      --workers int          number of workers for accounts (default 8)

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output

Use "aktool migrate policystore [command] --help" for more information about a command.
```

Changes

- **Optimization of the repository structure of sending policy counters has been performed.**
- **Metrics on API calls to the scenarios are now available in analytics.**

- **Special characters can now be used in entity names.**
- **Fixed some bugs in the functionality and interface of the Administrative Panel.**
- **Logging of some platform processes has been improved.**

Fixes

- **Fixed an error where push campaigns generated a large number of non-existent undeliveries.**
- **Statuses in SMS integration with Devino Telecom are now processed correctly.**
- **Fixed errors occurring when testing custom channel campaigns.**
- **Entity analytics with HTML tag in the name works correctly again.**
- **Fixed errors occurring when registering events in Viber integration with SMS Traffic.**
- **Improved stability of Web Push SDK methods.**
- **Fixed some interface bugs.**
- **Improved stability of `proctask` process.**

References

v2025.1.71.507: PL-17756 PL-17753 PL-17446 PL-17727 PL-17731 PL-17728 PL-17715 PL-17720 PL-17586 P 16354

---

v2025.1.71.627 - RMQ buffer base mode

Changes

- **Added buffer base mode for RMQ in `prochook`.**
- **Metrics on API calls to the scenarios are now available in analytics.**
- **Special characters can now be used in entity names.**
- **Added the option to configure `global qos` for RMQ.**
- **Fixed some bugs in the functionality and interface of the Administrative Panel.**
- **Logging of some platform processes has been improved.**

Fixes

- **Fixed an error where push campaigns generated a large number of non-existent undeliveries.**
- **Statuses in SMS integration with Devino Telecom are now processed correctly.**
- **Fixed errors occurring when testing custom channel campaigns.**
- **Fixed a bug related to incorrect display of inactive campaign status.**
- **Entity analytics with HTML tag in the name works correctly again.**
- **Fixed errors occurring when registering events in Viber integration with SMS Traffic.**
- **Improved stability of Web Push SDK methods.**
- **Fixed some interface bugs.**
- **Improved stability of `proctask` process.**

References

v2025.1.71.627: PL-17756 PL-17753 PL-17446 PL-17727 PL-17731 PL-17728 PL-17715 PL-17720 PL-17586 P 16354

---

v2025.1.71.873 - SSL support for CH

Major changes

SSL support for ClickHouse

Added support for SSL-secured connections to ClickHouse DB. To enable this, activate the corresponding

Minor changes

- **The global control group now retains groups assigned to the original profile database.**

Fixes

- **Custom channel events are now displayed correctly in profile history.**
- **Improved message sending via Telegram bots in scenarios.**
- **Campaign testing within scenarios is functioning correctly again.**
- **Fixed an issue where subscription fields for user channels were not saved.**
- **Relations now work correctly in message templates.**
- **Fixed an issue causing the `send_message_id` parameter to take different values for the same message in**

- **Phone number filtering in scenarios is functioning correctly again.**
- **Fixed an issue causing sending policy parameters to change randomly.**
- **Several improvements and fixes in the Admin Panel.**
- **Fixed issues related to formatting floating-point numbers.**
- **Improved profile history and certain logs.**
- **Fixed an unintended subscription priority change when modified via API.**

References

v2025.1.71.873: PL-17067 PL-17351 PL-17361 PL-17888 PL-17876 PL-17814 PL-17801 PL-17536 PL-17748 P 17707 PL-17729 PL-17472 PL-17708

---

v2025.1.71.1059 - Scenario API improvements

Major changes

Scenario-related API improvements

- **Added activation and deactivation requests for a scenario: `/workflows/activate` and `/workflows/deactivate`.**
- **Scenario API responses now include campaign and template identifiers, as well as information about**

Minor changes

- **The maximum length of scenario titles has been increased to 156 characters.**
- **Improved logging of certain platform processes.**

Fixes

- **Improved layout for several UI elements.**
- **Fixed an issue where test launches of campaigns using a dynamic segment as a subscriber source w**
- **LDAP authentication now works correctly again.**
- **The campaign filter now finds campaigns that were created within a scenario.**
- **Fixed errors occurring during Yandex.Audience segment resynchronization attempts.**
- **Fixed an issue where form publish and hide times were incorrectly set in the settings.**
- **Tracking services again correctly listen to the address and port passed through parameters.**

References

v2025.1.71.1059: PL-17896 PL-17938 PL-16077 PL-17925 PL-17832 PL-17831 PL-17823 PL-16045 PL-16896

---

v2025.1.71.1487 - API calls optimization, improved logging

Changes

- **Extended data type support in ClickHouse table integrations.**
- **Added the option to specify the launch time of global control groups in the platform configuration file.**
- **The platform now logs the processing time for each profile at every campaign stage.**
- **Added a button to stop segment recounting.**

Fixes

- **Fixed an issue where data from a deleted profile field was still returned via the API.**
- **Duplicate relations no longer appear when created via API.**
- **Optimized the `/profiles/import_batch` API method.**
- **The formatted date variable in the “API call” scenario element works correctly again.**
- **Fixed font rendering issues in popups.**
- **Links to the web versions of emails now work correctly in test sends.**
- **Fixed several UI-related bugs.**
- **Render errors are now properly displayed in the campaign log.**
- **Fixed an issue where templates and campaigns were not moved to the trash.**
- **The “Profile update” scenario element can now set a null value.**
- **The `blockquote` tag renders correctly in Telegram templates.**
- **Fixed several bugs related to custom channel.**

References

v2025.1.71.1487: PL-17984 PL-18003 PL-17901 PL-17947 PL-15365 PL-13140 PL-17926 PL-17900 PL-17945 17938 PL-16892 PL-17965 PL-17097 PL-17280 PL-17808 PL-17841 PL-17767 PL-17825 PL-17766 PL-17785 P

---

v2025.1.71.2447 - SberMobile and Intellin integrations

Major changes

New integrations: SberMobile and Intellin

The following integrations have been added:

- **integration with the Intellin SMS provider;**
- **integration with the SberMobile SMS gateway.**

Improved data logging for sends and scenarios

- **Logs for Notify requests have been added to retrieve integration statuses. These logs help you unders**
- **The USMID has been added to AKMTA headers. USMID is a unique message identifier that contains de**

Multiple links in “Link click” condition

You can now list multiple URLs in the segmentation condition Link click.

Minor changes

> **CAUTION**
>
> **If you specified an address in the RPC_HOST_CLI parameter, all clients (even from other hosts) will conn**

- **The `procworkflow` RPC client now connects using the address set in the dedicated configuration param**
- **Added the ability to batch-send web versions of messages to S3 storage.**

Fixes

- **Fixed issues related to two-factor authentication in the Admin Panel.**
- **Improved the stability of the Tilda integration.**
- **Fixed a bug where scheduled segments were not recalculated when a scenario was launched.**
- **Revised the logic for sending form submission reports, improving data accuracy.**
- **The `format` function now works correctly in loops and when used with `apicontent`.**
- **Improved click registration in Telegram bots.**
- **Delivery statuses and duplicate messages are now correctly registered for SMS campaigns.**
- **Resolved issues where users without appropriate roles could access platform entities.**
- **Test campaigns with dynamic subscriber segments now launch correctly.**
- **Fixed several UI/UX issues in the interface.**
- **Resolved errors related to using conditions in segments and scenarios.**
- **Custom bounce rules for AKMTA now function correctly again.**
- **Improved stability of the following processes: `webcontrol`, `procintegras`, `proctask`, `prochook`, and `procworkf`**

References

v2025.1.71.2447: PL-18148 PL-18022 PL-18189 PL-18131 PL-18102 PL-17352 PL-17981 PL-17931 PL-17397 15625 PL-17771 PL-17735 PL-17770 PL-17689 PL-17653 PL-15344 PL-14393 PL-12555 PL-18122 PL-18127 P 15009 PL-17861 PL-17977 PL-17874 PL-17400 PL-17003 PL-16578 PL-16589 PL-16022 PL-17105 PL-17489 P

---

v2025.1.71.2873 - Hive connector support

> **CAUTION**
>
> **You will need to migrate for the platform to work correctly after the upgrade. Before launching the new v**
>
> **1. Stop the required platform services:**
>
> ```
> ./akd stop {proctrigger,procworkflow,proctask}
> ```
>
> **2. Run `aktool` from a new build with the command:**
>
> ```
> ./aktool migrate policystore --config /opt/MarketingPlatform/config/main.json
> ```
>
> **3. Restart the platform services:**
>
> ```
> ./akd restart
> ```
>
> **Help for the migrator:**

```
Migrate policystore db to accounts dbs collections policy_contact, policy_profile

Usage:
  aktool migrate policystore [flags]
  aktool migrate policystore [command]

Available Commands:
  ensure_policystore_indexes Ensures policystore migrate indexes created

Flags:
      --account int          account id
      --batch_interval int   batch interval in seconds (default 1)
      --batch_size int       batch size (default 10000)
      --batch_workers int    number of batch workers (default 8)
  -h, --help                 help for policystore
      --workers int          number of workers for accounts (default 8)

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output

Use "aktool migrate policystore [command] --help" for more information about a command.
```

Major changes

SQL Connector Hive

Added support for the SQL Hive/Hadoop connector as an external database. Learn more about configuring

Minor changes

- **Updated APIs for the new versions of Google Ads and Facebook™* Ads Manager.**
- **Added support for Rabbit version 4+.**
- **The Altcraft platform has been certified for compatibility and proper operation with the domestic oper**
- **The `MTA_DEFAULT_TTL` parameter, which defines the delivery waiting time for SMS and Email, now default**
- **The `"not\\savailable"` blocking rule has been removed from the MTA sender rules because it caused er**

Fixes

- **Enhanced platform security mechanisms.**
- **Fixed various UI/UX issues.**
- **Improved stability of `akd`, `akmta`, `aktool`, `procpiper`, `proctrigger`, `procintegras`, `proctask`, and `procworkflow`**
- **Optimized `oxy_triggers` queue parsing.**
- **Custom bounce rules in AKMTA now work correctly.**
- **Optimized log rotation mechanism.**
- **Web versions of emails sent through the platform are now saved in the correct format.**
- **Improved 2FA stability.**
- **Fixed issues related to editing message templates within the platform.**
- **JSON filter now works correctly for nested objects.**
- **Fixed an issue where form data was mistakenly submitted twice.**
- **Added validation for certain fields in the Admin Panel.**
- **Fixed some issues related to access roles.**
- **Additional profile fields are now exported correctly in personal data exports.**
- **Fixed an issue where `apicontent` was incorrectly handled during bulk profile imports into a Scenario.**

References

v2025.1.71.2873: PL-17550 PL-18489 PL-18393 PL-18384 PL-18106 PL-18136 PL-15177 PL-18457 PL-18247 15410 PL-18294 PL-18289 PL-18135 PL-18235 PL-18314 PL-18290 PL-18198 PL-18145 PL-18175 PL-18013 P 18196

---

v2025.1.71.3092 - Event backward compatibility

Changes

- **Added backward compatibility for events from custom channels of legacy platform versions.**
- **Security mechanisms optimized.**

Fixes

- **Test sends to Telegram now work correctly.**
- **The value `0` in the “in data table” condition is now displayed when passed via query parameters.**
- **Fixed an issue in the form editor where incorrect access rules were applied.**
- **Added a timeout for hanging queries in PostgreSQL.**
- **The ability to specify a custom configuration file path for platform services now works properly.**
- **Improved readability of API logs.**
- **Tags and additional fields are now correctly processed when creating objects via API.**
- **The visual editor now correctly updates custom block settings.**
- **Fields of type “Integer” now work properly when forming a control group from a file.**
- **Reduced memory consumption by the `campaign` process when sending Telegram messages with attach**
- **Confirmation link check now works correctly in templates.**
- **Fixed the “Prioritize new subscriptions” option in resource settings.**
- **AKMTA now correctly switches to backup MX records when the primary is unavailable.**
- **The profile date is now correctly inserted in templates when using the `formatdate` function.**
- **Fixed issues related to handling events from Amazon SNS sender.**

References

v2025.1.71.3092: PL-10119 PL-18521 PL-18315 PL-17401 PL-18125 PL-18000 PL-17857 PL-17818 PL-18266

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated as a
Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

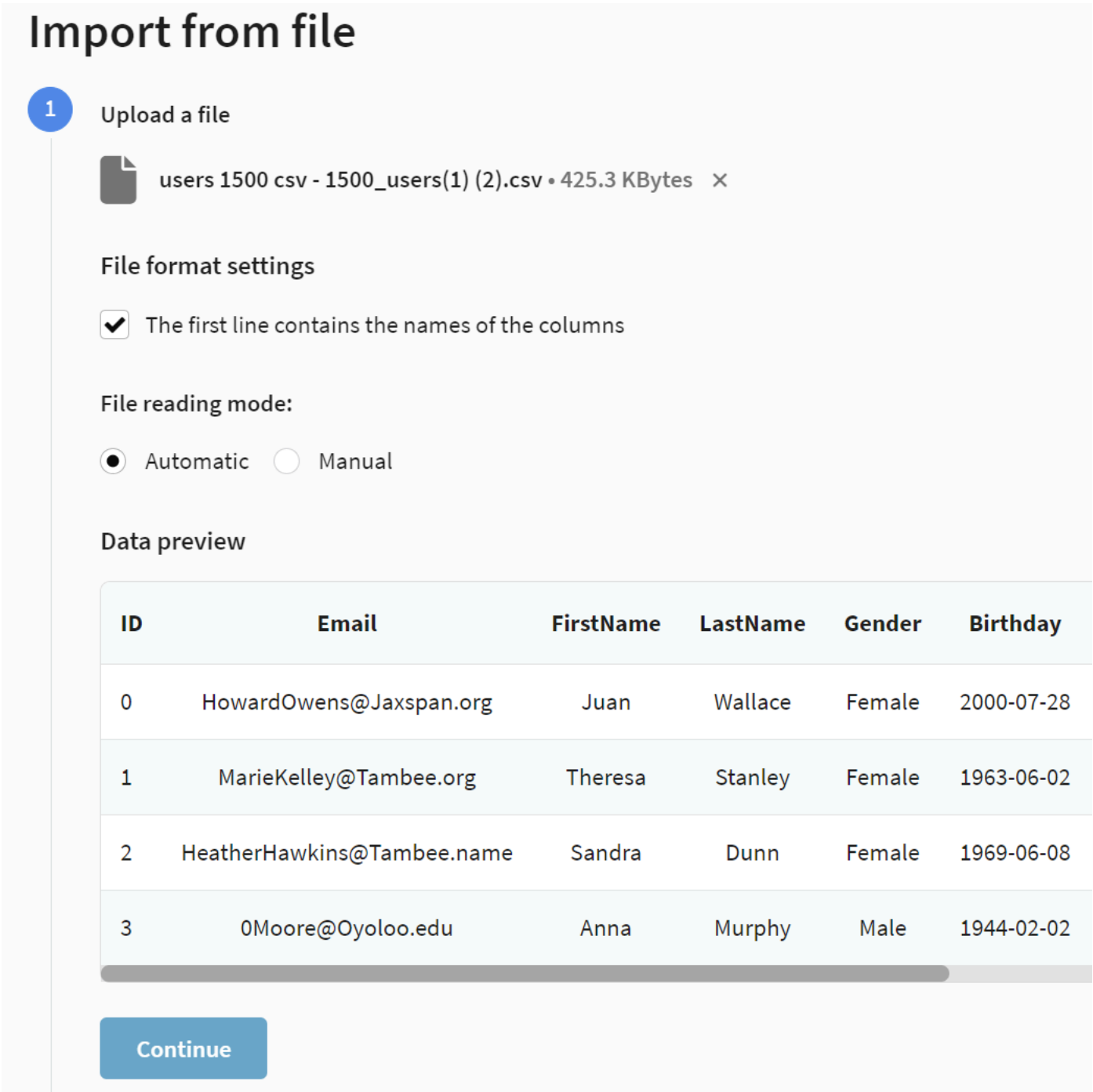
v2024.4.70

v2024.4.70.0 - New import menu, Rapporto integration, License info

Major changes

Improved import of profiles

Completely redesigned importing profiles from a file. Now when you select this item, a detailed step-by-st



You now have the possibility to select file encoding, delimiter, select date format and edit the format of pl
Also the import menu to the profile database has been updated:

Import

Import data

Manually

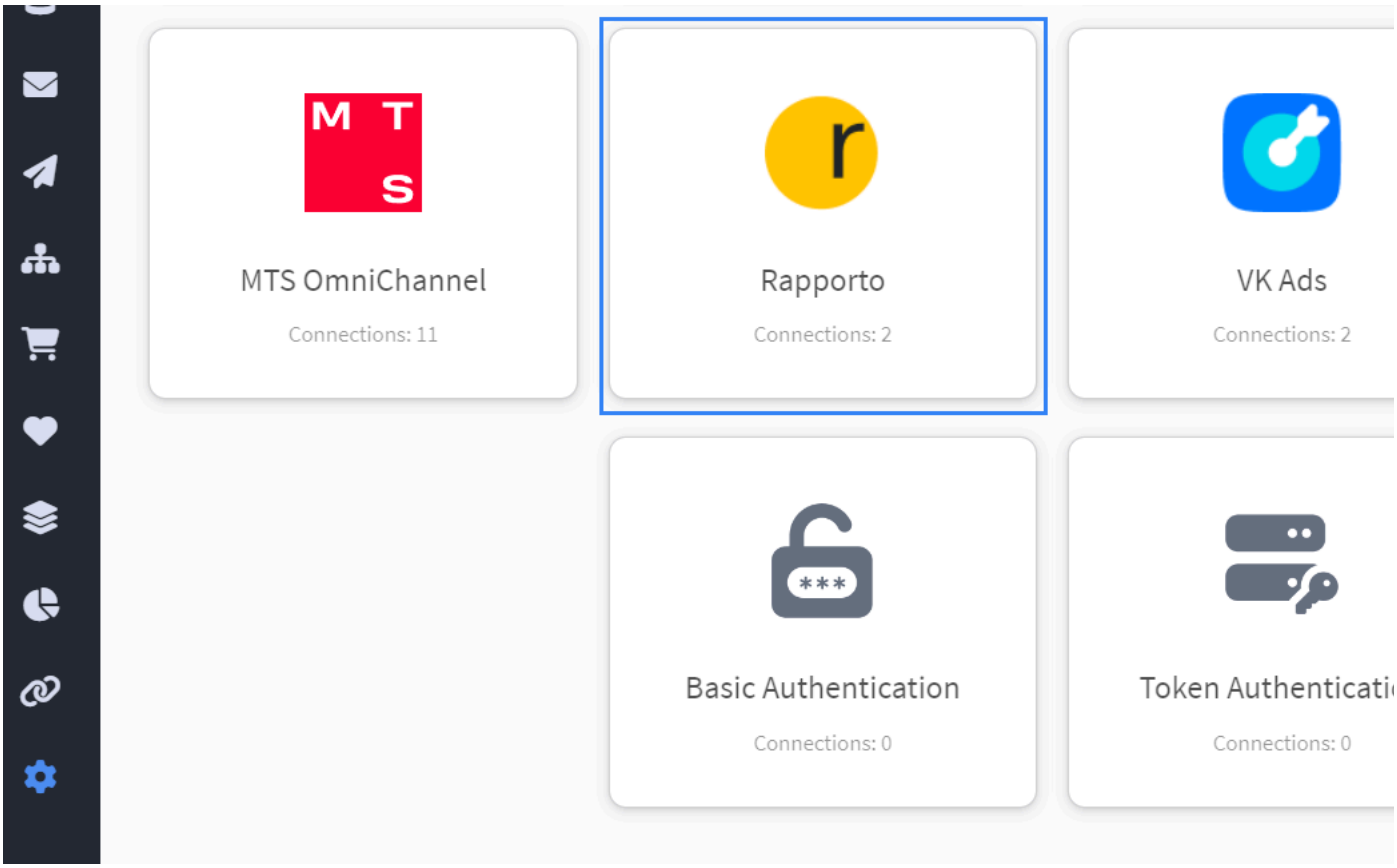
single profile creation

Import automation

On Schedule

SQL query to external datab

Added support for integration with Rapporto. Now you can use this platform as a gateway to send messag



Before setting up the integration, you need to connect your Rapporto account with Altcraft Platform via c

License information

Added License item in the Tools section of the Administrative Panel. In this item you can see detailed info
In the Notify about license changes to field, you can specify the email address to which you want to recei

Minor changes

- **New operators have been added to the "JSON filter" scenario element: `contains`, `in`, `notin` and `!~=`. R**
- **It is now possible to add custom headers to the "API request" scenario element.**
- **You can now send emails without the text version of the message.**
- **The `genders` directory has been updated.**

> INFO
> **If you need to keep the old `genders` directory configuration or make some changes to it, create a new file**

- **Optimized operation of the "Segment" scenario element.**
- **The mechanism of interrupting cycles in scenarios has been redesigned, now it works correctly in co**

Fixes

- **Fixed a bug where the exported template did not contain all source data.**
- **Fixed errors occurring during resynchronization with VK Ads service.**
- **Conditions of waiting for a certain day of the week now work correctly in scenarios.**
- **Fixed an issue where some channels could not be selected in trigger campaigns.**
- **Fixed an issue when exporting profiles with subscriptions.**
- **Fixed an issue where segmentation with negative conditions on Push deliveries was incorrectly selec**
- **Segmentation condition "Found in list" now correctly selects values containing a slash.**
- **Elements in scenarios now connect graphically to each other correctly.**
- **Fixed a bug where the delivery event for web push in Safari was logged before the actual delivery.**
- **Fixed false triggers of the "Timer" trigger in the Tag Manager.**
- **Fixed errors occurring when setting up connection to Facebook Ads.**
- **Improved analytics on test sends.**
- **Campaign filter now correctly detects campaigns created from scenario.**

- Fixed a bug where moving campaigns and templates to trash did not work correctly.
- Improved stability of `akmtad`, `procworkflow`, `proctrigger` processes.

References

PL-16980 PL-17004 PL-16990 PL-16934 PL-15925 PL-16939 PL-16170 PL-16565 PL-16649 PL-16306 PL-161
PL-17050 PL-17041 PL-17036 PL-17022 PL-17001 PL-16978 PL-16983 PL-16890 PL-16891 PL-16866 PL-169

---

v2024.4.70.963 - Action Hook SMID, push suspend method

Major changes

Changes in the identifiers of the action hook messages

The structure of the event capture message has been changed. Now such messages contain fields:

- `send_message_id` - unique identifier (USMID);
- `full_message_id` - FSMID for the event (if any);
- `email_message_id` - ESMID for an email channel.

Introduction of new identifiers allows you to find necessary events and thus avoid difficulties in error ana

New method to suspend push subscription

Implemented a new method to suspend push subscription of users to a resource via web push SDK. You c

Minor changes

- Added “Text + Image” combination for Viber templates.
- Added support for selecting authentication fields for MFMS EDNA SMS gateway.
- Expired web version collections are now deleted along with their contents.
- Added custom header editor for “API request” scenario element.

Fixes

- The counter of imported profiles in the Administrative panel is now automatically recalculated when t
- Attaching a file in the Segment section now works correctly.
- Fixed a bug where the platform was mistakenly registering more send and open events for some chan
- Images are now loaded correctly when importing an HTML template.
- Improved stability of the Analytics section, fixed some server errors related to analytics.
- Winner selection by clicks in multivariant tests now works correctly.
- Fixed a bug causing duplicate subscriptions for a profile when clicking the save button.
- Statuses on sends via WhatsApp* channel now arrive correctly.
- Improved validation when filling out fields in some sections of the platform.
- Import via file works correctly again for clients of the cloud version of the platform.
- Fixed an error occurring when connecting PostgreSQL connector to database on Yandex Cloud.
- Import via zip archive works correctly again.
- Fixed bug related to reaching the maximum limit of active sessions in MongoDB.
- Profiles caught in a scenarios are now parsed evenly by handlers without creating delays.
- Fixed a bug where a part of a message was cut off when previewing an SMS template.
- The Segment Participation condition now does not display the last created segment if the segment in
- Improved full screen mode of template editor on Safari.
- Email events in Sendsay integration are now logged correctly.
- Fractional number selection is now available again in the segment by numeric target data.
- Removed PostgreSQL queries to Market in scenarios where they don’t need to be initiated.
- Template with custom structure is now correctly imported into the platform.
- Fixed a bug where the platform would not start when one node in a ClickHouse cluster was unavailab
- Day of week wait condition now works correctly in scenarios again.
- Push notifications with special Unicode character sequences are now sent correctly on Firebase iOs
- Fixed auto-alignment algorithm in scenarios.
- When sending an empty SMS template, an error entry now appears in the campaign log.
- Event type in Notify is now correctly displayed when exporting profile history.
- Fixed a bug where it was possible to create a product without filling in mandatory fields.
- Fixed errors related to saving or deleting a subscription to a user channel resource.
- Disabling notifications in Safari browser now works correctly.

- **Supported file formats are now listed for importing from a file in Market.**
- **Fixed an issue where when copying a segment, the segment structure would retain incorrect conditio**
- **The irrelevant error “Config server db posgres is missing for account” has been removed from the can**
- **Fixed errors related to saving Telegram templates.**
- **Improved stability of `procworkflow`, `webcontrol` processes.**
- **Fixed some bugs in the platform interface.**

References

PL-17403 PL-17338 PL-17301 PL-16970 PL-17313 PL-17190 PL-17167 PL-17116 PL-13523 PL-17152 PL-1293
PL-17238 PL-17226 PL-16615 PL-17217 PL-16840 PL-17148 PL-17201 PL-17207 PL-17199 PL-15413 PL-1084
PL-16821 PL-16819 PL-17196 PL-17081 PL-17129 PL-17058 PL-17112 PL-16830 PL-13026 PL-16823 PL-1718
PL-16627 PL-16966 PL-16732 PL-16867 PL-16844 PL-16753 PL-2915 PL-5732 PL-11319 PL-16381 PL-16313

---

v2024.4.70.1088 - Web services handlers, stability improvements

Major changes

API requests to web services handlers

Correct balancing of web services requires from the user a constant and accurate transfer of their status `trkaction`, `trkwebctrl`, `cookiesaver`.
You can learn more about these API requests at link.

Minor changes

- **Added validation when uploading a video file in Telegram template.**
- **Profile history is now updated faster, ensuring correct operation of scenarios with complex logic.**

Fixes

- **Temporary platform files are now correctly cleared after the expiration of the lifespan period.**
- **Fixed incorrect display of growth report indicators.**
- **Fixed issues with the display of some elements in the interface.**
- **Fixed errors that caused the log of actions in the Administrative panel to record not all events.**
- **Improved stability of `procworkflow` and `proctask` processes.**

References

PL-17467 PL-17445 PL-17127 PL-17267 PL-17182 PL-17090 PL-16638 PL-16931 PL-17183 PL-17016 PL-1628

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated as a
Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

v2024.3.69

v2024.3.69.0 - Altcraft mSDK, Notify

Major changes

Mobile push SDK

Altcraft mSDK is a mobile programming interface (SDK) for integrating push notifications into mobile apps
Altcraft mSDK allows you to:

- **Automatically subscribe users to push notifications.**
- **Support sending notifications with text, images and action buttons.**
- **Update and refresh device tokens on the server for seamless notification delivery.**
- **Track notification delivery and user interaction with notifications.**
- **Automatically retry failed requests and work in the background.**

> TIP
> **Currently, the Altcraft mSDK is available upon request in the form of an archive.**

Migration of mobile push templates

The structure of mobile push templates has been updated in the following files:

- `json_consts/templates/AndroidFirebase.json`
- `json_consts/templates/IOSAPNS.json`
- `json_consts/templates/IOSFirebase.json`

If you are sending mobile push notifications through these providers and haven't replaced these files with
To do this, copy the contents of each original file into the corresponding `custom.` prefixed files:

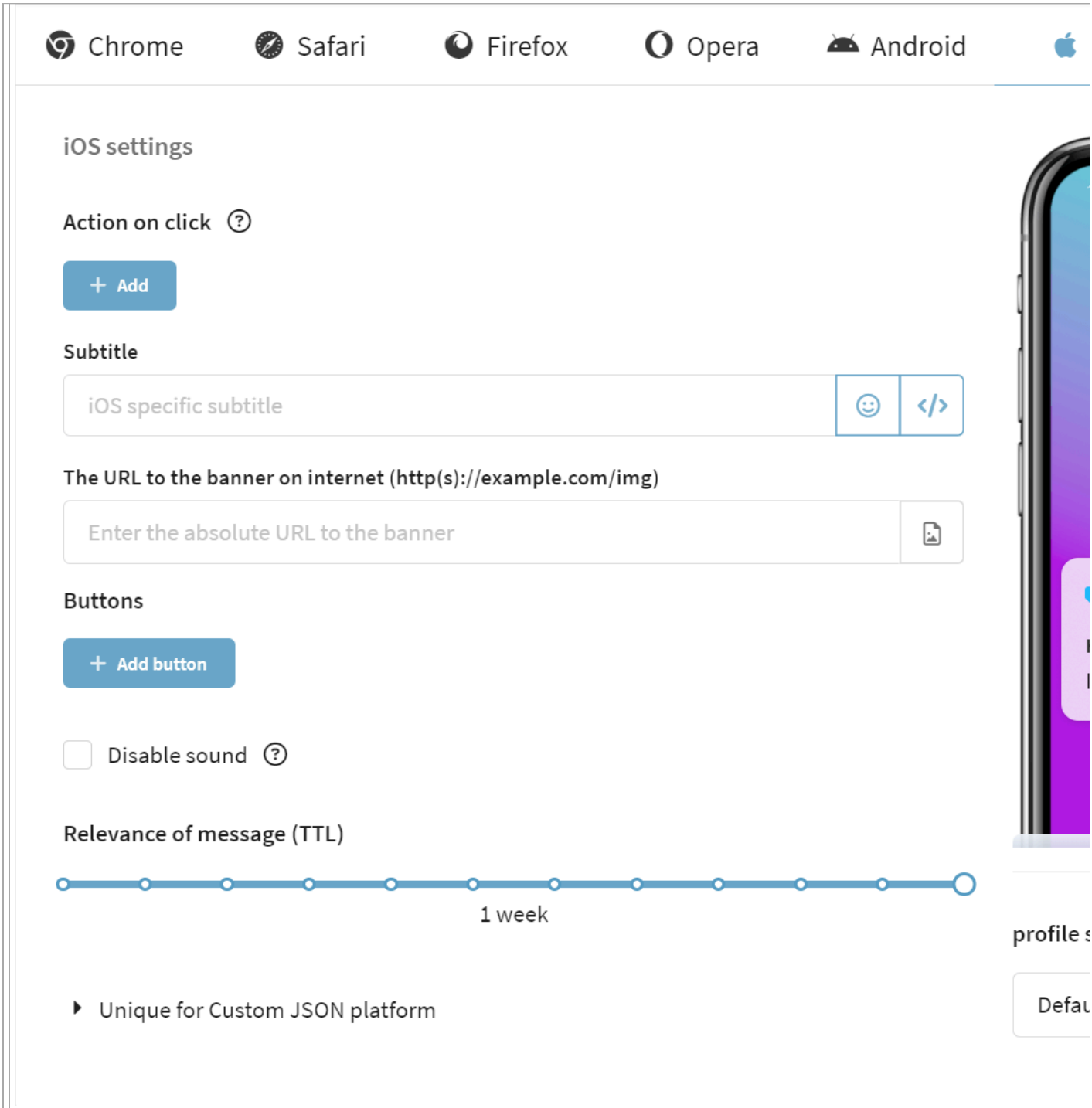
- `json_consts/templates/custom.AndroidFirebase.json`
- `json_consts/templates/custom.IOSAPNS.json`
- `json_consts/templates/custom.IOSFirebase.json`

Or copy them into files linked to your account:

- `json_consts/templates/custom.<account_number>.AndroidFirebase.json`
- `json_consts/templates/custom.<account_number>.IOSAPNS.json`
- `json_consts/templates/custom.<account_number>.IOSFirebase.json`

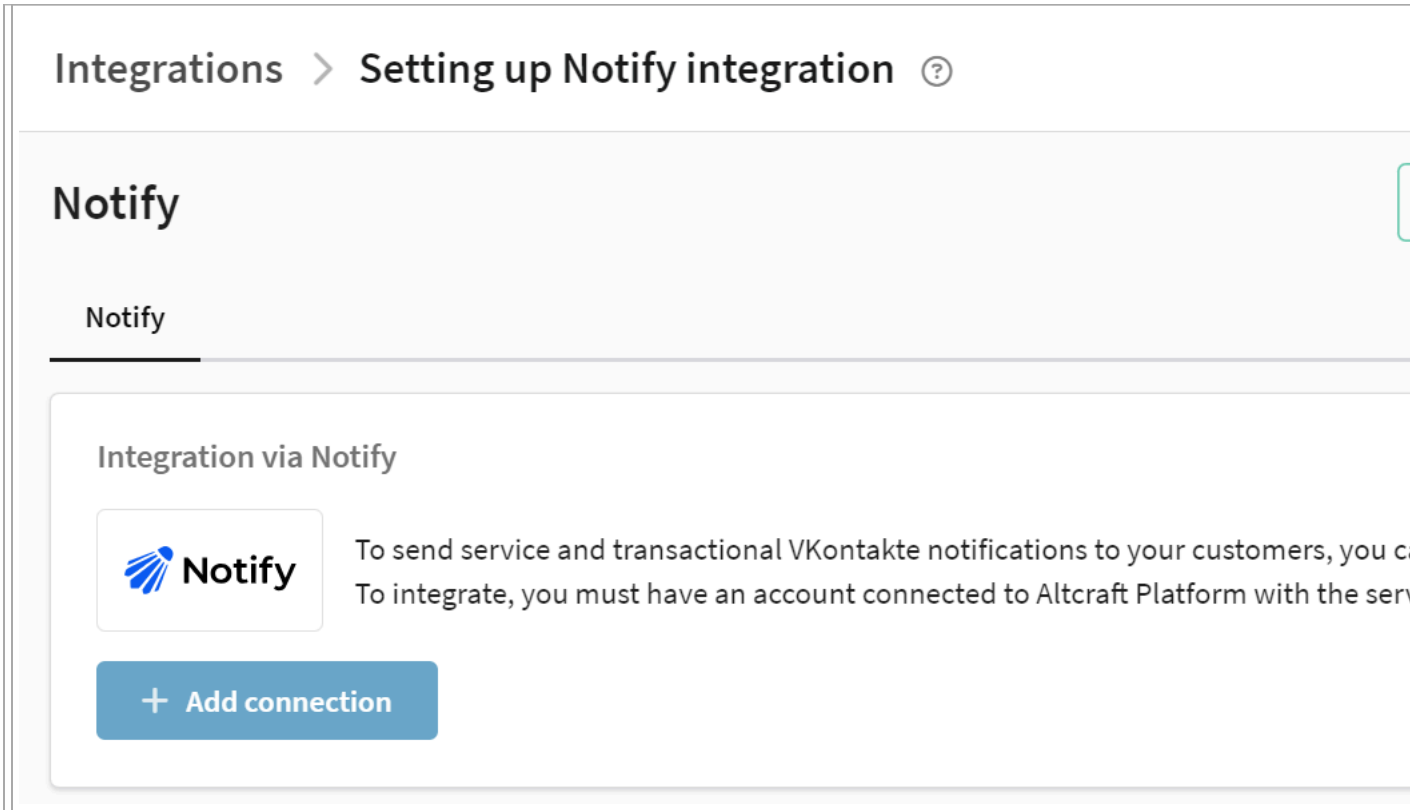
Mobile Push Editor update

Mobile Push Editor has been updated (editor β). Removed redundant customization fields, added new butt



Notify integration

The platform now features a direct integration with Notify, allowing you to send transactional and service



Minor changes

- **Added correct XML support to API 1.1 version of the platform. Learn more.**
- **Improved scenarios. Added possibility of scaling on the touchpad. Changed the system of setting link**
- **Template previews have been improved.**

Bug fixes

- **Fixed an issue causing campaigns to hang during the RMQ reconnection stage due to high memory co**
- **Export of profile history now executes faster.**
- **Changing the API token in the settings of the web push channel for Safari is now successfully saved.**
- **The analytics for undeliverables now correctly displays statistics for web push notifications.**
- **In the profile field of type array of objects, it is now possible to manually save an empty array by spec**
- **A warning has been added when saving a scenario if no conditions are selected for the "Scheduled se**
- **The HTTP(S) protocol is now displayed in the image URLs in the gallery.**
- **TypeScript has been updated for improved support and compatibility.**
- **The stability of the proctask and webcontrol processes has been improved.**
- **Minor interface issues in the platform have been fixed.**

References

PL-16878, PL-16717, PL-16767, PL-16799, PL-16860, PL-16798, PL-16843, PL-16020, PL-16831, PL-16814, P 16726, PL-15887, PL-15878, PL-15677, PL-16568, PL-16658, PL-16644, PL-15124, PL-16143, PL-12781, PL-1 13267, PL-15429, PL-15372, PL-15367, PL-15377, PL-15374, PL-14911

---

v2024.3.69.490 - Batch import API improvment

Major changes

Improved mass import of profiles via API

Optimized API method for mass import of profiles into the database (`/v1.1/profiles/import_batch`). It now tak

Minor changes

- **Updated integration with Google Ads due to the release of a new version of Google Ads API.**
- **Added validation for the "Limit" field in the Relations section.**
- **Added validation for the extension of the uploaded file in Telegram templates.**

Interface improvements

- **Fixed display of quotes in tags and groups.**
- **Corrected texts of many errors in platform logs.**
- **Fixed display of elements on the mobile version.**
- **Improved display of dropdown lists.**
- **Added a tooltip in the interface for the “Copy profiles from database” option.**
- **Improved layout in trigger activation conditions.**
- **Fixed a bug where the “Apply” and “Save” buttons were incorrectly displayed in different sections of t**

Fixes

- **Fixed a bug where some web push notifications were not sent to users.**
- **Request to Close method of MongoDB driver now work correctly.**
- **Fixed an error occurring when navigating between pages in profile activity history.**
- **Profile import requests now display the correct error code.**
- **Fixed a bug where it was not possible to use cyclic enumeration for Market entities and fields.**
- **SASL authorization mode in the Action Hooks setting now works correctly.**
- **Fixed a bug where it was impossible to save a resource for mobile push without an `appId` field.**
- **The sending mode selection for push notifications now displays correctly for all campaigns.**
- **Fixed a bug where some processes would not work when running the platform with SOCKETS enable**
- **Scenarios that have a Firebase API Legacy resource specified in the campaign are now saved correct**
- **Fixed errors occurring when saving funnels with tag filtering.**
- **Queues on the Sender are now removed correctly.**
- **Fixed an issue where part was missing in the html version while saving a template.**
- **Profiles participating in a static segment are now correctly included in a quick segment.**
- **Fixed a bug where the trigger for importing a profile via push did not work.**
- **Fixed an error when the “JSON-filter” scenario element did not work correctly.**
- **Improved stability of the “API Call” element in scripts.**
- **PostgreSQL connector now correctly connects to the database in Yandex Cloud.**
- **Fixed a bug where the “Email subscription: domain (not) in list” condition operator in scenarios searcl**
- **Quick Segment now correctly searches for profiles by specified conditions.**
- **Fixed a bug where a new item was created in an order instead of changing the data of the old one.**
- **Clearing temporary files in `data/segment` now works correctly.**
- **Fixed bugs related to WhatsApp integration.**
- **Fixed memory overload when a database query was added to the template.**
- **Template body is now not imaged when sending push notifications via IOS ANPs.**
- **Fixed errors occurring when sending push notifications with old structure.**
- **The option to send a push notification to multiple identical subscriptions now works correctly.**
- **Profiles with an incorrect phone number are now displayed correctly on export.**
- **Stability of the `akmtad` process has been improved.**

References

PL-16976 PL-16955 PL-16940 PL-16928 PL-16752 PL-16886 PL-16877 PL-15784 PL-15086 PL-16194 PL-166
PL-6070 PL-15583 PL-15438 PL-16925 PL-16916 PL-16919 PL-16909 PL-16207 PL-16747 PL-16784 PL-1680

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated as a
Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.

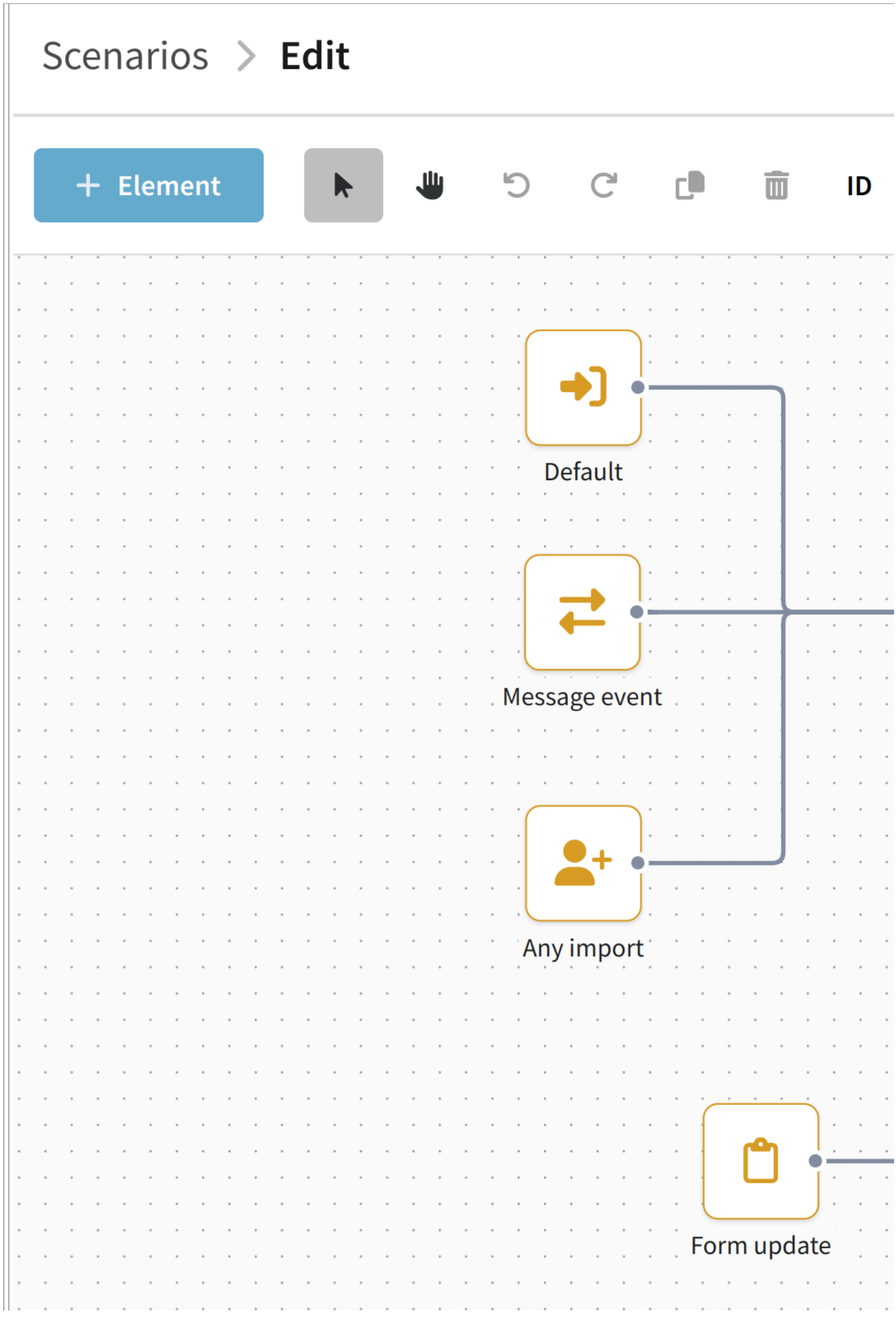
v2024.2.68.2

v2024.2.68.2.287 - Multi-capture scenario, Product feed, Pixel trigger improvements

Major changes

Combined profile capture into a scenario

Added the ability to configure capture profiles with multiple conditions within a single scenario. This capt


Scenarios
Edit
+ Element
ID
Default
Message event
Any import
Form update

Also statistics on captured profiles is available, including for each condition separately. This feature allow

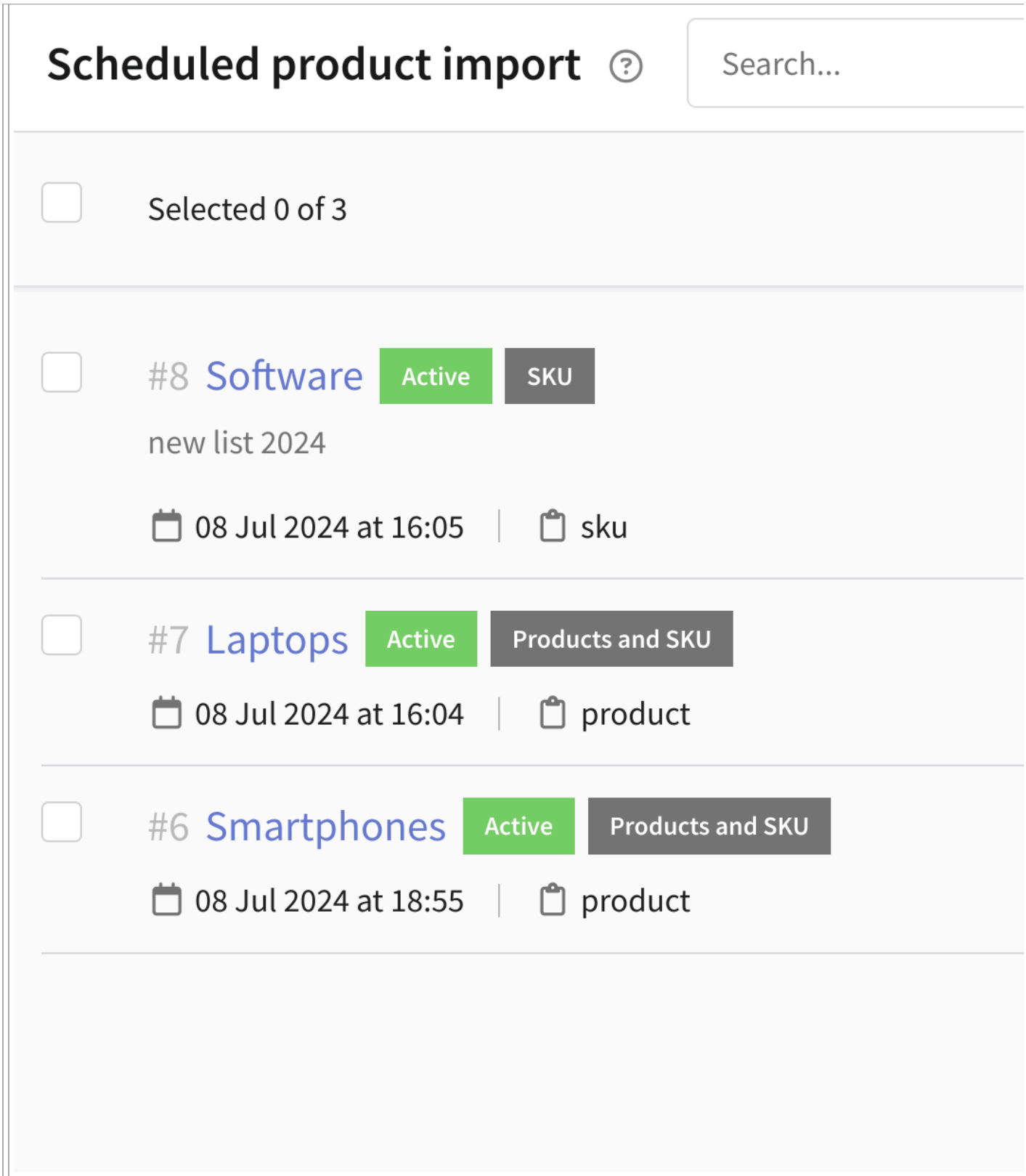
Importing a product feed by link in the Market

> CAUTION
>
> **After adding a product feed, you need to migrate for the Market module to work correctly. To do this, do t**
>
> 1. **Stop the `webcontrol` service.**
> 2. **Start `aktool` from new build with the command:**

```
./aktool migrate importfeed --config /opt/akd/config/main.json
```

> 3. **Start the `webcontrol` service.**

Now you can set up synchronization of products and SKUs in the Market with your product feed data. The feed [in this article](#).



Detailed import statistics are available for each synchronization:

#6 Smartphones Active

Latest launch completed

Total products

100

Updated 99

Added 1

Skipped 0

Import settings

General info

ururugu.yml

Not selected

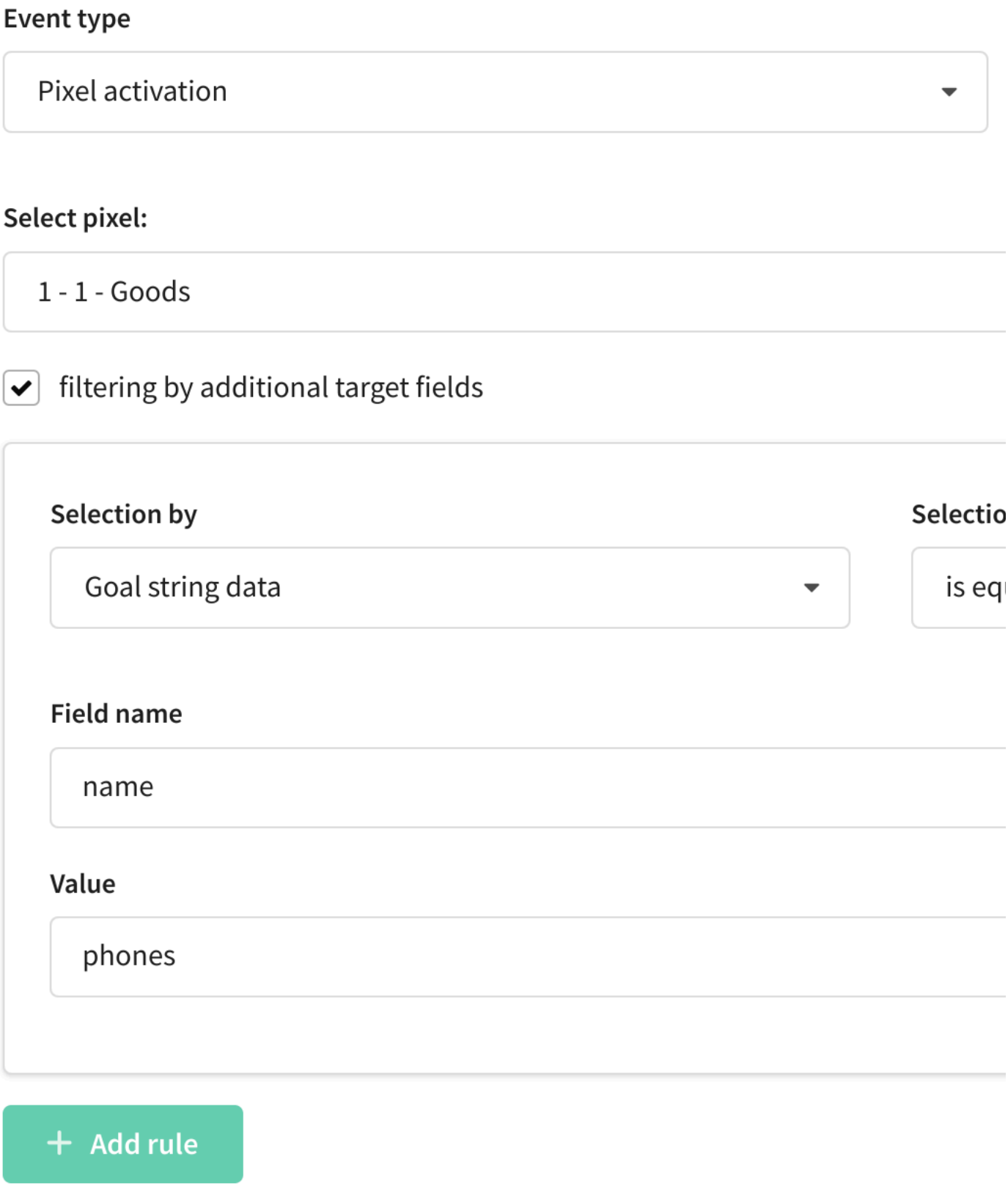
Imports history

Start	End	Products		
		Total	Added	Upd

07-06-2024 18:58	07-06-2024 18:58	100	1	9

Filtering by additional fields for pixel activation trigger

Added the ability to specify selection rules for additional fields when setting up a trigger by activating a p



Minor changes

- **New connection types have been added: Basic Authentication and Token Authentication.**
- **ClickHouse system driver package and connector have been updated.**

Interface improvements and fixes

- **Fixed a bug in the tooltip in the profile list.**
- **Scenario exit item ID is now displayed in the profile card.**
- **Fixed role management for master users.**
- **Pixel reference is now correctly displayed in the profile history.**
- **Market when importing products now displays the correct number of products.**
- **Forms now work correctly in preview mode.**
- **The image insertion panel in the template editor now displays the correct URL.**
- **Information about the creator and the date of the last change in integrations and connections is displ**
- **Outdated Firebase API has been removed from the resource settings for push notifications.**
- **List by metrics in popup analytics is now displayed correctly.**
- **Improved display of some data in the Analytics section.**
- **Improved localization.**
- **Fixed typos and incorrect links.**

Performance improvements

- **Fixed some issues in AKMTA.**
- **Redesigned mechanism of getting the name of global function `acpopup`.**
- **Fixed high memory consumption in `akmtad` and `procworkflow` processes.**
- **Fixed hang when sending email messages with some sending policies.**

Fixes

- **Fixed bug where the “Import via popup” trigger condition could not be selected.**
- **Profiles are now correctly included in the statistical segment by complex condition again.**
- **Fixed the error “Not calculating statistics with errors” when receiving a report in the Administrator P**
- **Traffic analytics with the “By popups” filter is now displayed correctly.**
- **Sorting by age in Databases now works correctly.**
- **Fixed a bug that occurred when adding 18 or more characters to an additional field with the data type**
- **Tracking now does not pause when the encryption key is too long.**
- **Link addition events are now correctly displayed in the profile history.**
- **Additional parameters are now correctly added to `custom_data` in Action hooks.**
- **Various segmentation issues have been fixed.**
- **Statistics in scenarios now work correctly.**
- **Added handling of query parameters when loading a script to control popups.**
- **Fixed an issue where editing campaigns with send policies was only available to master users.**
- **Added validation to save templates for sending via WhatsApp*.**
- **Exporting profile history by pooches now works correctly.**
- **Fixed bugs in the Market module.**
- **Added validation for deleting external databases.**
- **Fixed a bug where it was impossible to create a segment via API.**
- **Sending policies in the trigger campaign work correctly again.**
- **Improved the operation of Observer Databases when sending a campaign.**
- **Fixed a bug that occurs when copying a deleted order or creating an order with the same name as a d**
- **Added ttl for links in SMS messages.**
- **Fixed bug where LDAP users change authorization type after saving.**
- **2FA connection for LDAP-user works correctly again.**
- **An empty string can now be added as a delimiter in the `format` function in templates.**
- **Save and Apply buttons are no longer available for frozen accounts.**
- **Fixed a bug where tab settings were reset while changing mandatory fields in profiles.**
- **Connection to Devino Telecom using login and password works correctly.**
- **Fixed some product search errors.**
- **Pictures with names less than 12 characters are now loaded correctly when importing HTML templat**
- **Fixed errors occurring when updating a trigger campaign.**
- **Stop list now skips domains that contain more than 6 characters after a period.**
- **Fixed panic when trying to save a deleted object in segments.**
- **Fixed panic when adding a template using API.**
- **Fixed panics in `cookie_saver`, `procintegras`, `xls.go`, `proctask` and `tariffcontroller`.**

References

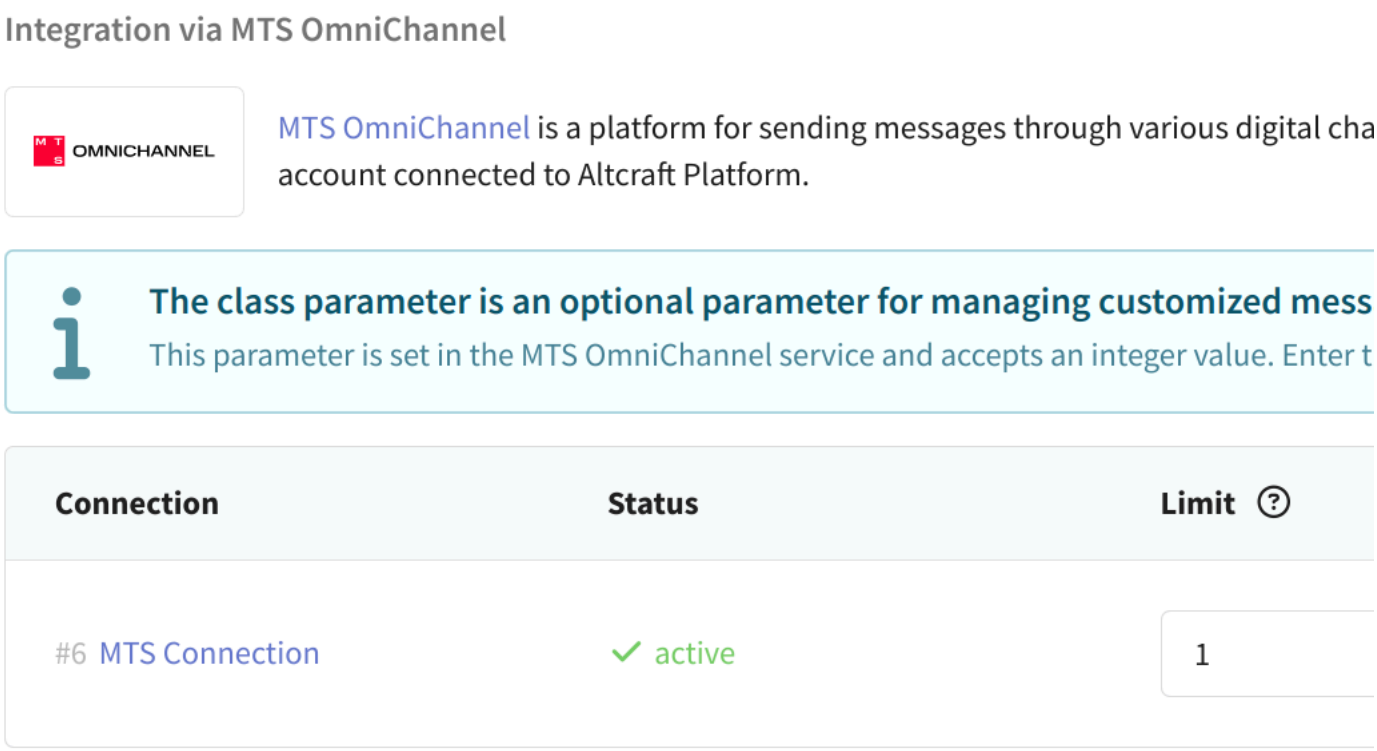
PL-16064 PL-16026 PL-15957 PL-15900 PL-15899 PL-15850 PL-15809 PL-15421 PL-15414 PL-15402 PL-153

v2024.2.68.2.749 - Viber integration updates

Major changes

Improvement of Viber channel integration through MTS OmniChannel

In the channel integration settings using MTS OmniChannel gateway, the `class` parameter has been added



Fixes

- **Adjusted texts and links in the platform interface, including in the mobile version.**
- **Improved html parsing when importing a message template.**
- **Fixed an error when it was impossible to delete email from a profile.**
- **Improved platform stability.**
- **Fixed Java Script errors in the “Segments” and “Integrations” sections.**
- **Fixed an error that occurred when opening the default template preview.**
- **Test form submission now correctly saves the result of filling out the form.**
- **Fixed a bug where MongoDB was generating multiple queries when saving scenarios.**
- **Deleting profiles by segment is now faster.**
- **Fixed errors that occurred when creating a resource.**
- **Filter by connection status in the connection list now works correctly.**
- **Fixed status of new connections to Yandex.Audience.**
- **AKPush now does not create duplicate profiles when clearing storage and cookies.**
- **Fixed errors when adding groups to the Database in bulk.**
- **Template search in the “Click Map” report now works correctly.**
- **Custom unsubscribe link is now displayed in the “Click Map” report.**
- **Fixed an issue where it was not possible to remove region and category in the Products section.**
- **AKMTA queues in RabbitMQ are now cleared after deleting a Sender in the Administrative panel.**
- **The message counter in the AKTMA queue now displays the correct values.**
- **Fixed some sending errors in AKMTA.**
- **Regular expressions in profile structure segmentation queries now work correctly.**
- **MongoDB driver has been updated to the current version.**
- **API query summary report now returns actual parameters.**
- **Improved control of privileges and user rights for some actions (e.g. exporting tasks).**
- **Fixed errors with HTTPS api in some libraries.**
- **Channel unsubscription is now correctly displayed in the profile’s action history when it enters the st**

- **Fixed a bug where a goal not registered before could not be specified in trigger campaigns.**
- **Forms now correctly save data after forced emailing.**
- **Fixed panics in akmtad.**

References

PL-15737 PL-16402 PL-16405 PL-16407 PL-16174 PL-8775 PL-16439 PL-16467 PL-16461 PL-16058 PL-15091 15466 PL-16395 PL-16187 PL-16409 PL-14836 PL-13393 PL-16269 PL-16220 PL-16208 PL-16110 PL-3435

---

v2024.2.68.2.950 - Improved bot filtering mechanism

Major changes

Bot filtration improvment

The mechanism for filtering bot actions in the platform has been improved. The filtering settings are conta

Minor changes

- **It is now possible to customize in the platoform configuration the timeout for connecting to the databa**
- **Improved performance of some platform processes.**

Fixes

- **Fixed an issue where the “API Request” node passed incorrect request parameter data to the platform**

> **CAUTION**
>
> **To fix this error, the user needs to migrate. To do this, do the following before launching the new version:**
>
> 1. **Stop `webcontrol` and `procworkflow` services**
> 2. **Start `aktool` from new build with the command:**

```
./aktool workflow api_nodes_convert --config /root/MarketingPlatform/config/main.json
```

> 3. **Start `webcontrol` and `procworkflow` services**
>
> **Help for the migrator:**

```
Usage:
  aktool workflow api_nodes_convert --accs=1,2... --wfs=1,2... [flags]

Examples:
./aktool workflow api_nodes_convert --accs=1,2 --wfs=1,2

Flags:
  -a, --accs ints   Account IDs (comma-separated list, empty means all)
  -h, --help        help for api_nodes_convert
  -w, --wfs ints    Workflow IDs (comma-separated list, empty means all)

Global Flags:
      --config string                    config path (default "/opt/akd/config/main.json")
  -o, --output string                    file path to output in the result of the command (default "stdout")
```

- **Fixed a bug where a space character was incorrectly encoded by the platform in the “API Request no**
- **AMP-version of the email template is now formed correctly.**
- **Fixed a bug where the scenario would not include profiles by activating pixels.**
- **Fixed a bug where the `procpush` and `tariffcontroller` processes would not terminate without a forced s**
- **Now data about previous popup form fills is not deleted after renaming fields of this form.**
- **Fixed errors related to incorrect deletion of entities from profiles.**
- **External variables of email templates are now correctly converted in the platform when importing ten**
- **Fixed a bug where location detection when importing profiles did not work correctly in some cases.**
- **Improved display of some interface elements in the mobile version.**
- **Corrected description of some parameters in the platform.**
- **Fixed some bugs in the platform interface.**
- **Fixed panic in `tzdata`.**

References

PL-16513 PL-16476 PL-16441 PL-15870 PL-16283 PL-16134 PL-16351 PL-16499 PL-16490 PL-15447 PL-1532

---

v2024.2.68.2.1131 - SSL support for MongoDB

Minor changes

- **Added SSL support for connecting to MongoDB.**
- **Added “Domain not listed” segmentation condition for the “Email subscription” field.**

Interface improvements and fixes

- **Profile card now shows the id of the element from which the profile has exited the scenario.**
- **Improved design of the form editor interface in the platform.**
- **Fixed errors in the text of some tooltips.**

Fixes

- **Fixed a bug where profiles were not included in scenarios with a scheduled segment under the “Dynar**
- **Information about occurring errors is now correctly displayed above scenario elements.**
- **Fixed a bug where profiles were not captured in scenarios configured to capture when a pixel is activ**
- **Fixed a bug in the `products` query that occurred when passing an identifier containing single quotes.**
- **It is now possible to pass an api variable for a field with type “Number” in the “Update Profile” script e**
- **Fixed “invalid syntax” error in GET-request to get action history of several profiles.**
- **Fixed some vulnerabilities in the interface**
- **Now when saving a profile with invalid data in one of the fields, the platform will display an error and**
- **Fixed a bug where a tag with a comma in its name was defined as two different tags.**
- **Updated some libraries to avoid potential vulnerabilities.**
- **Fixed panics in `proctask`, `webcontrol` and `procworkflow`.**

References

PL-16567 PL-16563 PL-16513 PL-16557 PL-16276 PL-14369 PL-16535 PL-16526 PL-16105 PL-16544 PL-1653

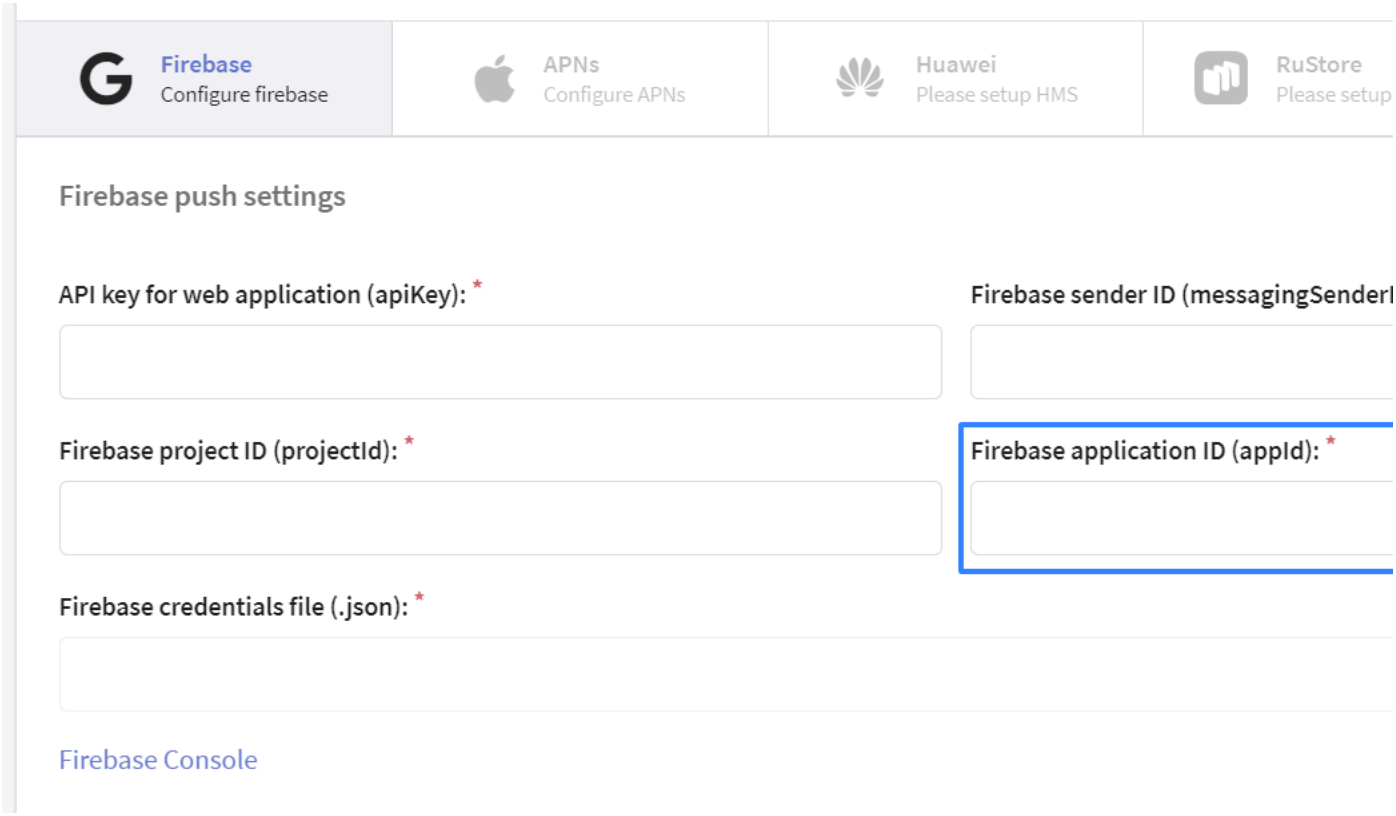
---

v2024.2.68.2.1378 - Firebase API updates

Restore the work of web push-notifications

Google Firebase service has changed some API handlers, in this connection for normal work of web push-

- **In all resources, scripts of which are hosted on sites, it is necessary to fill in the Application ID field i**



The Application ID value can be found in the project settings in the Firebase console.

- **If static ak-push.js and service-worker.js are placed on the sites, then after editing and saving the res**

Minor changes

- **Images in the template editor are now inserted with `https` at the beginning of the link, rather than `ht`**
- **Links between a sceanario start element and other elements are now added correctly.**
- **Fixed a bug where it was not possible to generate a Promo Code without a quantity limit.**
- **The maximum number of characters for a first-level domain in the platform has been increased to 128**

- **Fixed errors that occurred when interacting with an empty profile database.**
- **Fixed an error when data was not sent to email after a user fills out a form.**
- **It is now possible to specify a tracking domain with a single-letter subdomain in the resource settings**
- **Fixed a bug where scripts with scheduled segment did not work correctly after the first run.**
- **Faster page loading in the user interface when using LDAP authorization.**
- **Fixed a bug where changes made to the Funnel were not saved in the platform.**
- **Improved parameter validation in some API requests.**
- **Drop-down lists work correctly in the latest versions of Google Chrome.**
- **Traffic analytics report is now uploaded correctly.**
- **Fixed interface errors in some sections of the platform.**
- **The descriptions of some errors in the platform have been corrected.**

References

PL-16623 PL-16613 PL-16148 PL-16599 PL-16604 PL-7192 PL-16555 PL-16571 PL-16334 PL-16511 PL-1614

---

v2024.2.68.2.1509 - Firebase push fixes

Major changes

- **Fixed errors occurring when sending push notifications using Google Firebase.**

Minor changes

- **Added option to disable deleting link data from ClickHouse.**
- **Now the value of “Lock double trigger activation” in pixel settings cannot be lower than one.**

Interface improvements

- **Fixed display of filters on mobile version.**
- **Improved appearance of “Upload Photo” button in profiles.**
- **Improved appearance of the line break button in the template editor.**
- **Fixed layout errors in some Market elements.**

Fixes

- **Fixed a bug where links in push notifications did not work.**
- **Database IDs are now displayed correctly when launching test campaigns.**
- **Fixed some vulnerabilities in the platform.**
- **The “Capture profiles when changing an order” scenario element now works correctly.**
- **Fixed a bug where uploading 0-day history would result in uploading 1-day history.**
- **Market variables now work correctly in the push and Viber template editor.**
- **Fixed an error occurring when trying to send rich push notifications to profiles that don’t have fields u**
- **The `campaigns/triggers/import_and_start/` method now works with the unconditional send flag.**
- **Fixed panic in webcontrol.**

References

PL-16637 PL-16634 PL-15610 PL-16179 PL-16146 PL-16625 PL-16620 PL-16150 PL-16619 PL-14996 PL-1646

---

v2024.2.68.2.1788 - Order status API

Major changes

- **Added API method to check order status in the Market. Read more.**

Minor changes

Interface improvements

- **Fixed layout errors in “Forms” section on mobile devices.**
- **Drop-down lists in scenario modal windows close correctly when focus is reset.**
- **Fixed scroll sliders not working on some platform pages.**

Fixes

- **Fixed errors related to reaching the maximum limit of active sessions in MongoDB.**

- **Profile exit record from a scenario is no longer overwritten when it is cleared of all profiles.**
- **Fixed errors occurring when running dynamic segment with selection condition on a custom field.**
- **Fixed duplication of web push notifications in browsers.**
- **Relations are now correctly overwritten when editing from profile.**
- **Fixed bug where user creation time was reset when updating its data.**
- **Now when cloning a scenario, error messages are not copied with it.**
- **Fixed a bug where an unavailable resource was displayed in the list without being labeled accordingly**
- **Custom `custom_json` in push notification templates now works correctly.**
- **Fixed a bug where the last user who edited a script was incorrectly displayed.**
- **Unsubscription event when notifications are forbidden via Safari is now correctly logged by the platfo**
- **Fixed a bug where a remote ISP was available to a node in the Administrative Panel.**
- **Improved stability in `webcontrol`, `akmta` processes.**

References

PL-16629 PL-16658 PL-16650 PL-16646 PL-16593 PL-16425 PL-16446 PL-16566 PL-15758 PL-16056 PL-1488

---

v2024.2.68.2.1957 - Web version compression mode

Major changes

- **Added option to disable compression when creating a web version storage policy in the Administrativ**

- **Campaign log now shows the time a segment was received.**

Minor changes

Interface improvments

- **Added display of the time of the next launch of the Multivariate testing campaign.**
- **Fixed a bug where tooltip pop-ups did not disappear after some time.**

Fixes

- **Fixed a bug where the platform did not display the campaign queue on senders.**
- **Fixed a bug where the platform did not take summer and winter time into account for time zones in so**
- **The campaign log works correctly again when opened from the campaign editor.**
- **Fixed various bugs related to updating a profile in the platform.**
- **Profiles with Apple Safari Native push notification subscriptions are now saved correctly.**

- **Fixed a bug where you could create a campaign with a draft template.**
- **Web versions of emails are correctly compressed before saving.**
- **Importing to Market using a file now works correctly.**
- **Fixed a bug where the result cache in the preview of template/segmentation SQL queries was not bei**

References

PL-16095 PL-16671 PL-16655 PL-16281 PL-10310 PL-16553 PL-16609 PL-11837 PL-16132 PL-15971 PL-157

---

v2024.2.68.2.2206 - Sendsay, Fixes

Major changes

Sendsay integration

We've added the ability to send emails through the Sendsay SMTP relay. The platform administrator needs

Minor changes

- **Expanded API requests for retrieving profile history: `/api/v1.1/subscribers/history_get` and `/api/v1.1/subscr`**
- **Removed the `tariffcontroller` and `procnotify` processes; their functionality was migrated to `proctask`.**
- **Improved import of email templates uploaded from third-party services into the platform. Images can include the files `text.txt`, `subject.txt`, and `template.json`, as their presence may cause import errors. A**
- **The "Web imports" metric in the Audience growth report now includes the total number of profiles ad**

> CAUTION
>
> **Previously, the "Web imports" metric only included data from form imports. To update the metric to inclu**
> **Migration is performed after the platform is updated to the new version and does not require stopping an**
> **1. Drop the `engagement_report` view and table:**
>
> ```
> ./aktool stats delete engagement --config=/opt/akd/config/main.json
> ```
>
> **2. Initialize the table in the new format:**
>
> ```
> ./aktool stats init engagement --config=/opt/akd/config/main.json
> ```
>
> **3. Recalculate the records in the table:**
>
> ```
> ./aktool stats fill engagement --config=/opt/akd/config/main.json
> ```
>
> **If successful, the console will show how long it took to fill the table. Example: `complete fill stats table for`**

Bug fixes

- **Fixed the issue with sending web push notifications to the Safari browser via APNS.**
- **Restored the ability to select a pool in sender settings.**
- **Fixed a problem with adding extra parameters to `custom_data` in hooks.**
- **Resolved the issue that caused hooks to not retry when the `_createtime` field was added to `custom_data`.**
- **Segment filtering by "List" type fields using the "Equals" condition now works correctly**
- **The counter above the SMS campaign node in the scenario now correctly displays the number of profi**
- **Resolved the issue of duplicated weekly and daily reports for platform users.**
- **The campaign log now displays the correct time for receiving a segment during regular campaign test**
- **A trigger campaign set up for a custom popup event can now be successfully saved.**
- **The next_from_id parameter in the `/api/v1.1/subscribers/history_get_al` l response works correctly.**
- **Missing categories field in the /api/v1.1/market/sku/import request no longer causes errors.**
- **The `ampcarousel` component is now connected in the demo AMP template version.**
- **Improved validation in scenarios and subscriptions for Viber and WhatsApp* channels.**
- **Stability improvements for the `procactions` process.**
- **Minor UI fixes in the platform.**

References

PL-15845 PL-16842 PL-15693 PL-16694 PL-16796 PL-16763 PL-16805 PL-15149 PL-16743 PL-16737 PL-163
PL-16533 PL-16644 PL-16676 PL-14390 PL-16476 PL-16259 PL-16608 PL-15941

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated as a
****Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.***

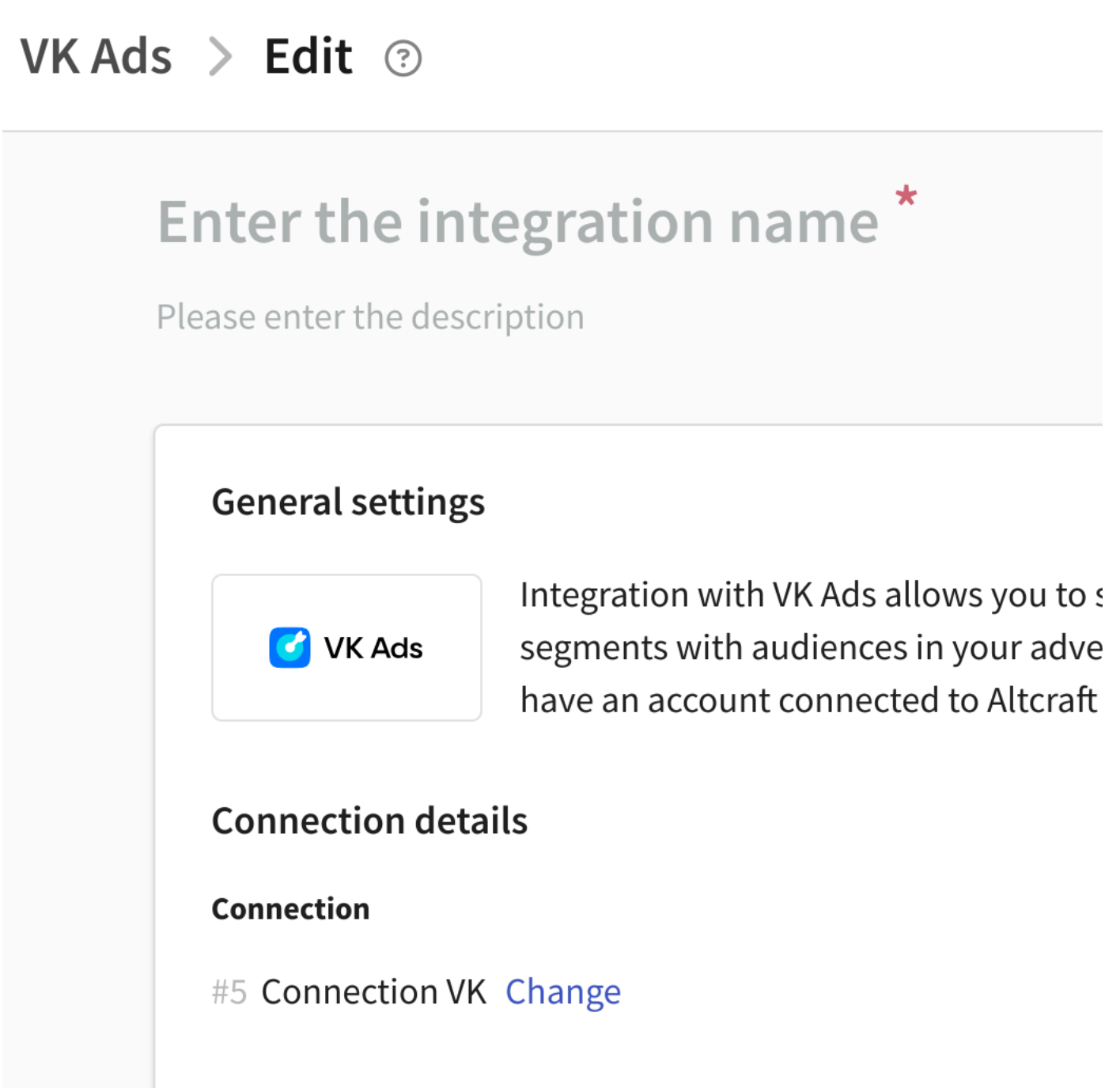
v2024.1.68

v2024.1.68.0 - VK Ads™, new AKMTA sender settings

Major changes

VK Ads™

	Added integration with **VK Ads™**. The new f



Updated connections menu in the dashboard

Removed Viber and WhatsApp* menu items, added separate buttons for creating connections using edna,

Connections

Facebook Ads

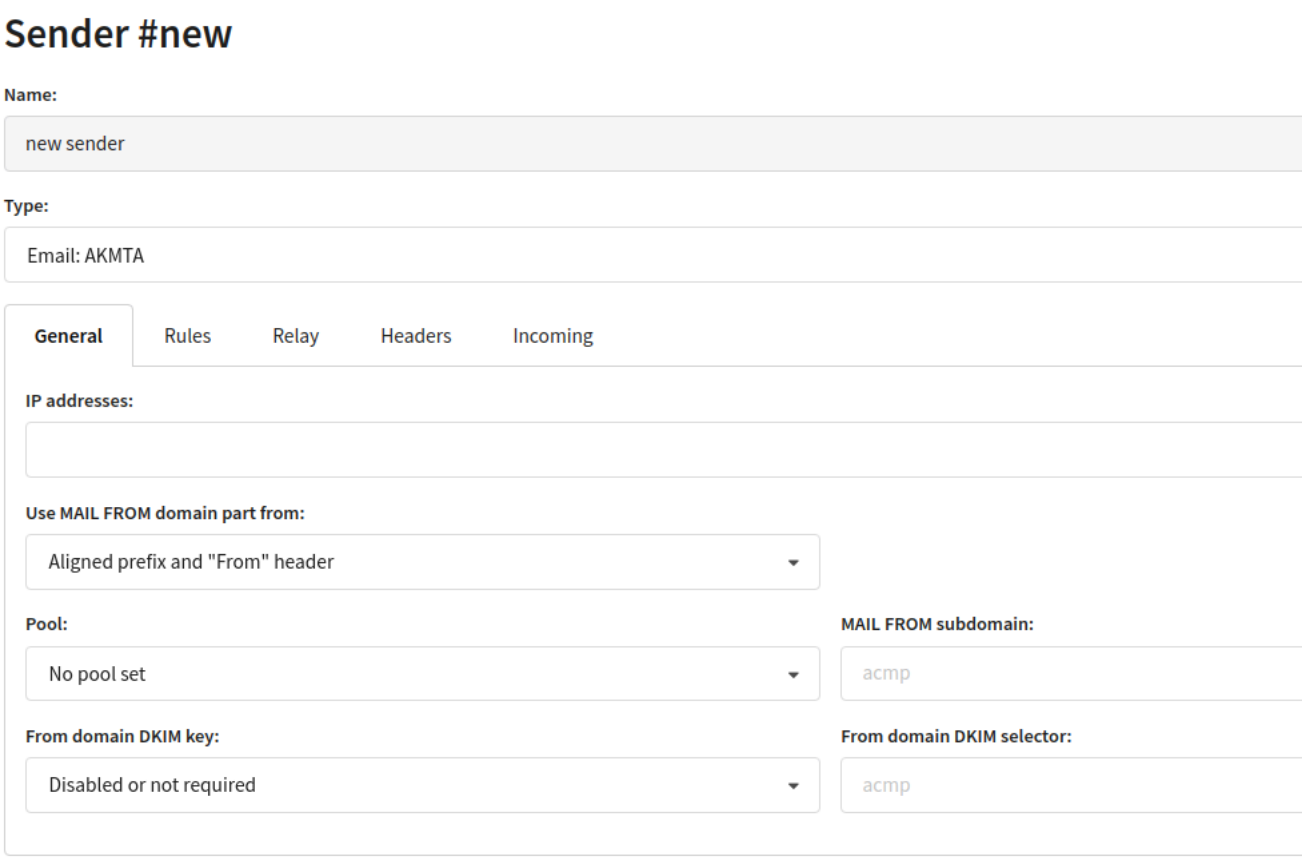
Connections: 0

Edna

Connections: 0

Updates of the administrative panel

- **When creating an AKMTA sender, a new type was added for the MAIL FROM domain part - Aligned pref**
- **Added pools. A pool is a set of resources needed to send emails from a specific domain. Detailed instr**
- **AKMTA settings in the “Senders Management” section are divided into General, Rules, Relay, Headers**



Minor changes

- **Improved search in the user list: added search by email and phone number, fixed various errors assoc**
- **The user’s last name is no longer visible to support specialists in Jivo.**
- **Removed the limit on the number of displayed columns in data tables.**
- **The ability to view and copy a column value has been added in cases where it does not fit completely**

Performance improvements

- **Optimized API request for bulk import and launch of profiles into the script**
- **Fixed extremely high memory consumption for akmtad**
- **Fixed freezes in the visual editor when clicking on elements**

Interface updates and fixes

- **Fixed display of unavailable actions for frozen accounts.**
- **Updated interface in the “Pixels” and “Click map” modules.**
- **The “% Opens” column is now available again in channel reports.**
- **Fixed errors with displaying the menu of additional fields.**
- **Now the connection creation/update time is displayed correctly.**
- **The scrollbar in pop-up menus is now displayed correctly.**
- **The audit log report is now uploaded correctly.**
- **Some names and translations have been corrected.**

Corrections

- **The non-delivery report now correctly displays the reason of failure.**
- **Fixed a bug where variables were not inserted into the pop-up editor field.**
- **When mass profiles updating, push subscriptions are added only to the specified provider.**
- **Fixed various errors in the “Loyalty” section.**

- **Fixed a situation where it was impossible to add text to an empty “Text” element in the visual editor.**
- **Fixed errors that occurred when deleting a database and associated segments.**
- **Improved logging.**
- **Saving a target in trigger campaigns now works correctly.**
- **Special characters are now correctly escaped in LDAP.**
- **Fixed a bug where sending more than 10 SMS on the MTS gateway did not work.**
- **Errors related to incorrect filling of date and time in various elements have been fixed.**
- **Fixed errors related to analytics, such as false clicks from bots, counting hardbounce responses, and**
- **An error message about the number of characters in the SMS has been exceeded has been added to t**
- **Now you can select the status “Unconfirmed” in trigger campaigns with the event type “When adding/**
- **Fixed repeated “context canceled” errors in procworkflow logs.**
- **Entity deletion is now correctly recorded as an update.**
- **Fixed panics in procintegras, webcontrol, proctask and regen_preview_msgs.**
- **Fixed panics when publishing a pop-up and when setting up 360dialog.**

References

PL-15764 PL-15754 PL-15741 PL-15735 PL-15731 PL-15730 PL-15727 PL-15725 PL-15711 PL-15703 PL-1569 PL-15286 PL-15016 PL-14793 PL-14439 PL-14291 PL-13485 PL-11892 PL-15573 PL-15572 PL-15463 PL-1538 PL-11972

---

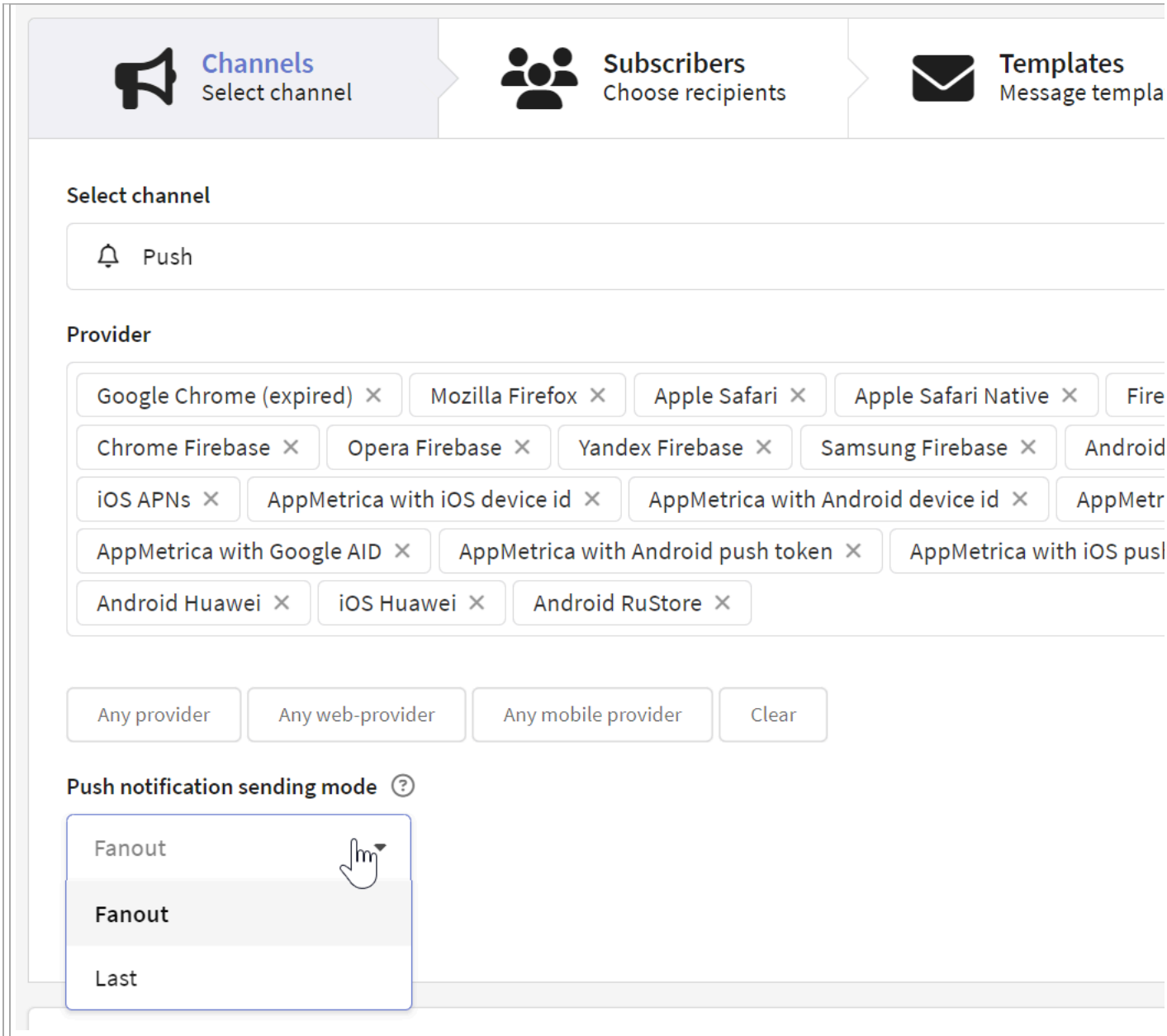
v2024.1.68.626 - New push sending mode, Fixes

Major changes

New mode for sending push notifications

When setting up campaigns using the push channel, you can now choose the sending mode for push mess

- **fanout (default) — osends messages to all subscriptions of profiles that match the campaign’s filters.**
- **last — sends the message to the subscription with the most recent creation date.**



Changes in the statistics table

Since the summary report was returning an incorrect number of events, the structure of the `channel_report`

```
./aktool stats delete channel_report
./aktool stats init channel_report
./aktool stats fill channel_report
```

> **CAUTION**
>
> **You must stop the `procevent` service before starting the migration.**

Minor changes

- **Improved validation of links in push templates. An error “Number of invalid links in the message” is nc**
- **In the campaign log, the entry “Number of processed subscribers” has been renamed to “Number of f**
- **Added the ability to save master users without specifying roles.**
- **Logging for relay sending has been added to syslog.**
- **Improved validation of fields in the Telegram Bot channel template.**
- **Updated the logic of the `./aktool stats` command, used for working with the report storage. Here’s hov**

Bug fixes

- **Fixed the operation of the segment by goals with a complex condition.**
- **In the scenarios, JSON filter now supports the OR operator and arithmetic operations.**
- **In the campaign settings, when deleting a segment used for sending, the correct segment is now disp**

- **Fixed bugs in searching in resources, roles, and the market.**
- **In test SMS messages, the date is inserted in the specified format.**
- **In the quick segment, when selecting profiles by phone number, spaces between digits do not affect t**
- **Images are now correctly displayed in both desktop and mobile versions in the visual image editor.**
- **The footer of the email with the unsubscribe link is displayed in the visual editor.**
- **Links in push templates after adding UTM tags redirect the user to the right page.**
- **It is now impossible to save an email template with an attachment larger than 12 MB.**
- **In forms, the option “Hide submit button” now works correctly and does not prevent form submission**
- **Export of the profile history of a static segment created during import now exports data without erro**
- **In scenarios, the “Condition” element with the “Wait for condition for a specified time” option enabled**
- **For one push message, the same `send_message_id` is now returned for all event types in event capture.**
- **Fixed an error when deleting a profile via the API.**
- **Fixed a situation where a push subscription could be created via API without specifying `subscription_id`**
- **Fixed the 2FA mechanism for administrators.**
- **Exporting audit logs in the admin panel no longer causes errors.**
- **Changes in sender settings with the type Rapporto SMS are now successfully saved.**
- **Fixed a situation where AKMTA logs contained duplicate rejection responses.**
- **Performance of some platform processes has been optimized.**
- **Some interface flaws have been fixed.**

References

PL-15986 PL-15983 PL-15964 PL-15959 PL-15952 PL-15886 PL-15876 PL-15849 PL-15828 PL-15823 PL-1582 PL-10097 PL-9688 PL-7740 PL-15918 PL-15917 PL-15914 PL-15912 PL-15911 PL-15905 PL-15903 PL-15898 15745 PL-15744 PL-15740 PL-15738 PL-15732 PL-15697 PL-15689 PL-15676 PL-15669 PL-15642 PL-15632 P

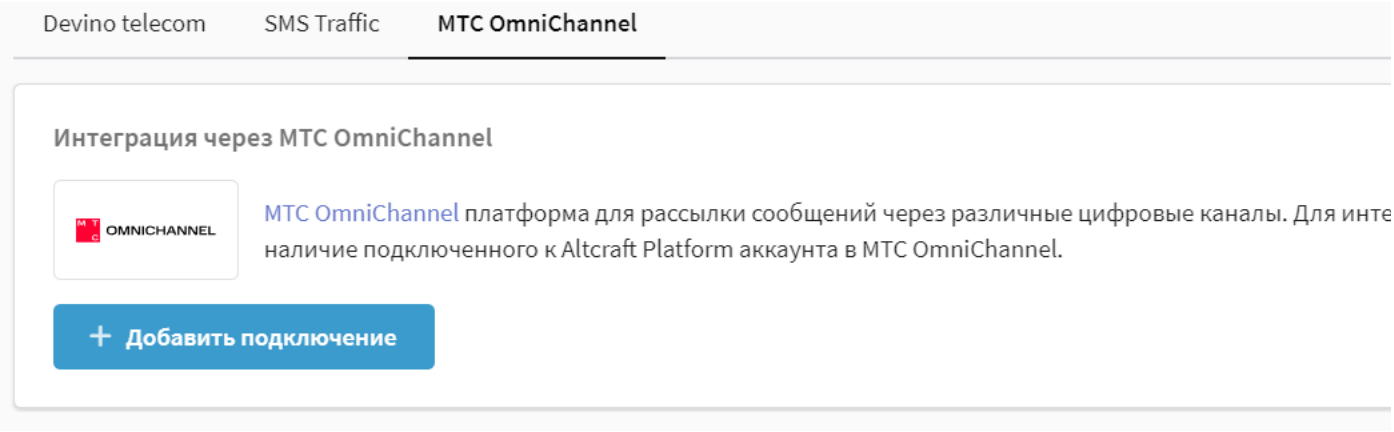
---

v2024.1.68.1354 - MTS OmniChannel integration, bug fixes

Major changes

Integration with MTS OmniChannel

Added support for integration with MTS OmniChannel. Now you can use this platform as a gateway to sen



Before setting up the integration, you need to connect your MTS OmniChannel account with Altcraft Platf

Other changes

- **The “Pixels” section has been moved to the “Web Layer” menu item.**
- **Updated geoip data for April 2024.**
- **Improved platform performance.**
- **Added option to remove validation for hidden fields in forms.**

Fixes

- **Fixed a bug related to duplicate tracking links.**
- **Scenario items with a condition are now saved with the correct id.**
- **Frozen accounts now have view permission restriction again.**
- **Fixed an issue where form validation errors were displayed in error level logs.**
- **Checking allowed domains for redirects when opening a pixel now works correctly.**
- **Improved traffic report performance when using grouping by pixel targets.**

- **Fixed incorrect platform behavior during deleting profile databases when the user role does not have**
- **Improved operation of critical push mode with APNS.**
- **Validation when using segments with link data collection and manual profile creation now works corr**
- **Platform login history now displays correct data.**
- **Fixed errors in logs.**
- **Popups work better on mobile display.**
- **Integrations in the platform are now correctly saved.**
- **Fixed a bug related to creating Facebook* connections.**
- **Fixed some analytics and reporting bugs.**
- **Correct mx is now used for sending to Yahoo mailboxes.**
- **Fixed bugs in Yandex.Appmetrica integration.**
- **Form opening and filling events in profile history are now displayed correctly.**
- **Fixed validation of fields in forms.**
- **Fixed errors when previewing multi-page forms.**
- `click_url` **in the web version of trigger push emails is now saved correctly.**
- **Fixed** `runtime error` **panic when sending a campaign with hidden copy.**
- **Fixed** `form_tst.go` **panic.**

References

PL-15888 PL-12880 PL-15993 PL-15991 PL-15865 PL-14353 PL-15869 PL-15926 PL-15840 PL-15979 PL-1045 PL-14015 PL-15104 PL-15989 PL-15712 PL-15693 PL-15700

Meta Platforms Inc., the company behind Instagram, Facebook, and WhatsApp, has been designated as a
****Viber is a trademark of Rakuten Group, Inc. and is blocked in the Russian Federation.***

API / Developer Guide: Contents

Contents of this section.

API interaction

Matching

Importing the API collection in Postman

List of API endpoints

Import profile

Update profile

Import multiple profiles

Update multiple profiles

Add multiple profiles

Add profile to database

Quick profile import

Import profile to RabbitMQ

Get profile data

Uploading profiles to a file

Suspend all database profiles

Unsuspend all profiles in a database

Delete profile

Subscription fields functional update

Database fields functional update

Merging multiple profiles

Unsubscribe profile from resource

Profile splitting

Get data for multiple profiles

Add or edit subscription

Get all profile subscriptions

Get all subscriptions from multiple profiles

Get profile subscription

Delete profile subscription

Restore deleted subscription

Suspend all subscriptions

Unsuspend all suspended subscriptions

Get profile action history

Get multiple profiles action history

Profile relations

Attach relation

Detach relation

Modify relation properties

Overwrite relation properties

Get profile relations info

Get profile relations info

Get database statistics

Update statistics on database

Get database list
Get database information
Get database fields
Database wipe
Get resource statistics
Update statistics on resource
Get resources list
Get resource information
Get resource subscription fields
Get statistics on resource
Update statistics on segment
Add or remove profile
Get profile data in a static segment
Create segment
Update segment
Get segment information
Get segments list
Delete segment
Create suppression list
Update suppression list
Get suppression list info
Get the list of suppression lists
Delete suppression list
Upload suppression list data to file
Suppression list actions
Check if email is suppressed
Add email to suppression list
Add multiple emails to suppression list
Remove email from suppression list
Remove all emails from suppression list
Check if domain is suppressed
Add domain to suppression list
Add multiple domains to suppression list
Remove domain from suppression list
Remove all domains from suppression list
Check if phone number is suppressed
Add phone number to suppression list
Add multiple phones to suppression list
Remove phone number from suppression list
Remove all phone numbers from suppression list
Get templates list
Get template info
Delete template
Add template

Update template
Channel object
Get campaign information
Get campaign list
Activate campaign
Complete campaign
Deactivate campaign
Get campaign status
Activate mailing
Deactivate mailing
Get mailing list
Get mailing information
Get mailing log
Clone mailing
Delete mailing
Get mailing status
Get broadcasts list
Get broadcast information
Create broadcast mailing
Update broadcast mailing
Launch a broadcast mailing
Get regular mailings list
Get regular mailing information
Create regular mailing
Update regular mailing
Launch a regular mailing
Get trigger mailings list
Get trigger mailing information
Create trigger mailing
Update trigger mailing
Trigger launch (API call)
Profile import + trigger mailing launch
Task for bulk trigger launch
Task for bulk profiles import + trigger launch
Bulk trigger launch
Bulk profiles import + trigger mailing launch
Clone a trigger mailing
Data array
Engage profile in scenario
Import and engage profile in scenario
Batch import and engage profiles in a scenario
Task for batch import and engaging profiles in the scenario
Get scenarios list
Activate scenario

Deactivate scenario
Get profile tier in a loyalty program
Export points transactions
Expiring points for a period
Get profile account transactions
Get trigger promotions list
Accrue points to a member
Redeem member points
Commit temporary transaction
Preliminary Order Calculation
Order Confirmation
Roll back temporary transaction
Cancel points transaction
Get points account balance
Register member in a loyalty program
Batch adding participants to loyalty program
Task for batch add participants to loyalty program
Remove member from loyalty program
Get form information
Get form list
Get form fills
Publish form
Unpublish form
Delete form
Import promo codes
Get promo code information
Activate promo code
Update promo code
Attach promo code
Detach promo code
Get all promo codes
Goals and goal values registration
Processing and adding a subscription
Add app push events
Order data object
Product data object
SKU data object
Categories array
Custom fields array
Import order and item statuses
Get orders list
Delete order
Get order status
Update order line status

Import products, SKUs and categories
Import SKUs and categories
Get products list
Get SKUs list
Delete products
Delete SKU
Get summary report
Get soft bounces report
Get undeliveries report
Get senders list
Get virtual senders list
Get virtual sender information
Clone virtual sender
Create virtual sender
Update virtual sender
Delete virtual sender
Add segmentation query
Update segmentation query
Get segmentation query information
Get segmentation queries list
Delete segmentation query
Add template query
Update template query
Get template query information
Get template queries list
Delete template query
AKMTA object
Content object
Email rule object
File object
Profile data object
Sender object
Sender types
SMS rule object
Start schedule object
Subscription object
Trigger types
Upload file
Get message web version
Push providers
Deduplication of requests
How to send API request with RabbitMQ
List of gender identifications
Obtain valid values for fields: browsers, devices, tz, oses, languages

mSDK
Managing JWT and Role Token
Quick Start
SDK Functionality
SDK Configuration
Public SDK API
Firebase Cloud Messaging
Huawei Mobile Services
RuStore
Quick Start
SDK Functionality
SDK Configuration
Public SDK API
Apple Push Notification Service
Firebase Cloud Messaging
Huawei Mobile Services
Quick Start
SDK Configuration
SDK Functionality
Public SDK API
Provider setup

API interaction

TIP

You can download the Altcraft API collection for Postman using this link.

Requests format

Requests are sent by POST method using HTTP/1.1 protocol. The method is also mentioned on each reque
Request parameters are placed in the sent structure. Some parameters can be sent in URL (API key, form
The input data format must be indicated in the Content-Type HTTP header.
Possible header values:

- **application/json — JSON format**

Symbols must have UTF-8 coding.

```
POST https://example.com/api/v1.1/<method name>
```

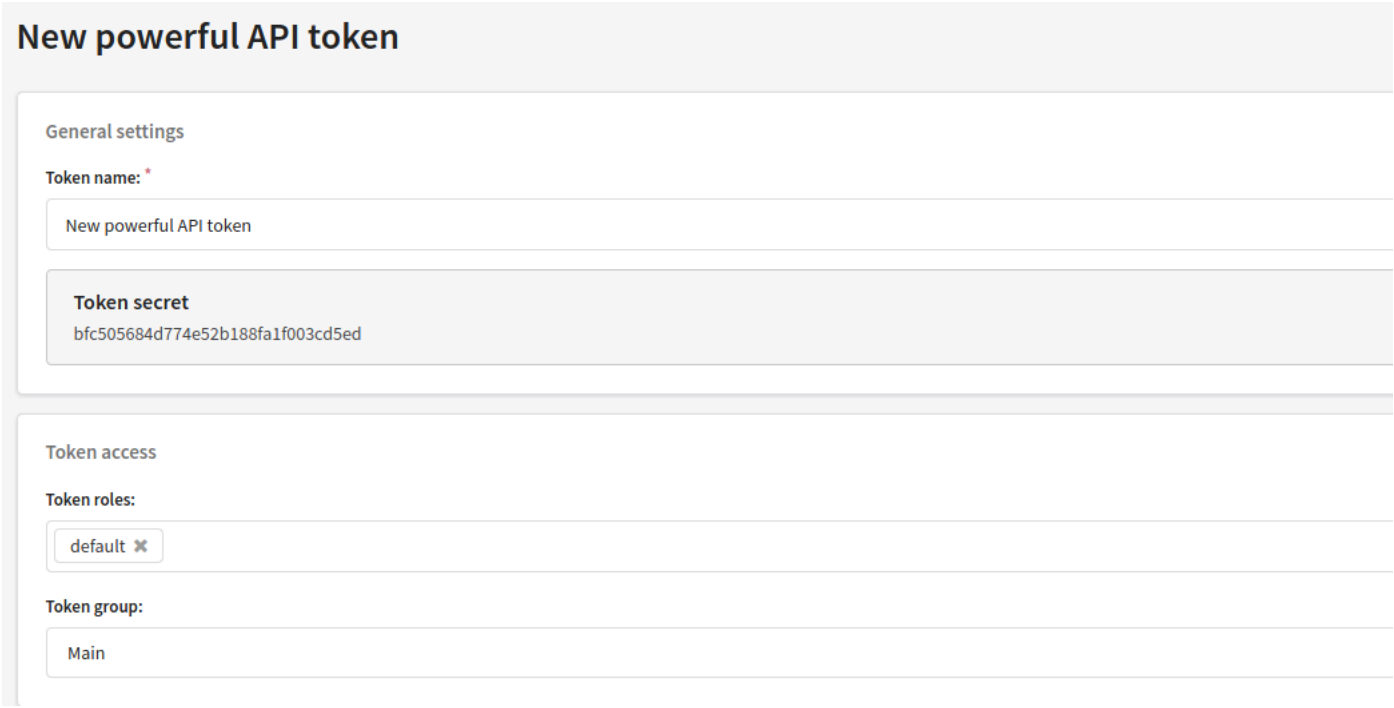
If you use an Altcraft Platform Cloud account, send API requests to the URL `ru.altkraft.com`. In the case of
A description of the JSON format can be found in RFC 7159: http://tools.ietf.org/html/rfc7159.
Don't forget to encode characters that cannot be directly written in JSON.

Authorization

For authorization when sending a request, you need to pass a token to the platform. There are several wa

```
{
        "token": "bfc505684d774e52b188fa1f003cd5ed",
        "db_id": 1,
        "resource_id": 1,
        "matching": "email",
        "email": "example@example.com",
        "data": {
                "_status": 0,
                "_fname": "Jim",
                "_lname": "Jones",
                "email": "example@example.com",
                "phones": ["+79000000000"]
        }
}
```

You can also pass the token in one of the request headers. To do this, create a header with the Authorizat
API token can be created in the user panel, in section "Settings" — "Tokens". Master user rights are neces



API token is automatically generated after saving. You can also select a token name and configure acces

CAUTION

We do not recommend passing a token in the URL of your request. This method is insecure and will not b

Response format

The response format can be selected in the request header or parameters.

Response example (Successful operation):

```
{
        "error": 0,
        "error_text": "Successful operation",
        "profile_id": "5f4fa1a5ce9448665fef548e"
}
```

The following parameters are given in responses:

- `error` — **error code;**
- `error_text` — **error description;**
- `profile_id` — **profile identifier (for successful operation).**

> INFO
>
> **Code 200 is the success status code in HTTP transport. However, if there is information about an error w**

Response codes

Code	
0	Operation is successful
400	Incorrect request
401	API token required
402	Tariff limit reached
403	No rights for this action
404	The object is not found in the token vision zone
409	Another record with the same unique attributes already exists
413	The database is not available in the selected resource
415	Requested Content-Type is not supported
429	The number of requests set in the configuration file has been exceeded (`API_MAX_REQUESTS_COUNT`)
435	Ambiguous search, there are several objects with given attributes
441	The object belongs to another group
450	Request validation error
500	Internal service error
501	The method does not exist

XML tags attributes

XML queries are built on top of JSON queries, so in some situations it is necessary to write additional attr

- `array='true'` **- an attribute indicating that the tag is an array. Used when there is either 1 element insid**
- `string='true'` **- attribute is used to indicate that the value used has a string data type. Required when t**
- `json='true'` **- attribute needed when we need to pass json request inside xml request (for example, sel**

Request deduplication

If the connection fails at the moment of receiving the data, a second request may be sent. The platform w

Matching

The `matching` field specifies the search mode for the profile to update. The default value is `email`.
When you update data, you should follow a general matching approach.

Search mode	matching	Required fields	
By email, from profile or subscription	**"email"**	**email**	**s**
By email from profile	**"email_profile"**	**email**	**s**
By email from subscriptions	**"email_sub"**	**email**	**s**
By phone, from profile or subscription	**"phone"**	**phone**	**s**
By phone from subscriptions	**"phone_sub"**	**phone**	**s**
By profile ID (only on update!)	**"profile_id"**	**profile_id**	**s**
By push subscription	**"push_sub"**	**provider**	**s**
		subscription_id	**s**
By custom field	**"custom"**	**field_name**	**s**
		field_value	**i**
Search for a profile by subscription to the custom channel	**"custom_sub"**	**field_name**	**s**
		field_value	**i**
By email or phone, from profile	**"email_phone"**	**email / phone**	**s**
By email or phone, from subscriptions	**"email_phone_sub"**	**email / phone**	**s**

INFO

Unclear matching

If the field you are passing in the request body is not unique, i.e. the value of this field is the same for se

INFO

Duplicate unique data

The “Duplicate unique data” error occurs if you use a custom unique field for matching when importing a
For example, a unique custom field `client_id` is used for matching. The database has a profile with fields:

- `email: test@example.com`,
- `client_id: 100`.

You import another profile with fields:

- `email: test@example.com`,
- `client_id: 101`.

The request will return a “Duplicate unique data” error, because the uniqueness of the `email` system fiel
In such cases, it is recommended to pass the `email` field in the `subscription` object:

```
{
    "token": "m1883gf9cacngh18c912016r7c7995a",
    "db_id": 1,
    "matching": "custom",
    "field_name": "client_id",
    "field_value": "100",
    "data": {
            "_fname": "John",
            "_lname": "Doe",
            "subscriptions": [
                {
                    "channel": "email",
                    "email": "test@example.com",
                    "resource_id": 1
                }
            ]
        }
}
```

Examples of importing/updating a subscription with certain matching

email, email_profile, email_sub, email_phone, email_phone_sub

```
{
     "token": "m1883gf9cacngh18c912016r7c7995a",
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "data": {
        "subscriptions": [
            {
                "channel": "email",
                "email": "example@example.com",
                "resource_id": 1
            }
        ]
     }
}
```

phone, phone_sub, email_phone, email_phone_sub

```
{
     "token": "m1883gf9cacngh18c912016r7c7995a",
     "db_id": 1,
     "matching": "phone",
     "phone": "+790000000000",
     "data": {
        "subscriptions": [
                            {
                "channel": "sms",
                "phone": "+790000000000",
                "resource_id": 1
            }
        ]
     }
}
```

profile_id

```
{
     "token": "91f1dfa81c264a938b475677c60ce967",
     "db_id": 1,
     "matching": "profile_id",
     "profile_id": "012452gf62fcd71fjh8ec0",
     "data": {
        "subscriptions": [
            {
                "channel": "email",
                "email": "example@example.com",
                "resource_id": 1
            }
        ]
    }
}
```

push_sub

```
{
     "token": "91f1dfa81c264a938b475677c60ce967",
     "db_id": 1,
     "matching": "push_sub",
     "provider": "android-firebase",
     "subscription_id": "<token>",
     "data": {
        "subscriptions": [
            {
                "channel": "push",
                "subscription_id": "<token>",
                "provider": "android-firebase",
                "resource_id": 1
            }
        ]
    }
}
```

custom, custom_sub

```
{
    "token": "91f1dfa81c264a938b475677c60ce967",
    "db_id": 1,
         "matching": "custom",
    "field_name": "CRM_id",
    "field_value": "12345",
    "data": {
       "subscriptions": [
           {
               "channel": "email",
               "email": "example@example.com",
               "resource_id": 1
           }
       ]
   }
}
```


TIP

The `subscription` object contains data about the profile's subscriptions to the resource. Learn more.

Importing the API collection in Postman

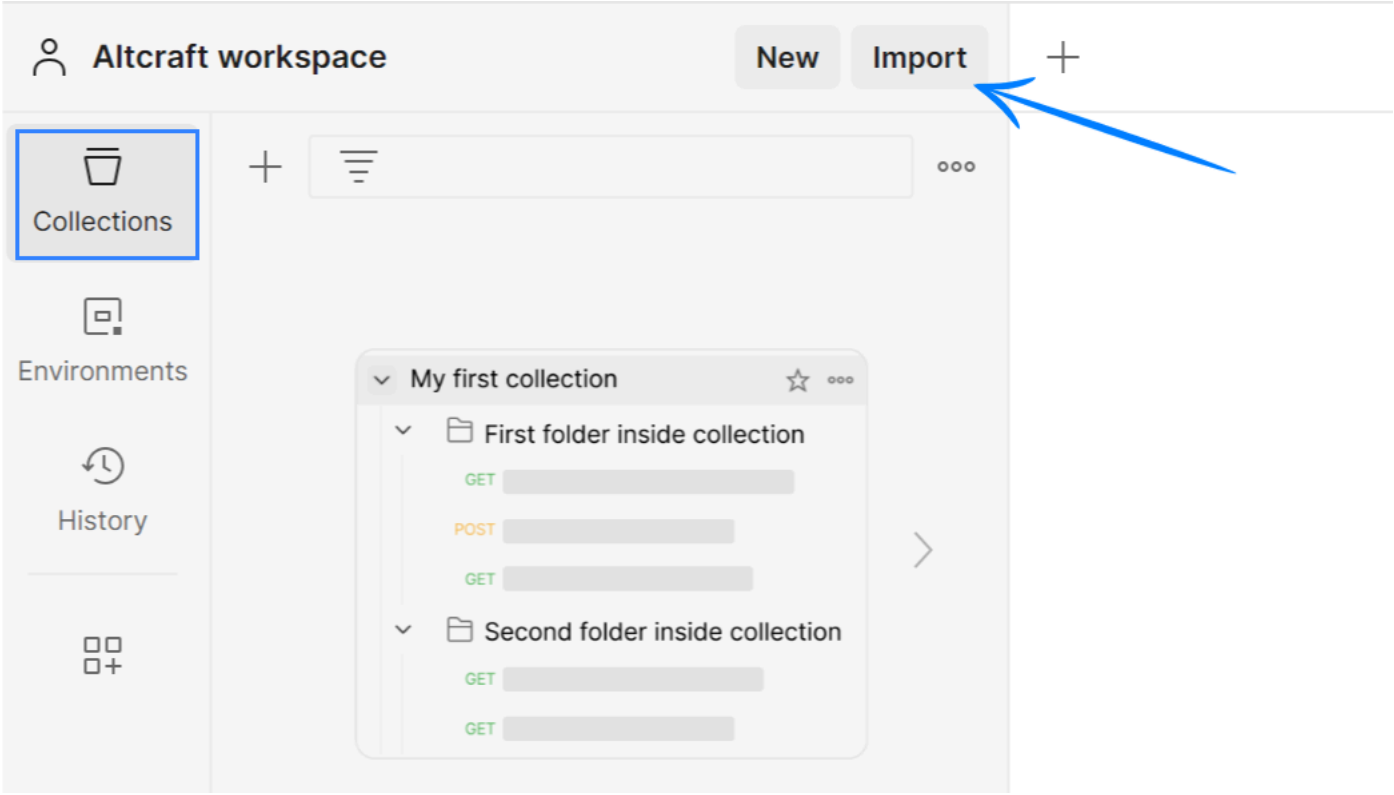
This guide will help you import and configure the API methods collection to work with Altcraft Platform vi

Importing the collection

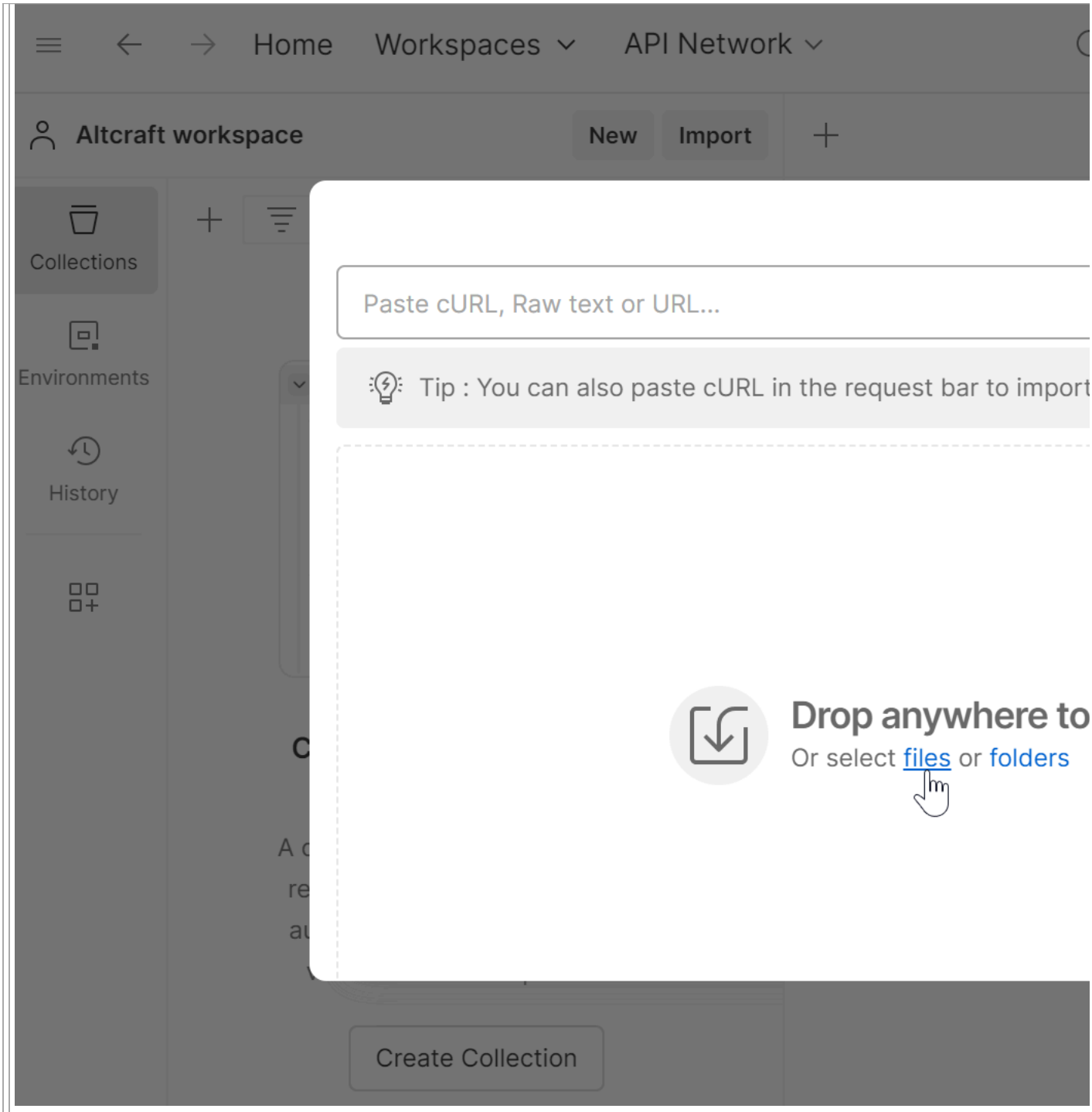
1. First, make sure you have Postman installed. You can download it from the official website.
2. Download the API methods collection to your device.

> TIP
> **If the file doesn't start downloading, right-click the link and choose "Save link as".**

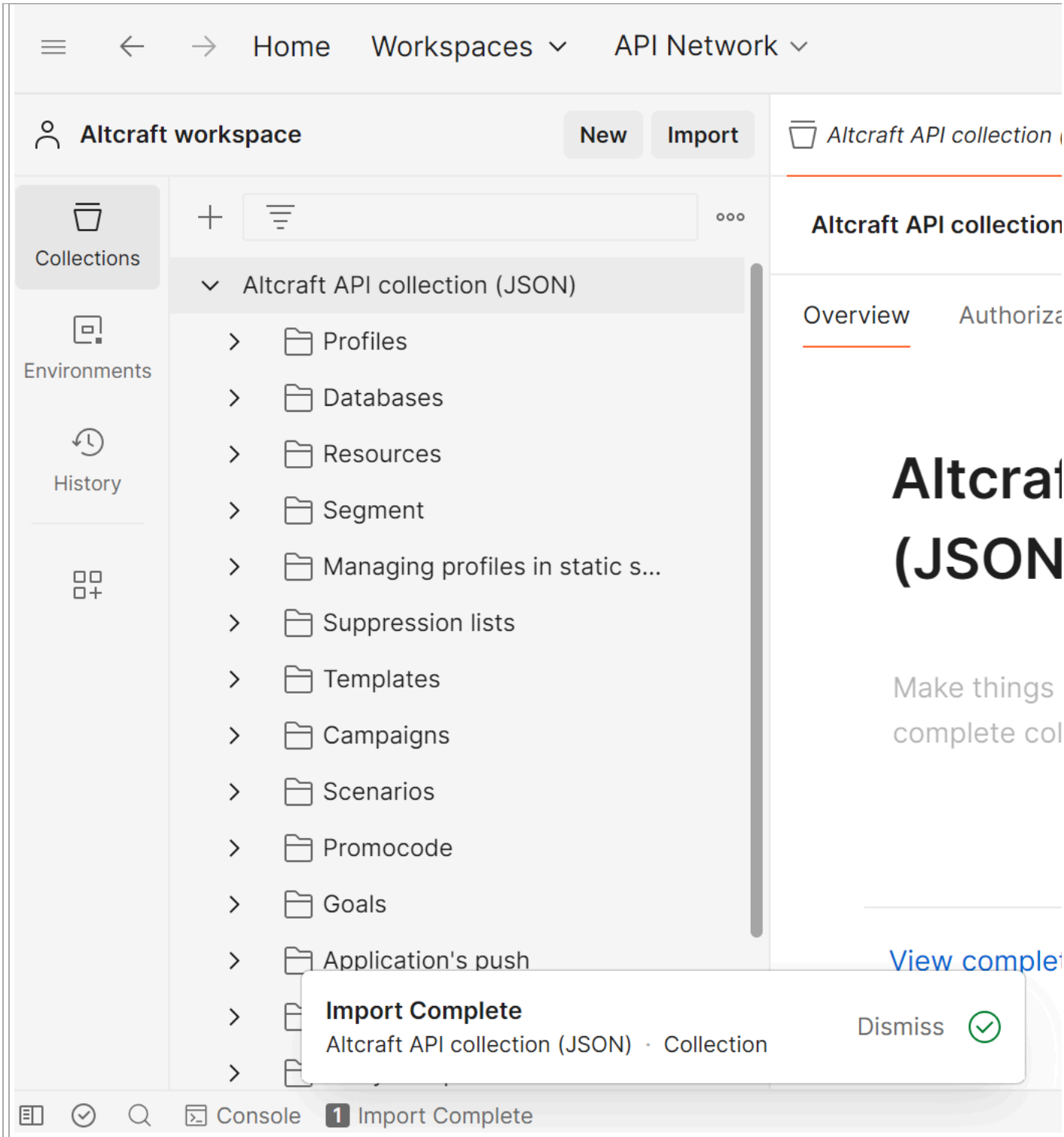
3. In Postman, go to the "Collections" section and click the "Import" button:



4. Import the collection file via the upload window:

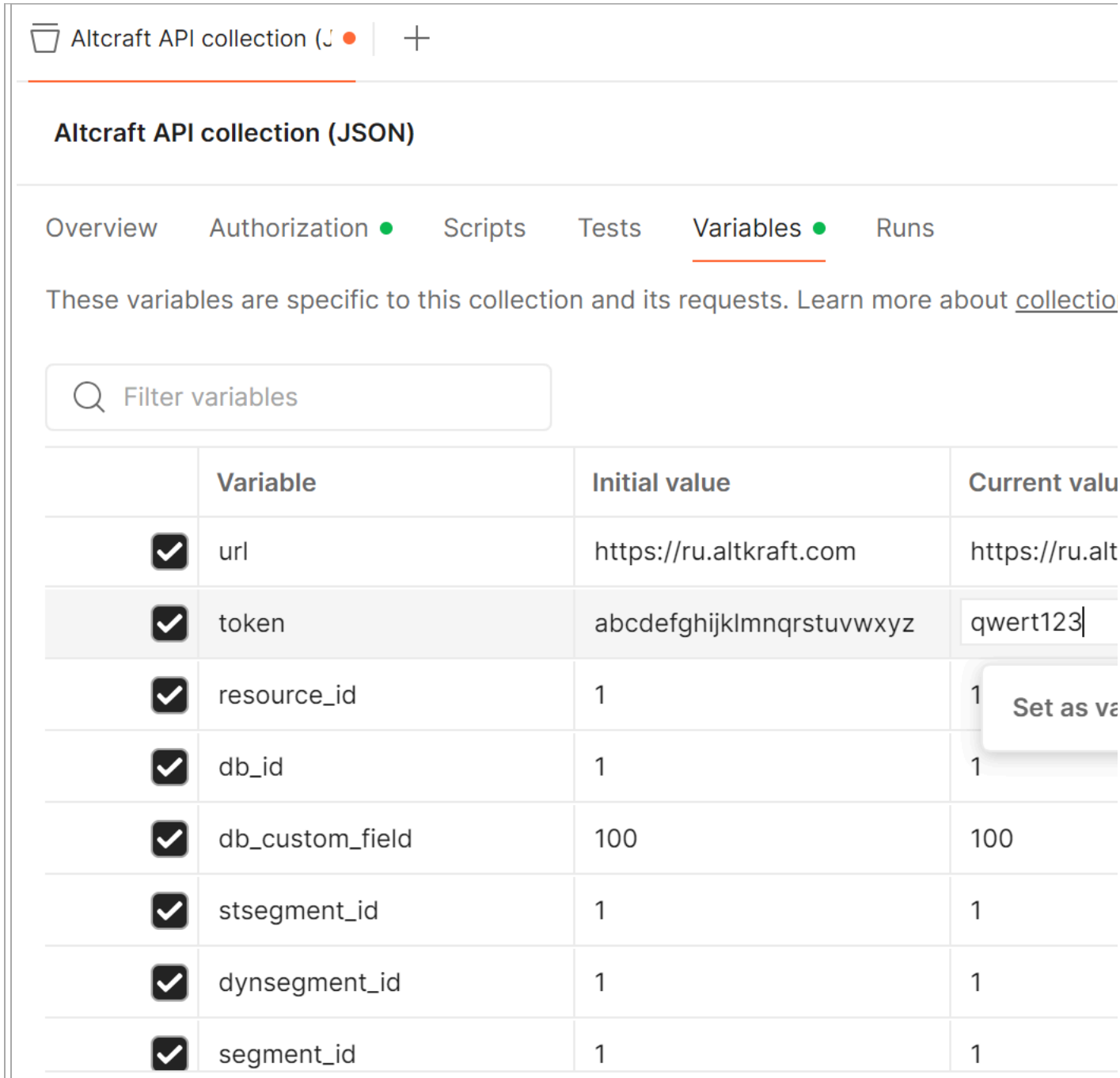


After the import is complete, all API methods will appear in the sidebar:



5. The collection includes Postman environment variables. Go to the “Variables” section to view the list of

6. The “Initial value” column contains default values for each variable. These act as examples. Replace th



7. Save the changes by clicking the “Save” button or using Ctrl + S.

CAUTION

Make sure to save any changes to the variable list after editing.

Collection Variables List

Postman Variable	
url	**The URL address of the platform's control panel** **If you are using a cloud account for Altcraft Platform, use `ru.altkraft.com`. For on-premise installations, use your own URL.**
token	**The platform token** **(created in the "Settings" section => "Tokens")**
resource_id	**Resource ID**
db_id	**Profile database ID**
db_custom_field	**Name of the custom field in the profile database** **Note that all collection requests use a custom field of type "Integer". If you are working with other field types, make sure**
stsegment_id	**Static segment ID**
dynsegment_id	**Dynamic segment ID**
segment_id	**Static or dynamic segment ID**
suplist_id	**Suppression list ID**

Postman Variable	
template_id	**Message template ID**
campaign_id	**Mailing ID**
workflow_id	**Workflow ID**
pixel_id	**Pixel ID**
loyalty_id	**Loyalty program ID**
vsender_id	**Virtual sender ID**
profile_id	**Profile ID**
smid	**Message send ID** **(can be found in the email header `X-AKMTA-FSMID`)**
subscription_id	**Profile push subscription ID (push token)**
rtoken	**Resource token** **(can be found in the resource editing window on the "Details" tab — button** **i** **in the top right corner)**
launch_id	**Internal send ID, unique within the mailing**
product_id	**Additional product identifier**
sku_id	**Additional SKU identifier**
order_external_id	**Additional order identifier (string)**
order_internal_id	**Order ID (integer)**
line_id	**Additional order line item ID**
market_endpoint	**Additional endpoint identifier**
market_region	**Additional region identifier**
market_category	**Additional product or SKU category identifier**
market_currency	**Currency identifier**

List of API endpoints

Profiles

Import profile
Import multiple profiles
Add profile to database
Add multiple profiles
Quick profile import
Update profile
Update multiple profiles
Get profile data
Get data for multiple profiles
Uploading profiles to a file
Delete profile
Merging multiple profiles
Profile splitting
Unsubscribe profile from resource
Suspend all database profiles
Unsuspend all profiles in a database
Get profile action history
Get multiple profiles action history
Attach relation
Detach relation
Modify relation properties
Overwrite relation properties
Get profile relations info
Add or edit subscription
Get all profile subscriptions
Get all subscriptions from multiple profiles
Get profile subscription
Delete profile subscription
Restore deleted subscription
Suspend all subscriptions
Unsuspend all suspended subscriptions

Databases

Get database list
Get database information
Get database fields
Database wipe

Resources

Get resources list
Get resource information
Get resource subscription fields

Segments

Create segment
Update segment
Get segment information
Get segments list
Delete segment
Refresh segment subscription statistics
Get profile subscription statistics for a segment
Add profile
Remove profile
Get profile data in a static segment

Suppression lists

- Create suppression list
- Update suppression list
- Get suppression list info
- Get the list of suppression lists
- Delete suppression list
- Upload suppression list data to file

- Check if email is suppressed
- Add email to suppression list
- Add multiple emails to suppression list
- Remove email from suppression list
- Remove all emails from suppression list

- Check if domain is suppressed
- Add domain to suppression list
- Add multiple domains to suppression list
- Remove domain from suppression list
- Remove all domains from suppression list

- Check if phone number is suppressed
- Add phone number to suppression list
- Add multiple phones to suppression list
- Remove phone number from suppression list
- Remove all phone numbers from suppression list

Templates

- Add template
- Update template
- Get template info
- Get templates list
- Delete template

Mailing

- Activate mailing
- Deactivate mailing
- Get mailing information
- Get mailing status
- Get mailing log
- Get mailing list
- Clone mailing
- Delete mailing

- Create broadcast mailing
- Update broadcast mailing
- Launch a broadcast mailing
- Get broadcast information
- Get broadcasts list

- Create regular mailing
- Update regular mailing
- Launch a regular mailing
- Get regular mailing information
- Get regular mailings list

- Create trigger mailing
- Update trigger mailing
- Clone a trigger mailing
- Trigger launch (API call)
- Bulk trigger launch
- Task for bulk trigger launch
- Profile import + trigger mailing launch
- Bulk profiles import + trigger mailing launch
- Task for bulk profiles import + trigger launch
- Get trigger mailing information

- Get trigger mailings list

Campaigns

- Get campaign information
- Get campaign list
- Activate campaign
- Deactivate campaign
- Complete campaign
- Get campaign status

Automation scenarios

- Engage profile in scenario
- Import and engage profile in scenario
- Batch import and engage profiles in a scenario
- Task for batch import and engaging profiles in the scenarioй
- Get scenarios list

Promo codes

- Import promo codes
- Get promo code information
- Activate promo code
- Update promo code
- Attach promo code
- Detach promo code
- Detach promo code

Loyalty Program

- Get points balance
- Get profile account transactions
- Get expiring points for a period
- Get profile loyalty tier
- Accrue points to member
- Redeem member points
- Cancel points transaction
- Export points transactions
- Register member in loyalty program
- Delete member from loyalty program
- Get trigger promotions list
- Preorder
- Confirm order
- Confirm temporary transaction
- Cancel temporary transaction

Goals

- Goals and goal values registration

Application push notifications

- Add app push events

Market

- Import order and item statuses
- Get orders list
- Delete order
- Update order line status
- Import products, SKUs and categories
- Import SKUs and categories
- Get products list
- Get SKUs list

Analytic reports

- Get summary report
- Get soft bounces report
- Get undeliveries report

Senders

- Get senders list
- Create virtual sender
- Update virtual sender
- Get virtual sender information
- Get virtual senders list
- Clone virtual sender
- Delete virtual sender

External datatables queries

- Add segmentation query
- Update segmentation query
- Delete segmentation query
- Get segmentation queries list
- Get segmentation query info
- Add templates query
- Update templates query
- Delete templates query
- Get templates queries list
- Get templates query info

Miscellaneous

- Upload file
- Get message web version

Import profile

Description

Imports a customer profile. It also can be used to update an existing profile.
Can delete custom fields, add or delete tags, increase or decrease field values. See update functions for r

> **TIP**
>
> **To launch a trigger mailing right after the import you can use Import + trigger request.**

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/import`

Request parameters

> **CAUTION**
>
> - **For JSON body use the following header: Content-Type: application/json.**
> - **For XML body use the following header: Content-Type: application/xml.**

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Ye**
db_id	**int**	**1**	**Ye**
matching	**string**	**“email” “email_profile” “email_sub” “phone” “phone_sub” “profile_id” “push_sub” “custom” “custom_sub” “email_phone” “email_phone_sub”**	**No**
skip_triggers	**bool**	**true**	**No**
skip_invalid_subscriptions	**bool**	**true**	**No**
detect_geo	**bool**	**true**	**No**
add_to_segments	**array**	`[1, 2]`	**No**
remove_from_segments	**array**	`[1, 2]`	**No**
data	**JSON object**	`{ “_fname”: “John”, “_lname”: “Doe” }`	**Ye**

Subscriptions array

The `subscriptions` array stores data about the profile’s subscriptions to resources. One object is one subsc

Parameter	Type	Example	
resource_id	**int**	**1**	**Resource ID**
status	**string**	**"subscribed"**	**Subscription status** **It is recommended to pass the parameter in the req**
priority	**int**	**1**	**Subscription priority** **If the subscription priority is passed in the request,**
custom_fields	**object**	`{ "_browser_name": "Chrome", "_device_type": "web" }`	**System and custom subscription fields.** **You can get information about available subscriptio**
cats	**array**	`[ "category_1", "category_2" ]`	**Resource categories to which to subscribe the prof**
Email channel			
channel	**string**	**"email"**	**Channel type**
email	**string**	**"john@example.com"**	**Email**
SMS channel			
channel	**string**	**"sms"**	**Channel type**
phone	**string**	**"+79000000000"**	**Phone number**
Push channel			
channel	**string**	**"push"**	**Channel type**
provider	**string**	**"Firefox"**	**Provider type**
subscription_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription ID**

Parameter	Type	Example	
Telegram Bot channel			
channel	**string**	**"telegram_bot"**	**Channel type**
cc_data	**object**	{ "id": 1 }	**Telegram Bot chat ID**
WhatsApp* channel			
channel	**string**	**"whatsapp"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on WhatsApp***
Viber channel			
channel	**string**	**"viber"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on Viber**

Request example

Importing a profile and adding an email channel subscription

JSON

```
{
     "token": "abcdefghijklmnqrstuvwxyz",
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": ["+79000000000"],
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            }
        ]
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <matching>email</matching>
    <detect_geo>true</detect_geo>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <subscriptions array='true'>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
    </data>
</xml>
```

Importing a profile with extended data

JSON

```
{
     "token": "abcdefghijklmnqrstuvwxyz",
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",

        "email": "example@example.com",
        "phones": ["+79000000000"],

        "_bdate": "1990-02-22T21:00:00Z",
        "_sex": 0,

        "_regdate": "2019-03-14T22:00:00Z",
        "_regip": "94.231.119.122",
        "_ip": "94.231.119.122",

        "_tz": "Europe/Moscow",
        "_postal_code": "390000",

        "_os": "Windows 10",
        "_browser": "Firefox",

        "_vendor": "form_#31",

        "custom_field": "custom_value",

        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            }
        ]
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <_bdate>1990-07-06</_bdate>
        <_sex>0</_sex>
        <_regdate>2023-07-06</_regdate>
        <_regip>192.168.0.179</_regip>
        <_ip>192.168.0.179</_ip>
        <_tz>Europe/Moscow</_tz>
        <_postal_code>390000</_postal_code>
        <_os>Windows 10</_os>
        <_browser>Firefox</_browser>
        <_vendor>form_#31</_vendor>
        <custom_field>custom_value</custom_field>
        <subscriptions array='true'>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
    </data>
</xml>
```

Importing a profile and adding subscriptions to multiple channels

JSON

```
{
     "token": "abcdefghijklmnqrstuvwxyz",
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",
        "email": "example@example.com",
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            },
            {
                "resource_id": 2,
                "channel": "sms",
                "phone": "+79000000000"
            },
            {
                "resource_id": 1,
                "channel": "push",
                "subscription_id": "a81c264a938b475",
                "provider": "android-firebase",
                "custom_fields": {
                    "_browser_name": "Firefox",
                    "_device_type": "web",
                    "custom_field_1": "test value"
                },
                "cats": [
                    "category_1",
                    "category_2"
                ]
            },
            {
                "resource_id": 1,
                "channel": "whatsapp",
                "cc_data": {
                    "phone": "+79000000000"
                }
            },
            {
                "resource_id": 1,
                "channel": "viber",
                "cc_data": {
                    "phone": "+79000000000"
                }
            },
            {
                "resource_id": 1,
                "channel": "telegram_bot",
                "cc_data": {
                    "id": 12345
                }
            },
            {
                "resource_id": 1,
                "channel": "notify",
                "cc_data": {
                    "phone": "+79000000000"
                }
            }
        ]
    }
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>sms</channel>
            <phone>+79000000000</phone>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>push</channel>
            <subscription_id>a81c264a938b475</subscription_id>
            <provider>android-firebase</provider>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>whatsapp</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>viber</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>telegram_bot</channel>
            <cc_data>
              <id>12345</id>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>notify</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "54759eb3c090d83494e2d804"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>c1are24fu213lwith21that2axe9<profile_id>
</xml>
```

Response parameters

Parameter
error
error_text
profile_id

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as*

Update profile

Description

Updates a customer profile.
Can delete custom fields, add or delete tags, increase or decrease field values. See update functions for r

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/update`

Request parameters

CAUTION

- **For JSON body use the following header: Content-Type: application/json.**
- **For XML body use the following header: Content-Type: application/xml.**

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Ye**
db_id	**int**	**1**	**Ye**
matching	**string**	**“email” “email_profile” “email_sub” “phone” “phone_sub” “profile_id” “push_sub” “custom” “custom_sub” “email_phone” “email_phone_sub”**	**No**
skip_triggers	**bool**	**true**	**No**
skip_invalid_subscriptions	**bool**	**true**	**No**
detect_geo	**bool**	**true**	**No**
add_to_segments	**array**	`[1, 2]`	**No**
remove_from_segments	**array**	`[1, 2]`	**No**
data	**JSON object**	`{ “_fname”: “John”, “_lname”: “Doe” }`	**Ye**

Subscriptions array

The `subscriptions` array stores data about the profile’s subscriptions to resources. One object is one subsc

Parameter	Type	Example	
resource_id	**int**	**1**	**Resource ID**
status	**string**	**"subscribed"**	**Subscription status** **It is recommended to pass the parameter in the req**
priority	**int**	**1**	**Subscription priority** **If the subscription priority is passed in the request,**
custom_fields	**object**	`{ "_browser_name": "Chrome", "_device_type": "web" }`	**System and custom subscription fields.** **You can get information about available subscriptio**
cats	**array**	`[ "category_1", "category_2" ]`	**Resource categories to which to subscribe the prof**
Email channel			
channel	**string**	**"email"**	**Channel type**
email	**string**	**"john@example.com"**	**Email**
SMS channel			
channel	**string**	**"sms"**	**Channel type**
phone	**string**	**"+79000000000"**	**Phone number**
Push channel			
channel	**string**	**"push"**	**Channel type**
provider	**string**	**"Firefox"**	**Provider type**
subscription_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription ID**
Telegram Bot channel			
channel	**string**	**"telegram_bot"**	**Channel type**
cc_data	**object**	`{ "id": 1 }`	**Telegram Bot chat ID**

Parameter	Type	Example	
WhatsApp* channel			
channel	**string**	**"whatsapp"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on WhatsApp***
Viber channel			
channel	**string**	**"viber"**	**Channel type**
cc_data	**object**	{ "phone": "+79000000000" }	**Profile phone number on Viber**

Request example

Updating basic profile data and adding an email channel subscription

JSON

```
{
     "token": "abcdefghijklmnqrstuvwxyz",
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": ["+79000000000"],
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            }
        ]
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <matching>email</matching>
    <detect_geo>true</detect_geo>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <subscriptions array='true'>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
    </data>
</xml>
```

Updating additional profile data

JSON

```
{
     "token": "abcdefghijklmnqrstuvwxyz",
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",

        "email": "example@example.com",
        "phones": ["+79000000000"],

        "_bdate": "1990-02-22T21:00:00Z",
        "_sex": 0,

        "_regdate": "2019-03-14T22:00:00Z",
        "_regip": "94.231.119.122",
        "_ip": "94.231.119.122",

        "_tz": "Europe/Moscow",
        "_postal_code": "390000",

        "_os": "Windows 10",
        "_browser": "Firefox",

        "_vendor": "form_#31",

        "custom_field": "custom_value"
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79511059000</phones>
        <_bdate>1990-07-06</_bdate>
        <_sex>0</_sex>
        <_regdate>2023-07-06</_regdate>
        <_regip>192.168.0.179</_regip>
        <_ip>192.168.0.179</_ip>
        <_tz>Europe/Moscow</_tz>
        <_postal_code>390000</_postal_code>
        <_os>Windows 10</_os>
        <_browser>Firefox</_browser>
        <_vendor>form_#31</_vendor>
        <custom_field>custom_value</custom_field>
    </data>
</xml>
```

Adding subscriptions to multiple channels to a profile

JSON

```
{
     "token": "abcdefghijklmnqrstuvwxyz",
     "db_id": 1,
     "matching": "email",
     "email": "example@example.com",
     "detect_geo": true,
     "data": {
        "_fname": "Olly",
        "_lname": "Lambert",
        "email": "example@example.com",
        "subscriptions": [
            {
                "resource_id": 1,
                "channel": "email",
                "email": "example@example.com"
            },
            {
                "resource_id": 2,
                "channel": "sms",
                "phone": "+79000000000"
            },
            {
                "resource_id": 1,
                "channel": "push",
                "subscription_id": "a81c264a938b475",
                "provider": "android-firebase",
                "custom_fields": {
                    "_browser_name": "Firefox",
                    "_device_type": "web",
                    "custom_field_1": "test value"
                },
                "cats": [
                    "category_1",
                    "category_2"
                ]
            },
            {
                "resource_id": 1,
                "channel": "whatsapp",
                "cc_data": {
                    "phone": "+79000000000"
                }
            },
            {
                "resource_id": 1,
                "channel": "viber",
                "cc_data": {
                    "phone": "+79000000000"
                }
            },
            {
                "resource_id": 1,
                "channel": "telegram_bot",
                "cc_data": {
                    "id": 12345
                }
            },
            {
                "resource_id": 1,
                "channel": "notify",
                "cc_data": {
                    "phone": "+79000000000"
                }
            }
        ]
    }
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <detect_geo>true</detect_geo>
    <email>example@example.com</email>
    <matching>email</matching>
    <data>
        <_fname>Olly</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>email</channel>
            <email>example@example.com</email>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>sms</channel>
            <phone>+79000000000</phone>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>push</channel>
            <subscription_id>a81c264a938b475</subscription_id>
            <provider>android-firebase</provider>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>whatsapp</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>viber</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>telegram_bot</channel>
            <cc_data>
              <id>12345</id>
            </cc_data>
        </subscriptions>
        <subscriptions>
            <resource_id>1</resource_id>
            <channel>notify</channel>
            <cc_data>
              <phone>+79000000000</phone>
            </cc_data>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "54759eb3c090d83494e2d804"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>c1are24fu213lwith21that2axe9<profile_id>
</xml>
```

Response parameters

Parameter
error
error_text
profile_id

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as*

Import multiple profiles

Description

Adds or updates multiple profiles to or in a database.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/import_batch`

Request parameters

CAUTION

- **For JSON body use the following header: Content-Type: application/json.**
- **For XML body use the following header: Content-Type: application/xml.**

Parameter	Type	Example	Required
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**

data	**JSON array**	<pre>[
		  {
		  "_fname": "John",
		  "_lname": "Doe",
		  "custom_id": "abcdefghijklmnqrstuvwxyz",
		  "email": "john@example.com",
		  "subscriptions": [
		  {
		  "channel": "email",
		  "email_data": {
		  "email": "john@example.com"
		  },
		  "resource_id": 1
		  }
		  ]
		  }
		]</pre>
			Yes

db_id	**int**	**1**	**Yes**
skip_invalid_subscriptions	**bool**	**true**	**No**
skip_triggers	**bool**	**true**	**No**
detect_geo	**bool**	**true**	**No**
Profile matching mode			
matching	**string**	**"email"** **"email_profile"** **"email_sub"** **"phone"** **"phone_sub"** **"profile_id"** **"push_sub"** **"custom"** **"custom_sub"** **"email_phone"** **"email_phone_sub"**	**No, if matched by email**
email	**string**	**"john@example.com"**	**"matching":"email"**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Subscriptions array

The `subscriptions` array stores data about the profile's subscriptions to resources. One object is one subsc

Parameter	Type	Example	
resource_id	**int**	**1**	**Resource ID**
status	**string**	**"subscribed"**	**Subscription status** **It is recommended to**
priority	**int**	**1**	**Subscription priority**

custom_fields	**object**	<pre>{
		  "_browser_name": "Chrome",
		  "_device_type": "web"
		}</pre>
			System and custom s
			
			You can get informat

cats	**array**	<pre>[
		  "category_1",
		  "category_2"
		]</pre>
			Resource categories

Email channel			

Parameter	Type	Example	
channel	**string**	**"email"**	**Channel type**
email	**string**	**"john@example.com"**	**Email**
SMS channel			
channel	**string**	**"sms"**	**Channel type**
phone	**string**	**"+79000000000"**	**Phone number**
Push channel			
channel	**string**	**"push"**	**Channel type**
provider	**string**	**"Firefox"**	**Provider type**
subscription_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription ID**
Telegram Bot channel			
channel	**string**	**"telegram_bot"**	**Channel type**

cc_data	**object**	<pre>{
		"id": 1
		}</pre>
			Telegram Bot chat ID

WhatsApp* channel			
channel	**string**	**"whatsapp"**	**Channel type**

cc_data	**object**	<pre>{
		"phone": "+79000000000"
		}</pre>
			Profile phone number

Viber channel			
channel	**string**	**"viber"**	**Channel type**

cc_data	**object**	<pre>{
		"phone": "+79000000000"
		}</pre>
			Profile phone number

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "matching": "custom",
    "field_name": "Custom_ID",
    "skip_triggers": true,
    "data": [
        {
            "_fname": "Lala",
            "_lname": "Doe",
            "Custom_ID": 77,
            "email": "example@example.com",
            "subscriptions": [
                {
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.com"
                    },
                    "resource_id": 28,
                    "custom_fields": {
                        "custom_field": "test value"
                    },
                    "cats": [
                        "category_1",
                        "category_2"
                    ]
                }
            ]
        },
        {
            "_fname": "Victor",
            "Custom_ID": 88,
            "email": "example@example.com"
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <skip_triggers>true</skip_triggers>
    <data>
        <_fname>Lala</_fname>
        <_lname>Doe</_lname>
        <Custom_ID>77</Custom_ID>
        <email>example@example.com</email>
        <subscriptions array='true'>
            <channel>email</channel>
            <email_data>
                <email>example@example.com</email>
            </email_data>
            <resource_id>28</resource_id>
            <custom_fields>
                <custom_field>test value</custom_field>
            </custom_fields>
            <cats>category_1</cats>
            <cats>category_2</cats>
        </subscriptions>
    </data>
    <data>
        <_fname>Alex</_fname>
        <Custom_ID>345</Custom_ID>
        <email>example@example.com</email>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "result": [
    {
      "error": 0,
      "error_text": "Successful operation",
      "profile_id": "abcdefghijklmnqrstuvwxyz"
    }
  ]
}
```

XML

```
<xml>
 <error>0</error>
  <error_text>Successful operation</error_text>
  <result>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  </result>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array of object**	**Update results by profiles** **The order of the objects in the returned `result` array**
result.error	**int**	**Error code, for profile**
result.error_text	**string**	**Error text, for profile**
result.profile_id	**string**	**Profile ID**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Update multiple profiles

Description

Updates (only) multiple profiles to or in a database.

> **INFO**
>
> **If you pass the `status` and `priority` parameters of the `subscriptions` object with a value of `0`, then the val**
> **In order not to change or update the values of these parameters, do not pass them, or set the values to**

> **CAUTION**
>
> **In one request it is not recommended:**
>
> - **transfer more than 10,000 profiles;**
> - **transfer more than 10 MB of data. It is possible totransfer more data using multiple streams.**

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/update_batch`

Request parameters

Parameter	Type	Example	Required
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**

data	**JSON array**	<pre>[
		  {
		  "_fname": "John",
		  "_lname": "Doe",
		  "custom_id": "abcdefghijklmnqrstuvwxyz",
		  "email": "john@example.com",
		  "subscriptions": [
		  {
		  "channel": "email",
		  "email_data": {
		  "email": "john@example.com"
		  },
		  "resource_id": 1
		  }
		  ]
		  }
		]</pre>
			Yes

db_id	**int**	**1**	**Yes**
skip_invalid_subscriptions	**bool**	**true**	**No**
Profile matching mode			
matching	**string**	**"email"** **"email_profile"** **"email_sub"** **"phone"** **"phone_sub"** **"profile_id"** **"push_sub"** **"custom"** **"custom_sub"** **"email_phone"** **"email_phone_sub"**	**No, if matched by email**
email	**string**	**"john@example.com"**	**"matching":"email"**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Subscriptions array

The `subscriptions` array stores data about the profile's subscriptions to resources. One object is one subsc

Parameter	Type	Example	
resource_id	**int**	**1**	**Resource ID**

Parameter	Type	Example	

status	**string**	**"subscribed"**	**Subscription status**
			It is recommended to pass the parameter in the request only in cases where it is necessary to update the subscription status.
			</td>
			</tr>
			<tr>
			<td>priority</td>
			<td>int</td>
			<td>1</td>
			<td>Subscription priority</td>
			</tr>
			<tr>
			<td>custom_fields</td>
			<td>object</td>
			<td><pre>{
			   "_browser_name": "Chrome",
			   "_device_type": "web"
			}</pre>
			<td>System and custom subscription fields.
			
			You can get information about available subscription fields in a resource
			</tr>
			<tr>
			<td>cats</td>
			<td>array</td>
			<td><pre>[
			   "category_1",
			   "category_2"
			]</pre></td>
			<td>Resource categories to which to subscribe the profile.</td>
			</tr>
			<tr>
			<td colspan="4">Email channel</td>
			</tr>
			<tr>
			<td>channel</td>
			<td>string</td>
			<td>"email"</td>
			<td>Channel type</td>
			</tr>
			<tr>
			<td>email</td>
			<td>string</td>
			<td>"john@example.com"</td>
			<td>Email</td>
			</tr>
			<tr>
			<td colspan="4">SMS channel</td>
			</tr>
			<tr>
			<td>channel</td>
			<td>string</td>
			<td>"sms"</td>
			<td>Channel type</td>
			</tr>
			<tr>
			<td>phone</td>
			<td>string</td>
			<td>"+79000000000"</td>
			<td>Phone number</td>
			</tr>
			<tr>
			<td colspan="4">Push channel</td>
			</tr>
			<tr>
			<td>channel</td>
			<td>string</td>
			<td>"push"</td>
			<td>Channel type</td>
			</tr>
			<tr>
			<td>
			provider
			</td>
			<td>string</td>
			<td>"Firefox"</td>
			<td>Provider type</td>
			</tr>
			<tr>
			<td>subscription_id</td>
			<td>string</td>
			<td>"abcdefghijklmnqrstuvwxyz"</td>
			<td>Subscription ID</td>
			</tr>
			<tr>
			<td colspan="4">Telegram Bot channel</td>
			</tr>
			<tr>
			<td>channel</td>
			<td>string</td>
			<td>"telegram_bot"</td>
			<td>Channel type</td>
			</tr>
			<tr>
			<td>cc_data</td>
			<td>object</td>
			<td><pre>{
			   "id": 1
			}</pre></td>
			<td>Telegram Bot chat ID</td>
			</tr>
			<tr>
			<td colspan="4">WhatsApp* channel</td>
			</tr>
			<tr>
			<td>channel</td>
			<td>string</td>
			<td>"whatsapp"</td>
			<td>Channel type</td>
			</tr>
			<tr>
			<td>cc_data</td>
			<td>object</td>
			<td><pre>{
			   "phone": "+79000000000"
			}</pre></td>
			<td>Profile phone number on WhatsApp*</td>
			</tr>
			<tr>
			<td colspan="4">Viber channel</td>
			</tr>
			<tr>
			<td>channel</td>
			<td>string</td>
			<td>"viber"</td>
			<td>Channel type</td>
			</tr>
			<tr>
			<td>cc_data</td>
			<td>object</td>
			<td><pre>{
			   "phone": "+79000000000"
			}</pre></td>
			<td>Profile phone number on Viber</td>
			</tr>

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "matching": "custom",
    "field_name": "Custom_ID",
    "skip_triggers": true,
    "data": [
        {
            "_fname": "Lala",
            "_lname": "Doe",
            "Custom_ID": 77,
            "email": "example@example.com",
            "subscriptions": [
                {
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.com"
                    },
                    "resource_id": 28,
                    "custom_fields": {
                        "custom_field": "test value"
                    },
                    "cats": [
                        "category_1",
                        "category_2"
                    ]
                }
            ]
        },
        {
            "_fname": "Victor",
            "Custom_ID": 88,
            "email": "example@example.com"
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <skip_triggers>true</skip_triggers>
    <data>
        <_fname>Lala</_fname>
        <_lname>Doe</_lname>
        <Custom_ID>77</Custom_ID>
        <email>example@example.com</email>
        <subscriptions array='true'>
            <channel>email</channel>
            <email_data>
                <email>example@example.com</email>
            </email_data>
            <resource_id>28</resource_id>
            <custom_fields>
                <custom_field>test value</custom_field>
            </custom_fields>
            <cats>category_1</cats>
            <cats>category_2</cats>
        </subscriptions>
    </data>
    <data>
        <_fname>Alex</_fname>
        <Custom_ID>345</Custom_ID>
        <email>example@example.com</email>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "error": 0,
            "error_text": "Successful operation",
            "profile_id": "abcdefghijklmnqrstuvwxyz"
        },
        {
            "error": 0,
            "error_text": "Successful operation",
            "profile_id": "abcdefghijklmnqrstuvwzzz"
        }
    ]
}
```


XML


```
<xml>
<error>0</error>
  <error_text>Successful operation</error_text>
  <result>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  </result>
  <result>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>abcdefghijklmnqrstuvwzzz</profile_id>
  </result>
</xml>
```


Response parameters

Parameter
error
error_text
profile_id

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as

Add multiple profiles

Description

Adds multiple new profiles to a database.
Profiles list is contained in `"data"` parameter as a JSON array. Server response will contain `"result"` parar

> **CAUTION**
>
> **In one request it is not recommended:**
>
> - **transfer more than 10,000 profiles;**
> - **transfer more than 10 MB of data. It is possible totransfer more data using multiple streams.**

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/add_batch/`

Request parameters

Parameter	Type
token	**string**
data	**JSON object**
db_id	**int**
skip_triggers	**bool**
skip_invalid_subscriptions	**bool**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "skip_triggers": true,
    "data": [
        {
            "_fname": "John",
            "_lname": "Doe",
            "subscriptions": [
                {
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.com"
                    },
                    "resource_id": 27
                }
            ]
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <skip_triggers>true</skip_triggers>
    <data array='true'>
        <_fname>Fillipp</_fname>
        <_lname>Doe</_lname>
        <subscriptions array='true'>
            <channel>email</channel>
            <email_data>
                <email>example@example.com</email>
            </email_data>
            <resource_id>1</resource_id>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "result": [
    {
      "error": 0,
      "error_text": "Successful operation",
      "profile_id": "663a138cee44bb50a67115cd"
    }
  ]
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>663a138cee44bb50a67115cd</profile_id>
  </xml>
</xml>
```

Response parameters

Parameter
error
error_text
profile_id

Add profile to database

TIP

The request does not look for existing records and adds any data other than those that duplicate the pro

Description

Adds a new customer profile to a database.

CAUTION

If no data is specified — an empty profile will be created.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/add`

Request parameters

Parameter	Type
token	**string**
db_id	**int**
data	**JSON object**
skip_triggers	**bool**
skip_invalid_subscriptions	**bool**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "data": {
        "_fname": "Fillipp",
        "_lname": "Doe"
    },
    "skip_triggers": true
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <skip_triggers>true</skip_triggers>
    <data>
        <_fname>Fillipp</_fname>
        <_lname>Doe</_lname>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "",
  "profile_id": "6638e1e8ee44bb50a67115c4"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>6638e1e8ee44bb50a67115c4</profile_id>
</xml>
```

Response parameters

Parameter
error
error_text

Quick profile import

Description

Imports a profile to a database in a quick way. The request has a linear structure and fits simple integrati
To have access to extended import options please use /api/v1.1/profiles/import.
The request imports or updates the profile;if you pass the resource ID and contact details, it adds subscri

Request URL

Method: POST. Data can be passed both in URL parameters and in the body of the request.

`https://example.com/api/integrations/any/profile_import`

Request parameters

Parameter	Type	Example	Required	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**API token**
db_id	**int**	**1**	**Yes**	**Database ID**
matching	**string**	**“email”**	**No**	**Profiles matching mode:** • **email** • **email_profile** • **phone** • ***field_name*** **By default – email**
resource_id	**int**	**1**	**No**	**Subscribe to a resource using request contact dat**
email	**string**	**“example@example.com”**	**No**	**Email address**
phone	**string**	**“+75551111444”**	**No**	**Phone number**
_regip	**string**	**“172.16.0.1”**	**No**	**Registration IPv4 or IPv6. For IPv4 geodata will be**
_regurl	**string**	**“https://example.com/?”**	**No**	**Registration URL or HTTP head “referer” value.**
*			**Yes**	**Specify at least one database field. To retrieve fie**
trigger_id	**int**	**13**	**No**	**The ID of the trigger mailing. If the `trigger_id` is s**
workflow_id	**int**	**1**	**No**	**Scenario ID. If `workflow_id` is set, there will be a si**

Response parameters

Quick import returns HTTP code and response description.

HTTP code	
200	**Import successful**
400	**Bad request, please verify request parameters**
402	**Tariff account limit reached**
403	**Invalid API key**
404	**No database or subscription resource found**
402	**Account exceeds tariff limits**
409	**More than one profiles matches, inaccurate matching**
435	**Unable to keep field unique - this can be email while matching by other parameters**
500	**Internal server error**

Request example

```
curl "https://example.com/api/integrations/any/profile_import" \
    -d token="abcdefghijklmnqrstuvwxyz" \
    -d db_id=1 \
    -d resource_id=1 \
    -d email="example@example.com" \
    -d _fname="John" \
    -d _lname="Doe"
```

Response example

```
Successfully added 5beeea0ad861602a078a6f34
```

Import profile to RabbitMQ

Description

Imports or updates customer profiles via RabbitMQ message broker queue. Useful for extensive data trans
You will need a queue named “`database_import`”. Therefore it must be available in the same message broker
Import results are processed in another queue – “`database_import_result`”. The default queue time-to-live is 2
`RABBITMQ_X_MESSAGE_TTL : "%seconds%"`
Use one of these parameters to identify and match profiles:

- **email**
- **field_name, field_value**
- **phone**
- **profile_id**

To identify your requests you can use an extra field – “`feedback_id`”. It will appear in server response and in

Message parameters

Parameter	Type	Example	Required	
account_id	**int**	**1**	**Yes**	**Account ID**
data	**JSON object**	`{ "_fname": "John", "_lname": "Doe" }`	**Yes**	**Profile fields data**
db_id	**int**	**1**	**Yes**	**Database ID**
feedback_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**No**	**Request ID**
skip_triggers	**bool**	**true**	**No**	**Skip trigger events. By default – false**
update_only	**bool**	**true**	**No**	**Only update. By default – false**
Profile matching mode				
matching	**string**	**"email" "phone" "profile_id" "custom"**	**No, if matched by email**	**Profile matching mod By default - email**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data fi**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data fi** TIP **Searching by combi**
resource_id	**int**	**24**	**No**	**Resource ID for email**

Message example

```
{
    "account_id": 1,
    "data": {
        "_fname": "John",
        "_lname": "Doe"
    },
    "db_id": 1,
    "email": "example@example.com",
    "feedback_id": "abcdefghijklmnqrstuvwxyz",
    "update_only": true
}
```

Response example

```
{
    "error": 0,
    "error_text": "",
    "feedback_id": "abcdefghijklmnqrstuvwxyz",
    "matching_field": "email",
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

Response parameters

Parameter
error

Parameter
error_text
feedback_id
matching_field
profile_id

Get profile data

Description

Retrieves customer profile data, including database fields and subscriptions.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/get/`

Request parameters

Parameter	Type	Example	Required	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**	**API token**
db_id	**int**	**1**	**Yes**	**Database ID**
Profile matching mode				
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"example@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field na**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field va TIP Searching by combination**
resource_id	**int**	**24**	**No**	**Resource ID for email and p**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "email": "example@example.com"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <email>example@example.com</email>
</xml>
```

Response example

JSON

```
{
    "data": {
        "LOYALTY_SCORE": 500,
        "New_field_7": "00016",
        "_acid": "FBFLR5Z8BxgvNPxBJZgL",
        "_bdate": "1990-07-06T11:15:25.526Z",
        "_browser": "Chrome",
        "_city": "Beijing",
        "_click_exist": false,
        "_click_push_exist": false,
        "_click_sms_exist": false,
        "_confirm_exist": false,
        "_country": "CN",
        "_createtime": "2024-04-08T10:48:05.418Z",
        "_deliv_date": "2024-04-18T10:18:29.809Z",
        "_deliv_exist": true,
        "_deliv_push_exist": false,
        "_deliv_sms_date": "2024-05-06T10:00:05.548Z",
        "_deliv_sms_exist": true,
        "_domaingroup": "other",
        "_emaildomain": "example.com",
        "_fname": "Christopher",
        "_ip": "106.12.21.70",
        "_is_desktop": true,
        "_is_test": true,
        "_lat": 34.7732,
        "_lname": "Lambert",
        "_lon": 113.722,
        "_md5": "1118e73b34d3f9a9030b9e4da6dcfb29",
        "_open_date": "2024-04-18T10:18:29.727Z",
        "_open_exist": true,
        "_open_push_exist": false,
        "_os": "Windows 10",
        "_pixel_open_exist": false,
        "_platform": "Other",
        "_postal_code": "17",
        "_ps_wf": [
            12,
            11,
            10
        ],
        "_ps_wfs": [
            {
                "di": "2024-04-08T10:48:07.093Z",
                "do": "2024-04-13T10:48:22.359Z",
                "et": "",
                "ln": 8,
                "s": 1,
                "wId": 12
            },
            {
                "di": "2024-04-08T19:35:04.15Z",
                "do": "2024-04-12T19:37:19.821Z",
                "et": "",
                "ln": 8,
                "s": 1,
                "wId": 11
            },
            {
                "di": "2024-04-16T14:42:49.063Z",
                "do": "2024-04-17T14:42:52.438Z",
                "et": "",
                "ln": 8,
                "s": 1,
                "wId": 10
            }
        ],
        "_read_date": "2024-04-18T10:18:29.727Z",
        "_read_exist": true,
        "_regdate": "2024-04-08T10:48:05.418Z",
        "_region": "Beijing",
        "_reply_exist": false,
        "_send_date": "2024-04-10T09:32:06.275960407Z",
        "_send_exist": true,
        "_send_push_exist": false,
        "_send_sms_date": "2024-04-12T12:17:05.995339508Z",
        "_send_sms_exist": true,
        "_sex": 1,
        "_status": 0,
        "_subscribetime": "2024-04-08T10:48:05.418Z",
```

```
        "_tz": "Asia/Shanghai",
        "_undeliv_date": "2024-04-17T11:49:42.098Z",
        "_undeliv_exist": true,
        "_updatetime": "2024-04-19T14:25:22.185Z",
        "_xxh": "5ce324d8407a8e2b",
        "_xxm": 8507,
        "db_id": 1,
        "email": "example@example.com",
        "id": "6613cb65dc4594c2934db63f",
        "phones": [
            "+79568150396"
        ],
        "subscriptions": [
            {
                "resource_id": 24,
                "channel": "email",
                "email_data": {
                    "email": "example@example.com",
                    "domain": "example.com",
                    "domain_group": "other",
                    "md5": "1118e73b34d3f9a9030b9e4da6dcfb29"
                },
                "priority": 0,
                "status": "unsubscribed",
                "reg_info": {
                    "date": "2024-04-10T09:28:55.743Z"
                },
                "hash_id": "4a3c5813",
                "is_delete": false
            },
            {
                "resource_id": 25,
                "channel": "sms",
                "phone_data": {
                    "phone": "+79568150396",
                    "md5": "06ec7f059dae6f002d06a2f62a4320d8"
                },
                "priority": 0,
                "status": "subscribed",
                "reg_info": {
                    "date": "2024-04-12T12:16:44.086Z"
                },
                "hash_id": "f67a97c1",
                "is_delete": false
            }
        ],
        "unsub": {
            "c": 115,
            "dt": "2024-04-18T10:18:29.727Z",
            "m": 15
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <CRM_ID>66</CRM_ID>
        <LOYALTY_SCORE>500</LOYALTY_SCORE>
        <_acid>HWbGhUN1jKAdY8iNjdYf</_acid>
        <_browser>Chrome</_browser>
        <_city>St Petersburg</_city>
        <_click_exist>false</_click_exist>
        <_click_push_exist>false</_click_push_exist>
        <_click_sms_exist>false</_click_sms_exist>
        <_confirm_exist>false</_confirm_exist>
        <_country>RU</_country>
        <_createtime>2024-04-08T10:24:09.087Z</_createtime>
        <_deliv_date>2024-05-06T10:00:06.623Z</_deliv_date>
        <_deliv_exist>true</_deliv_exist>
        <_deliv_push_exist>false</_deliv_push_exist>
        <_deliv_sms_date>2024-05-06T10:00:05.274Z</_deliv_sms_date>
        <_deliv_sms_exist>true</_deliv_sms_exist>
        <_domaingroup>other</_domaingroup>
        <_emaildomain>example.com</_emaildomain>
        <_fname>Alina</_fname>
        <_ip>92.53.90.8</_ip>
        <_is_desktop>true</_is_desktop>
        <_is_mobile>true</_is_mobile>
        <_is_test>true</_is_test>
        <_lat>59.939095</_lat>
        <_lname>Lambert</_lname>
        <_lon>30.315868</_lon>
        <_md5>c5278382ce8972b4cd23b447c0e58989</_md5>
        <_open_date>2024-05-06T10:00:06.481Z</_open_date>
        <_open_exist>true</_open_exist>
        <_open_push_exist>false</_open_push_exist>
        <_os>Windows 10</_os>
        <_pixel_open_exist>false</_pixel_open_exist>
        <_platform>Other</_platform>
        <_postal_code>1042</_postal_code>
        <_ps_wf>12</_ps_wf>
        <_ps_wf>11</_ps_wf>
        <_ps_wfs>
            <di>2024-04-08T10:24:10.069Z</di>
            <do>2024-04-19T10:24:43.381Z</do>
            <ln>8</ln>
            <s>1</s>
            <wId>12</wId>
        </_ps_wfs>
        <_ps_wfs>
            <di>2024-04-08T19:35:04.091Z</di>
            <do>2024-04-18T19:37:37.428Z</do>
            <ln>8</ln>
            <s>1</s>
            <wId>11</wId>
        </_ps_wfs>
        <_read_date>2024-05-06T10:00:06.481Z</_read_date>
        <_read_exist>true</_read_exist>
        <_regdate>2024-04-08T10:24:09.087Z</_regdate>
        <_region>Saint Petersburg</_region>
        <_reply_exist>false</_reply_exist>
        <_sbounce_date>2024-04-28T10:00:03.452Z</_sbounce_date>
        <_sbounce_exist>true</_sbounce_exist>
        <_send_date>2024-04-10T09:32:05.531223661Z</_send_date>
        <_send_exist>true</_send_exist>
        <_send_push_exist>false</_send_push_exist>
        <_send_sms_date>2024-04-18T11:14:02.172438444Z</_send_sms_date>
        <_send_sms_exist>true</_send_sms_exist>
        <_status>0</_status>
        <_subscribetime>2024-04-08T10:24:09.087Z</_subscribetime>
        <_tz>Europe/Moscow</_tz>
        <_undeliv_date>2024-04-17T11:49:39.091Z</_undeliv_date>
        <_undeliv_exist>true</_undeliv_exist>
        <_undeliv_sms_date>2024-04-22T10:00:02.62Z</_undeliv_sms_date>
        <_undeliv_sms_exist>true</_undeliv_sms_exist>
        <_updatetime>2024-04-18T12:36:31.7Z</_updatetime>
        <_xxh>507a01069df7046e</_xxh>
        <_xxm>7902</_xxm>
        <db_id>1</db_id>
        <email>example@example.com</email>
        <id>6613c5c94a5d12bd6cc09171</id>
        <phones>+79006574081</phones>
        <subscriptions>
            <channel>email</channel>
```

```
            <email_data>
                <domain>example.com</domain>
                <domain_group>other</domain_group>
                <email>example@example.com</email>
                <md5>c5278382ce8972b4cd23b447c0e58989</md5>
            </email_data>
            <hash_id>fcc69fe5</hash_id>
            <is_delete>false</is_delete>
            <priority>0</priority>
            <reg_info>
                <date>2024-04-10T09:28:55.706Z</date>
            </reg_info>
            <resource_id>24</resource_id>
            <status>subscribed</status>
        </subscriptions>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	
data	**JSON object**
error	**int**
error_text	**string**

Uploading profiles to a file

Description

Retrieves information about all database or segment profiles.
Contains 3 requests.

Step 1. Creating profiles export task

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/get_all`

Request parameters

Parameter	Type	
token	**string**	**“abcde**
source_type	**string**	**“databa “segme**
source_id	**int**	**1**
file_format	**string**	**“csv”**
custom_request	**JSON object**	`{“occupa`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "source_type": "database",
    "source_id": 1,
    "file_format": "csv"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <source_type>database</source_type>
  <source_id>1</source_id>
  <file_format>csv</file_format>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
task_id	**string**
error	**int**
error_text	**string**

Step 2. Task status check

Request URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
task_id	**string**	**“task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "task_id": "ecd371a8-2229-48bb-bc3e-de7d39cd3051"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <task_id>ecd371a8-2229-48bb-bc3e-de7d39cd3051</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": {
            "errors": [],
            "source_id": 1,
            "total": 1343,
            "type": "database",
            "urls": [
                "/api/download/data/4cf435e8-9921-4666-a39e-03ecbf57527e/1_database_export.csv"
            ]
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <progress>100</progress>
        <result>
            <source_id>1</source_id>
            <total>1343</total>
            <type>database</type>
            <urls>/api/download/data/4cf435e8-9921-4666-a39e-03ecbf57527e/1_database_export.csv</urls>
        </result>
        <status>completed</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
status	**string**
progress	**int**
total	**int**
urls	**array**
error	**int**

Parameter	
error_text	**string**

Step 3. Downloading result file

The link to download the file looks like following:

`https://example.com/api/download/6b248de2-33b9-4715-9720-5f7e23f4a296/1_database_export.csv?token=abcdefghijklmnqrstuvwxyz`

- `https://example.com` **- link to the platform panel**
- `api/download/6b248de2-33b9-4715-9720-5f7e23f4a296/1_database_export.csv` **- link from the** `urls` **parameter of the p**
- `?token=abcdefghijklmnqrstuvwxyz` **- user's API token (after the equal sign).**

To download a file with profiles you should either open this link in your browser or send an empty GET req

Suspend all database profiles

> NOTE
> **This API is deprecated. It is described here for information only. Please do not set up new integrations fo**

Description

Changes the global subscription status from `"subscribed"` to `"suspended"` for all database profiles.

Request URL

Method: POST.

`https://example.com/api/v1.0/lists/{list_id}/suspend_all?token={token}`

Note that this is API 1.0 request. For better security use replacement API 1.1 request.

Request parameters

`list_id` – **database ID**
`token` — **API token**

Unsuspend all profiles in a database

> NOTE
> **This API is deprecated. It is described here for information only. Please do not set up new integrations fo**

Description

Changes global subscription status from “suspended” to “subscribed” for all profiles in a database.

Request URL

Method: POST

`https://example.com/api/v1.0/lists/{list_id}/unsuspend_all?token={token}`

Note that this is API 1.0 request. For better security use replacement API 1.1 request.

Request parameters

`list_id` – **database ID**
`token` — **API token**

Delete profile

Description

This request deletes customer profile data, including its events history and relations.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/delete`

Request parameters

Parameter	Type	Example	Required	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**	**API token**
db_id	**int**	**1**	**Yes**	**Database ID**
Profile matching mode				
matching	**string**	**"email" "phone" "profile_id" "custom"**	**No, if matched by email**	**Profile matching mode By default - email**
email	**string**	**"john@example.com"**	**"matching":"email"**	**Email address**
phone	**string**	**"+79000000000"**	**"matching":"phone"**	**Phone number**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**	**Profile ID**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**	**Custom profile data field na**
field_value	**int/string**	**"123420"**	**"matching":"custom"**	**Custom profile data field va** TIP **Searching by combination**
resource_id	**int**	**24**	**No**	**Resource ID for email and p**

CAUTION

Special permission is necessary for deleting profiles. It can be configured in user role, and this role can l

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <matching>email</matching>
  <email>example@example.com</email>
</xml>
```

Response example

JSON

```
{
    "data": {
        "profile_id": "5f33d33ece94481aa9382110"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <profile_id>5f33d33ece94481aa9382110</profile_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
data	**JSON object**
error	**int**
error_text	**string**
profile_id	**string**

Subscription fields functional update

When you use requests to update profile's subscription fields, you can include functions in field values.

A function here is a set of an operator and a value:

```
{
    "action": "set",
    "value": "x",
}
```

`action` **operator can be:**

action	Usage
set	**Set is a default operator for request parameters**
delete	**Deletes a field from a profile. For custom fields only.**
incr	**Increases field value. Integer type fields only**

TIP

You can find a description of the resource subscription fields at this link.

Example request /v1.1/subscriptions/set to update subscription system fields

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "channel": "email",
    "db_id": 1,
    "email": "example@example.com",
    "profile_id": "6527e4e2097c54377ee0e41d",
    "resource_id": 1,
    "status": "subscribe",
     "cats": [
        "category_1",
        "category_2"
    ],
    "custom_fields": {
        "_ad_id": {
            "action":"set",
            "value":"qwerty123"
        },
        "_ad_track":{
            "action":"set",
            "value":true
        },
        "_app_id": {
            "action":"delete"
        },
        "_app_iid": {
            "action":"set",
            "value":"example iid"
        },
        "_app_ver": {
            "action":"set",
            "value": "v1.2.3.4"
        },
        "_browser_name":{
            "action":"set",
            "value":"Chrome"
        },
        "_browser_ver": {
            "action":"set",
            "value":"4.5.6.7.8"
        },
        "_device_model": {
            "action":"set",
            "value":"Device model"
        },
        "_device_name":{
            "action":"set",
            "value":"Device name"
        },
        "_device_type": {
            "action":"set",
            "value":"Mobile"
        },
        "_os": {
            "action":"set",
            "value":"Android"
        },
        "_os_language": {
            "action":"set",
            "value":"ru"
        },
        "_os_tz": {
            "action":"set",
            "value":"Europe/Moscow"
        },
        "_os_ver": {
            "action":"set",
            "value": "v1.2.3.4"
        }
    }
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <channel>email</channel>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <profile_id>66102b984a5d12bd6cc09132</profile_id>
    <resource_id>29</resource_id>
    <status>subscribe</status>
    <cats>category_1</cats>
    <cats>category_2</cats>
    <custom_fields>
        <_ad_id>
            <action>set</action>
            <value>qwerty123</value>
        </_ad_id>
        <_ad_track>
            <action>set</action>
            <value>true</value>
        </_ad_track>
        <_app_id>
            <action>delete</action>
        </_app_id>
        <_app_iid>
            <action>set</action>
            <value>example iid</value>
        </_app_iid>
        <_app_ver>
            <action>set</action>
            <value>v1.2.3.4</value>
        </_app_ver>
        <_browser_name>
            <action>set</action>
            <value>Chrome</value>
        </_browser_name>
        <_browser_ver>
            <action>set</action>
            <value>4.5.6.7.8</value>
        </_browser_ver>
        <_device_model>
            <action>set</action>
            <value>Device model</value>
        </_device_model>
        <_device_name>
            <action>set</action>
            <value>Device name</value>
        </_device_name>
        <_device_type>
            <action>set</action>
            <value>Mobile</value>
        </_device_type>
        <_os>
            <action>set</action>
            <value>Android</value>
        </_os>
        <_os_language>
            <action>set</action>
            <value>ru</value>
        </_os_language>
        <_os_tz>
            <action>set</action>
            <value>Europe/Moscow</value>
        </_os_tz>
        <_os_ver>
            <action>set</action>
            <value>v1.2.3.4</value>
        </_os_ver>
    </custom_fields>
</xml>
```

Example request /v1.1/subscriptions/set to update custom subscription fields

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "channel": "email",
    "db_id": 1,
    "email": "example@example.com",
    "profile_id": "6527e4e2097c54377ee0e41d",
    "resource_id": 1,
    "status": "subscribe",
    "cats": [
        "category_1",
        "category_2"
    ],
    "custom_fields": {
        "custom_str_field": {
            "action": "set",
            "value": "example value"
        },
        "custom_int_field_1": {
            "action": "incr",
            "value": 1000
        },
        "custom_int_field_2": {
            "action": "incr",
            "value": -1000
        },
        "custom_bool_field": {
            "action": "delete"
        }
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <channel>email</channel>
    <db_id>1</db_id>
    <email>example@example.com</email>
    <profile_id>6613c4bf4a5d12bd6cc0916b</profile_id>
    <resource_id>29</resource_id>
    <status>subscribe</status>
    <cats>category_1</cats>
    <cats>category_2</cats>
    <custom_fields>
        <custom_str_field>
            <action>set</action>
            <value>example value</value>
        </custom_str_field>
        <custom_int_field_1>
            <action>incr</action>
            <value>1000</value>
        </custom_int_field_1>
        <custom_int_field_2>
            <action>incr</action>
            <value>-1000</value>
        </custom_int_field_2>
        <custom_bool_field>
            <action>delete</action>
        </custom_bool_field>
    </custom_fields>
</xml>
```

Example request /v1.1/profiles/import to update subscription system fields

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com",
    "data": {
        "_fname": "John",
        "_lname": "Doe",
        "email": "example@example.com",
        "subscriptions": [
            {
                "channel": "push",
                "subscription_id": "9f135a38-aa7e-4bf3-8d75-de971d597cf0",
                "provider": "MR",
                "resource_id": 1,
                "custom_fields": {
                    "_ad_id": {
                        "action": "set",
                        "value": "qwerty123"
                    },
                    "_ad_track": {
                        "action": "set",
                        "value": true
                    },
                    "_app_id": {
                        "action": "set",
                        "value": "123"
                    },
                    "_app_iid": {
                        "action": "set",
                        "value": "123"
                    },
                    "_app_ver": {
                        "action": "set",
                        "value": "v1.2.3.4"
                    },
                    "_browser_name": {
                        "action": "set",
                        "value": "Chrome"
                    },
                    "_browser_ver": {
                        "action": "set",
                        "value": "4.5.6.7.8"
                    },
                    "_device_model": {
                        "action": "set",
                        "value": "Device model"
                    },
                    "_device_name": {
                        "action": "set",
                        "value": "Device name"
                    },
                    "_device_type": {
                        "action": "set",
                        "value": "Mobile"
                    },
                    "_os": {
                        "action": "set",
                        "value": "Android"
                    },
                    "_os_language": {
                        "action": "set",
                        "value": "ru"
                    },
                    "_os_tz": {
                        "action": "set",
                        "value": "Europe/Moscow"
                    },
                    "_os_ver": {
                        "action": "set",
                        "value": "v1.2.3.4"
                    }
                }
            }
        ]
    }
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <matching>email</matching>
    <email>example@example.com</email>
    <data>
        <_fname>Fillippina</_fname>
        <_lname>Doe</_lname>
        <email>example@example.com</email>
        <subscriptions array='true'>
            <channel>push</channel>
            <subscription_id>9f135a38-aa7e-4bf3-8d75-de971d597cf0</subscription_id>
            <provider>MR</provider>
            <resource_id>26</resource_id>
            <custom_fields>
                <_ad_id>
                    <action>set</action>
                    <value>qwerty123</value>
                </_ad_id>
                <_ad_track>
                    <action>set</action>
                    <value>true</value>
                </_ad_track>
                <_app_id>
                    <action>delete</action>
                </_app_id>
                <_app_iid>
                    <action>set</action>
                    <value>example iid</value>
                </_app_iid>
                <_app_ver>
                    <action>set</action>
                    <value>v1.2.3.4</value>
                </_app_ver>
                <_browser_name>
                    <action>set</action>
                    <value>Chrome</value>
                </_browser_name>
                <_browser_ver>
                    <action>set</action>
                    <value>4.5.6.7.8</value>
                </_browser_ver>
                <_device_model>
                    <action>set</action>
                    <value>Device model</value>
                </_device_model>
                <_device_name>
                    <action>set</action>
                    <value>Device name</value>
                </_device_name>
                <_device_type>
                    <action>set</action>
                    <value>Mobile</value>
                </_device_type>
                <_os>
                    <action>set</action>
                    <value>Android</value>
                </_os>
                <_os_language>
                    <action>set</action>
                    <value>ru</value>
                </_os_language>
                <_os_tz>
                    <action>set</action>
                    <value>Europe/Moscow</value>
                </_os_tz>
                <_os_ver>
                    <action>set</action>
                    <value>v1.2.3.4</value>
                </_os_ver>
            </custom_fields>
        </subscriptions>
    </data>
</xml>
```

Example request /v1.1/profiles/import to update custom subscription fields

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "matching": "email",
    "email": "example@example.com",
    "data": {
        "_fname": "John",
        "_lname": "Doe",
        "email": "example@example.com",
        "subscriptions": [
            {
                "channel": "email",
                "channel_sid": "email",
                "email": "example@example.com",
                "resource_id": 1
            },
            {
                "channel": "sms",
                "phone": "+79203214111",
                "resource_id": 1
            },
            {
                "channel": "push",
                "subscription_id": "9f135a38-aa7e-4bf3-8d75-de971d597cf0",
                "provider": "MR",
                "resource_id": 1,
                "custom_fields": {
                    "custom_str_field": {
                        "action": "set",
                        "value": "example value"
                    },
                    "custom_int_field": {
                        "action": "incr",
                        "value": 1000
                    },
                    "custom_bool_field": {
                        "action": "delete"
                    }
                }
            }
        ]
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <matching>email</matching>
    <email>example@example.com</email>
    <data>
        <_fname>Fillippina</_fname>
        <_lname>Doe</_lname>
        <email>example@example.com</email>
        <subscriptions>
            <channel>email</channel>
            <channel_sid>email</channel_sid>
            <email>example@example.com</email>
            <resource_id>1</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>sms</channel>
            <phone>+79203214111</phone>
            <resource_id>1</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>push</channel>
            <subscription_id>9f135a38-aa7e-4bf3-8d75-de971d597cf0</subscription_id>
            <provider>MR</provider>
            <resource_id>1</resource_id>
            <custom_fields>
                <custom_str_field>
                    <action>set</action>
                    <value>example value</value>
                </custom_str_field>
                <custom_int_field>
                    <action>incr</action>
                    <value>1000</value>
                </custom_int_field>
                <custom_bool_field>
                    <action>delete</action>
                </custom_bool_field>
            </custom_fields>
        </subscriptions>
    </data>
</xml>
```

Database fields functional update

When you use requests to update profile fields, you can include functions in field values.
A function here is a set of an operator and a value:

```
{
    "action": "set",
    "value": "x",
}
```

Action operator can be:

action	
set	**Set is a default operator for request parameters**
unset	**Deletes a field from a profile (for all fields except `email`)**
incr	**Increases field value. Integer type fields only**
add	**Adds tags to a profile field**
delete	**Deletes tags from a profile field**
upsert	**Sets the value from the *value* field to the property specified in the *submatch* field**

INFO

For `object` and `arr_object` data types, you must pass the `value` and `submatch` fields.
`submatch` is the path to the property whose value needs to be updated.

Request examples

profiles/update increases one value and decreases the other. The same operator with different value sign

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 24,
    "profile_id": "5a5b64a969d4265b958b9ac3",
    "data": {
        "checks": {
            "action": "incr",
            "value": 1000
        },
        "returns": {
            "action": "incr",
            "value": -1000
        }
    }
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <profile_id>6613c4754a5d12bd6cc0915f</profile_id>
  <data>
    <checks>
      <action>incr</action>
      <value>1000</value>
    </checks>
    <returns>
      <action>incr</action>
      <value>-1000</value>
    </returns>
  </data>
</xml>
```

profiles/update adds tags to profile “category” field:

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "profile_id": "5a5b64a969d4265b958b9ac3",
    "data": {
        "category": {
            "action": "add",
            "value": [
                "cars",
                "cats"
            ]
        }
    }
}
```

XML

```
<xml>
 <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <profile_id>6613c4bf4a5d12bd6cc0916b</profile_id>
  <data>
    <category>
      <action>add</action>
      <value array='true'>cars</value>
      <value>cats</value>
    </category>
  </data>
</xml>
```

An example of a profiles/update request that will update the “x” field in an object:

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "db_id": 1,
  "profile_id": "6613c4bf4a5d12bd6cc0916b",
  "data": {
    "myObject": {
       "action": "upsert",
       "value": "y",
       "submatch": "x"
    }
  }
}
```

XML

```
<xml>
    <token>2fefb577533d4cae919350c450755239</token>
    <db_id>1</db_id>
    <profile_id>66102b984a5d12bd6cc0912f</profile_id>
    <data>
        <myObject>
            <action>upsert</action>
            <value>y</value>
            <submatch>x</submatch>
        </myObject>
    </data>
</xml>
```

profiles/update sets a new value for all elements of the object that contain the specified property (submat

Request

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "profile_id": "66102b984a5d12bd6cc0912f",
    "data": {
        "myObject": {
            "action": "upsert",
            "value": {
                "x": "y",
                "z": 1
            },
            "submatch": "x"
        }
    }
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <profile_id>66102b984a5d12bd6cc09132</profile_id>
  <data>
    <myObject>
      <action>upsert</action>
      <value>
        <x>y</x>
        <z>1</z>
      </value>
      <submatch>x</submatch>
    </myObject>
  </data>
</xml>
```

Object

JSON

```
{
    {
       "A": 1,
       "x": false,
    },
    {
       "B": "test"
    },
    {
       "x": "test",
       "C": true
    }
 }
```

XML

```
<xml>
    <value>
        <A>1</A>
        <x>false</x>
    </value>
    <value>
        <B>test</a>
    </value>
    <value>
        <x>test</x>
        <c>true</c>
    </value>
</xml>
```

Updated object

JSON

```
{
    {
       "A": 1,
       "x": {"x": "y", "z": 1},
    },
    {
       "B": "test"
    },
    {
       "x": {"x": "y", "z": 1},
       "C": true
    }
}
```

XML

```
<xml>
    <value>
        <A>1</A>
        <x>
            <m>y</m>
            <z>1</z>
        </x>
    </value>
    <value>
        <B>test</B>
    </value>
    <value>
        <x>
            <m>y</m>
            <z>1</z>
        </x>
        <c>true</c>
    </value>
</xml>
```

INFO

If fields with type `object` and `object_arr` do not have a property that you want to change, this property is

Merging multiple profiles

Description

Combining multiple profiles into a new profile or existing one.

> **CAUTION**
>
> - **The profiles generated by merging are not imports. Import trigger does not work on them.**
> - **Email: localhost - as an absent field, it is created if necessary and is not copied to another profile.**
> - **Not localhost email is copied like any other field, but in order to avoid any conflicts, it disappears fro**

URL

Method: POST

`https://example.com/api/v1.1/profiles/merge`

Request parameters

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**

data	**JSON object**	“data”:{
		“_fname”: “John”,
		“_lname”: “Doe”
		}
			No

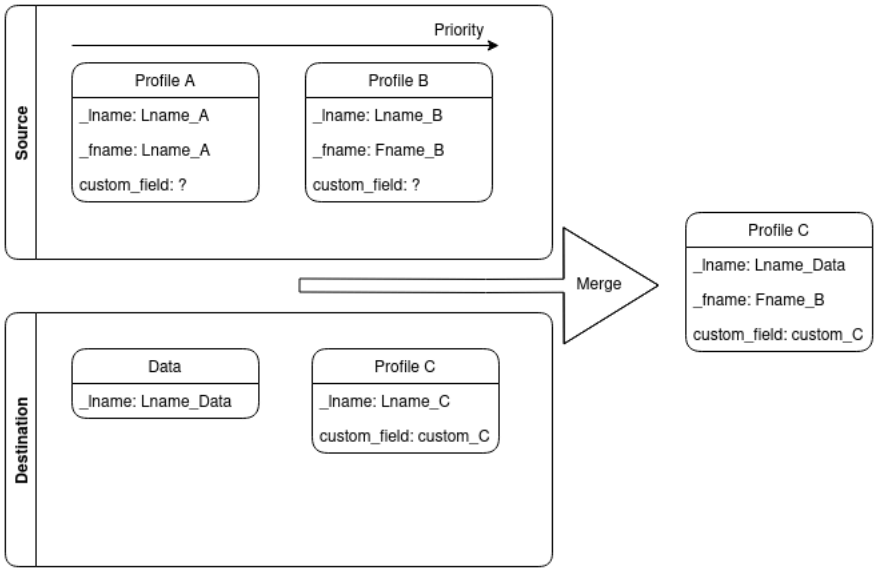
db_id	**int**	**1**	**Yes**
skip_triggers	**bool**	**true**	**No**

source	**array**	“source”: [
		{
		“db_id”: 1,
		“email”: “test1@mail.ru”
		},
		{
		“db_id”: 2,
		“email”: “test2@mail.ru”
		}
		]
			Yes

destination	**JSON object**	“destination”:{
		“db_id”: 3,
		“email”: “test3@mail.ru”
		}
			Yes

matching	**string**	**“email”** **“email_profile”** **“email_sub”** **“phone”** **“phone_sub”** **“profile_id”** **“push_sub”** **“custom”** **“custom_sub”** **“email_phone”** **“email_phone_sub”**	**No, if matched by email from the profile or subscrip**
field_name	**string**	**custom_ID**	**Yes, if** **“matching”: “custom”**
field_value	**int/string**	**custom_string**	**Yes, if** **“matching”: “custom”**
subscriptions	**bool**	**true**	**No**
history	**bool**	**true**	**No**
static_segments	**bool**	**true**	**No**
overwrite	**bool**	**true**	**No**
delete	**bool**	**true**	**No**

Copying profile fields



Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "matching": "custom",
    "field_name": "field_1",
    "source": [
        {
            "db_id": 1,
            "field_value": "1",
            "data": {
                "_fname": "Profile1",
                "_lname": "LnameProfile1"
            }
        },
        {
            "db_id": 1,
            "matching": "email",
            "email": "test2@mail.ru",
            "data": {
                "_fname": "Profile2",
                "_lname": "LnameProfile2"
            }
        }
    ],
    "destination": {
        "db_id": 2,
        "matching": "email",
        "email": "test3@mail.ru",
        "data": {
            "_fname": "Profile3",
            "_lname": "LnameProfile3"
        }
    },
    "subscriptions": true,
    "static_segments": true,
    "history": false
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <matching>custom</matching>
    <field_name>field_1</field_name>
    <source>
        <db_id>1</db_id>
        <field_value>1</field_value>
        <data>
            <_fname>Profile1</_fname>
            <_lname>LnameProfile1</_lname>
        </data>
    </source>
    <source>
        <db_id>1</db_id>
        <matching>email</matching>
        <email>test2@mail.ru</email>
        <data>
            <_fname>Profile2</_fname>
            <_lname>LnameProfile2</_lname>
        </data>
    </source>
    <destination>
        <db_id>2</db_id>
        <matching>email</matching>
        <email>test3@mail.ru</email>
        <data>
            <_fname>Profile2</_fname>
            <_lname>LnameProfile2</_lname>
        </data>
    </destination>
    <subscriptions>true</subscriptions>
    <static_segments>true</static_segments>
    <history>false</history>
</xml>
```

Response example

JSON

```
{
    "destination_id": {
        "db_id": 1,
        "profile_id": "6613cb65dc4594c2934db63b"
    },
    "error": 0,
    "error_text": "",
    "source_ids": [
        {
            "db_id": 1,
            "profile_id": "6613cb65dc4594c2934db63f"
        },
        {
            "db_id": 1,
            "profile_id": "6613cb65dc4594c2934db63b"
        }
    ]
}
```

XML

```
<xml>
    <destination_id>
        <db_id>2</db_id>
        <profile_id>60a7855930b8bc8535b2acfb</profile_id>
    </destination_id>
    <error>0</error>
    <error_text></error_text>
    <source_ids>
        <db_id>1</db_id>
        <profile_id>60a7839330b8bc86a5bc5f62</profile_id>
    </source_ids>
    <source_ids>
        <db_id>1</db_id>
        <profile_id>60a7845230b8bc8535b2acf7</profile_id>
    </source_ids>
</xml>
```

Response parameters

Parameter	Type	
destination_id	**JSON object**	**Co** db pr
source_ids	**JSON object**	**Co** db pr
error	**int**	**Er**
error_text	**string**	**Er**

Unsubscribe profile from resource

Description

Sends the information of the profile unsubscribtion to the platform. The subscription status changes to “U

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/message_unsub`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
rtoken	**string**	**“abcdefghijklmnqrstuvwxyz”**
smid	**string**	**“abcdefghijklmnqrstuvwxyz”**
categories	**array**	**[“cat_to_unsub”]**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "rtoken":"2fHLS9YR9daS-a90b371eecc2952f",
  "smid": "fGmXiWU9Y4NSIpCwl4QG6QXAAAQEQsQQQEQEAABCRA",
  "categories": ["cat_to_unsub"]
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <rtoken>2fHLS9YR9daS-a90b371eecc2952f</rtoken>
  <smid>fGmXiWU9Y4NSIpCwl4QG6QXAAAQEQsQQQEQEAABCRA</smid>
  <categories>cat_to_unsub</categories>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "profile_id": "658997338e8f4032d92fd56d"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <profile_id>658997338e8f4032d92fd56d</profile_id>
</xml>
```

Response parameters

Parameter
error
error_text
profile_id

Profile splitting

Description

Splitting a profile into multiple new or existing profiles.

> **CAUTION**
>
> **The profiles generated by splitting are not imports. Import trigger does not work on them.**

URL

Method: POST

`https://example.com/api/v1.1/profiles/split`

Request parameters

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**

data	**JSON object**	“data”:{
		“_fname”: “John”,
		“_lname”: “Doe”
		}
			No

skip_triggers	**bool**	**true**	**No**

source	**JSON object**	“source”:{
		“db_id”: 1,
		“email”: “test1@mail.ru”
		}
			Yes

destination	**array**	“destination”: [
		{
		“db_id”: 3,
		“email”: “test3@mail.ru”
		}
		]
			Yes

matching	**string**	**“email”** **“email_profile”** **“email_sub”** **“phone”** **“phone_sub”** **“profile_id”** **“push_sub”** **“custom”** **“custom_sub”** **“email_phone”** **“email_phone_sub”**	**No, if matched by email from the profile or su**
field_name	**string**	**custom_ID**	**Yes, if** **“matching”: “custom”**
field_value	**int/string**	**custom_string**	**Yes, if** **“matching”: “custom”**
subscriptions	**bool**	**true**	**No**
history	**bool**	**true**	**No**
static_segments	**bool**	**true**	**No**

subscription_filters	**JSON object**	“subscription_filters”: [
		{
		“resource_id”: 1,
		“channel”: “push”,
		“provider”: “Safari”
		}
		]
			No

overwrite	**bool**	**true**	**No**
delete	**bool**	**true**	**No**

Request example

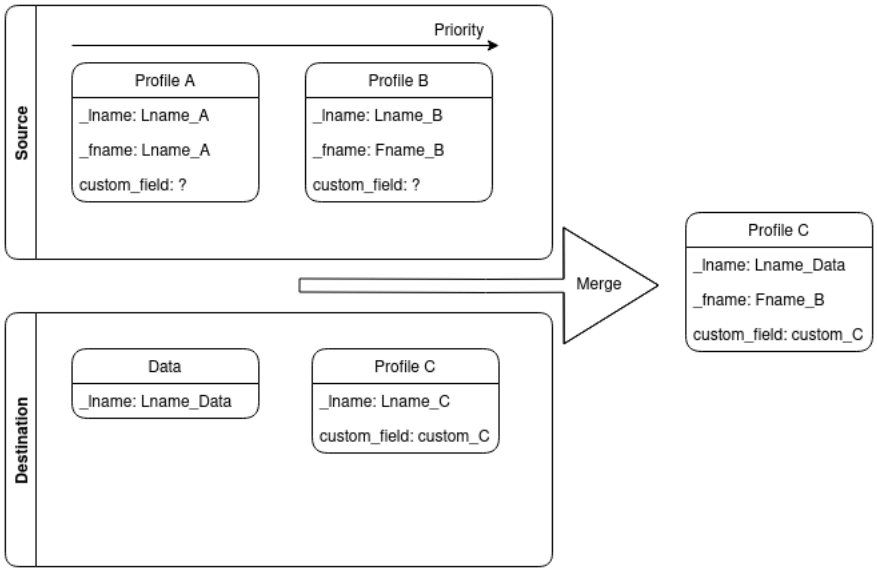
JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "matching": "custom",
    "field_name": "field_1",
    "source": {
        "db_id": 1,
        "field_value": 1,
        "data": {
            "_fname": "Profile1",
            "_lname": "LnameProfile1"
        }
    },
    "destination": [
        {
            "db_id": 2,
            "matching": "email",
            "email": "test2@mail.ru",
            "data": {
                "_fname": "Profile2",
                "_lname": "LnameProfile2"
            }
        },
        {
            "db_id": 2,
            "field_value": 3,
            "data": {
                "_fname": "Profile3",
                "_lname": "Lname3"
            },
            "subscription_filters": [
                {
                    "resource_id": 1,
                    "channel": "push",
                    "provider": "Safari"
                },
                {
                    "resource_id": 2,
                    "channel": "email"
                }
            ]
        }
    ],
    "subscriptions": true,
    "static_segments": true,
    "history": false
}
```


XML

```
<xml>
 <token>abcdefghijklmnqrstuvwxyz</token>
    <matching>custom</matching>
    <field_name>field_1</field_name>
    <delete>true</delete>
    <source>
        <db_id>1</db_id>
        <field_value>1</field_value>
        <data>
            <_fname>Profile1</_fname>
            <_lname>LnameProfile1</_lname>
        </data>
    </source>
    <destination>
        <db_id>2</db_id>
        <matching>email</matching>
        <email>test2@mail.ru</email>
        <data>
            <_fname>Profile2</_fname>
            <_lname>LnameProfile2</_lname>
        </data>
    </destination>
    <destination>
        <db_id>2</db_id>
        <field_value>3</field_value>
        <data>
            <_fname>Profile3</_fname>
            <_lname>LnameProfile3</_lname>
        </data>
        <subscription_filters>
            <resource_id>1</resource_id>
            <channel>push</channel>
            <provider>Safari</provider>
        </subscription_filters>
        <subscription_filters>
            <resource_id>2</resource_id>
            <channel>email</channel>
        </subscription_filters>
    </destination>
    <subscriptions>true</subscriptions>
    <static_segments>true</static_segments>
    <history>false</history>
</xml>
```

Copying profile fields



Response example

JSON

```
{
    "destination_ids": [
        {
            "db_id": 2,
            "profile_id": "60a7b150fa3adb6f1179fe7f"
        },
        {
            "db_id": 2,
            "profile_id": "60a7b150fa3adb6f1179fe80"
        }
    ],
    "error": 0,
    "error_text": "",
    "source_id": {
        "db_id": 1,
        "profile_id": "60a7a45bfa3adb7132dcceaa"
    }
}
```


XML


```
<xml>
    <destination_ids>
        <db_id>1</db_id>
        <profile_id>66432f6cee44bb50a6711634</profile_id>
    </destination_ids>
    <destination_ids>
        <db_id>1</db_id>
        <profile_id>6613cb65dc4594c2934db63f</profile_id>
    </destination_ids>
    <error>0</error>
    <source_id>
        <db_id>1</db_id>
        <profile_id>6643405b41c3a674897dd9cc</profile_id>
    </source_id>
</xml>
```


Response parameters

Parameter	Type
destination_ids	**JSON object**
source_id	**JSON object**
error	**int**
error_text	**string**

Get data for multiple profiles

Description

Gets data for multiple profiles in the database.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/list`

Request parameters

Parameter	Type	Example	Required	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**API token**
format	**string**	**“json”**	**No**	**Response data forn By default – json**
db_id	**int**	**20**	**Yes**	**Database ID**
from_id	**string**	**“0014a2feb331ace2.20200124134127”**	**No**	**The pointer to the**
limit	**int**	**3**	**No**	**Limit on the numbe Default - 1000**

query	**JSON обект**	“query”:{
		“$or”:[
		{
		“_lname”: “Black”
		},
		{
		“_city”: “Bay City”
		}
		]
		}
			No	**Refinement of the
				
				To get a list of field**

fields	**Array of string**	“fields”: [“_id”, “_lname”, “_fname”, “_city”, “subscriptions”],	**No**	**Contains fields for**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "from_id": "65117f62736a0e15d273fae9",
    "limit": 1,
    "query": {
        "$or": [
            {
                "_lname": "Doe"
            },
            {
                "_city": "Moscow"
            }
        ]
    },
    "fields": [
        "_id",
        "_lname",
        "_fname",
        "_city",
        "subscriptions"
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <from_id>65117f62736a0e15d273fae9</from_id>>
    <limit>1</limit>
    <fields>_id</fields>
    <fields>_lname</fields>
    <fields>_fname</fields>
    <fields>_city</fields>
    <fields>subscriptions</fields>
</xml>
```

Response example

JSON


```
{
    "data": [
        {
            "_city": "Moscow",
            "_fname": "Anna",
            "_lname": "Murphy",
            "id": "65117f62736a0e15d273faee",
            "subscriptions": [
                {
                    "resource_id": 5,
                    "channel": "email",
                    "email_data": {
                        "email": "example@example.edu",
                        "domain": "example.edu",
                        "domain_group": "other",
                        "md5": "7dbeb0466131e377e58988a97196d18e"
                    },
                    "priority": 0,
                    "status": "unsubscribed",
                    "reg_info": {
                        "date": "2024-01-09T12:12:18.172Z"
                    },
                    "hash_id": "e70dbabe",
                    "is_delete": false
                },
                {
                    "resource_id": 26,
                    "channel": "push",
                    "push_data": {
                        "subscription_id": "abcdefghijklmnqrstuvwxyz",
                        "provider": "Firefox"
                    },
                    "priority": 0,
                    "status": "subscribed",
                    "reg_info": {
                        "date": "2024-04-18T12:30:44.592Z"
                    },
                    "hash_id": "e70dbabe",
                    "is_delete": false
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": "65117f62736a0e15d273faf4"
}
```


XML

```
<xml>
    <data>
        <_city>Toronto</_city>
        <_fname>Juan</_fname>
        <_lname>Wallace</_lname>
        <id>65117f62736a0e15d273fae9</id>
        <subscriptions>
            <channel>email</channel>
            <email_data>
                <domain>example.com</domain>
                <domain_group>other</domain_group>
                <email>example@example.com</email>
                <md5>18450b1edfef75279fc4f4f60c78d97e</md5>
            </email_data>
            <hash_id>a0cc2672</hash_id>
            <is_delete>false</is_delete>
            <priority>0</priority>
            <reg_info>
                <date>2024-01-09T12:12:18.172Z</date>
            </reg_info>
            <resource_id>5</resource_id>
            <status>hardbounced</status>
        </subscriptions>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>65117f62736a0e15d273faea</next_from_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**string**	**The value must be passed in the next request to get another piece of**
data	**Array of objects**	**Array of received data on profiles**

Add or edit subscription

Description

Adds or edits profile subscriptions. Changes subscription status.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/set`

Request parameters

Parameter	Type	Example	Required
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
format	**string**	**"json"**	**No**
db_id	**int**	**1**	**Yes**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
resource_id	**int**	**1**	**Yes**
status	**string**	**"subscribed"**	**No**
priority	**int**		**No**
custom_fields	**object**	`{ "_browser_name": "Chrome", "_device_type": "web" }`	**No**
cats	**array**	`[ "category_1", "category_2" ]`	**No**
date	**string**	**2018-08-22T00:00:00.000Z**	**No**
ip	**string**	**10.250.13.5**	**No**
url	**string**	**https://example.com/**	**No**
skip_triggers	**bool**	**true**	**No**

Additionally for the Email channel:

Parameters	Type	
channel	**string**	**“ema**
email	**string**	**“johr**

Additionally for the Push channel:

Parameters	Type	
channel	**string**	**“**
provider	**string**	**“**
subscription_id	**string**	**“**

Additionally for the SMS channel:

Parameters	Type
channel	**string**
phone	**string**

Additionally for the Telegram channel:

Parameters	Type	Example
channel	**string**	**“telegram_bot”**
cc_data	**object**	`{ “id”: 1 }`

Additionally for the WhatsApp* channel:

Parameters	Type	Example
channel	**string**	**“whatsapp”**
cc_data	**object**	`{ “phone”: “+79000000000” }`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "db_id": 1,
  "resource_id": 1,
  "channel": "email",
  "email": "example@example.com",
  "priority": 10,
  "profile_id": "abcdefghijklmnqrstuvwxyz",
  "status": "suspended"
}
```


XML


```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <resource_id>29</resource_id>
  <channel>email</channel>
  <email>example@example.com</email>
  <priority>10</priority>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  <status>suspended</status>
</xml>
```


Response example

JSON


```
{
  "data": {
    "action": "subscribed",
    "is_new_channel": false,
    "is_new_contact": false,
    "is_new_subscription": true,
    "is_restored": false,
    "is_safe": true
  },
  "error": 0,
  "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <action>subscribed</action>
    <is_new_channel>false</is_new_channel>
    <is_new_contact>false</is_new_contact>
    <is_new_subscription>true</is_new_subscription>
    <is_restored>false</is_restored>
    <is_safe>true</is_safe>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


CAUTION

Please note that the “Increase priority for new subscriptions” option in the resource works only if a prior change.

Response parameters

Parameter	Type
error	**int**
error_text	**string**
action	**string**
is_safe	**bool**
is_new_subscription	**bool**
is_new_channel	**bool**
is_new_contact	**bool**
is_restored	**bool**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Get all profile subscriptions

Description

Retrieves all subscriptions of a single profile.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/list`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
db_id	**int**	**1**
format	**string**	**“json”**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "resource_id": 29,
            "channel": "email",
            "email": "example@example.com",
            "priority": 10,
            "status": "suspended",
            "custom_fields": {
                "custom_int_field_2": 20000
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <channel>email</channel>
        <custom_fields>
            <custom_int_field_2>20000</custom_int_field_2>
        </custom_fields>
        <email>example@example.com</email>
        <priority>10</priority>
        <resource_id>29</resource_id>
        <status>suspended</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	type	
error	**int**	**Error code**
error_text	**string**	**Error text**
channel	**string**	**Channel type (email, push or sms)**
priority	**int**	**Subscription priority (0 for the highest priority)**
resource_id	**int**	**Subscription resource ID**
status	**string**	**Subscription status**
custom_fields	**object**	**System and custom subscription fields.** **You can get information about subscription fields in a resource using the /v1.1/resou**
cats	**array**	**The resource categories to which the profile is subscribed.**

Additionally for the Email channel:

Parameter	
email	**string**

Additionally for the Push channel:

Parameter
provider
subscription_id

Additionally for the SMS channel:

Parameter	
phone	**string**

Additionally for the Telegram channel:

Parameter	Type	
cc_data	**object**	**Chat ID betwee**

Additionally for the WhatsApp* channel:

Parameter	Type
cc_data	**object**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Get all subscriptions from multiple profiles

Description

Retrieves all subscription entries from several customer profiles.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/list_all`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
resource_id	**int**	**1**
db_id	**int**	**1**
channel	**string**	**“email”**
status	**string**	**“subscribed”**
from_id	**string**	**“abcdefghijklmnqrstuvwxyz”**
limit	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "resource_id": 29,
    "from_id": "abcdefghijklmnqrstuvwxyz",
    "limit": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <resource_id>1</resource_id>
  <from_id>abcdefghijklmnqrstuvwxyz</from_id>
  <limit>1</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "db_id": 1,
            "profile_id": "6643052fee44bb50a671161d",
            "_xxh": "026e82a364dd17e6",
            "hash_id": "fcf06ea1",
            "resource_id": 29,
            "channel": "email",
            "email": "zanitrevitteu-5912@1xp.fr",
            "priority": 0,
            "status": "subscribed",
            "custom_fields": {
                "custom_int_field_2": 20000
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": "cPydNpzRRGLhjKziMHRQ3awnGtyo8hjzn7Jar5mhd"
}
```

XML

```
<xml>
    <data>
        <_xxh>026e82a364dd17e6</_xxh>
        <channel>email</channel>
        <custom_fields>
            <custom_int_field_2>20000</custom_int_field_2>
        </custom_fields>
        <db_id>1</db_id>
        <email>zanitrevitteu-5912@1xp.fr</email>
        <hash_id>fcf06ea1</hash_id>
        <priority>0</priority>
        <profile_id>6643052fee44bb50a671161d</profile_id>
        <resource_id>29</resource_id>
        <status>subscribed</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>cPydNpzRRGLhjKziMHRQ3awnGtyo8hjzn7Jar5mhdm</next_from_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**string**	**The index must be passed in the next request to receive another piece of data, until the result**
resource_id	**int**	**Subscription resource ID**
channel	**string**	**channel type**
custom_fields	**object**	**System and custom subscription fields.** **You can get information about subscription fields in a resource using the /v1.1/resources/fields**
cats	**array**	**The resource categories to which the profile is subscribed.**

Additionally for the Email channel:

Parameter	
email	**string**

Additionally for the Push channel:

Parameter
provider
subscription_id

Additionally for the SMS channel:

Parameters	
phone	**string**

Additionally for the Telegram channel:

Parameter	Type	
cc_data	**object**	**Chat ID betwee**

Additionally for the WhatsApp* channel:

Parameter	Type
cc_data	**object**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Get profile subscription

Description

Retrieves detailed information about one profile subscription.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
db_id	**int**	**1**
format	**string**	**“json”**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**
resource_id	**int**	**1**

Additionally for the Email channel:

Parameter	Type	
channel	**string**	**“email”**
email	**string**	**“john@exa**

Additionally for the Push channel:

Parameters	Type	
channel	**string**	**“**
provider	**string**	**“**
subscription_id	**string**	**“**

Additionally for the SMS channel:

Parameters	Type
channel	**string**
phone	**string**

Additionally for the Telegram channel:

Parameters	Type	
channel	**string**	**“telegram_bot”**
cc_data	**object**	`{ “id”: 1 }`

Additionally for the WhatsApp* channel:

Parameters	Type	
channel	**string**	**“telegram_bot”**
cc_data	**object**	`{ “phone”: “+79000000000” }`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "resource_id": 26,
    "channel": "push",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "provider": "Firefox"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <resource_id>26</resource_id>
  <channel>push</channel>
  <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  <provider>Firefox</provider>
</xml>
```

Response example

JSON

```
{
  "data": {
    "channel": "push",
    "priority": 10,
    "provider": "Firefox",
    "resource_id": 1,
    "status": "subscribed",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "custom_fields": {
      "_device_type": "mob",
      "_os": "Android",
      "custom_field_1": "test value"
    },
    "cats": ["news_category", "transaction_category"]
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <channel>push</channel>
        <priority>0</priority>
        <provider>Firefox</provider>
        <resource_id>26</resource_id>
        <status>subscribed</status>
        <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
channel	**string**	**Channel type (email, push or sms)**
priority	**int**	**Subscription priority (0 is the highest priority)**
resource_id	**int**	**Subscription resource ID**
status	**string**	**Subscription status**
custom_fields	**object**	**System and custom subscription fields.** **You can get information about subscription fields in a resource using the /v1.1/resou**
cats	**array**	**The resource categories to which the profile is subscribed.**

Additionally for the Email channel:

Parameter	
email	**string**

Additionally for the Push channel:

Parameter
provider
subscription_id

Additionally for the SMS channel:

Parameter	
phone	**string**

Additionally for the Telegram channel:

Parameter	Type	
cc_data	**object**	**Chat ID betwee**

Additionally for the WhatsApp* channel:

Parameter	Type
cc_data	**object**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Delete profile subscription

Description

Deletes profile subscription.
Subscription remains but gets `"is_delete : true"` property.

Request URL

Method: POST
`https://example.com/api/v1.1/subscriptions/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**
resource_id	**int**	**1**
db_id	**int**	**1**

Additionally for the Email channel:

Parameters	Type	
channel	**string**	**“ema**
email	**string**	**“johr**

Additionally for the Push channel:

Parameters	Type	
channel	**string**	**“**
provider	**string**	**“**
subscription_id	**string**	**“**

Additionally for the SMS channel:

Parameters	Type
channel	**string**
phone	**string**

Additionally for the Telegram channel:

Parameters	Type	Example
channel	**string**	**“telegram_bot”**
cc_data	**object**	`{ “id”: 1 }`

Additionally for the WhatsApp* channel:

Parameters	Type	Example
channel	**string**	**“whatsapp”**
cc_data	**object**	`{ “phone”: “+79000000000” }`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "resource_id": 26,
    "channel": "push",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "provider": "Firefox"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <resource_id>26</resource_id>
    <channel>push</channel>
    <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <provider>Firefox</provider>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as

Restore deleted subscription

Description

Restores deleted subscription.
Changes `"is_delete"` parameter from `"false"` to `"true"`.

Request URL

Method: POST
`https://example.com/api/v1.1/subscriptions/undelete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**
resource_id	**int**	**1**
db_id	**int**	**1**

Additionally for the Email channel:

Parameters	Type	
channel	**string**	**“ema**
email	**string**	**“john**

Additionally for the Push channel:

Parameters	Type	
channel	**string**	**“**
provider	**string**	**“**
subscription_id	**string**	**“**

Additionally for the SMS channel:

Parameters	Type
channel	**string**
phone	**string**

Additionally for the Telegram channel:

Parameters	Type	Example
channel	**string**	**“telegram_bot”**

cc_data	**object**	{
		“id”: 1
		}

Additionally for the WhatsApp* channel:

Parameters	Type	Example
channel	**string**	**“whatsapp”**

cc_data	**object**	{
		“phone”: “+79000000000”
		}

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "resource_id": 26,
    "channel": "push",
    "subscription_id": "abcdefghijklmnqrstuvwxyz",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "provider": "Firefox"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <resource_id>26</resource_id>
    <channel>push</channel>
    <subscription_id>abcdefghijklmnqrstuvwxyz</subscription_id>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <provider>Firefox</provider>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as

Suspend all subscriptions

Description

Changes profile subscription status from `"subscribed"` to `"suspended"`.
Does not affect any other subscription status.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/suspend_all`

Request example

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
db_id	**int**	**1**
channel	**int**	**“email”**
resource_id	**int**	**1**

If the `"channel"` parameter is passed, all subscriptions within the specified channel will be suspended.
If the `"resource_id"` parameter is passed, all subscriptions within the specified resource will be suspended
If both the `"channel"` and `"resource_id"` parameters are passed, subscriptions for the specified channel with

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "channel": "email"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
  <channel>email</channel>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "updated": 4080
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <updated>1344</updated>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Er**
error_text	**string**	**Er**
updated	**int**	**Nu**

Unsuspend all suspended subscriptions

Description

Changes profile subscription status from `"suspended"` to `"subscribed"`.
Does not affect any other subscription status.

Request URL

Method: POST

`https://example.com/api/v1.1/subscriptions/unsuspend_all`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
db_id	**int**	**1**
channel	**int**	**“email”**
resource_id	**int**	**1**

If `"channel"` parameter is specified, all channel subscriptions for a database will be unsuspended.
If `"resource_id"` parameter is specified, all resource subscriptions for a database will be unsuspended.
If both parameters are specified — resource and channel subscriptions will be unsuspended.
If none of these parameters are specified — all database subscriptions will be unsuspended.

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <db_id>1</db_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "updated": 5428
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <updated>5428</updated>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error cod**
error_text	**string**	**Error text**
updated	**int**	**Number o**

Get profile action history

Description

Retrieves a single profile action history.

Request URL

Method: POST

`https://example.com/api/v1.1/subscribers/history_get`

Request parameters

Parameter	Type	Example	Required
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
format	**string**	**“json”**	**No**
list_id	**Int**	**20**	**Yes**
xxh	**string**	**“0eb51aefd919f90a”**	**No, of you use `matching` for search**
matching	**string**	**“email” “email_profile” “email_sub” “phone” “phone_sub” “profile_id” “push_sub” “custom” “custom_sub” “email_phone” “email_phone_sub”**	**No, of you use `xxh` for search**
date_from	**string**	**“2016-11-25”**	**No**
date_to	**string**	**“2016-11-29”**	**No**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "list_id": 20,
  "xxh": "0eb51aefd919f90a"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <list_id>20</list_id>
  <xxh>0eb51aefd919f90a</xxh>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "xxh": "9202595093f237d4",
            "profile_id": "66f66973096b3b454bbbccec",
            "email": "example@example.com",
            "list_id": 66,
            "action": "send",
            "count": 1,
            "datetime": "2024-09-27T15:52:57.004Z",
            "message_id": 5,
            "campaign_id": 137,
            "event_id": "66f6aaa9da00a9db1c0b2683",
            "smid": "uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5",
            "campaign_name": "New astral mailing",
            "message_name": "testing",
            "subscription": {
                "resource_id": 5,
                "resource_name": "res1",
                "channel": "email",
                "email": "sub1@yandex.ru",
                "priority": 0,
                "status": "subscribed",
                "custom_fields": {
                    "_device_type": "other"
                },
                "cats": []
            },
            "channel": "email",
            "subscriptions": [
                {
                    "resource_id": 5,
                    "resource_name": "res1",
                    "channel": "email",
                    "email": "sub1@yandex.ru",
                    "priority": 0,
                    "status": "subscribed",
                    "custom_fields": {
                        "_device_type": "other"
                    },
                    "cats": []
                },
                {
                    "resource_id": 5,
                    "resource_name": "res1",
                    "channel": "email",
                    "email": "sub2@yandex.ru",
                    "priority": 0,
                    "status": "subscribed",
                    "custom_fields": {
                        "_device_type": "mob"
                    },
                    "cats": []
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
        <item>
            <xxh>9202595093f237d4</xxh>
            <profile_id>66f66973096b3b454bbbccec</profile_id>
            <email>example@example.ru</email>
            <list_id>66</list_id>
            <action>send</action>
            <count>1</count>
            <datetime>2024-09-27T15:52:57.004Z</datetime>
            <message_id>5</message_id>
            <campaign_id>137</campaign_id>
            <event_id>66f6aaa9da00a9db1c0b2683</event_id>
            <smid>uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5</smid>
            <campaign_name>New astral mailing</campaign_name>
            <message_name>testing</message_name>
            <subscription>
                <resource_id>5</resource_id>
                <resource_name>res1</resource_name>
                <channel>email</channel>
                <email>sub1@yandex.ru</email>
                <priority>0</priority>
                <status>subscribed</status>
                <custom_fields>
                    <_device_type>other</_device_type>
                </custom_fields>
            </subscription>
            <channel>email</channel>
            <subscriptions>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub1@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>other</_device_type>
                    </custom_fields>
                </subscription>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub2@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>mob</_device_type>
                    </custom_fields>
                </subscription>
            </subscriptions>
        </item>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	
data	**string**	**An array of received da**
data.xxh	**string**	**Profile xxhash ID**
data.profile_id	**string**	**Profile ID**
data.email	**string**	**Profile email**
data.list_id (db_id)	**int**	**Database ID**
data.action	**string**	**Profile action (More de**
data.count	**int**	**Number of actions tak**
data.datetime	**string**	**Date and time of the a**
data.message_id	**int**	**Message template ID** **Returns 0 if the action**
data.campaign_id	**int**	**Mailing ID** **Returns 0 if the action**
data.event_id	**string**	**Event ID**

Parameter	Type	
data.smid	string	Unique identifier of the Returns an empty strin
data.campaign_name	string	Name of the mailing Returns an empty strin
data.message_name	string	Name of the message Returns an empty strin
data.subscription	string	Object containing infor
data.channel	string	Communication channe Returns an empty strin
data.subscriptions	array of objects	All subscriptions of the
error	int	Error code
error_text	string	Error message

Profile history events (action)

Email channel events	
send	Email sent
deliv	Email delivered
undeliv	Email not delivered
open	Email opened
read	Email read. The message was opened by the user for 8 seconds or more
click	Email clicked. Registered when the user clicks a link in the email. Each link click is registered as a separate click event,
confirm	Click on the subscription confirmation link in the email (confirm-link)
subscribe_email	Subscribed to the Email channel of the resource
unsubscribe_email	Unsubscribed from the Email channel of the resource
reply	Received a reply to the Email
complain	Received a Complain status
hbounce	Received a Hard Bounce status
SMS channel events	
send_sms	SMS message sent to the gateway for subscriber delivery
deliv_sms	SMS message delivered
undeliv_sms	SMS message not delivered
click_sms	Non-unique click on the link in the SMS message
subscribe_sms	Subscribed to the SMS channel of the resource
unsubscribe_sms	Unsubscribed from the SMS channel of the resource
Push channel events	
send_push	Push sent
deliv_push	Push delivered
open_push	Non-unique opening of the push message
click_push	Click on the Push message
undeliv_push	Push message not delivered
subscribe_push	Subscribed to the Push channel of the resource
unsubscribe_push	Unsubscribed from the Push channel of the resource
Telegram Bot Channel Events	
telegram_bot_send	Telegram bot event: Telegram bot sent
telegram_bot_deliv	Telegram bot event: Telegram bot delivered
telegram_bot_undeliv	Telegram bot event: Telegram bot not delivered
telegram_bot_click	Telegram bot event: Telegram bot click
telegram_bot_subscribe	Telegram bot event: Telegram bot subscription
WhatsApp* channel events	
whatsapp_send	Message sent for delivery to the WhatsApp* chat
whatsapp_deliv	Message delivered to the WhatsApp* chat
whatsapp_undeliv	Message not delivered to the WhatsApp* chat
whatsapp_click	Non-unique click on the link in the WhatsApp* chat message
whatsapp_read	Message read by the user
whatsapp_subscribe	WhatsApp* channel event: WhatsApp* subscription
Viber channel events	
viber_send	Message sent for delivery to the Viber chat
viber_deliv	Message delivered to the recipient
viber_undeliv	Message not delivered, rejected by Viber or Devino.Online
viber_click	Click on the link in the message. If the recipient clicks on the same link several times, each click event is recorded.
viber_read	Message read by the user
viber_subscribe	Viber channel event: Viber subscription
Events for all channels	
offence	Message not sent due to sending policy restrictions

suppress	**Message not sent by the mailing because the profile is on the stop list**
Subscriber Profile Import Events	
import_manual	**Profile created manually**
import_api	**Profile created by API import**
import_file	**Profile created by import from a file**
import_form	**Profile created by import from a form**
import_push	**Profile created as a result of push import**
import_popup	**Profile created by import via popup**
Unsubscribe Events	
unsub_api	**Subscriber unsubscribed from emails via API**
unsub_manual	**Unsubscribed manually (global profile status changed to "Unsubscribed")**
Pixel events	
pixel_open	**Goal achieved**
Promo codes events	
promocode_attach	**Promo code attached**
promocode_detach	**Promo code detached**
promocode_activate	**Promo code activated**
Segment events	
segs_add	**Entered a static segment**
segs_remove	**Exited a static segment**
Profile Relationship Events	
rel_attach	**Profile receives a connection with another profile**
rel_detach	**Profile loses a connection with another profile**
rel_strengthen	**Weight of the connection between profiles increases**
Form Events	
form_load	**Form loaded**
form_page_show	**Form page loaded**
form_post	**Form filled and submitted**
form_abandon	**Form abandoned. The profile loads the form page and does not close it for an hour, without submitting the form.**
form_bounce	**Error in filling out the form (attempted re-filling, technical issues, etc.)**

**** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as***

Get multiple profiles action history

Description

Retrieves multiple profiles action history.

Request URL

Method: POST

`https://example.com/api/v1.1/subscribers/history_get_all`

Request parameters

Parameter	Type	Example	Required	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**	**API to**
format	**string**	**“json”**	**No**	**Respo By de**
list_id	**int**	**20**	**Yes**	**Datab**
from_id	**string**	**“005a06ded879ecd7.20170406”**	**No**	**The fi**
limit	**int**	**300**	**No**	**Respo By de**
date_from	**string**	**“2016-11-25”**	**Yes**	**Time YYYY**
date_to	**string**	**“2016-11-29”**	**Yes**	**Time YYYY**
fields	**string array**	**[“CustomID”]**	**No**	**Retur**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "list_id": 4,
    "limit": 3,
    "date_from": "2024-07-15",
    "date_to": "2024-10-09",
    "format": "json",
    "from_id": "",
    "fields": [
        "CustomNumber",
        "CustomEnum"
    ]
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <list_id>4</list_id>
  <limit>3</limit>
  <date_from>2024-07-15</date_from>
  <date_to>2024-10-09</date_to>
  <format>xml</format>
  <from_id></from_id>
  <fields>CustomNumber</fields>
  <fields>CustomEnum</fields>
</xml>
```

Response example

JSON

```
{
"data": [
    {
        "xxh": "9202595093f237d4",
        "profile_id": "66f66973096b3b454bbbccec",
        "email": "example@example.com",
        "list_id": 66,
        "action": "send",
        "count": 1,
        "datetime": "2024-09-27T15:52:57.004Z",
        "message_id": 5,
        "campaign_id": 137,
        "event_id": "66f6aaa9da00a9db1c0b2683",
        "smid": "uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5",
        "campaign_name": "New astral mailing",
        "message_name": "testing",
        "subscription": {
            "resource_id": 5,
            "resource_name": "res1",
            "channel": "email",
            "email": "sub1@yandex.ru",
            "priority": 0,
            "status": "subscribed",
            "custom_fields": {
                "_device_type": "other"
            },
            "cats": []
        },
        "channel": "email",
        "subscriptions": [
            {
                "resource_id": 5,
                "resource_name": "res1",
                "channel": "email",
                "email": "sub1@yandex.ru",
                "priority": 0,
                "status": "subscribed",
                "custom_fields": {
                    "_device_type": "other"
                },
                "cats": []
            },
            {
                "resource_id": 5,
                "resource_name": "res1",
                "channel": "email",
                "email": "sub2@yandex.ru",
                "priority": 0,
                "status": "subscribed",
                "custom_fields": {
                    "_device_type": "mob"
                },
                "cats": []
            }
        ]
    }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
        <item>
            <xxh>9202595093f237d4</xxh>
            <profile_id>66f66973096b3b454bbbccec</profile_id>
            <email>example@example.ru</email>
            <list_id>66</list_id>
            <action>send</action>
            <count>1</count>
            <datetime>2024-09-27T15:52:57.004Z</datetime>
            <message_id>5</message_id>
            <campaign_id>137</campaign_id>
            <event_id>66f6aaa9da00a9db1c0b2683</event_id>
            <smid>uBAhiRBRQhARg7y489hr5F2b2aXMJaztFS7vM7DE1NGJhZWM5</smid>
            <campaign_name>New astral mailing</campaign_name>
            <message_name>testing</message_name>
            <subscription>
                <resource_id>5</resource_id>
                <resource_name>res1</resource_name>
                <channel>email</channel>
                <email>sub1@yandex.ru</email>
                <priority>0</priority>
                <status>subscribed</status>
                <custom_fields>
                    <_device_type>other</_device_type>
                </custom_fields>
            </subscription>
            <channel>email</channel>
            <subscriptions>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub1@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>other</_device_type>
                    </custom_fields>
                </subscription>
                <subscription>
                    <resource_id>5</resource_id>
                    <resource_name>res1</resource_name>
                    <channel>email</channel>
                    <email>sub2@yandex.ru</email>
                    <priority>0</priority>
                    <status>subscribed</status>
                    <custom_fields>
                        <_device_type>mob</_device_type>
                    </custom_fields>
                </subscription>
            </subscriptions>
        </item>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	
data	**string**	**An array of received data regarding events in the histo**
data.xxh	**string**	**Profile xxhash ID**
data.profile_id	**string**	**Profile ID**
data.email	**string**	**Profile email**
data.list_id (db_id)	**int**	**Database ID**
data.action	**string**	**Profile action (More details)**
data.count	**int**	**Number of actions taken in this second**
data.datetime	**string**	**Date and time of the action in RFC 3339 (ISO 8601) for**
data.message_id	**int**	**Message template ID** **Returns 0 if the action does not involve using a templa**
data.campaign_id	**int**	**Mailing ID** **Returns 0 if the action does not involve using a mailing**
data.event_id	**string**	**Event ID**

Parameter	Type	
data.smid	**string**	**Unique identifier of the send within the mailing** **Returns an empty string if the action does not involve**
data.campaign_name	**string**	**Name of the mailing** **Returns an empty string if the action does not involve**
data.message_name	**string**	**Name of the message template** **Returns an empty string if the action does not involve**
data.subscription	**string**	**Object containing information about the subscription u**
data.channel	**string**	**Communication channel in which the event was record** **Returns an empty string if the channel is not involved.**
data.subscriptions	**array of objects**	**All subscriptions of the profile. For CSV format, it will**
data.fields	**object**	**Additional profile fields specified in the fields paramet**
next_from_id	**string**	**The first profile in the next batch of the databas**
error	**int**	**Error code**
error_text	**string**	**Error message**

Profile relations

TIP

Profile relations are used to describe interactions between your customers, or between customers and e

Attach relation

Description

Creates a relation between two profiles. In case a relation already exists an error will be returned.

URL

Method: POST

`https://example.com/api/v1.1/relations/attach`

Request parameters

Parameter	Type	Example	Required
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
rel_id	**int**	**3**	**Yes**
profile_a	**object**	`{"profile_id": "c1wefohoi1312"}`	**Yes**
profile_b	**object**	`{"profile_id": "c1wefohoi13123"}`	**Yes**
weights	**object**	`{"user_like": 2, "donate": 343, "is_online": true}`	**Yes**
Profile matching mode			
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email from profile or subscriptions**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions** **"matching":"email_profile" - profile data only**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    },
    "weights": {
        "user_like": 2,
        "donate": 343,
        "is_online": true
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <rel_id>3</rel_id>
    <profile_a>
        <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
    </profile_a>
    <profile_b>
        <profile_id>5d441038439e7363f3c26539</profile_id>
    </profile_b>
    <weights>
        <user_like>2</user_like>
        <donate>343</donate>
        <is_online>true</is_online>
    </weights>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```

XML

```
<xml>
  <error>0</error>
  <error_text></error_text>
  <profile_a>5c910f6bc484d023b1309bee</profile_a>
  <profile_b>5c910938c484d023b1309bac</profile_b>
  <relation_id>1</relation_id>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
profile_a	**string**
profile_b	**string**
relation_id	**int**

Detach relation

Description

Removes a relation between two profiles.

URL

Method: POST

`https://example.com/api/v1.1/relations/detach`

Request parameters

Parameter	Type	Example	Required
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
rel_id	**int**	**3**	**Yes**
profile_a	**object**	{"profile_id": "c1wefohoi1312"}	**Yes**
profile_b	**object**	{"profile_id": "c1wefohoi13123"}	**Yes**
Profile matching mode			
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email from profile or subscriptions**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions** **"matching":"email_profile" - profile data only**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    }
}
```

XML

```
<xml>
<token>abcdefghijklmnqrstuvwxyz</token>
  <rel_id>3</rel_id>
  <profile_a>
    <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
  </profile_a>
  <profile_b>
    <profile_id>5d441038439e7363f3c26539</profile_id>
  </profile_b>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```


XML


```
<xml>
    <error>0</error>
    <error_text></error_text>
    <profile_a>5c910f6bc484d023b1309bee</profile_a>
    <profile_b>5c910938c484d023b1309bac</profile_b>
    <relation_id>1</relation_id>
</xml>
```


Response parameters

Parameter	
error	**int**
error_text	**string**
profile_a	**string**
profile_b	**string**
relation_id	**int**

Modify relation properties

Description

Modifies (increases or decreases) relation properties' values by those you send in `weights` parameter of th

> **NOTE**
>
> **If no relation is found between the two profiles, a new relation will be created.**

> **TIP**
>
> **If you need to replace properties' values instead of increasing/decreasing, use Overwrite relation proper**

URL

Method: POST.

`https://example.com/api/v1.1/relations/strengthen`

Request parameters

Parameter	Type	Example	Required
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
rel_id	**int**	**3**	**Yes**
profile_a	**object**	{"profile_id": "c1wefohoi1312"}	**Yes**
profile_b	**object**	{"profile_id": "c1wefohoi13123"}	**Yes**
weights	**object**	{"user_like": 2, "donate": 343, "is_online": true}	**Yes**
Profile matching mode			
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email from profile or subscriptions**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions** **"matching":"email_profile" - profile data only**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    },
    "weights": {
        "user_like": 2,
        "donate": 343,
        "is_online": true
    }
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <rel_id>3</rel_id>
  <profile_a>
    <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
  </profile_a>
  <profile_b>
    <profile_id>5d441038439e7363f3c26539</profile_id>
  </profile_b>
  <weights>
    <user_like>2</user_like>
    <donate>343</donate>
    <is_online>true</is_online>
  </weights>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```

XML

```
<xml>
  <error>0</error>
  <error_text></error_text>
  <profile_a>5c910f6bc484d023b1309bee</profile_a>
  <profile_b>5c910938c484d023b1309bac</profile_b>
  <relation_id>1</relation_id>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
profile_a	**string**
profile_b	**string**
relation_id	**int**

Overwrite relation properties

Description

Replaces relation properties' values with the ones you send with this request.

> **NOTE**
>
> **In case no relation between the specified profiles exists — a new relation will be created.**

> **TIP**
>
> **If you need to increase or decrease relation properties, use Modify relation properties endpoint.**

URL

Method: POST.

`https://example.com/api/v1.1/relations/edit`

Request parameters

Parameter	Type	Example	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
rel_id	**int**	**3**	**Yes**
profile_a	**object**	{"profile_id": "c1wefohoi1312"}	**Yes**
profile_b	**object**	{"profile_id": "c1wefohoi13123"}	**Yes**
weights	**object**	{"user_like": 2, "donate": 343, "is_online": true}	**Yes**
Profile matching mode			
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email from pro**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile** **"matching":"email_profile" - profi**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "rel_id": 3,
    "profile_a": {
        "profile_id": "5d63dbdb439e7306b596c5d0"
    },
    "profile_b": {
        "profile_id": "5d441038439e7363f3c26539"
    },
    "weights": {
        "user_like": 2,
        "donate": 343,
        "is_online": true
    }
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <rel_id>3</rel_id>
  <profile_a>
    <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
  </profile_a>
  <profile_b>
    <profile_id>5d441038439e7363f3c26539</profile_id>
  </profile_b>
  <weights>
    <user_like>2</user_like>
    <donate>343</donate>
    <is_online>true</is_online>
  </weights>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_a": "5c910f6bc484d023b1309bee",
    "profile_b": "5c910938c484d023b1309bac",
    "relation_id": 1
}
```

XML

```
<xml>
<error>0</error>
  <error_text></error_text>
  <profile_a>5c910f6bc484d023b1309bee</profile_a>
  <profile_b>5c910938c484d023b1309bac</profile_b>
  <relation_id>1</relation_id>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
profile_a	**string**
profile_b	**string**
relation_id	**int**

Get profile relations info

Description

Get information about a Database A profile's relations (initiated by Database B profiles) and their properti

URL

Method: POST.

`https://example.com/api/v1.1/relations/get_counter`

Request parameters

Parameter	Type	Example	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
rel_id	**int**	**3**	**Yes**
Profile matching mode			
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscriptions "matching":"email_profile" - profile data only**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "rel_id": 3,
    "profile_id": "5d63dbdb439e7306b596c5d0"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <rel_id>3</rel_id>
  <profile_id>5d63dbdb439e7306b596c5d0</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "profile_id": "669e6968ebf4ac52b623a997",
        "profile_xxh": "5af1023c84f66195",
        "props": {}
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <profile_id>669e6968ebf4ac52b623a997</profile_id>
        <profile_xxh>5af1023c84f66195</profile_xxh>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text
data
data.profile_id
data.profile_xxh
data.props

Relation properties: numeric

Parameter	Type
top	**object**
sum	**int/float**

Get profile relations info

Description

The request brings a list of all existing links for the specified profile from the database A, including identif

URL

Method: POST.

`https://example.com/api/v1.1/relations/get`

Request parameters

Parameter	Type	Example	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
rel_id	**int**	**3**	**Yes**
profile_id	**string**	**"669e6968ebf4ac52b623a997"**	**Yes, if no other search type is specified**
Profile matching mode			
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptions**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscripti "matching":"email_profile" - profile data only**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**

Request example

JSON

```
{
    "token": "8a28168486844c8bae2673a51d4433ff",
    "rel_id": 2,
    "profile_id": "669e6968ebf4ac52b623a997"
}
```

XML

```
<xml>
  <token>8a28168486844c8bae2673a51d4433ff</token>
  <rel_id>2</rel_id>
  <profile_id>669e6968ebf4ac52b623a997</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "profile_id_a": "669e6968ebf4ac52b623a997",
            "xxh_a": "5af1023c84f66195",
            "profile_id_b": "6751b8f1c5e3db9bcb75befa",
            "xxh_b": "879cebbb0e6b6723",
            "db_b": 9,
            "props": {
                "exr": {
                    "value": 11323123
                }
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 1
}
```

XML

```
<xml>
    <data>
        <profile_id_a>669e6968ebf4ac52b623a997</profile_id_a>
            <xxh_a>5af1023c84f66195</xxh_a>
            <profile_id_b>6751b8f1c5e3db9bcb75befa</profile_id_b>
            <xxh_b>879cebbb0e6b6723</xxh_b>
            <db_b>9</db_b>
            <props>
                <exr>
                    <value>11323123</value>
                </exr>
            </props>
        </relation>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>1</total_count>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error message**
data	**object**	**Information about pr**
data.profile_id_a	**string**	**ID of profile A (the pr**
data.xxh_a	**string**	**Hashed ID of profile**
data.profile_id_b	**string**	**ID of profile B (the pr**
data.xxh_b	**string**	**Hashed ID of profile**
data.db_b	**int**	**ID of the database co**
data.props	**object**	**Connection propertie**
data.total_count	**int**	**Number of relations**

Get database statistics

Description

Get the statistics of subscriptions to a specific resource within the base.

URL

Method: POST

`https://example.com/api/v1.1/databases/get_counters`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
resource_id	**int**	**1**
format	**int**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 5,
    "resource_id": 24,
    "format": "json"
}
```

XML

```
 <xml>
     <token>abcdefghijklmnqrstuvwxyz</token>
     <id>5</id>
     <resource_id>24</resource_id>
     <format>json</format>
 </xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "status": {
                "all": 1336,
                "subscribed": 925,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 411,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 925,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "sms",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "push",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <channel>email</channel>
        <status>
            <all>1336</all>
            <subscribed>925</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>411</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>925</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>sms</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>push</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
data	**array of objects**
error	**int**
error_text	**string**

Update statistics on database

Description

Update subscription statistics of profiles in the database.
Returns a response with task ID without waiting for execution.

URL

Method: POST

`https://example.com/api/v1.1/databases/refresh_counters`

Request parameters

```
    <tr>
        <td>format</td>
        <td>JSON object </td>
        <td>"json"</td>
        <td>No</td>
        <td> Response data format<br/>By default – json </td>
    </tr>
    <tr>
        <td>feedback</td>
        <td>JSON object </td>
        <td>
"feedback": {
    "type": "JSON",
    "url": "http://example.com/receiver"
}
            </td>
        <td>No</td>
        <td>Parameter defining the method of sending data about statistics update. <br/> Parameter "type" - the format or method used to send t
    </tr>
</tbody>
```

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**1**
resource_id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 5,
    "resource_id": 24,
    "format": "json",
    "feedback": {
        "type": "JSON",
        "url": "http://example.com/receiver"
    }
}
```

XML

```
<xml>
     <token>abcdefghijklmnqrstuvwxyz</token>
     <id>5</id>
     <resource_id>24</resource_id>
     <format>json</format>
     <feedback>
         <type>JSON</type>
         <url>http://example.com/receiver</url>
     </feedback>
 </xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "6908adeb-346e-4f5e-9f68-df062bf795cf"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
     <data>
         <task_id>6908adeb-346e-4f5e-9f68-df062bf795cf</task_id>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
 </xml>
```

Response parameters

Parameter	Type
data	**JSON objects**
error	**int**
error_text	**string**

Checking the task status

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
task_id	**string**	**“518f4997-ec43-4df5-bfd0-093aaacf679f”**

Request example

JSON

```
{
    "task_id": "930bfdb0-a5a8-4207-a77d-6b989bdea326",
    "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
     <task_id>930bfdb0-a5a8-4207-a77d-6b989bdea326</task_id>
     <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": [
            {
                "channels": [
                    {
                        "channel": "email",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 411,
                            "invalid": 0,
                            "subscribed": 925,
                            "subscribed_sendable": 925,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "push",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "sms",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    }
                ],
                "in_progress": false,
                "last_count_time": "counted at 2024-04-17T21:36:01Z in 82.746959ms",
                "refresh_task_id": "",
                "resource_id": 24,
                "total_count": 1336,
                "updated": "2024-04-17T21:36:01.449Z"
            }
        ]
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <channels>
                <channel>email</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>411</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>925</subscribed>
                    <subscribed_sendable>925</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>push</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>sms</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <in_progress>false</in_progress>
            <last_count_time>counted at 2024-04-17T21:36:01Z in 82.746959ms</last_count_time>
            <refresh_task_id></refresh_task_id>
            <resource_id>24</resource_id>
            <total_count>1336</total_count>
            <updated>2024-04-17T21:36:01.449Z</updated>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Get database list

Description

Retrieves a list of databases.

Request URL

Method: POST

`https://example.com/api/v1.1/databases/list/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**int**	**“json”**
from_id	**int**	**1**
limit	**int**	**1**
ui_tags	**[]String**	**[“tag1”, “tag2”]**
groups	**[]String**	**[“group1”, “group2”]**
groups_id	**[]int**	**[0, 1]**

Request example

JSON

```
{
  "limit": 2,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <limit>2</limit>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 1,
      "name": "Database #1",
      "ui_tags": [],
      "Groups": [
        {
          "Id": 0,
          "Name": "main"
        }
      ]
    },
    {
      "id": 2,
      "name": "Database #2",
      "ui_tags": ["tag1", "tag2"],
      "Groups": [
        {
          "Id": 1,
          "Name": "subaccount"
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "next_from_id": 6,
  "total_count": 4
}
```


XML


```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <id>1</id>
    <name>Database #1</name>
    <Groups>
      <Id>0</Id>
      <Name>main</Name>
    </Groups>
  </data>
  <data>
    <id>2</id>
    <name>Database #2</name>
    <ui_tags>tag1</ui_tags>
    <ui_tags>tag2</ui_tags>
    <Groups>
      <Id>1</Id>
      <Name>subaccount</Name>
    </Groups>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <next_from_id>6</next_from_id>
  <total_count>4</total_count>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result**
total_count	**int**	**Total database count**
id	**int**	**Database ID**
name	**string**	**Database name**
ui_tags	**[]String**	**List of tags**
groups	**[]String**	**List of groups**

Get database information

Description

Retrieves a database information.

Request URL

Method: POST

`https://example.com/api/v1.1/databases/get/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
format	**int**	**“json”**

Request example

JSON

```
{
  "id": 1,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": {
    "id": 1,
    "name": "B2C customers",
    "suppress_id": 1,
    "description": "May contain info"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <id>1</id>
    <name>B2C customers</name>
    <suppress_id>1</suppress_id>
    <description>May contain info</description>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**

Parameter	Type
suppress_id	**int**
description	**string**

Get database fields

Description

Retrieves database fields information. The types of fields and allowed values are described in this article.

Request URL

Method: POST

`https://example.com/api/v1.1/databases/fields_get/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
format	**int**	**“json”**

Request example

JSON

```
{
  "id": 1,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "category": "system",
      "title": "Id",
      "name": "_id",
      "required": true,
      "submanager": false,
      "type": "unicode",
      "enum_values": null
    },
    {
      "category": "default",
      "title": "FirstName",
      "name": "_fname",
      "required": false,
      "submanager": false,
      "type": "unicode",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_String",
      "name": "Custom_String",
      "required": false,
      "submanager": false,
      "type": "unicode",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Int",
      "name": "Custom_Int",
      "required": false,
      "submanager": false,
      "type": "int",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Bool",
      "name": "Custom_Bool",
      "required": false,
      "submanager": false,
      "type": "bool",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Date",
      "name": "Custom_Date",
      "required": false,
      "submanager": false,
      "type": "date",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Tags",
      "name": "Custom_Tags",
      "required": false,
      "submanager": false,
      "type": "tags",
      "enum_values": null
    },
    {
      "category": "custom",
      "title": "Custom_Enum",
      "name": "Custom_Enum",
      "required": false,
      "submanager": false,
      "type": "enum",
      "enum_values": [
        {
          "name": "GMail",
          "value": 2
        },
        {
          "name": "HotMail",
          "value": 3
```

```
        },
        {
          "name": "Yahoo!",
          "value": 1
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <category>system</category>
    <title>Id</title>
    <name>_id</name>
    <required>true</required>
    <submanager>false</submanager>
    <type>unicode</type>
  </data>
  <data>
    <category>default</category>
    <title>FirstName</title>
    <name>_fname</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>unicode</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_String</title>
    <name>Custom_String</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>unicode</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Int</title>
    <name>Custom_Int</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>int</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Bool</title>
    <name>Custom_Bool</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>bool</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Date</title>
    <name>Custom_Date</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>date</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Tags</title>
    <name>Custom_Tags</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>tags</type>
  </data>
  <data>
    <category>custom</category>
    <title>Custom_Enum</title>
    <name>Custom_Enum</name>
    <required>false</required>
    <submanager>false</submanager>
    <type>enum</type>
    <enum_values>
      <name>GMail</name>
      <value>2</value>
    </enum_values>
    <enum_values>
      <name>HotMail</name>
      <value>3</value>
    </enum_values>
    <enum_values>
      <name>Yahoo!</name>
      <value>1</value>
    </enum_values>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
```

```
</xml>
```

Response parameters

Parameter	Typ
error	**int**
error_text	**string**
category	**string**
title	**string**
name	**string**
required	**bool**
submanager	**bool**
type	**string**
enum_values	

Database wipe

Description

Wipes all data from a database:

1. All customer or seed profiles.
2. All associated data except statistics.

To use this request you need to reconfigure `main.json` file in `config` directory:

```
{
    "ALLOW_DATA_WIPE": true
}
```

Request URL

Method: POST

`https://example.com/api/v1.1/databases/wipe`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
format	**int**	**“json”**

Request example

JSON

```
{
  "id": 1,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <id>1</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": {
    "count_action": 0,
    "count_profile": 0
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <count_action>0</count_action>
    <count_profile>0</count_profile>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
data
data.count_action

Parameter
data.count_profile
error
error_text

Get resource statistics

Description

Get the statistics of subscriptions on resource.

URL

Method: POST

`https://example.com/api/v1.1/resources/get_counters`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
list_id	**int**	**1**
id	**int**	**1**
format	**JSON object**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "list_id": 24,
    "id": 5,
    "format": "json"
}
```

XML

```
 <xml>
     <token>abcdefghijklmnqrstuvwxyz</token>
     <list_id>24</list_id>
     <id>5</id>
     <format>json</format>
 </xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "status": {
                "all": 1336,
                "subscribed": 925,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 411,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 925,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "sms",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "push",
            "status": {
                "all": 1336,
                "subscribed": 0,
                "unsubscribed": 0,
                "complained": 0,
                "hardbounced": 0,
                "unconfirmed": 0,
                "suspended": 0,
                "invalid": 0,
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <channel>email</channel>
        <status>
            <all>1336</all>
            <subscribed>925</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>411</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>925</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>sms</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <data>
        <channel>push</channel>
        <status>
            <all>1336</all>
            <subscribed>0</subscribed>
            <unsubscribed>0</unsubscribed>
            <complained>0</complained>
            <hardbounced>0</hardbounced>
            <unconfirmed>0</unconfirmed>
            <suspended>0</suspended>
            <invalid>0</invalid>
            <subscribed_sendable>0</subscribed_sendable>
            <unconfirmed_sendable>0</unconfirmed_sendable>
        </status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
data	**array of objects**
error	**int**
error_text	**string**

Update statistics on resource

Description

Update subscription statistics of resource.
Returns a response with task ID without waiting for execution.

URL

Method: POST

`https://example.com/api/v1.1/resources/refresh_counters`

Request

Parameter	Type	Exam
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**1**
list_id	**int**	**1**
format	**JSON object**	**"json"**
feedback	**JSON object**	**```json "feedback": { "type": "JSON", "url": "http://example.com/receiver" } ```**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 24,
    "list_id": 1,
    "format": "json",
    "feedback": {
        "type": "JSON",
        "url": "http://example.com/receiver"
    }
}
```

XML

```
<xml>
     <token>abcdefghijklmnqrstuvwxyz</token>
     <id>24</id>
     <list_id>1</list_id>
     <format>json</format>
     <feedback>
         <type>JSON</type>
         <url>http://example.com/receiver</url>
     </feedback>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "c9c3ec48-1bef-4969-9c1a-df39f0c179bb"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
     <data>
         <task_id>c9c3ec48-1bef-4969-9c1a-df39f0c179bb</task_id>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
data	**JSON objects**
error	**int**
error_text	**string**

Checking the task status

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
task_id	**string**	**“518f4997-ec43-4df5-bfd0-093aaacf679f”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "task_id": "c9c3ec48-1bef-4969-9c1a-df39f0c179bb"
}
```

XML

```
<xml>
     <token>abcdefghijklmnqrstuvwxyz</token>
     <task_id>c9c3ec48-1bef-4969-9c1a-df39f0c179bb</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": [
            {
                "channels": [
                    {
                        "channel": "sms",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "push",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 0,
                            "invalid": 0,
                            "subscribed": 0,
                            "subscribed_sendable": 0,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 0
                        }
                    },
                    {
                        "channel": "email",
                        "status": {
                            "all": 1336,
                            "complained": 0,
                            "hardbounced": 434,
                            "invalid": 0,
                            "subscribed": 842,
                            "subscribed_sendable": 842,
                            "suspended": 0,
                            "unconfirmed": 0,
                            "unconfirmed_sendable": 0,
                            "unsubscribed": 60
                        }
                    }
                ],
                "in_progress": false,
                "last_count_time": "counted at 2024-04-19T13:31:08Z in 231.374251ms",
                "list_id": 1,
                "refresh_task_id": "",
                "updated": "2024-04-19T13:31:08.927Z"
            }
        ]
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <channels>
                <channel>sms</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>push</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>0</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>0</subscribed>
                    <subscribed_sendable>0</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>0</unsubscribed>
                </status>
            </channels>
            <channels>
                <channel>email</channel>
                <status>
                    <all>1336</all>
                    <complained>0</complained>
                    <hardbounced>434</hardbounced>
                    <invalid>0</invalid>
                    <subscribed>842</subscribed>
                    <subscribed_sendable>842</subscribed_sendable>
                    <suspended>0</suspended>
                    <unconfirmed>0</unconfirmed>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                    <unsubscribed>60</unsubscribed>
                </status>
            </channels>
            <in_progress>false</in_progress>
            <last_count_time>counted at 2024-04-19T13:31:08Z in 231.374251ms</last_count_time>
            <list_id>1</list_id>
            <refresh_task_id></refresh_task_id>
            <updated>2024-04-19T13:31:08.927Z</updated>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Get resources list

Description

Retrieves all created resources list.

Request URL

Method: POST

`https://example.com/api/v1.1/resources/list`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**1**
groups	**[]String**	`[ “group1”, “group2” ]`
groups_id	**[]int**	`[ 0, 1 ]`

Request example

JSON

```
{
  "from_id": 1,
  "limit": 1,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <from_id>1</from_id>
  <limit>1</limit>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 1,
      "name": "Resource",
      "groups": [
        {
          "id": 1,
          "name": "subaccount"
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "next_from_id": 4,
  "total_count": 1
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <id>1</id>
    <name>Resource</name>
    <groups>
      <id>1</id>
      <name>subaccount</name>
    </groups>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <next_from_id>4</next_from_id>
  <total_count>1</total_count>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result**
id	**int**	**Resource ID**
name	**string**	**Resource name**
total_count	**int**	**Total resources count**
groups	**[]String**	**List of groups**

Get resource information

Description

Retrieves detailed information about a single resource, including channels and associated databases.

Request URL

Method: POST

`https://example.com/api/v1.1/resources/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
  "id": 1,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": {
    "channels": ["email", "push"],
    "dbs_ids": [1],
    "id": 1,
    "name": "Resource",
    "trk_domain": "tracking.example.com",
    "url": "https://blog.example:8080"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <channels>email</channels>
    <channels>push</channels>
    <dbs_ids>1</dbs_ids>
    <id>1</id>
    <name>Resource</name>
    <trk_domain>tracking.example.com</trk_domain>
    <url>https://blog.example:8080</url>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
channel	**JSON array**
dbs_ids	**JSON array**
id	**int**
name	**string**
trk_domain	**string**
url	**string**

Get resource subscription fields

Description

Get information about system and additional subscription fields.

Request URL

Method: POST

`https://example.com/api/v1.1/resources/fields_get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
channel	**string**	**“email”** **“sms”** **“push”** **“telegram_bot”** **“whatsapp”**

Request example

JSON

```
{
  "id": 1,
  "channel": "email",
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <id>1</id>
  <channel>email</channel>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "category": "custom",
      "title": "New_field_1",
      "name": "api_new_field_1",
      "required": false,
      "type": "int"
    },
    {
      "category": "system",
      "title": "Device type",
      "name": "_device_type",
      "required": false,
      "type": "unicode",
      "dictionary": "devices"
    },
    {
      "category": "system",
      "title": "Device model",
      "name": "_device_model",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Device name",
      "name": "_device_name",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Platform",
      "name": "_os",
      "required": false,
      "type": "unicode",
      "dictionary": "oses"
    },
    {
      "category": "system",
      "title": "Platform version",
      "name": "_os_ver",
      "required": false,
      "type": "version"
    },
    {
      "category": "system",
      "title": "Platform time zone",
      "name": "_os_tz",
      "required": false,
      "type": "unicode",
      "dictionary": "tz"
    },
    {
      "category": "system",
      "title": "Platform language",
      "name": "_os_language",
      "required": false,
      "type": "unicode",
      "dictionary": "languages"
    },
    {
      "category": "system",
      "title": "Ad tracking allowed",
      "name": "_ad_track",
      "required": false,
      "type": "bool"
    },
    {
      "category": "system",
      "title": "Ad identifier",
      "name": "_ad_id",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Application ID",
      "name": "_app_id",
```

```
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Application installation ID",
      "name": "_app_iid",
      "required": false,
      "type": "unicode"
    },
    {
      "category": "system",
      "title": "Application version",
      "name": "_app_ver",
      "required": false,
      "type": "version"
    },
    {
      "category": "system",
      "title": "Browser",
      "name": "_browser_name",
      "required": false,
      "type": "unicode",
      "dictionary": "browsers"
    },
    {
      "category": "system",
      "title": "Browser version",
      "name": "_browser_ver",
      "required": false,
      "type": "version"
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<xml>
  <data>
    <category>custom</category>
    <title>New_field_1</title>
    <name>api_new_field_1</name>
    <required>false</required>
    <type>int</type>
  </data>
  <data>
    <category>system</category>
    <title>Device type</title>
    <name>_device_type</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>devices</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Device model</title>
    <name>_device_model</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Device name</title>
    <name>_device_name</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Platform</title>
    <name>_os</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>oses</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Platform version</title>
    <name>_os_ver</name>
    <required>false</required>
    <type>version</type>
  </data>
  <data>
    <category>system</category>
    <title>Platform time zone</title>
    <name>_os_tz</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>tz</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Platform language</title>
    <name>_os_language</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>languages</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Ad tracking allowed</title>
    <name>_ad_track</name>
    <required>false</required>
    <type>bool</type>
  </data>
  <data>
    <category>system</category>
    <title>Ad identifier</title>
    <name>_ad_id</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Application ID</title>
    <name>_app_id</name>
```

```
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Application installation ID</title>
    <name>_app_iid</name>
    <required>false</required>
    <type>unicode</type>
  </data>
  <data>
    <category>system</category>
    <title>Application version</title>
    <name>_app_ver</name>
    <required>false</required>
    <type>version</type>
  </data>
  <data>
    <category>system</category>
    <title>Browser</title>
    <name>_browser_name</name>
    <required>false</required>
    <type>unicode</type>
    <dictionary>browsers</dictionary>
  </data>
  <data>
    <category>system</category>
    <title>Browser version</title>
    <name>_browser_ver</name>
    <required>false</required>
    <type>version</type>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
category	**string**	**Subscription field category** `"system","custom"`
title	**string**	**UI field name**
name	**string**	**API field name**
required	**bool**	**Required flag (true)**
type	**string**	**Field type**
default_value	*	**Field default value if it exists**

dictionary	**string**	**Field name used in the request /v1.1/dictionary (a request to obtain the values of system fields that are displ**
		
		`"browsers"`
		`"devices"`
		`"tz"`
		`"oses"`
		`"languages"`

Subscription system fields

Parameter	Type	Example	
_device_type	**string**	**“mob”**	**Device type. Recommended to pass specific values. G**
_device_model	**string**	**“MHA-L29”**	**Device model.**
_device_name	**string**	**“HUAWEI Mate 9”**	**Device name.**
_os	**string**	**“Android”**	**Platform’s operating system. Recommended to pass s**
_os_ver	**string**	**“v11.0”**	**OS version.**
_os_tz	**string**	**“+0300”**	**Platform time zone. Recommended to pass specific va**
_os_language	**string**	**“ru”**	**Platform language. Recommended to pass specific val**
_ad_track	**bool**	**true**	**Ad tracking allowed.**
_ad_id	**string**	**“ad 123”**	**Ad identifier.**
_app_id	**string**	**“app 123”**	**Application ID.**
_app_iid	**string**	**“example iid”**	**Installation identifier of the application.**
_app_ver	**string**	**“v1.2.3.4”**	**Application version.**
_browser_name	**string**	**“Chrome”**	**Browser. Recommended to pass specific values. Get t**
_browser_ver	**string**	**“4.5.6.7.8”**	**Browser version.**

Get statistics on resource

Description

Get statistics on profile subscriptions in segment.

> **INFO: Please note**
>
> **The `get_counters` and `refresh_counters` methods work together. Segment statistics are not updated automa updated data becomes available in the `get_counters` method.**

URL

Method: POST

`https://example.com/api/v1.1/segments/get_counters`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
list_id	**int**	**3**
resource_id	**int**	**2**
id	**int**	**1**
format	**int**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "list_id": 1,
    "resource_id": 24,
    "id": 7,
    "format": "json"
}
```

XML

```
<xml>
     <token>abcdefghijklmnqrstuvwxyz</token>
     <list_id>1</list_id>
     <resource_id>24</resource_id>
     <id>7</id>
     <format>json</format>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "status": {
                "subscribed_sendable": 209,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "push",
            "status": {
                "subscribed_sendable": 0,
                "unconfirmed_sendable": 0
            }
        },
        {
            "channel": "sms",
            "status": {
                "subscribed_sendable": 108,
                "unconfirmed_sendable": 0
            }
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
     <data>
         <channel>email</channel>
         <status>
             <subscribed_sendable>209</subscribed_sendable>
             <unconfirmed_sendable>0</unconfirmed_sendable>
         </status>
     </data>
     <data>
         <channel>push</channel>
         <status>
             <subscribed_sendable>0</subscribed_sendable>
             <unconfirmed_sendable>0</unconfirmed_sendable>
         </status>
     </data>
     <data>
         <channel>sms</channel>
         <status>
             <subscribed_sendable>108</subscribed_sendable>
             <unconfirmed_sendable>0</unconfirmed_sendable>
         </status>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
data	**array of objects**
error	**int**
error_text	**string**

Update statistics on segment

Description

Update the subscription statistics of the profiles in the segment.
Returns a response with task ID without waiting for execution.

URL

Method: POST

`https://example.com/api/v1.1/segments/refresh_counters`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
list_id	**int**	**3**
resource_id	**int**	**2**
id	**int**	**1**
format	**int**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "list_id": 1,
    "resource_id": 24,
    "id": 7,
    "format": "json"
}
```

XML

```
<xml>
     <token>abcdefghijklmnqrstuvwxyz</token>
     <list_id>1</list_id>
     <resource_id>24</resource_id>
     <id>7</id>
     <format>json</format>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "55a345c8-3d79-47ef-b03e-b84ce8e0c126"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
     <data>
         <task_id>55a345c8-3d79-47ef-b03e-b84ce8e0c126</task_id>
     </data>
     <error>0</error>
     <error_text>Successful operation</error_text>
</xml>
```

Parameter	Type
data	**JSON objects**
error	**int**
error_text	**string**

Checking the task status

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
task_id	**string**	**“518f4997-ec43-4df5-bfd0-093aaacf679f”**

Request example

JSON

```
{
    "task_id": "930bfdb0-a5a8-4207-a77d-6b989bdea326",
    "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
     <task_id>930bfdb0-a5a8-4207-a77d-6b989bdea326</task_id>
     <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": [
            {
                "channels": [
                    {
                        "channel": "email",
                        "status": {
                            "subscribed_sendable": 209,
                            "unconfirmed_sendable": 0
                        }
                    },
                    {
                        "channel": "push",
                        "status": {
                            "subscribed_sendable": 0,
                            "unconfirmed_sendable": 0
                        }
                    },
                    {
                        "channel": "sms",
                        "status": {
                            "subscribed_sendable": 0,
                            "unconfirmed_sendable": 0
                        }
                    }
                ],
                "in_progress": false,
                "list_id": 1,
                "refresh_task_id": "",
                "resource_id": 27,
                "updated": "2024-04-20T09:30:01.583Z"
            }
        ]
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <channels>
                <channel>email</channel>
                <status>
                    <subscribed_sendable>209</subscribed_sendable>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                </status>
            </channels>
            <channels>
                <channel>push</channel>
                <status>
                    <subscribed_sendable>0</subscribed_sendable>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                </status>
            </channels>
            <channels>
                <channel>sms</channel>
                <status>
                    <subscribed_sendable>0</subscribed_sendable>
                    <unconfirmed_sendable>0</unconfirmed_sendable>
                </status>
            </channels>
            <in_progress>false</in_progress>
            <list_id>1</list_id>
            <refresh_task_id></refresh_task_id>
            <resource_id>27</resource_id>
            <updated>2024-04-20T09:30:01.583Z</updated>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Add or remove profile

Description

Adds a profile to a static segment or removes a profile by ID (`profile_id`).

Request URL

Method: POST
Add a profile to a segment: `https://example.com/api/v1.1/profiles/segstatic/add`
Remove a profile from a segment: `https://example.com/api/v1.1/profiles/segstatic/remove`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
segment_id	**int**	**1**
db_id	**int**	**1**
profile_id	**String**	**“abcdefghijklmnqrstuvwxyz”**
skip_triggers	**bool**	**true**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "segment_id": 2,
  "db_id": 8,
  "profile_id": "abcdefghijklmnqrstuvwxyz",
  "skip_triggers": true
}
```

XML

```
<xml>
  <token>0</token>
  <segment_id>2</segment_id>
  <db_id>8</db_id>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
  <skip_triggers>true</skip_triggers>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get profile data in a static or updatable segment

Description

Get the data of profiles that are in a static or updatable segment, page by page.

Request URL

Method: POST

`https://example.com/api/v1.1/profiles/segstatic/list`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
from_id	**string**	**“abcdefghijklmnqrstuvwxyz”**
from_db	**int**	**1**
count	**int**	**20**
fields	**[string]**	`[“custom_field_1”]`

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "id": 13,
   "from_id": "abcdefghijklmnqrstuvwxyz",
   "from_db": 3,
   "count": 25,
   "fields": [email]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>33</id>
    <from_id>abcdefghijklmnqrstuvwxyz</from_id>
    <from_db>75</from_db>
    <count>25</count>
    <fields array="true">email</fields>
</xml>
```

Response example

JSON

```
{
   "next_id": 26,
   "data": [  {
       "profile_id": "abcdefghijklmnqrstuvwxyz",
       "db_id": 3,
       "xxh": "abcdefghijklmxyz",
       "captured": "ISODateString",
       "profile": {profile Object}
   }
]}
```

XML

```
<xml>
  <next_id>26</next_id>
  <data>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <db_id>3</db_id>
    <xxh>abcdefghijklmxyz</xxh>
    <captured>ISODateString</captured>
    <profile></profile>
  </data>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
next_id	**int**
data	**JSON object**

Create segment

Description

Creates a new segment.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/add/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
name	**string**	**“New segment”**
segment	**JSON array**	

Segment type is determined by a combination of parameters:

- `is_static` is `false` — the segment is dynamic
- `is_static` is `true` and `is_refresh_daily` is `false` — the segment is static
- `is_static` is `true` and `is_refresh_daily` is `true` — the segment is updatable

For static segments

When creating a static segment with the `is_refresh_daily` flag set to `true`, the presence of segmentation ru

JSON

```
{
    "name": "Segment 2",
    "is_static": true,
    "list_id": 5
    "is_refresh_daily": false,
    "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <list_id>2</list_id>
  <name>New segment 2</name>
  <is_static>true</is_static>
  <is_refresh_daily>false</is_refresh_daily>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

For dynamic segments

When creating a dynamic segment, the presence of segmentation rules in the `segment` field is required.

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz"
  "name": "Dynamic segment",
  "is_static": false,
  "segment": [
    {
      "condition": "and",
      "limit": 0,
      "list_id": 1,
      "percent_from": 0,
      "percent_to": 99,
      "queries": [
        {
          "compo": "default",
          "name": "statseg",
          "operator": "once",
          "type": "statseg",
          "value": [
            {
                "name": "segment_id",
                "value": 1
            }
          ]
        },
        {
          "compo": "default",
          "name": "statseg",
          "operator": "once",
          "type": "statseg",
          "value": [
            {
                "name": "segment_id",
                "value": 2
            }
          ]
        }
      ]
    }
  ],
}
```

XML

```
<xml>
    <name>Dynamic segment</name>
    <is_static>false</is_static>
    <segment array="true">
        <condition>and</condition>
        <limit>0</limit>
        <list_id>1</list_id>
        <percent_from>0</percent_from>
        <percent_to>99</percent_to>
        <queries>
            <compo>default</compo>
            <name>statseg</name>
            <operator>once</operator>
            <type>statseg</type>
            <value array="true">
                <name>segment_id</name>
                <value>1</value>
            </value>
        </queries>
        <queries>
            <compo>default</compo>
            <name>statseg</name>
            <operator>once</operator>
            <type>statseg</type>
            <value array="true">
                <name>segment_id</name>
                <value>2</value>
            </value>
        </queries>
    </segment>
    <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

For updatable segments

When creating an updatable segment, segmentation rules in the `segment` field are required.

JSON

```
{
  "name": "Updatable segment",
  "is_static": true,
  "is_refresh_daily": true,
  "segment": [
    {
      "condition": "and",
      "limit": 0,
      "list_id": 1,
      "percent_from": 0,
      "percent_to": 99,
      "queries": [
        {
          "compo": "default",
          "name": "statseg",
          "operator": "once",
          "type": "statseg",
          "value": [
            {
                "name": "segment_id",
                "value": 1
            }
          ]
        },
        {
          "compo": "default",
          "name": "statseg",
          "operator": "once",
          "type": "statseg",
          "value": [
            {
                "name": "segment_id",
                "value": 2
            }
          ]
        }
      ]
    }
  ],
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
  <name>New segment 2</name>
  <is_static>true</is_static>
  <is_refresh_daily>true</is_refresh_daily>
  <list_id>1</list_id>
  <segment array="">
    <condition>and</condition>
    <limit>0</limit>
    <list_id>1</list_id>
    <percent_from>0</percent_from>
    <percent_to>99</percent_to>
    <queries array="">
      <compo>domaingroup</compo>
      <name>email</name>
      <operator>equal</operator>
      <type>email</type>
      <value array="">
        <name>domaingroup</name>
        <value>Yahoo! (US)</value>
      </value>
    </queries>
  </segment>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 1
  },
  "error": 0,
  "error_text": "Successful operation"
}
```


XML


```
<xml>
    <data>
        <saved_id>14</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Update segment

Description

Updates segment building rules.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/update/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
list_id	**int**	**5**
id	**int**	**1**
name	**string**	**“New segment”**
segment	**JSON array**	

Request example

> **TIP**
>
> **When updating segmentation conditions, the specified profile database cannot be changed**

For static segments

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz"
    "list_id":5
    "name": "Static segment 2",
    "is_static": true,
    "is_refresh_daily": false
}
```

XML

```
<xml>
  <id>2</id>
  <list_id>5</list_id>
  <name>New segment 2</name>
  <is_static>true</is_static>
  <is_refresh_daily>false</is_refresh_daily>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

For dynamic segments

When updating a dynamic segment, the presence of segmentation rules in the `segment` field is required.

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 14,
    "list_id": 6,
    "name": "Static segment 2",
    "is_static": false,
    "segment": [
        {
            "condition": "and",
            "limit": 0,
            "percent_from": 1,
            "list_id": 6,
            "percent_to": 99,
            "queries": [
                {
                    "compo": "domaingroup",
                    "name": "email",
                    "operator": "equal",
                    "type": "email",
                    "value": [
                        {
                            "name": "domaingroup",
                            "value": "Yahoo! (US)"
                        }
                    ]
                }
            ]
        }
    ]
}
```


XML


```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>14</id>
  <name>New segment 2</name>
  <is_static>false</is_static>
  <segment array="">
    <list_id>6</list_id>
    <condition>and</condition>
    <limit>0</limit>
    <percent_from>0</percent_from>
    <percent_to>99</percent_to>
    <queries array="">
      <compo>domaingroup</compo>
      <name>email</name>
      <operator>equal</operator>
      <type>email</type>
      <value array="">
        <name>domaingroup</name>
        <value>Yahoo! (US)</value>
      </value>
    </queries>
  </segment>
</xml>
```


For updatable segments

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 28,
    "list_id": 6,
    "name": "Static segment 2",
    "is_static": true,
    "is_refresh_daily": true,
    "segment": [
        {
            "condition": "and",
            "limit": 0,
            "percent_from": 1,
            "list_id": 6,
            "percent_to": 99,
            "queries": [
                {
                    "compo": "domaingroup",
                    "name": "email",
                    "operator": "equal",
                    "type": "email",
                    "value": [
                        {
                            "name": "domaingroup",
                            "value": "Yahoo! (US)"
                        }
                    ]
                }
            ]
        }
    ]
}

#### XML

```xml
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>28</id>
  <name>New segment 2</name>
  <is_static>true</is_static>
  <is_refresh_daily>true</is_refresh_daily>
  
    <list_id>6</list_id>
    <condition>and</condition>
    <limit>0</limit>
    <percent_from>0</percent_from>
    <percent_to>99</percent_to>
    <queries array="">
      <compo>domaingroup</compo>
      <name>email</name>
      <operator>equal</operator>
      <type>email</type>
      <value array="">
        <name>domaingroup</name>
        <value>Yahoo! (US)</value>
      </value>
    </queries>
  
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 1
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML
```

```
<xml>
    <data>
        <saved_id>14</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text
saved_id

Get segment information

Description

Retrieves a segment detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/get/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>10</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "count": 100,
        "id": 1,
        "name": "Segment A",
        "segment": [
            {
                "condition": "or",
                "limit": 0,
                "list_id": 1,
                "percent_from": 0,
                "percent_to": 99,
                "queries": [
                    {
                        "child": [],
                        "compo": "domaingroup",
                        "condition": "",
                        "name": "email",
                        "operator": "equal",
                        "type": "email",
                        "value": [
                            {
                                "name": "domaingroup",
                                "value": "Yahoo! (US)"
                            }
                        ]
                    }
                ]
            }
        ]
     },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <count>0</count>
        <description>""</description>
        <id>14</id>
        <name>"Test segment1"</name>
        <segment>
            <condition>"or"</condition>
            <limit>0</limit>
            <list_id>1</list_id>
            <percent_from>0</percent_from>
            <percent_to>99</percent_to>
            <queries>
                <compo>"default"</compo>
                <condition>""</condition>
                <name>"custom_tags"</name>
                <object_path>""</object_path>
                <object_type>""</object_type>
                <operator>"in_data"</operator>
                <type>"subscription_email"</type>
                <value>
                    <name>"url"</name>
                    <value>"http:127.0.0.1:8080/json.email.subscriptions"</value>
                </value>
                <value>
                    <name>"field"</name>
                    <value>"subscriptions"</value>
                </value>
                <value>
                    <name>"resource_id"</name>
                    <value>"1"</value>
                </value>
            </queries>
        </segment>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
count	**int**
id	**int**
list_id	**int**
name	**string**
type	**string**
segment	**JSON array**
descr	**string**

Get segments list

Description

Retrieves all segments list.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/list/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**1**
ui_tags	**[string]**	`[ “tag1”, “tag2” ]`
groups	**[string]**	`[ “group1”, “group2” ]`
groups_id	**[int]**	`[ 0, 1 ]`

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "limit": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
   <limit>1</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 1,
      "name": "Segment A",
      "ui_tags": ["tag1", "tag2"],
      "Groups": [
        {
          "Id": 1,
          "Name": "subaccount"
        }
      ]
    },
    {
      "id": 2,
      "name": "Segment B",
      "ui_tags": [],
      "Groups": [
        {
          "Id": 0,
          "Name": "main"
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "next_from_id": 2,
  "total_count": 2
}
```


XML


```
<xml>
    <data>
        <groups>
            <id>1</id>
            <name>"subaccount"</name>
        </groups>
        <id>2</id>
        <name>"Women age 35-60"</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>5</next_from_id>
    <total_count>9</total_count>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result c**
total_count	**int**	**Total segments count**
id	**int**	**Segment ID**
list_id	**int**	**Database id**
name	**string**	**Segment name**
ui_tags	**[string]**	**List of tags**
type	**string**	**Type of a segment**
groups	**[string]**	**List of groups**

Delete segment

Description

Deletes a segment.

Request URL

Method: POST

`https://example.com/api/v1.1/segments/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>10</id>
</xml>
```

Response parameters

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Create suppression list

Description

Creates a new suppression list.

Request URL

Method POST

`https://example.com/api/v1.1/suppresses/add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
name	**string**	**“New suppression list”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "name": "New suppression list"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <name>New suppression list</name>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 1
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<result>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</result>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
saved_id	**int**

Update suppression list

Description

Renames a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/update`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
name	**string**	**“Suppression list v.2”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 2,
    "name": "New suppression list"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>2</id>
  <name>Suppression list v.2</name>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 2
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>2</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get suppression list info

Description

Retrieves suppression list details.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 2
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
</xml>
```

JSON

```
{
    "data": {
        "id": 3,
        "name": "New suppression list",
        "is_global": false,
        "description": ""
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <description>""</description>
        <id>1</id>
        <is_global>true</is_global>
        <name>"Global suppress list"</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**
is_global	**bool**

Get the list of suppression lists

Description

Retrieves multiple suppression lists information.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/list`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**5**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`
groups	**[]String**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "from_id": 1,
    "limit":2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <from_id>1</from_id>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 1,
            "name": "Global suppress list",
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 3,
            "name": "New suppression list",
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 5
}
```


XML


```
<xml>
    <data>
        <groups>
            <id>0</id>
            <name>"main"</name>
        </groups>
        <id>1</id>
        <name>"Global suppress list"</name>
    </data>
    <data>
        <groups>
            <id>0</id>
            <name>"main"</name>
        </groups>
        <id>3</id>
        <name>"New suppression list"</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>5</total_count>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**
total_count	**int**
ui_tags	**[]String**
groups	**[]String**

Delete suppression list

Description

Deletes a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 2
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>4</id>
</xml>
```

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Upload suppression list data to file

Description

Uploads a csv file containing md5 data from a suppression list.
Includes 3 requests.

Step 1 - creating profiles export task

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/export`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
type	**string**	**“emails”** **“phones”** **“domains”** **“all”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "type": "emails"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>3</id>
  <type>emails</type>
</xml>
```

Response example

JSON

```
{
    "data": {
        "task_id": "c110f0-06d0-4695-9aee-a9bab742d"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>c110f0-06d0-4695-9aee-a9bab742d</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
task_id	**string**
error	**int**
error_text	**string**

Step 2 - checking export status and getting link

Request URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
task_id	**string**	**“c110f0-06d0-4695-9aee-a9bab742d”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "task_id": "c110f0-06d0-4695-9aee-a9bab742d"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <task_id>c110f0-06d0-4695-9aee-a9bab742d</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": {
            "hashed_file": "suppress/b5bb73d7-530d-4a64-89d4-8ad22bad73/suppress_list_name.md5.zip",
            "is_api": true,
            "regular_file": "",
            "suppress_id": 1,
            "total": 32
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <status>completed</status>
    <progress>100</progress>
    <xml>
      <hashed_file>suppress/b5bb73d7-530d-4a64-89d4-8ad22bad73/suppress_list_name.md5.zip</hashed_file>
      <is_api>true</is_api>
      <regular_file></regular_file>
      <suppress_id>1</suppress_id>
      <total>32</total>
    </xml>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
status	**string**	**Task status**
progress	**int**	**Task progress**
hashed_file	**array**	**Link to download the file with data in md5 format**
is_api	**bool**	**Export via API**
suppress_id	**int**	**Suppression list ID**
total	**int**	**Number of lines (email addresses and domains, phones) in the suppr**
error	**int**	**Error code**
error_text	**string**	**Error text**

Step 3 - downloading result file

The link to download the file looks like following:

`https://example.com/api/download/suppress/b5bb73d7-530d-4a64-89d4-8ad22bad73/suppress_list_name.md5.zip?token=abcdefghijkl`

- `https://example.com` **- link to the platform panel**
- `api/download/suppress/b5bb73d7-530d-4a64-89d4-8ad22bad73/suppress_list_name.md5.zip` **- link from the** `urls` **paramet**
- `?token=abcdefghijklmnqrstuvwxyz` **- user's API token (after the equal sign).**

To download a file with profiles you should either open this link in your browser or send an empty GET req

Suppression list actions

Email address

- **Check if email is suppressed**
- **Add email to suppression list**
- **Add multiple emails to suppression list**
- **Remove email from suppression list**
- **Remove all emails from suppression list**

Email domain

- **Check if domain is suppressed**
- **Add domain to suppression list**
- **Add multiple domains to suppression list**
- **Remove domain from suppression list**
- **Remove all domains from suppression list**

Phone number

- **Check if phone number is suppressed**
- **Add phone number to suppression list**
- **Add multiple phones to suppression list**
- **Remove phone number from suppression list**
- **Remove all phone numbers from suppression list**

Check if email is suppressed

Description

Checks if an email address is in a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/email_check`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
email	**string**	**“john@example.com”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "email": "john@example.com",
    "id": 1
}
```

XML

```
<xml>
  <email>john@example.com</email>
  <id>1</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

JSON

```
{
    "domain_matched": false,
    "email_matched": true,
    "error": 0,
    "error_text": "Successful operation",
    "matched": true,
    "phone_matched": false
}
```

XML

```
<xml>
    <domain_matched>false</domain_matched>
    <email_matched>true</email_matched>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <matched>true</matched>
    <phone_matched>false</phone_matched>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**error text**
domain_matched	**bool**	**true, if email domain is on the s**
email_matched	**bool**	**true, if email domain is on the s**
matched	**bool**	**true, if email address is on the**

Add email to suppression list

Description

Adds an email address to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/email_add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
email	**string**	**“john@example.com”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "email": "john@example.com",
    "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <email>john@example.com</email>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "john@example.com": true
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <john@example.com>false</john@example.com>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Add multiple emails to suppression list

Description

Adds single or multiple email addresses to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/emails_add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
emails	**JSON array**	`["john@example.com"]`
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "emails": [
        "example@example.com",
        "example1@example.com",
        "example2@example.com"
    ],
    "id": 3
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <emails>example@example.com</emails>
  <emails>example1@example.com</emails>
  <emails>example2@example.com</emails>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 3,
    "total_duplicated": 0,
    "total_imported": 3,
    "total_invalid": 0
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>3</total_count>
    <total_duplicated>0</total_duplicated>
    <total_imported>3</total_imported>
    <total_invalid>0</total_invalid>
</xml>
```

Response parameters

Parameter	Type
error	**int**

Parameter	Type
error_text	**string**
total_count	**int**
total_duplicated	**int**
total_imported	**int**
total_updated	**int**

Remove email from suppression list

Description

Removes an email address from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/email_remove`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**
email	**string**	**“example@example.com”**

CAUTION

You can use MD5 hash for an email address.

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "email": "john@example.com",
    "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <email>john@example.com</email>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Remove all emails from suppression list

Description

Removes all email addresses from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/emails_remove`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "removed_count": 1000
}
```

XML

```
<result>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <removed_count>1000</removed_count>
</result>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
remove_count	**int**

Check if domain is suppressed

Description

Check if an email domain is in a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domain_check`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
domain	**string**	**“example.com”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "domain": "example.com",
    "id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <domain>example.com</domain>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "matched": true
}
```

XML

```
<xml>
    <domain_matched>true</domain_matched>
    <email_matched>false</email_matched>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <matched>true</matched>
    <phone_matched>false</phone_matched>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
matched	**bool**	**true, if domain**

Add domain to suppression list

Description

Adds an email domain to a specified suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domain_add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
domain	**string**	**“example.com”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "domain": "example.com",
    "id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <domain>example.com</domain>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "example.com": true
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <example.com>false</example.com>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Add multiple domains to suppression list

Description

Adds one or multiple email domains to a specified suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domains_add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
domains	**JSON array**	**[“example.com”]**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "domains": [
        "example.com",
        "example1.com"
    ],
    "id": 3
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <domains>example1.com</domains>
  <domains>example2.com</domains>
  <domains>example3.com</domains>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 3,
    "total_duplicated": 0,
    "total_imported": 3,
    "total_invalid": 0
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>3</total_count>
    <total_duplicated>0</total_duplicated>
    <total_imported>3</total_imported>
    <total_invalid>0</total_invalid>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**

Parameter	Type
total_count	**int**
total_duplicated	**int**
total_imported	**int**
total_updated	**int**

Remove domain from suppression list

Description

Removes an email domain from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domain_remove`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
domain	**string**	**“example.com”**

CAUTION

You can use domain MD5-hash sum.

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "domain": "example.com",
    "id": 3
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <domain>test.com</domain>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Remove all domains from suppression list

Description

Removes all email domains from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/domains_remove`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 3
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>3</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "removed_count": 100
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <removed_count>100</removed_count>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
removed_count	**int**

Check if phone number is suppressed

Description

Checks if a phone number is in a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phone_check`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
value	**string**	**“+79000000001”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token":"abcdefghijklmnqrstuvwxyz",
    "id":1,
    "value":"+79000000001"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <value>+79000000001</value>
</xml>
```

Response example

JSON

```
{
    "domain_matched": false,
    "email_matched": false,
    "error": 0,
    "error_text": "Successful operation",
    "matched": true,
    "phone_matched": true
}
```

XML

```
<xml>
  <domain_matched>false</domain_matched>
  <email_matched>false</email_matched>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <matched>true</matched>
  <phone_matched>true</phone_matched>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
domain_matched	**bool**	**true, if value is in the suppres**
email_matched	**bool**	**true, if value is in the suppres**
phone_matched	**bool**	**true, if value is in the list of su**
matched	**bool**	**true, if value is in the suppres**

Add phone number to suppression list

Description

Adds a phone number to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phone_add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
value	**string**	**“+79000000001”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1,
  "value":"+79000000001"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
    <value>+79000000001</value>
</xml>
```

Response example

JSON

```
{
    "data": {
        "+79000000991": true
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data><+79000000001>false</+79000000001>
</data>
<error>0</error>
<error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
data	**obj**	**Phone number added to**

Add multiple phones to suppression list

Description

Adds single or multiple phone numbers to a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phones_add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
values	**JSON array**	[“+79009009001”, “+79009009002”]
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "values": [
        "+79009009001",
        "+79009009002"
    ]
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <values>+79009009001</values>
  <values>+79009009002</values>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 2,
    "total_duplicated": 0,
    "total_imported": 2,
    "total_invalid": 0
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>2</total_count>
  <total_duplicated>0</total_duplicated>
  <total_imported>2</total_imported>
  <total_invalid>0</total_invalid>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Err**

Parameter	Type	
error_text	**string**	**Err**
total_count	**int**	**Tot**
total_duplicated	**int**	**Ph**
total_imported	**int**	**Ph**
total_invalid	**int**	**Nu**

Remove phone number from suppression list

Description

Removes a phone number from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phone_remove`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**
value	**string**	**“+79009009001”**

Request example

JSON

```
{
    "token":"62c1336c402e429ab2e59d84d089b1cc",
    "id":1,
    "value":"+79009009001"
}
```

XML

```
<xml>
  <token>62c1336c402e429ab2e59d84d089b1cc</token>
  <id>1</id>
  <value>+79009009001</value>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Remove all phone numbers from suppression list

Description

Removes all phone numbers from a suppression list.

Request URL

Method: POST

`https://example.com/api/v1.1/suppresses/phones_remove`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "removed_count": 500
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <removed_count>4</removed_count>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "removed_count": 500
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <removed_count>500</removed_count>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
remove_count	**int**

Get templates list

Description

Retrieves a list of message templates.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/list`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**100**
channel	**string**	**“email”** **“sms”** **“push”** **“telegram_bot”** **“telegram_group”** **“custom_channel”**
ui_tags	**[string]**	`[ "tag1", "tag2" ]`
groups	**[string]**	`[ "group1", "group2" ]`
groups_id	**[int]**	`[ 0, 1 ]`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
           "limit": 2
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 1,
            "updated_user": "ivan_ivanov",
            "update_time": "2025-05-21T09:17:53.362Z",
            "name": "Message A",
                        "channels": [
                "email"
            ]
                },
        {
            "id": 2,
            "name": "Message B",
                        "channels": [
                "push"
            ]
                }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 10
}
```


XML


```
<xml>
  <data>
    <id>1</id>
    <updated_user>ivan_ivanov</updated_user>
    <update_time>2025-05-21T09:17:53.362Z</update_time>
    <name>Message A</name>
    <channels>email</channels>
  </data>
  <data>
    <id>2</id>
    <name>Message B</name>
    <channels>push</channels>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>10</total_count>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
total_count	**int**
id	**int**
name	**string**
channels	**array**
is_draft	**bool**
is_confirm	**bool**
ui_tags	**[]String**
groups	**array of objects**
updated_user	**string**
update_time	**string**

Get template info

Description

Retrieves a single message template information.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/get`

Request parameters

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
format	**string**	**“json”**	**No**
id	**int**	**1**	**Yes**
channel_sid	**array of strings**	**[“email”, “sms”, “cc_channel”]**	**No**
get_variable	**boolean**	**true**	**No**

Available variable types

Profile data

`lead.fname` , `lead.lname` , `lead.email` , `lead.phone`
`lead.birthdate` , `lead.gender` , `lead._city` , `lead._country`
`lead._xxh` , `lead_custom.*`

Parameters

`campid` , `workflow_id` , `workflow_name` , `msg_type`
`attribute_value.*` , `node_id`

Date and time

`datenow` , `yearlong` , `hourse` , `minutes` , `ampm`

Apicontent variables

`apicontent.field_1` , `apicontent.field_2` , `apicontent.field_N`

Segmentation

`segname`

Learn more about template variables in this article.

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "channel_sid": ["email", "sms", "cc_channel"],
    "get_variable": true
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <channel_sid>email</channel_sid>
  <channel_sid>sms</channel_sid>
  <channel_sid>cc_channel</channel_sid>
  <get_variable>true</get_variable>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 1,
        "updated_user": "example_user",
        "update_time": "2025-05-21T09:17:53.362Z",
        "name": "example template name",
        "description": "example template description",
        "channels": [
            {
                "id": 1,
                "sid": "email",
                "fields": {
                    "text": "example text version",
                    "html": "example html content",
                    "subject": "example subject",
                    "from_email": "example@example.com",
                    "from_name": "example from name",
                    "replyto_email": "reply@example.com",
                    "replyto_name": "example reply name",
                    "is_not_inline": true,
                    "attach": []
                },
                "params": [
                    "lead.fname",
                    "datenow",
                    "segname",
                    "apicontent.field_name",
                    "workflow_name"
                ],
                "presets": null
            },
            {
                "id": 2,
                "sid": "sms",
                "fields": {
                    "sms": "example sms content",
                    "sms_max_count": 0,
                    "add_sms_url_scheme": false
                },
                "params": [
                    "lead.fname",
                    "datenow",
                    "segname",
                    "apicontent.field_name",
                    "workflow_name"
                ],
                "presets": null
            }
        ],
        "json_url": "https://example.com/examples/message.json",
        "is_draft": true,
        "is_confirm": true,
        "ui_tags": [
            "example_tag_1",
            "example_tag_2"
        ],
        "groups": [
            {
                "id": 2,
                "name": "example_group_1"
            },
            {
                "id": 3,
                "name": "example_group_2"
            }
        ],
        "suppress_id": 1
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```xml
<xml>
  <data>
    <id>1</id>
    <updated_user>example_user</updated_user>
    <update_time>2025-05-21T09:17:53.362Z</update_time>
    <name>example template name</name>
    <description>example template description</description>
    <channels>
      <id>1</id>
      <sid>email</sid>
      <fields>
        <text>example text version</text>
        <html>example html content</html>
        <subject>example subject</subject>
        <from_email>example@example.com</from_email>
        <from_name>example from name</from_name>
        <replyto_email>reply@example.com</replyto_email>
        <replyto_name>example reply name</replyto_name>
        <is_not_inline>true</is_not_inline>
        <attach array='true'></attach>
      </fields>
      <params>lead.fname</params>
      <params>datenow</params>
      <params>segname</params>
      <params>apicontent.field_name</params>
      <params>workflow_name</params>
      <presets></presets>
    </channels>
    <channels>
      <id>2</id>
      <sid>sms</sid>
      <fields>
        <sms>example sms content</sms>
        <sms_max_count>0</sms_max_count>
        <add_sms_url_scheme>false</add_sms_url_scheme>
      </fields>
      <params>lead.fname</params>
      <params>datenow</params>
      <params>segname</params>
      <params>apicontent.field_name</params>
      <params>workflow_name</params>
    </channels>
    <json_url>https://example.com/examples/message.json</json_url>
    <is_draft>true</is_draft>
    <is_confirm>true</is_confirm>
    <ui_tags>example_tag_1</ui_tags>
    <ui_tags>example_tag_2</ui_tags>
    <groups>
      <id>2</id>
      <name>example_group_1</name>
    </groups>
    <groups>
      <id>3</id>
      <name>example_group_2</name>
    </groups>
    <suppress_id>1</suppress_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Template ID**
name	**string**	**Template name**
description	**string**	**Template description**
channels	**array of objects**	**Template settings data f**
channels[].id	**int**	**Channel ID in template**
channels[].sid	**string**	**System channel identifie**
channels[].fields	**object**	**Template content setting**
channels[].params	**array of strings**	**List of variables used in**
channels[].presets	**object|null**	**Preset channel settings**
json_url	**string**	**Dynamic JSON URL**
is_draft	**bool**	**Draft template flag. If thi**

Parameter	Type	
is_confirm	**bool**	**Confirm subscription link**
ui_tags	**[]String**	**List of tags to search**
groups	**array of objects**	**List of groups**
suppress_id	**int**	**Suppression list ID**
updated_user	**string**	**By whom the template w**
update_time	**string**	**When the template was**

Delete template

Description

Deletes a message template.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
format	**string**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
         "id": 1
}
```

XML

```
<xml>
  <token>97</token>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": 97,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>97</data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
data	**int**
error	**int**
error_text	**string**

Add template

Description

Adds a new message template.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/add`

Request parameters

Parameter	Type	Ex
token	**string**	`"abcdefghijklmnqrstuvwxyz"`
format	**string**	`"json"`
name	**string**	`"Subscription confirmation template"`
descr	**string**	`"Template used in Double Opt-In scenario"`
channels	**[]channels**	`“channels”: [ { “sid”: “email”, “fields”:{ “text”: “Hello, world!”, “html”: “Hello, world!”, “subject”: “Subject line”, “from_email”: “”, “from_name”: “”, “replyto_email”: “”, “replyto_name”: “” } } ]`
json_url	**string**	`"https://altcraft.com/public/assets/example/test.json"`
ui_tags	**[string]**	`["tag_1","tag_2"]`
groups	**[int]**	`[1,2]`
is_draft	**bool**	`false`
is_confirm	**bool**	`true`
suppress_id	**int**	`2`

Request example

INFO

Every template should contain unsubscribe link.

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "name": "Subscription confirmation template",
    "descr":"Template used in Double Opt-In scenario",
    "channels": [
        {
            "sid": "email",
            "fields": {
                "text": "Hello, world!",
                "html": "<html>Hello, world!</html>",
                "subject": "Тема письма",
                "from_email": "",
                "from_name": "",
                "replyto_email": "",
                "replyto_name": "",
                "is_not_inline": true
            }
        }
    ]
    "json_url": "https://altcraft.com/example/message.json",
    "ui_tags": ["tag_1","tag_2"],
    "groups": [1,2],
         "is_draft": false,
    "is_confirm": false,
    "suppress_id": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <name>Template</name>
    <descr>Template description</descr>
    <channels array = 'true'>
        <fields>
            <body>Hello, world!</body>
            <click_url>https://example.com</click_url>
            <icon>https://example.com/images/picture.jpg</icon>
            <title>Welcome</title>
        </fields>
        <sid>push</sid>
    </channels>
    <is_confirm>false</is_confirm>
    <is_draft>false</is_draft>
    <json_url>https://altcraft.com/example/message.json</json_url>
    <suppress_id>2</suppress_id>
    <groups>1</groups>
    <groups>2</groups>
    <ui_tags>tag_1</ui_tags>
    <ui_tags>tag_2</ui_tags>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 10
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <saved_id>10</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**string**

Update template

Description

Add a new message template.

Request URL

Method: POST

`https://example.com/api/v1.1/templates/update`

Request parameters

Parameter	Type	Ex
token	**string**	`"abcdefghijklmnqrstuvwxyz"`
format	**string**	`"json"`
id	**int**	`1`
name	**string**	`"Subscription confirmation template"`
descr	**string**	`"Template used in Double Opt-In scenario"`
channels	**[]channels**	`“channels”: [ { “sid”: “email”, “fields”:{ “text”: “Hello, world!”, “html”: “Hello, world!”, “subject”: “Subject line”, “from_email”: “”, “from_name”: “”, “replyto_email”: “”, “replyto_name”: “” } } ]`
json_url	**string**	`"https://altcraft.com/public/assets/example/test.json"`
ui_tags	**[string]**	`["tag_1","tag_2"]`
groups	**[int]**	`[1,2]`
is_draft	**bool**	`false`
is_confirm	**bool**	`true`
suppress_id	**int**	`2`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 10,
  "name": "Название шаблона",
  "descr": "Описание шаблона",
  "channels": [
    {
      "sid": "email",
      "fields": {
        "text": "Текстовая версия письма",
        "html": "<html>Hello, world!</html>",
        "subject": "Шаблон сообщения",
        "from_email": "",
        "from_name": "",
        "replyto_email": "",
        "replyto_name": "",
        "is_not_inline": true
      }
    }
  ],
  "json_url": "https://altcraft.com/public/assets/example/test.json",
  "ui_tags": ["tag_1", "tag_2"],
  "groups": [],
  "is_draft": false,
  "is_confirm": false,
  "suppress_id": 4
}
```

XML

```
<xml>
  <channels array = 'true'>
    <fields>
      <body>Hello, world!</body>
      <click_url>https://example.com</click_url>
      <icon>https://example.com/images/picture.jpg</icon>
      <title>Welcome</title>
    </fields>
    <sid>push</sid>
  </channels>
  <descr>Template description</descr>
  <groups>1</groups>
  <groups>2</groups>
  <is_confirm>false</is_confirm>
  <is_draft>false</is_draft>
  <json_url>https://altcraft.com/example/message.json</json_url>
  <id>10</id>
  <name>Template</name>
  <suppress_id>2</suppress_id>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <ui_tags>tag_1</ui_tags>
  <ui_tags>tag_2</ui_tags>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 10
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <saved_id>10</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**string**

Channel object

Used in API requests related to adding and updating message templates.

JSON

```
{
	"sid": "email",
	"fields": {
		"text": "Hello, world!",
		"html": "<html>Hello, world!</html>",
		"subject": "Welcome"
	}
},
{
	"sid": "sms",
	"fields": {
		"sms": "Hello, world!"
	}
},
{
	"sid": "push",
	"fields": {
		"title": "Welcome",
		"body": "Hello, world!",
		"icon": "https://example.com/images/picture.jpg",
		"click_url": "https://example.com"
	}
}
```

Description

Parameter	Type	Example
sid	**string**	**“email”** **“sms”** **“push”** **“telegram_bot”** **“telegram_group”** **“custom_channel”**

fields	**object**	“fields”:{
		“text”: “Hello, world”,
		“html”: “Hello, world!”,
		“subject”: “Welcome”
		}

Fields object

Each `channel` object contains a `fields` object:

Email fields

JSON

```
{
	"sid": "email",
	"fields": {
		"text": "Hello, world!",
		"html": "<html>Hello, world!</html>",
		"subject": "Subject line",
		"from_email": "test@example.com",
		"from_name": "Jenny's List",
		"replyto_email": "test_reply@example.com",
		"replyto_name": "Jenny",
		"is_not_inline": true
	}
}
```

Description

Parameter	Type	
text	**string**	`"Hello, world!"`
html	**string**	`"<html>Hello, world!</html>"`
is_amp	**bool**	`false`
amp	**string**	`"<html>Hello, world!</html>"`
subject	**string**	`"Weather alert"`
from_email	**string**	`"test@example.com"`

Parameter	Type	
from_name	**string**	`"Jenny's List"`
replyto_email	**string**	`"test_reply@example.com"`
replyto_name	**string**	`"Jenny"`
is_not_inline	**bool**	`true`

SMS fields

JSON

```
{
	"sid": "sms",
	"fields": {
		"sms": "Hello, world!",
		"sms_max_count": 0,
		"add_sms_url_scheme": true
	}
}
```

Description

Parameter	Type
sms	**string**
sms_max_count	**int**
add_sms_url_scheme	**bool**

Push fields

JSON

```
{
	"sid": "push",
	"fields": {
		"title": "Welcome",
		"body": "Hello, world!",
		"icon": "https://example.com/images/icon.jpg",
		"click_url": "https://example.com",
		"custom_push_json_raw": "{}"
	}
}
```

Description

Parameter	Type	Example
title	**string**	`"Welcome"`
body	**string**	`"Hello, world!"`
icon	**string**	`"https://example.com/images/tes`
click_url	**string**	`"https://example.com"`
custom_push_json_raw	**string**	`"{}"`

Notify fields

JSON

```
{
  "sid": "notify",
  "fields": {
    "template": {
      "id": 1,
      "name": "message_after_purchase",
      "type": "NOTIFY",
      "status": "NOT_SENT",
      "is_draft": true,
      "edna_account_id": "",
      "notify_service_name": "altcraft_testing",
      "vk_data": {
        "body": "Notify message content",
        "buttons": [],
        "attachments": [],
        "enabled": true,
        "inline_keyboard": false,
        "hide_keyboard": false,
        "reply_enabled": false
      },
      "ok_data": {
        "enabled": true,
        "body": "Notify message content"
      },
      "sms_data": {
        "enabled": true,
        "body": "Notify message content",
        "type": "advert"
      },
      "routes": [
        {
          "route": "vk",
          "deliv_timeout": 86400,
          "read_timeout": 86400
        },
        {
          "route": "ok",
          "deliv_timeout": 86400,
          "read_timeout": 86400
        },
        {
          "route": "sms"
        }
      ]
    },
    "info": {
      "is_obsolete": false,
      "not_found": false
    },
    "params": {
      "name": "{{$.Lead._fname}}",
      "order_number": "{{$.Lead.order_number}}"
    }
  }
}
```


XML

```
<xml>
  <sid>notify</sid>
  <fields>
                <info>
                    <is_obsolete>false</is_obsolete>
                    <not_found>false</not_found>
                </info>
                <params>
                    <name>{{$.Lead._fname}}</name>
                    <order_number>{{$.Lead.order_number}}</order_number>
                </params>
                <template>
                    <id>1</id>
                    <is_draft>true</is_draft>
                    <name>message_after_purchase</name>
                    <notify_service_name>altcraft_testing</notify_service_name>
                    <ok_data>
                        <body>Notify message content</body>
                        <enabled>true</enabled>
                    </ok_data>
                    <routes>
                        <deliv_timeout>86400</deliv_timeout>
                        <read_timeout>86400</read_timeout>
                        <route>vk</route>
                    </routes>
                    <routes>
                        <deliv_timeout>86400</deliv_timeout>
                        <read_timeout>86400</read_timeout>
                        <route>ok</route>
                    </routes>
                    <routes>
                        <route>sms</route>
                    </routes>
                    <sms_data>
                        <body>Notify message content</body>
                        <enabled>true</enabled>
                        <type>advert</type>
                    </sms_data>
                    <status>NOT_SENT</status>
                    <type>NOTIFY</type>
                    <vk_data>
                        <body>Notify message content</body>
                        <enabled>true</enabled>
                        <hide_keyboard>false</hide_keyboard>
                        <inline_keyboard>false</inline_keyboard>
                        <reply_enabled>false</reply_enabled>
                    </vk_data>
                </template>
            </fields>
</xml>
```

Description

Parameter	Type	
name	**string**	**“message_after_purcl**
type	**string**	**“NOTIFY”**
status	**string**	**“NOT_SENT”**
is_draft	**boolean**	**true**
edna_account_id	**string**	**“123456”**
notify_service_name	**string**	**“altcraft_testing”**
body	**string**	**“Notify message cont**
buttons	**array**	**[]**
attachments	**array**	**[]**
enabled	**boolean**	**true**
inline_keyboard	**boolean**	**false**
hide_keyboard	**boolean**	**false**
reply_enabled	**boolean**	**false**
enabled	**boolean**	**true**
body	**string**	**“Notify message cont**
enabled	**boolean**	**true**
body	**string**	**“Notify message cont**
type	**string**	**“advert”**
route	**string**	**“vk”**
deliv_timeout	**integer**	**86400**

Parameter	Type	
read_timeout	**integer**	**86400**
route	**string**	**“ok”**
route	**string**	**“sms”**
is_obsolete	**boolean**	**false**
not_found	**boolean**	**false**

Telegram fields

JSON

```
{
  "sid": "telegram_bot",
  "fields": {
    "image": {
      "attach": [
        {
          "content": "/9j/bcvb4234hgdffsd45",
          "filekey": "67863659182463338076 2bee",
          "name": "example.jpg",
          "size": 65101,
          "type": "image_path"
        }
      ]
    },
    "image_caption": "Telegram bot message description",
    "markup_style": "HTML"
  }
}
```

XML

```
<xml>
  <sid>telegram_bot</sid>
  <fields>
    <image>
      <attach>
          <content>/9j/bcvb4234hgdffsd45</content>
          <filekey>67863659182463338076 2bee</filekey>
          <name>example.jpg</name>
          <size>65101</size>
          <type>image_path</type>
      </attach>
    </image>
    <image_caption>Telegram bot message description</image_caption>
    <markup_style>HTML</markup_style>
  </fields>
</xml>
```

Description

Parameter	Type	
legacy_button_link	**string**	**“https://example.com”**
legacy_button_name	**string**	**“button1”**
markdown_type	**string**	**“HTML”**
text	**string**	**“example”**
content	**string**	**“/9j/bcvb4234hgdffsd45”**
filekey	**string**	**“67863659182463338076 2bee”**
name	**string**	**“example.jpg”**
size	**integer**	**65101**
type	**string**	**“image_path”**
image_caption	**string**	**“Image description”**
audio_caption	**string**	**“Audio file description”**
video_caption	**string**	**“Video file description”**
markup_style	**string**	**“HTML”**

Whatsapp fields

JSON

```
{
  "id": 9,
  "sid": "whatsapp",
  "fields": {
    "type": "TEMPLATE",
    "template": {
      "type": "360DIALOG",
      "name": "initiate_message_en",
      "hsm_external_id": "c466caf6_c340_4d85_8659_6c9685618a9f_en_initiate_message_en",
      "language": "en",
      "wa_account_id": "+79999999999",
      "hsm_hash_sum": "0b7ac0a91b0ba52e961b87c2297d2bd06a3f5bc1",
      "is_invalid": false,
      "hsm_not_found": false,
      "account_not_found": false
    },
    "raw_hsm": {
      "dialog_template": {
        "components": [
          {
            "type": "BODY",
            "text": "Hello! Thank you for waiting. I needed some time to clarify the information regarding your request. Could you please let m
          }
        ],
        "name": "initiate_message_en",
        "namespace": "c466caf6_c340_4d85_8659_6c9685618a9f",
        "category": "ISSUE_RESOLUTION",
        "language": "en",
        "rejected_reason": "NONE",
        "status": "approved"
      },
      "edna_template": null
    }
  }
}
```


XML


```
<xml>
  <id>9</id>
  <sid>whatsapp</sid>
  <fields>
    <type>TEMPLATE</type>
    <template>
      <type>360DIALOG</type>
      <name>initiate_message_en</name>
      <hsm_external_id>c466caf6_c340_4d85_8659_6c9685618a9f_en_initiate_message_en</hsm_external_id>
      <language>en</language>
      <wa_account_id>+79999999999</wa_account_id>
      <hsm_hash_sum>0b7ac0a91b0ba52e961b87c2297d2bd06a3f5bc1</hsm_hash_sum>
      <is_invalid>false</is_invalid>
      <hsm_not_found>false</hsm_not_found>
      <account_not_found>false</account_not_found>
    </template>
    <raw_hsm>
      <dialog_template>
        <components>
          <component>
            <type>BODY</type>
            <text>Hello! Thank you for waiting. I needed some time to clarify the information regarding your request. Could you please let me k
          </component>
        </components>
        <name>initiate_message_en</name>
        <namespace>c466caf6_c340_4d85_8659_6c9685618a9f</namespace>
        <category>ISSUE_RESOLUTION</category>
        <language>en</language>
        <rejected_reason>NONE</rejected_reason>
        <status>approved</status>
      </dialog_template>
    </raw_hsm>
  </fields>
</xml>
```


Description

Parameter	Type	
type	**string**	**“360DIALOG”**

Parameter	Type	
name	**string**	**“initiate_message_en”**
hsm_external_id	**string**	**“c466caf6_c340_4d85_8659_6c9685618a9f_en_initiate_message_en”**
language	**string**	**“en”**
wa_account_id	**string**	**“+79999999999”**
hsm_hash_sum	**string**	**“0b7ac0a91b0ba52e961b87c2297d2bd06a3f5bc1”**
is_invalid	**boolean**	**false**
hsm_not_found	**boolean**	**false**
account_not_found	**boolean**	**false**
name (in raw_hsm)	**string**	**“initiate_message_en”**
namespace	**string**	**“c466caf6_c340_4d85_8659_6c9685618a9f”**
category	**string**	**“ISSUE_RESOLUTION”**
status	**string**	**“approved”**
rejected_reason	**string**	**“NONE”**
components.type	**string**	**“BODY”**
components.text	**string**	**“Hello! Thank you for waiting...”**
edna_template	**string**	**“null”**

Viber™ fields

JSON

```
"id": 10,
{
  "sid": "viber",
  "fields": {
    "series": [
      {
        "type": "TEXT",
        "text": "message text"
      },
      {
        "type": "IMAGE",
        "image": "/849074172/3/example.jpg"
      },
      {
        "type": "VIDEO",
        "thumbnail": "/848985842/3/6786323458246333807 62bee/example.jpg",
        "duration": 6,
        "file_size": 2,
        "action": "https://example.com/lib/preview/mp4/example.mp4"
      },
      {
        "type": "TEXT_BUTTON",
        "text": "Sample text",
        "caption": "button text",
        "action": "https://example.com"
      },
      {
        "type": "FILE",
        "file_type": "pdf",
        "file_name": "example.pdf",
        "action": "https://www.example.org/testfiles/resources/pdf/example.pdf"
      }
    ]
  }
}
```

XML

```
<xml>
    <id>10</id>
    <sid>viber</sid>
    <fields>
      <series>
        <element>
          <type>TEXT</type>
          <text>message text</text>
        </element>
        <element>
          <type>IMAGE</type>
          <image>/849074172/3/example.jpg</image>
        </element>
        <element>
          <type>VIDEO</type>
          <thumbnail>/848985842/3/6786323458246333807 62bee/example.jpg</thumbnail>
          <duration>6</duration>
          <file_size>2</file_size>
          <action>https://example.com/lib/preview/mp4/example.mp4</action>
        </element>
        <element>
          <type>TEXT_BUTTON</type>
          <text>Sample text</text>
          <caption>button text</caption>
          <action>https://example.com</action>
        </element>
        <element>
          <type>FILE</type>
          <file_type>pdf</file_type>
          <file_name>example.pdf</file_name>
          <action>https://www.example.org/testfiles/resources/pdf/example.pdf</action>
        </element>
      </series>
    </fields>
</xml>
```

Description

Parameter	Type	
type	**string**	**“TEXT”**
text	**string**	**“message text”**
image	**string**	**“/849074172/3/example.jpg”**
thumbnail	**string**	**“/848985842/3/6786323458246333807 62bee/example.jpg”**
duration	**integer**	**6**
file_size	**integer**	**2**
action	**string**	**“https://example.com/lib/preview/mp4/example.mp4”**
caption	**string**	**“button text”**
file_type	**string**	**“pdf”**
file_name	**string**	**“example.pdf”**

**** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

Get campaign information

Description

Retrieve detailed information about a campaign, including audience settings, LCG, marketing activities an

> **CAUTION: Access rights**
>
> **To use this method, the API token must have:**
>
> - **Campaigns — View object**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings — Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/get
```

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**22**
format	**string**	**"json"**
marketing_activities	**bool**	**false**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 22,
  "marketing_activities": true
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>22</id>
  <marketing_activities>true</marketing_activities>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 22,
      "is_delete": false,
      "createtime": "2025-08-22T09:00:21.654Z",
      "updatetime": "2025-08-22T11:08:10.216Z",
      "updated_user": "api.user",
      "name": "Seasonal sale",
      "description": "Campaign for promoting the spring collection",
      "tags": [],
      "state": "active",
      "audience": {
        "state": "ready",
        "seg_id": 4,
        "list_id": 2,
        "extension_conf": {
          "workflows": {"ids": []},
          "triggers": {"ids": []},
          "broadcasts": {"ids": [1]},
          "regulars": {"ids": []},
          "abns": {"ids": []},
          "placements": {"ids": []}
        },
        "lcg": {
          "stratification_field_settings": [
            {
              "field_name": "_sex",
              "field_type": "enum",
              "condition_type": "list",
              "conditions": [
                {"list": [1]},
                {"list": [2]}
              ]
            },
            {
              "field_name": "_region",
              "field_type": "unicode",
              "condition_type": "list",
              "conditions": [
                {"list": ["Moscow", "Saint Petersburg"]},
                {"list": ["Kazan"]}
              ]
            }
          ],
          "ignore_entities": {
            "workflows": {"ids": []},
            "triggers": {"ids": []},
            "broadcasts": {"ids": [1]},
            "regulars": {"ids": []},
            "abns": {"ids": []},
            "placements": {"ids": []}
          },
          "size_formation_settings": {
            "limit": 133333,
            "percent": "16.00"
          }
        }
      },
      "active_period": {
        "settings": {
          "tz": "UTC",
          "end_at": "2025-08-24T12:03:21.93Z"
        },
        "history": {
          "start": "2025-08-22T09:00:47.786Z"
        }
      },
      "counters": {
        "segment": 631,
        "test_group": 530,
        "local_control_group": 101,
        "global_control_group": 0
      },
      "marketing_activities_counter": 2,
      "groups": [
        {
          "id": 0,
          "name": "main"
        }
```

```
      ],
      "attributes": {
        "season": "spring_2025",
        "branch": "marketing"
      },
      "marketing_activities": [
        {
          "workflows": [
            {
              "id": 9,
              "name": "Welcome Series",
              "is_active": true
            }
          ],
          "triggers": [
            {
              "id": 6,
              "name": "Thank You Purchase",
              "is_active": true
            }
          ],
          "broadcasts": [
            {
              "id": 1,
              "name": "Seasonal Sale Announcement",
              "is_active": true,
              "is_completed": false
            }
          ],
          "regulars": [
            {
              "id": 2,
              "name": "Weekly Newsletter",
              "is_active": true
            }
          ],
          "abns": [],
          "placements": []
        }
      ]
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <id>22</id>
    <is_delete>false</is_delete>
    <createtime>2025-08-22T09:00:21.654Z</createtime>
    <updatetime>2025-08-22T11:08:10.216Z</updatetime>
    <updated_user>api.user</updated_user>
    <name>Seasonal sale</name>
    <description>Campaign for promoting the spring collection</description>
    <state>active</state>
    <audience>
      <state>ready</state>
      <seg_id>4</seg_id>
      <list_id>2</list_id>
      <extension_conf>
        <workflows>
        </workflows>
        <triggers>
        </triggers>
        <broadcasts>
          <ids>
            <id>1</id>
          </ids>
        </broadcasts>
        <regulars>
        </regulars>
        <abns>
        </abns>
        <placements>
        </placements>
      </extension_conf>
      <lcg>
        <stratification_field_settings>
          <field>
            <field_name>_sex</field_name>
            <field_type>enum</field_type>
            <condition_type>list</condition_type>
            <conditions>
              <condition>
                <list>
                  <value>1</value>
                </list>
              </condition>
              <condition>
                <list>
                  <value>2</value>
                </list>
              </condition>
            </conditions>
          </field>
        </stratification_field_settings>
        <ignore_entities>
          <broadcasts>
            <ids>
              <id>1</id>
            </ids>
          </broadcasts>
        </ignore_entities>
        <size_formation_settings>
          <limit>133333</limit>
          <percent>16.00</percent>
        </size_formation_settings>
      </lcg>
    </audience>
    <active_period>
      <settings>
        <tz>UTC</tz>
        <end_at>2025-08-24T12:03:21.93Z</end_at>
      </settings>
      <history>
        <start>2025-08-22T09:00:47.786Z</start>
      </history>
    </active_period>
    <counters>
      <segment>631</segment>
      <test_group>530</test_group>
      <local_control_group>101</local_control_group>
      <global_control_group>0</global_control_group>
```

```
    </counters>
    <marketing_activities_counter>2</marketing_activities_counter>
    <groups>
      <group>
        <id>0</id>
        <name>main</name>
      </group>
    </groups>
    <attributes>
      <season>spring_2025</season>
      <branch>marketing</branch>
    </attributes>
  </data>
</xml>
```

Returned parameters

Top-level fields

Parameter	Type	
`error`	**int**	**Er**
`error_text`	**string**	**Er**
`data`	**array**	**Ar**

The data object

Contains the main campaign information.

Parameter	Type	
`id`	**int**	**Campaign ID**
`name`	**string**	**Campaign na**
`description`	**string**	**Campaign de**
`list_id`	**int**	**Profile databa**
`seg_id`	**int**	**Segment ID**
`state`	**string**	**Current camp**
`createtime`	**string**	**Campaign cr**
`updatetime`	**string**	**Last update**
`updated_user`	**string**	**User who ma**
`is_delete`	**boolean**	**Deletion flag**
`tags`	**array**	**Campaign tag**
`groups`	**array**	**Access group**
`marketing_activities_counter`	**int**	**Number of m**
`attributes`	**object**	**Campaign att**
`counters`	**object**	**Profile count**

The counters object

Contains profile counts for different groups.

Parameter	Type	
`segment`	**int**	**Number of profiles in the segment**
`test_group`	**int**	**Number of profiles in the test group (TG)**
`local_control_group`	**int**	**Number of profiles in the Local control group**
`global_control_group`	**int**	**Number of profiles in the Global control group**

The active_period object

Contains information about the campaign timeframe.

Parameter	Type
`settings`	**object**
`history`	**object**

settings

Parameter	Type
`tz`	**string**
`start_at`	**string**
`end_at`	**string**

history

Parameter	Typ
`start`	**string**

The audience object

Contains campaign audience settings and information.

Parameter	
`state`	**string**
`seg_id`	**int**
`list_id`	**int**
`extension_conf`	**object**
`lcg`	**object**

extension_conf

Activities included in audience expansion.

Parameter	Type	
`workflows`	**object**	**Scenarios that expand**
`triggers`	**object**	**Triggered mailings that**
`broadcasts`	**object**	**Broadcast mailings tha**
`regulars`	**object**	**Regular mailings that e**
`abns`	**object**	**A/B tests that expand t**

Parameter	Type	
placements	object	Placements that expan

Nested extension_conf objects

Each object contains:

Parameter	
ids	array

lcg (Local control group)

Parameter
stratification_field_settings
ignore_entities
size_formation_settings

stratification_field_settings

Parameter	Type
field_name	string
field_type	string
condition_type	string
conditions	array

ignore_entities

Activities that ignore LCG audience building.

Parameter	Type
workflows	object
triggers	object
broadcasts	object
regulars	object
abns	object
placements	object

size_formation_settings

LCG size settings.

Parameter	Type	
limit	int	LCG size lim
percent	string	LCG share o

The marketing_activities object

Contains the list of all campaign marketing activities.

Parameter	
workflows	array
triggers	array
broadcasts	array
regulars	array
abns	array
placements	array

Common fields of marketing activities

Parameter	
id	int
name	string
is_active	boolean
is_completed	boolean

Get campaign list

Description

Retrieve the list of campaigns — combinations of mailings and scenarios into unified marketing strategies

CAUTION: Access rights

To use this method, the API token must have:

- **Campaigns — Run**
- **Access to the groups where the campaigns are located**

Configure permissions in Settings — Tokens.

Request URL

Method: `POST`

`https://example.com/api/v1.1/scampaigns/list`

Request parameters

Parameter	Type	Example
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
format	**string**	**"json"**
from_id	**int**	**1**
limit	**int**	**2**
state	**string**	**"active"**
ui_tags	**[]string**	**["tag1", "tag2"]**
groups	**[]string**	**["group1", "group2"]**
groups_id	**[]int**	**[0, 1]**
attributes	**JSON object**	`"attributes": \{ "season": ["spring_2025", "summer_2025"] \}`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "limit": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <limit>1</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 42,
      "is_delete": false,
      "createtime": "2025-08-22T07:15:29.711Z",
      "updatetime": "2025-08-22T09:13:24.264Z",
      "updated_user": "test",
      "name": "Seasonal sale",
      "description": "",
      "tags": [],
      "state": "inactive",
      "audience": {
        "state": "ready",
        "seg_id": 124,
        "list_id": 49,
        "extension_conf": {
          "workflows": {"ids": []},
          "triggers": {"ids": []},
          "broadcasts": {"ids": []},
          "regulars": {"ids": [556]},
          "abns": {"ids": []},
          "placements": {"ids": []}
        },
        "lcg": {
          "ignore_entities": {
            "workflows": {"ids": []},
            "triggers": {"ids": []},
            "broadcasts": {"ids": []},
            "regulars": {"ids": []},
            "abns": {"ids": []},
            "placements": {"ids": []}
          },
          "size_formation_settings": {
            "limit": 0,
            "percent": "10.00"
          }
        }
      },
      "active_period": {
        "settings": {
          "tz": "Europe/Moscow",
          "end_at": "2025-08-23T10:36:25.099+03:00"
        },
        "history": {
          "start": "2025-08-22T12:13:18.813+03:00",
          "stop": "2025-08-22T12:13:24.264+03:00"
        }
      },
      "control_group_counter": 3002,
      "marketing_activities_counter": 1,
      "refresh_data": {
        "state": "completed",
        "progress": 100,
        "task_start_date": "2025-08-22T09:13:19.926Z",
        "task_end_date": "2025-08-22T09:13:20.048Z"
      },
      "group_id": 0,
      "groups_id": [0],
      "groups": [
        {
          "id": 0,
          "name": "main"
        }
      ],
      "attributes": {
        "season": "spring_2025",
        "branch": "marketing"
      }
    }
  ],
  "error": 0,
  "error_text": "",
  "total_count": 5
}
```


XML

```xml
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <id>42</id>
    <is_delete>false</is_delete>
    <createtime>2025-08-22T07:15:29.711Z</createtime>
    <updatetime>2025-08-22T09:13:24.264Z</updatetime>
    <updated_user>test</updated_user>
    <name>Seasonal sale</name>
    <state>inactive</state>
    <audience>
      <state>ready</state>
      <seg_id>124</seg_id>
      <list_id>49</list_id>
      <extension_conf>
        <workflows><ids/></workflows>
        <triggers><ids/></triggers>
        <broadcasts><ids/></broadcasts>
        <regulars><ids><id>556</id></ids></regulars>
        <abns><ids/></abns>
        <placements><ids/></placements>
      </extension_conf>
      <lcg>
        <ignore_entities>
          <workflows><ids/></workflows>
          <triggers><ids/></triggers>
          <broadcasts><ids/></broadcasts>
          <regulars><ids/></regulars>
          <abns><ids/></abns>
          <placements><ids/></placements>
        </ignore_entities>
        <size_formation_settings>
          <limit>0</limit>
          <percent>10.00</percent>
        </size_formation_settings>
      </lcg>
    </audience>
    <active_period>
      <settings>
        <tz>Europe/Moscow</tz>
        <end_at>2025-08-23T10:36:25.099+03:00</end_at>
      </settings>
      <history>
        <start>2025-08-22T12:13:18.813+03:00</start>
        <stop>2025-08-22T12:13:24.264+03:00</stop>
      </history>
    </active_period>
    <control_group_counter>3002</control_group_counter>
    <marketing_activities_counter>1</marketing_activities_counter>
    <refresh_data>
      <state>completed</state>
      <progress>100</progress>
      <task_start_date>2025-08-22T09:13:19.926Z</task_start_date>
      <task_end_date>2025-08-22T09:13:20.048Z</task_end_date>
    </refresh_data>
    <group_id>0</group_id>
    <groups_id>0</groups_id>
    <groups>
      <group>
        <id>0</id>
        <name>main</name>
      </group>
    </groups>
    <attributes>
      <season>spring_2025</season>
      <branch>marketing</branch>
    </attributes>
  </data>
  <total_count>5</total_count>
</xml>
```


Returned parameters

Top-level fields

Parameter	Type	
`error`	**int**	**Error code. (
`error_text`	**string**	**Error descri**

Parameter	Type	
`total_count`	**int**	**Total numbe**
`next_from_id`	**int**	**Pointer for r**
`data`	**array**	**Array of can**

The data object

Contains core campaign information.

Parameter	Type
`id`	**int**
`name`	**string**
`description`	**string**
`list_id`	**int**
`seg_id`	**int**
`state`	**string**
`createtime`	**string**
`updatetime`	**string**
`updated_user`	**string**
`is_delete`	**boolean**
`tags`	**array**
`groups`	**array**
`group_id`	**int**
`groups_id`	**array**
`marketing_activities_counter`	**int**
`control_group_counter`	**int**
`attributes`	**object**
`refresh_data`	**object**

The active_period object

Contains information about the campaign timeframe.

Parameter	Type
`settings`	**object**
`history`	**object**

settings — planned campaign timing

Parameter	Type
`tz`	**string**
`start_at`	**string**
`end_at`	**string**

history — actual campaign data

Parameter	Type
`start`	**string**
`stop`	**string**

The audience object

Contains campaign audience settings and information.

Parameter	Type	
`state`	**string**	**Audience building stat**
`seg_id`	**int**	**Segment ID**
`list_id`	**int**	**Profile database ID**
`extension_conf`	**object**	**Audience expansion se**
`lcg`	**object**	**Local control group se**

extension_conf

Activities that expand the campaign audience.

Parameter	Type	
`workflows`	**object**	**Scenarios that expand**
`triggers`	**object**	**Triggered mailings that**
`broadcasts`	**object**	**Broadcast mailings tha**
`regulars`	**object**	**Regular mailings that e**
`abns`	**object**	**A/B tests that expand t**
`placements`	**object**	**Placements that expan**

Each of these objects contains an array of IDs of the corresponding activities.

lcg — Local control group settings

Parameter
`ignore_entities`
`size_formation_settings`

ignore_entities — activities excluded from LCG audience building

Parameter	Type
`workflows`	**object**
`triggers`	**object**
`broadcasts`	**object**
`regulars`	**object**
`abns`	**object**
`placements`	**object**

size_formation_settings — LCG size settings

Parameter	Type	
`limit`	**int**	**Maximum number of profil**
`percent`	**string**	**LCG share of the total auc**

The refresh_data object

Contains information about the campaign data refresh process.

Parameter	Type
`state`	**string**
`progress`	**int**
`task_start_date`	**string**
`task_end_date`	**string**

The groups object

Contains information about access groups for the campaign.

Parameter	
`id`	**int**
`name`	**string**

Activate campaign

Description

Request to activate a campaign.

> CAUTION: Access rights
>
> **To use this method, the API token must have:**
>
> - **Campaigns — Run**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/activate
```

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**1**
activate_date	**string**	**"2025-03-15T10:00:00Z"**
deactivate_date	**string**	**"2025-03-20T15:30:00Z"**
completed_date	**string**	**"2025-04-01T18:45:00Z"**

> TIP
>
> **If you send `start_date` without a date value, the campaign is activated immediately.**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1,
  "start_date": "2025-04-01T18:45:00Z",
  "deactivate_date": "2025-05-01T18:45:00Z",
  "completed_date": "2025-05-01T18:45:00Z"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <start_date>2025-04-01T18:45:00Z</start_date>
  <deactivate_date>2025-05-01T18:45:00Z</deactivate_date>
  <completed_date>2025-05-01T18:45:00Z</completed_date>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Returned parameters

Parameter	
error	**int**
error_text	**string**
data	**null**

Complete campaign

Description

Complete a campaign immediately. Completion stops sending messages for running mailings and switches

> CAUTION: Access rights
>
> **To use this method, the API token must have:**
>
> - **Campaigns — Run**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/completed
```

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**18**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 18
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>18</id>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Returned parameters

Parameter	
error	**int**
error_text	**string**
data	**null**

> WARNING
>
> **Completing a campaign is irreversible. After completion, the campaign cannot be restarted; you can only**

Deactivate campaign

Description

Deactivate a campaign immediately. Deactivation stops sending messages from running mailings.

> CAUTION: Access rights
>
> **To use this method, the API token must have:**
>
> - **Campaigns — Run**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/deactivate
```

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**18**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 18
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>18</id>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Returned parameters

Parameter	
error	**int**
error_text	**string**
data	**null**

> INFO
>
> **Deactivation temporarily pauses the campaign. All settings and data are preserved for potential reactiva**

Get campaign status

Description

Retrieve the current status of a campaign.

> **CAUTION: Access rights**
>
> **To use this method, the API token must have:**
>
> - **Campaigns — View object**
> - **Access to the groups where the campaigns are located**
>
> **Configure permissions in Settings → Tokens.**

Request URL

Method: `POST`

```
https://example.com/api/v1.1/scampaigns/get_status
```

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**10**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 10
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>10</id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "status": "active"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <status>active</status>
  </data>
</xml>
```

Returned parameters

Parameter	Type	
error	**int**	**E**
error_text	**string**	**E**
data	**object**	**O**

Status fields

Parameter	Type
status	**string**

Possible statuses

Status
cloning
inactive
waiting
active
completed
deleted

Activate mailing

Description

Activates a mailing.
Can be used to activate and launch a mailing instantly.
Also you can override mailing audience source for a different database or segment.
If `run_now` = `true`, the specified mailing launches instantly.
`override_source_type` and `override_source_id` are valid only if `run_now` = `true`.
`override_limit` parameter sets sending limitations for the current run.
If no override options are set - mailings will use their default audience sources and sending limitations.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/activate/`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**12**
run_now	**bool**	**true**
override_source_type	**enum**	**“list” or “segment”**
override_source_id	**int**	**5**
override_limit	**int**	**300**
content	**JSON object**	`{"field":"value"}`
attach	**JSON array**	`[]`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 82,
  "override_source_id": 1,
  "override_source_type": "list",
  "run_now": true
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>82</id>
    <override_source_id>1</override_source_id>
    <override_source_type>list</override_source_type>
    <run_now>true</run_now>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Deactivate mailing

Description

Deactivates a mailing.

Can also abort message sending, if a mailing is active:

`abort_if_running` = `true` **property immediately aborts all messages in sending queue. If the value is** `false` **tl**

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/deactivate`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
abort_if_running	**bool**	**true**
id	**int**	**12**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": "15",
    "abort_if_running": "true"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>15</id>
    <abort_if_running>true</abort_if_running>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get mailing list

Description

Retrieves mailing list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/list`

Request parameters

Parameter	Type	
token	**string**	`"abcdefghijklmnqrstuvwxyz"`
format	**string**	`"json"`
from_id	**int**	`1`
limit	**int**	`2`
type	**string**	`"trigger"`
is_active	**bool**	`true`
ui_tags	**[]String**	`[ "tag1", "tag2" ]`
groups	**[]String**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "limit": 2
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 5,
            "name": "New meteoric mailing",
            "type": "broadcast",
            "is_active": false,
            "subid": "",
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 7,
            "name": "Copy of New meteoric mailing",
            "type": "broadcast",
            "is_active": false,
            "subid": "",
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": 8,
    "total_count": 134
}
```


XML


```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <id>5</id>
        <name>New meteoric mailing</name>
        <type>broadcast</type>
        <is_active>false</is_active>
        <subid></subid>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
    </data>
    <data>
        <id>7</id>
        <name>Copy of New meteoric mailing</name>
        <type>broadcast</type>
        <is_active>false</is_active>
        <subid></subid>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
    </data>
    <next_from_id>8</next_from_id>
    <total_count>134</total_count>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the resu**

Parameter	Type	
id	**int**	**Mailing ID**
is_active	**bool**	**Mailing active flag**
name	**string**	**Mailing name**
type	**string**	**Mailing type**
total_count	**int**	**Total mailing count**
for_confirm	**bool**	**Messages with opt-in link flag**
ui_tags	**[]String**	**List of tags**
groups	**[]String**	**List of access groups**
attributes	**object**	**Attributes assigned to the mailing and their values**
scampaign_id	**int**	**Campaign ID in which the mailing is engaged** **(The parameter is returned if the mailing is engaged in a campaign)**

Get mailing information

Description

Retrieves detailed mailing information.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 2
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>2</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 118,
        "name": "Daily weather notification",
        "description": "",
        "type": "regular",
        "updated_user": "roman",
        "update_time": "2024-05-20T09:17:53.362Z",
        "is_active": false,
        "is_completed": false,
        "data_id": 1,
        "data_type": "list",
        "content": [
            {
                "subject": "",
                "from_email": "",
                "from_name": "",
                "replyto_email": "",
                "replyto_name": "",
                "message_id": 21
            }
        ],
        "status": "Inactive",
        "subid": "",
        "virtual_sender_id": 2,
        "providers": [
            "google",
            "mozilla",
            "safari",
            "safari-native",
            "firefox-firebase",
            "chrome-firebase",
            "opera-firebase",
            "yandex-firebase",
            "samsung-firebase",
            "android-firebase",
            "ios-firebase",
            "ios-apns",
            "yandex-Appmetrica-device-id-ios",
            "yandex-Appmetrica-device-id-android",
            "yandex-Appmetrica-ios-ifa",
            "yandex-Appmetrica-google-aid",
            "yandex-Appmetrica-android-token",
            "yandex-Appmetrica-ios-token",
            "android-huawei",
            "ios-huawei",
            "android-rustore"
        ],
        "push_sending_mode": "fanout",
        "resource_id": 26,
        "channel": "push",
        "for_confirm": false
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <id>118</id>
        <name>Daily weather notification</name>
        <description></description>
        <type>regular</type>
        <updated_user>roman</updated_user>
        <update_time>2024-05-20T09:17:53.362Z</update_time>
        <is_active>false</is_active>
        <is_completed>false</is_completed>
        <data_id>1</data_id>
        <data_type>list</data_type>
        <content>
            <subject></subject>
            <from_email></from_email>
            <from_name></from_name>
            <replyto_email></replyto_email>
            <replyto_name></replyto_name>
            <message_id>21</message_id>
        </content>
        <status>Inactive</status>
        <subid></subid>
        <virtual_sender_id>2</virtual_sender_id>
        <providers>google</providers>
        <providers>mozilla</providers>
        <providers>safari</providers>
        <providers>safari-native</providers>
        <providers>firefox-firebase</providers>
        <providers>chrome-firebase</providers>
        <providers>opera-firebase</providers>
        <providers>yandex-firebase</providers>
        <providers>samsung-firebase</providers>
        <providers>android-firebase</providers>
        <providers>ios-firebase</providers>
        <providers>ios-apns</providers>
        <providers>yandex-Appmetrica-device-id-ios</providers>
        <providers>yandex-Appmetrica-device-id-android</providers>
        <providers>yandex-Appmetrica-ios-ifa</providers>
        <providers>yandex-Appmetrica-google-aid</providers>
        <providers>yandex-Appmetrica-android-token</providers>
        <providers>yandex-Appmetrica-ios-token</providers>
        <providers>android-huawei</providers>
        <providers>ios-huawei</providers>
        <providers>android-rustore</providers>
        <push_sending_mode>fanout</push_sending_mode>
        <resource_id>26</resource_id>
        <channel>push</channel>
        <for_confirm>false</for_confirm>
    </data>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameters	Type	
error	**int**	**Erro**
error_text	**string**	**Erro**
content	**JSON array**	**Mes**
data_id	**int**	**Audi**
data_type	**string**	**Audi**
id	**int**	**Mail**
update_time	**string**	**Date**
updated_user	**string**	**Nam**
is_active	**bool**	**Mail**
is_completed	**bool**	**Mail**
name	**string**	**Mail**
resource_id	**int**	**Reso**
send_email_content	**bool**	**Ema**
send_push_content	**bool**	**Push**
send_sms_content	**bool**	**SMS**
push_sending_mode	**string**	**Push**
for_confirm	**bool**	**Mes**
status	**string**	**Mail**

Parameters	Type	
subid	**string**	**Mail**
type	**string**	**Mail**
virtual_sender_id	**int**	**Virtu**
descr	**string**	**Mail**
attributes	**object**	**Attri**
scampaign_id	**int**	**Cam (The**

Get mailing log

Description

Retrieves mailing logs.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/log_v2`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_time	**string**	**“2018-08-10T00:00:00.000Z”**
id	**int**	**1**
limit	**int**	**2**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "limit": 2
}
```

XML

```
<xml>
   <token>abcdefghijklmnqrstuvwxyz</token>
   <id>1</id>
   <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "level": "INFO",
            "message": "Mailing finished",
            "time": "2024-04-02T17:57:01.504Z",
            "state": "end",
            "value": ""
        },
        {
            "level": "INFO",
            "message": "Total time spent",
            "time": "2024-04-02T17:57:01.505Z",
            "state": "",
            "value": "138.351985ms"
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 140
}
```

XML

```
<xml>
    <data>
        <level>"INFO"</level>
        <message>"Mailing finished"</message>
        <state>"end"</state>
        <time>"2024-05-28T16:40:59.207Z"</time>
        <value>""</value>
    </data>
    <data>
        <level>"INFO"</level>
        <message>"Total time spent"</message>
        <state>""</state>
        <time>"2024-05-28T16:40:59.208Z"</time>
        <value>"351.155547ms"</value>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>9</total_count>
</xml>
```


Response parameters

Parameter	
error	**int**
error_text	**string**
level	**string**
message	**string**
state	**string**
time	**date**
value	**string**

Logging levels:

- `DEBUG`
- `INFO`
- `WARNING`
- `ERROR`

Event entry status:

- `start` – **mailing start**
- `end` – **mailing finish**

Clone mailing

Description

Clones mailing content and settings to a newly created mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/clone`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
name	**string**	**“Clone”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "name": "Clone"
}
```

XML

```
<xml>
   <token>abcdefghijklmnqrstuvwxyz</token>
   <id>1</id>
   <name>Clone</name>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 87
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>88</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Delete mailing

Description

Deletes a mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get mailing status

Description

Retrieves a mailing status — whether it is active or inactive.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/get_status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**12**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 82
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>82</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "is_active": false,
        "is_completed": false,
        "status": "Inactive"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <is_active>false</is_active>
        <is_completed>false</is_completed>
        <status>"Inactive"</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
is_active	**bool**
is_completed	**bool**
status	**string**

Get broadcasts list

Description

Retrieves broadcast mailings list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/list`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
from_id	**int**	**2**
limit	**int**	**10**
format	**string**	**“json”**
type	**string**	**“broadcast”**
is_active	**bool**	**true**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`
groups	**[]String**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "limit": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 3,
            "name": "New mailing 1",
            "type": "broadcast",
            "is_active": false,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 5,
            "name": "New mailing 2",
            "type": "broadcast",
            "is_active": false,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 17
}
```


XML


```
<xml>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
        <id>3</id>
        <is_active>false</is_active>
        <name>New mailing 1</name>
        <type>broadcast</type>
    </data>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
        <id>5</id>
        <is_active>false</is_active>
        <name>New mailing 2</name>
        <type>broadcast</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>17</total_count>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Mailing ID**
name	**string**	**Mailing name**
type	**string**	**Mailing type**
is_active	**bool**	**Mailing active flag**
for_confirm	**bool**	**Messages with opt-in link flag**
total_count	**int**	**Total mailings count**

Parameter	Type	
ui_tags	**[]String**	**List of tags**
groups	**[]String**	**List of groups**
scampaign_id	**int**	**Campaign ID in which the mailing is enga** **(The parameter is returned if the mailing**

Get broadcast information

Description

Retrieves detailed information about a broadcast mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**11768**
format	**string**	**“json”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": "87"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>87</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "name": "sms ",
        "is_active": false,
        "data_id": 13,
        "data_type": "segment",
        "content": [
            {
                "subject": "",
                "from_email": "",
                "from_name": "",
                "replyto_email": "",
                "replyto_name": "",
                "message_id": 27
            }
        ],
        "send_channel": "sms",
        "channel": "sms",
        "push_sending_mode": "fanout",
        "resource_id": 36,
        "subid": "",
        "limit": 0,
        "aglimit_count": 0,
        "aglimit_percent": 0,
        "start_schedule": {
            "start_mins": [
                38
            ],
            "start_hours": [
                16
            ],
            "start_mdays": [],
            "start_wdays": [],
            "start_months": [],
            "start_years": [],
            "tz": "",
            "profile": false,
            "losers": false
        },
        "for_confirm": false,
        "policy_id": 0,
        "description": "",
        "virtual_sender_id": 0,
        "is_random_tracking_prefix": false,
        "custom_tracking_domains": [],
        "watcher_list_id": 0,
        "watcher_limit": 0,
        "blind_carbon_copy": [],
        "unconditional_sending": false,
        "skip_unique_checking": false,
        "sub_cats_filter": null,
        "sub_fields_filter": null,
        "id": 179,
        "updated_user": "roman",
        "update_time": "2024-05-28T13:40:57.3Z"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <aglimit_count>0</aglimit_count>
        <aglimit_percent>0</aglimit_percent>
        <channel>"sms"</channel>
        <content>
            <from_email>""</from_email>
            <from_name>""</from_name>
            <message_id>27</message_id>
            <replyto_email>""</replyto_email>
            <replyto_name>""</replyto_name>
            <subject>""</subject>
        </content>
        <data_id>13</data_id>
        <data_type>"segment"</data_type>
        <description>""</description>
        <for_confirm>false</for_confirm>
        <id>179</id>
        <is_active>false</is_active>
        <is_random_tracking_prefix>false</is_random_tracking_prefix>
        <limit>0</limit>
        <name>"sms "</name>
        <policy_id>0</policy_id>
        <push_sending_mode>"fanout"</push_sending_mode>
        <resource_id>36</resource_id>
        <send_channel>"sms"</send_channel>
        <skip_unique_checking>false</skip_unique_checking>
        <start_schedule>
            <losers>false</losers>
            <profile>false</profile>
            <start_hours>16</start_hours>
            <start_mins>38</start_mins>
            <tz>""</tz>
        </start_schedule>
        <subid>""</subid>
        <unconditional_sending>false</unconditional_sending>
        <update_time>"2024-05-28T13:40:57.3Z"</update_time>
        <updated_user>"roman"</updated_user>
        <virtual_sender_id>0</virtual_sender_id>
        <watcher_limit>0</watcher_limit>
        <watcher_list_id>0</watcher_list_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**
is_active	**bool**
resource_id	**int**
data_id	**int**
data_type	**string**
content	**JSON array**
send_channel	**string**
channel	**string**
for_confirm	**bool**
subid	**string**
start_schedule	**JSON object**
limit	**int**
aglimit_count	**int**
agllimit_percent	**float**
policy_id	**int**
push_sending_mode	**string**
descr	**string**
attributes	**object**
scampaign_id	**int**
Smart settings	
virtual_sender_id	**int**

Parameter	Type
is_random_tracking_prefix	**bool**
custom_tracking_domains	**JSON array**
blind_carbon_copy	**JSON array**
unconditional_sending	**bool**
watcher_list_id	**int**
watcher_limit	**int**

Create broadcast mailing

Description

Creates a new broadcast mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/add`

Request parameters

Parameter	Type	Example	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	Y
format	**string**	**"json"**	N
name	**string**	**"Broadcast mailing name"**	Y
is_active	**bool**	**true**	N
data_type	**string**	**"list"**	Y
data_id	**int**	**12**	Y
resource_id	**int**	**24**	N
content	**JSON array**		Y
send_channel	**string**	**"email"** **"sms"** **"push"**	Y
content_type	**string**	**"email"**	N
send_email_content	**bool**	**true**	N
send_push_content	**bool**	**true**	N
send_sms_content	**bool**	**true**	N
subid	**string**	**"Adv_1"**	N
start_schedule	**JSON object**		N
limit	**int**	**0**	N
aglimit_count	**int**	**0**	N
agllimit_percent	**float**	**99.0**	N
policy_id	**int**	**1**	N
push_sending_mode	**string**	**"fanout" (default)** **"last"**	N
attributes	**object**	**"season":"summer_2024"**	N
Smart settings			
virtual_sender_id	**int**	**12**	N
is_random_tracking_prefix	**bool**	**false**	N
custom_tracking_domains	**JSON array**	**["trk.example.com", "trk2.example.com"]**	N
blind_carbon_copy	**JSON array**	**["john@example.com", "george@example.com"]**	N
unconditional_sending	**bool**	**true**	N
watched_list_id	**int**	**12**	N
watcher_limit	**int**	**100**	N

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "example@example.com",
            "from_name": "Victor",
            "message_id": 15,
            "replyto_email": "example@example.com",
            "replyto_name": "Victoria",
            "subject": "Hello"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": 1,
    "data_type": "list",
    "is_active": true,
    "limit": 1,
    "name": "New broadcast",
    "resource_id": 21,
    "send_channel": "email",
    "start_schedule": {
        "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```

XML

```
<xml>
    <token>{{token}}</token>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>example@example.com</from_email>
        <from_name></from_name>
        <message_id>14</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name></replyto_name>
        <subject>Hello!</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <is_active>true</is_active>
    <limit>1</limit>
    <name>Broadcast</name>
    <resource_id>21</resource_id>
    <send_channel>email</send_channel>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_hours>4</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 90
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>90</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Update broadcast mailing

Description

Updates mailing details, changes an existing mailing settings.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/update`

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**1**
format	**string**	**"json"**
name	**string**	**"Broadcast mailing name"**
is_active	**bool**	**true**
data_type	**string**	**"list"**
data_id	**int**	**12**
resource_id	**int**	**24**
content	**JSON array**	
subid	**string**	**"Adv_1"**
start_schedule	**JSON object**	
limit	**int**	**0**
aglimit_count	**int**	**0**
agllimit_percent	**float**	**99.0**
policy_id	**int**	**1**
push_sending_mode	**string**	**"fanout" (default)** **"last"**
attributes	**object**	**"season":"summer_2024"**
Smart settings		
virtual_sender_id	**int**	**12**
is_random_tracking_prefix	**bool**	**false**
custom_tracking_domains	**JSON array**	**["trk.example.com", "trk2.example.c**
blind_carbon_copy	**JSON array**	**["john@example.com", "george@exar**
unconditional_sending	**bool**	**true**
watched_list_id	**int**	**12**
watcher_limit	**int**	**100**

INFO

You cannot change the channel in a saved mailing. When passing such parameters as send_channel, con

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "example@example.com",
            "from_name": "Alexandra",
            "message_id": 14,
            "replyto_email": "example@example.com",
            "replyto_name": "Victoria",
            "subject": "Hi!"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": 1,
    "data_type": "list",
    "id": 90,
    "is_active": true,
    "limit": 10,
    "name": "Broadcast",
    "resource_id": 20,
    "start_schedule": {
        "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>broadcast1@altcraft.com</from_email>
        <from_name>Margorita</from_name>
        <message_id>14</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name>Victoria</replyto_name>
        <subject>Good morning!</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <id>89</id>
    <is_active>true</is_active>
    <limit>10</limit>
    <name>Broadcast</name>
    <resource_id>20</resource_id>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_hours>4</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 90
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>90</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text
saved_id

Launch a broadcast mailing

Description

Launches a broadcast mailing instantly. A mailing must be active.
`override_source_type` and `override_source_id` parameters override mailing audience for the run.
`override_limit` parameter sets sending limitations for the current run.
If no override options are set — mailings will use their default audience sources and sending limitations.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/broadcasts/start`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**12**
override_source_type	**enum**	**“list” или “segment”**
override_source_id	**int**	**5**
override_limit	**int**	**300**
content	**JSON object**	`{"field":"value"}`
attach	**JSON array**	**[]**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 90,
    "override_source_id": 1,
    "override_source_type": "list"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>90</id>
  <override_source_id>1</override_source_id>
  <override_source_type>list</override_source_type>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get regular mailings list

Description

Retrieves regular mailings list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/list/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
from_id	**int**	**11768**
limit	**int**	**10**
format	**string**	**“json”**
is_active	**bool**	**true**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`
groups	**[]String**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "limit": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 15,
            "name": "Regular mailing 1",
            "type": "regular",
            "is_active": true,
            "for_confirm": false,
            "ui_tags": [
                "B2C"
            ],
            "groups": [
                {
                    "id": 1,
                    "name": "subaccount"
                }
            ]
        },
        {
            "id": 78,
            "name": "Regular mailing 2",
            "type": "regular",
            "is_active": false,
            "for_confirm": false,
            "ui_tags": [
                "B2C"
            ],
            "groups": [
                {
                    "id": 1,
                    "name": "subaccount"
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 3
}
```


XML


```
<xml>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>1</id>
            <name>subaccount</name>
        </groups>
        <id>15</id>
        <is_active>true</is_active>
        <name>Regular mailing 1</name>
        <type>regular</type>
        <ui_tags>B2C</ui_tags>
    </data>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>1</id>
            <name>subaccount</name>
        </groups>
        <id>78</id>
        <is_active>false</is_active>
        <name>Regular mailing 2</name>
        <type>regular</type>
        <ui_tags>B2C</ui_tags>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>3</total_count>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Mailing ID**

Parameter	Type	
name	**string**	**Mailing name**
type	**string**	**Mailing type**
is_active	**bool**	**Mailing active flag**
for_confirm	**bool**	**Messages with opt-in link flag**
total_count	**int**	**Total mailing count**
ui_tags	**[]String**	**Tag list**
groups	**[]String**	**Group list**
scampaign_id	**int**	**Campaign ID in which the mailing is enga (The parameter is returned if the mailing**

Get regular mailing information

Description

Retrieves a regular mailing detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**11768**
format	**string**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 15
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>15</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "name": "Regular mailing name",
        "is_active": true,
        "data_id": 2,
        "data_type": "segment",
        "content": [
            {
                "subject": "Hi!",
                "from_email": "broadcast@altcraft.com",
                "from_name": "",
                "replyto_email": "broadcast@altcraft.com",
                "replyto_name": "",
                "message_id": 15
            }
        ],
        "send_channel": "email",
        "channel": "email",
        "resource_id": 20,
        "subid": "",
        "limit": 10,
        "aglimit_count": 0,
        "aglimit_percent": 0,
        "start_schedule": {
            "start_mins": [
                10
            ],
            "start_hours": [
                12
            ],
            "start_mdays": [],
            "start_wdays": [],
            "start_months": [],
            "start_years": [],
            "tz": "",
            "profile": false,
            "losers": false
        },
        "for_confirm": false,
        "policy_id": 0,
        "description": "",
        "virtual_sender_id": 1,
        "is_random_tracking_prefix": false,
        "custom_tracking_domains": [
            "trk.example.com",
            "trk.example.org"
        ],
        "watcher_list_id": 0,
        "watcher_limit": 0,
        "blind_carbon_copy": [],
        "unconditional_sending": false,
        "skip_unique_checking": false,
        "sub_cats_filter": [],
        "sub_fields_filter": null,
        "id": 15,
        "updated_user": "roman",
        "update_time": "2024-04-03T15:35:54.024Z"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <aglimit_count>0</aglimit_count>
        <aglimit_percent>0</aglimit_percent>
        <channel>"email"</channel>
        <content>
            <from_email>"broadcast@altcraft.com"</from_email>
            <from_name>""</from_name>
            <message_id>15</message_id>
            <replyto_email>"broadcast@altcraft.com"</replyto_email>
            <replyto_name>""</replyto_name>
            <subject>"Hi!"</subject>
        </content>
        <custom_tracking_domains>"trk.example.com"</custom_tracking_domains>
        <custom_tracking_domains>"trk.example.org"</custom_tracking_domains>
        <data_id>2</data_id>
        <data_type>"segment"</data_type>
        <description>""</description>
        <for_confirm>false</for_confirm>
        <id>15</id>
        <is_active>false</is_active>
        <is_random_tracking_prefix>false</is_random_tracking_prefix>
        <limit>10</limit>
        <name>"Regular mailing name"</name>
        <policy_id>0</policy_id>
        <push_sending_mode>"fanout"</push_sending_mode>
        <resource_id>20</resource_id>
        <send_channel>"email"</send_channel>
        <skip_unique_checking>false</skip_unique_checking>
        <start_schedule>
            <losers>false</losers>
            <profile>false</profile>
            <start_hours>12</start_hours>
            <start_mins>10</start_mins>
            <tz>""</tz>
        </start_schedule>
        <subid>""</subid>
        <unconditional_sending>false</unconditional_sending>
        <update_time>"2024-05-20T09:17:53.326Z"</update_time>
        <updated_user>"roman"</updated_user>
        <virtual_sender_id>1</virtual_sender_id>
        <watcher_limit>0</watcher_limit>
        <watcher_list_id>0</watcher_list_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**
is_active	**bool**
resource_id	**int**
data_id	**int**
data_type	**string**
content	**JSON array**
send_channel	**string**
channel	**string**
for_confirm	**bool**
subid	**string**
start_schedule	**JSON object**
limit	**int**
aglimit_count	**int**
agllimit_percent	**float**
policy_id	**int**
push_sending_mode	**string**
descr	**string**
Smart settings	
virtual_sender_id	**int**

Parameter	Type
is_random_tracking_prefix	**bool**
custom_tracking_domains	**JSON array**
blind_carbon_copy	**JSON array**
unconditional_sending	**bool**
watcher_list_id	**int**
watcher_limit	**int**
attributes	**object**
scampaign_id	**int**

Create regular mailing

Description

Creates a new regular mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/add/`

Request parameters

Parameter	Type	Example	
token	string	"abcdefghijklmnqrstuvwxyz"	Y
format	string	"json"	N
name	string	"Broadcast mailing name"	Y
is_active	bool	true	N
data_type	string	"list"	Y
data_id	int	12	Y
resource_id	int	24	N
content	JSON array		Y
send_channel	string	"email" "sms" "push"	Y
content_type	string	"email"	N
send_email_content	bool	true	N
send_push_content	bool	true	N
send_sms_content	bool	true	N
subid	string	"Adv_1"	N
start_schedule	JSON object		N
limit	int	0	N
aglimit_count	int	0	N
agllimit_percent	float	99.0	N
policy_id	int	1	N
push_sending_mode	string	"fanout" (default) "last"	N
attributes	object	"season":"summer_2024"	N
Smart settings			
virtual_sender_id	int	12	N
is_random_tracking_prefix	bool	false	N
custom_tracking_domains	JSON array	["trk.example.com", "trk2.example.com"]	N
blind_carbon_copy	JSON array	["john@example.com", "george@example.com"]	N
unconditional_sending	bool	true	N
watched_list_id	int	12	N
watcher_limit	int	100	N

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "regular@altcraft.com",
            "from_name": "Paul",
            "message_id": 15,
            "replyto_email": "regular@altcraft.com",
            "replyto_name": "Roman",
            "subject": "Hello"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": 1,
    "data_type": "list",
    "is_active": true,
    "limit": 1,
    "name": "Regular mailing #1",
    "resource_id": 20,
    "send_channel": "email",
    "start_schedule": {
         "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>example@example.com</from_email>
        <from_name>Paul</from_name>
        <message_id>15</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name>Roman</replyto_name>
        <subject>Hello</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <is_active>true</is_active>
    <limit>1</limit>
    <name>Regular mailing #2</name>
    <resource_id>20</resource_id>
    <send_channel>email</send_channel>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_hours>4</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 92
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>93</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Update regular mailing

Description

Updates a regular mailing. Changes mailing settings.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/update`

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**int**	**1**
format	**string**	**"json"**
name	**string**	**"Broadcast mailing name"**
is_active	**bool**	**true**
data_type	**string**	**"list"**
data_id	**int**	**12**
resource_id	**int**	**24**
content	**JSON array**	
subid	**string**	**"Adv_1"**
start_schedule	**JSON object**	
limit	**int**	**0**
aglimit_count	**int**	**0**
agllimit_percent	**float**	**99.0**
policy_id	**int**	**1**
push_sending_mode	**string**	**"fanout" (default)** **"last"**
attributes	**object**	**"season":"summer_2024"**
Smart settings		
virtual_sender_id	**int**	**12**
is_random_tracking_prefix	**bool**	**false**
custom_tracking_domains	**JSON array**	**["trk.example.com", "trk2.example.c**
blind_carbon_copy	**JSON array**	**["john@example.com", "george@exar**
unconditional_sending	**bool**	**true**
watched_list_id	**int**	**12**
watcher_limit	**int**	**100**

INFO

You cannot change the channel in a saved mailing. When passing such parameters as send_channel, con

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "virtual_sender_id": 5,
    "content": [
        {
            "from_email": "regular@altcraft.com",
            "from_name": "Roman",
            "message_id": 14,
            "replyto_email": "regular@altcraft.com",
            "replyto_name": "Romana",
            "subject": "Hi!"
        }
    ],
    "custom_tracking_domains": [
        "trk.example.com",
        "trk.example.org"
    ],
    "data_id": {{db_id}},
    "data_type": "list",
    "id": 93,
    "is_active": true,
    "limit": 1,
    "name": "Regular mailing 1",
    "resource_id": 20,
    "start_schedule": {
        "start_hours": [
            1,
            2,
            3,
            4
        ],
        "start_mins": [
            1,
            2,
            3,
            4
        ],
        "start_mdays": [
            1
        ],
        "start_months": [
            12
        ],
        "start_wdays": [],
        "start_years": [
            2024
        ]
    }
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <virtual_sender_id>5</virtual_sender_id>
    <content array='true'>
        <from_email>example@example.com</from_email>
        <from_name>Roman</from_name>
        <message_id>14</message_id>
        <replyto_email>example@example.com</replyto_email>
        <replyto_name>Romana</replyto_name>
        <subject>Hi!</subject>
    </content>
    <custom_tracking_domains>trk.example.com</custom_tracking_domains>
    <custom_tracking_domains>trk.example.org</custom_tracking_domains>
    <data_id>1</data_id>
    <data_type>list</data_type>
    <id>93</id>
    <is_active>true</is_active>
    <limit>1</limit>
    <name>Regular mailing 1</name>
    <resource_id>20</resource_id>
    <start_schedule>
        <start_hours>1</start_hours>
        <start_hours>2</start_hours>
        <start_hours>3</start_hours>
        <start_mins>1</start_mins>
        <start_mins>2</start_mins>
        <start_mins>3</start_mins>
        <start_mins>4</start_mins>
        <start_mdays array='true'>1</start_mdays>
        <start_months array='true'>12</start_months>
        <start_wdays array='true'></start_wdays>
        <start_years array='true'>2024</start_years>
    </start_schedule>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 93
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>93</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text
saved_id

Launch a regular mailing

Description

Instantly starts a regular mailing if it is active. There is a one-time send.
`override_source_type` **and** `override_source_id` **parameters override mailing audience for the run.**
`override_limit` **parameter sets sending limitations for the current run.**
If no override options are set, mailings will use their default audience sources and sending limitations.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/regulars/start`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**12**
override_source_type	**enum**	**“list” или “segment”**
override_source_id	**int**	**5**
override_limit	**int**	**300**
content	**JSON object**	`{"field":"value"}`
attach	**JSON array**	**[]**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 94,
    "override_source_id": 1,
    "override_source_type": "list"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>93</id>
    <override_source_id>1</override_source_id>
    <override_source_type>list</override_source_type>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get trigger mailings list

Description

Retrieves trigger mailings list.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/list/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
from_id	**int**	**11768**
limit	**int**	**10**
format	**string**	**“json”**
is_active	**bool**	**true**
ui_tags	**[]String**	`[ "tag1", "tag2" ]`
groups	**[]String**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "limit": 2
}
```

XML

```
<xml>
   <token>abcdefghijklmnqrstuvwxyz</token>
   <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 12,
            "name": "New trigger mailing 3",
            "type": "api_call",
            "is_active": true,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 0,
                    "name": "main"
                }
            ]
        },
        {
            "id": 14,
            "name": "New planetary mailing",
            "type": "api_call",
            "is_active": true,
            "for_confirm": false,
            "ui_tags": [],
            "groups": [
                {
                    "id": 1,
                    "name": "subaccount"
                }
            ],
            "providers": [
                "google",
                "mozilla",
                "safari",
                "safari-native",
                "firefox-firebase",
                "chrome-firebase",
                "opera-firebase",
                "yandex-firebase",
                "samsung-firebase",
                "android-firebase",
                "ios-firebase",
                "ios-apns",
                "yandex-Appmetrica-device-id-ios",
                "yandex-Appmetrica-device-id-android",
                "yandex-Appmetrica-ios-ifa",
                "yandex-Appmetrica-google-aid",
                "yandex-Appmetrica-android-token",
                "yandex-Appmetrica-ios-token",
                "mr",
                "android-huawei",
                "ios-huawei",
                "android-rustore"
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 41
}
```


XML

```
<xml>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>0</id>
            <name>main</name>
        </groups>
        <id>12</id>
        <is_active>true</is_active>
        <name>New trigger mailing 3</name>
        <type>api_call</type>
    </data>
    <data>
        <for_confirm>false</for_confirm>
        <groups>
            <id>1</id>
            <name>subaccount</name>
        </groups>
        <id>14</id>
        <is_active>true</is_active>
        <name>New planetary mailing</name>
        <providers>google</providers>
        <providers>mozilla</providers>
        <providers>safari</providers>
        <providers>safari-native</providers>
        <providers>firefox-firebase</providers>
        <providers>chrome-firebase</providers>
        <providers>opera-firebase</providers>
        <providers>yandex-firebase</providers>
        <providers>samsung-firebase</providers>
        <providers>android-firebase</providers>
        <providers>ios-firebase</providers>
        <providers>ios-apns</providers>
        <providers>yandex-Appmetrica-device-id-ios</providers>
        <providers>yandex-Appmetrica-device-id-android</providers>
        <providers>yandex-Appmetrica-ios-ifa</providers>
        <providers>yandex-Appmetrica-google-aid</providers>
        <providers>yandex-Appmetrica-android-token</providers>
        <providers>yandex-Appmetrica-ios-token</providers>
        <providers>mr</providers>
        <providers>android-huawei</providers>
        <providers>ios-huawei</providers>
        <providers>android-rustore</providers>
        <type>api_call</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>41</total_count>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
id	**int**	**Trigger mailing ID**
name	**string**	**Mailing name**
type	**string**	**Trigger type**
for_confirm	**bool**	**Messages with opt-in links flag**
is_active	**bool**	**Mailing active flag**
ui_tags	**[]String**	**Tag list**
groups	**[]String**	**Group list**
workflow_id	**int**	**Scenario identifier in which the trigger ma (The parameter is returned if the mailing**
node_id	**int**	**Scenario node identifier in which the trigg (The parameter is returned if the mailing**
scampaign_id	**int**	**Campaign ID in which the mailing is enga (The parameter is returned if the mailing**

Get trigger mailing information

Description

Retrieves detailed information about a trigger mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
format	**string**	**“json”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 12
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>12</id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "call_url": "trk.test.test.altcraft.com/trg/465ea6fedf0a75fafe1a413943aae3696d95939038311994/<email|md5|xxh>",
    "content": [
      {
        "from_email": "",
        "from_name": "",
        "message_id": 41,
        "replyto_email": "",
        "replyto_name": "",
        "subject": ""
      }
    ],
    "send_channel": "email",
    "custom_tracking_domains": [],
    "id": 7,
    "is_active": true,
    "is_random_tracking_prefix": false,
    "list_id": 20,
    "name": "API call trigger",
    "resource_id": 44,
    "skip_minutes": 0,
    "subid": "API",
    "trigger_type": "api_call",
    "virtual_sender_id": 98
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <call_url>http://trk.test.test.altcraft.com:8095/trg/65512f8ec196edd63e50585b4bfd377365512f8ec196edd62f3de6b87e2969e3/
      <email|md5|xxh>
        </call_url>
          <channel>0</channel>
          <content>
              <message_id>8</message_id>
          </content>
          <for_confirm>false</for_confirm>
          <id>12</id>
          <is_active>true</is_active>
          <is_new_ch>false</is_new_ch>
          <is_new_contact>false</is_new_contact>
          <is_new_sub>false</is_new_sub>
          <is_random_tracking_prefix>false</is_random_tracking_prefix>
          <is_status_changed>false</is_status_changed>
          <list_id>1</list_id>
          <name>New trigger mailing 3</name>
          <policy_id>0</policy_id>
          <resource_id>4</resource_id>
          <send_channel>email</send_channel>
          <skip_minutes>0</skip_minutes>
          <skip_unique_checking>false</skip_unique_checking>
          <start_schedule>
              <losers>false</losers>
              <profile>false</profile>
          </start_schedule>
          <subscribe_resource_id>0</subscribe_resource_id>
          <trigger_type>api_call</trigger_type>
          <unconditional_sending>false</unconditional_sending>
          <update_time>2024-02-15T20:45:00.705Z</update_time>
          <updated_user>roman</updated_user>
          <virtual_sender_id>0</virtual_sender_id>
  </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**
is_active	**bool**
content	**JSON array**
trigger_type	**string**
skip_minutes	**int**
resource_id	**int**
send_channel	**string**
for_confirm	**bool**
policy_id	**int**
descr	**string**
push_sending_mode	**string**
attributes	**object**
workflow_id	**int**
node_id	**int**
scampaign_id	**int**
Smart settings	
virtual_sender_id	**string**
is_random_tracking_prefix	**bool**
custom_tracking_domains	**JSON array**
blind_carbon_copy	**JSON array**
unconditional_sending	**bool**

Trigger type parameters

These parameters depend on trigger type specified for a mailing (`trigger_type`)

Parameter	Type	
action	**string**	**Action based trigger typ**
update_query	**JSON array**	
link_name	**string**	**Message link name**
message_id	**int**	**Template ID**
list_id	**int**	**Database ID**
subscribe_resource_id	**int**	**Resource ID**
channel	**string**	**Trigger channel**
is_new_sub	**bool**	**New resource trigger**
is_new_ch	**bool**	**New channel trigger**
is_new_contact	**bool**	**New contact trigger**
is_status_changed	**bool**	**Subscription status cha**
trigger_statuses	**array(int)**	**Subscription status cha**
pixel_id	**int**	**Pixel ID**
pixel_target	**string**	**Pixel goal**

trigger_type: action

Represents message action based trigger.

It has 3 additional parameters.

Two of them are: `"action": "<action name>"` and `"message_id": 123`. They specify associated message and a me

`"link_name": "String"` property cpecifies link name when `"action": "click_sms"` or `"action": "clicklink_sms"`.

- **`"action": "open"` — email opened.**
- **`"action": "read"` — email read**
- **`"action": "click"` — any link clicked in email message**
- **`"action": "confirm"` — opt-in link clicked**
- **`"action": "click_sms"` — any click in SMS message**
- **`"action": "clicklink"` — a specific email link clicked, `"link_name": "String"`**
- **`"action": "clicklink_sms"` — a specific SMS link clicked, `"link_name": "String"`**

trigger_type: api_import

Represents API profile import trigger.

Retrieves import database ID `"list_id": 32`.

trigger_type: pixel

Represents pixel activation trigger type.

Has two additional parameters:

- **`"pixel_id": 64` — Altcraft pixel unique ID.**
- **`"pixel_target": "Target-name"` — Pixel goal name.**

trigger_type: update

Represents profile fields update trigger.

Has two additional parameters:

- **`"list_id": 32` — profile database ID.**
- **`"update_query": {"fieldname": "_postal_code", "fieldtype": "unicode", "operator": "equal", "value": "800028"}` — trig**

Update query

- **`"fieldname": "_postal_code"` — field name. Custom fields are used by their custom names. System and de**
- **`"fieldtype": "unicode"` — data type for custom and default database fields.**
- **`"operator": "equal"` — logical operator.**
- **`"value": "800028"` — value expected by trigger.**
- **`"submatch": "object_key"` — property path. When the value of this property changes, the trigger will fire.**

Field types

field types can be the following:

- **`unicode` — a string of characters or numbers, `"a1b2c"`**
- **`int` — an integer number, `-1234`**
- **`bool` — Boolean, `true or false`**
- **`date` — date in string format, `"2000-10-24T00:00:00.000Z"`**

- `tags` — **tags, an array of strings,** `["T1", "Tag2"]`
- `enum` — **a map composed of key-value pairs** `{"item_one": 1, "item_two": 2}`
- `object` — **object,** `{"A": "B", "C": "D"};`
- `arr_object` — **array of objects,** `[{"A": 1},{"B": 2}]`

Query logical operators

Operator	
“equal”	**value gets equal**
“not_equal”	**value gets not equal**
“not_exists_or_empty”	**value unset or field removed**
“not_empty_or_exists”	**value set or field added**
“tag_added_any”	**any tag added**
“tag_removed_any”	**any tag removed**
“equal_property”	**if the value of the specified property becomes equal to**
“not_equal_property”	**if the value of the specified property becomes not equal to**
“added_element”	**if a new element is added to the array**
“deleted_element”	**if one of the elements is removed from the array**

Create trigger mailing

Description

Creates a new trigger mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/add/`

Request parameters

Parameter	Type	Example	Required
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
format	**string**	**"csv"**	**No**
name	**string**	**"New Awesome Mailing"**	**Yes**
is_active	**bool**	**true**	**No**
content	**JSON array**	`[ { "from_email": "", "from_name": "", "message_id": 12, "replyto_email": "", "replyto_name": "", "subject": "Weather alert" } ]`	**Yes**
subid	**string**	**"Adv_1"**	**No**
trigger_type	**string**	**"api_call"**	**Yes**
skip_minutes	**int**	**0**	**No**
resource_id	**int**	**24**	**No**
send_channel	**string**	**"email" "sms" "push"**	**Yes**
content_type	**string**	**"email"**	**No**
send_email_content	**bool**	**true**	**No**
send_sms_content	**bool**	**true**	**No**
send_push_content	**bool**	**true**	**No**
policy_id	**int**	**1**	**No**
attributes	**object**	**"season":"summer_2024"**	**No**
push_sending_mode	**string**	**"fanout" (default) "last"**	**No**
Smart settings			
virtual_sender_id	**int**	**12**	**No**
is_random_trk_prefix	**bool**	**false**	**No**
custom_trk_domain	**JSON array**	`["trk.example.com", "trk2.example.com"]`	**No**
blind_carbon_copy	**JSON array**	`["john@example.com", "george@example.com"]`	**No**
unconditional_sending	**bool**	**true**	**No**

INFO

Unconditional sending is available only for On-premise and Off-premise licensed instances. To activate it

Trigger type parameters

Parameter	Type	
action	**string**	**Action based trigger**
update_query	**JSON array**	`{ “fieldname”: “custom_ID”, “fieldtype”: “int”, “operator”: “equal”, “value”: “2000” }`
link_name	**string**	**Message link name**
message_id	**int**	**Template ID**
list_id	**int**	**Database ID**
subscribe_resource_id	**int**	**Resource ID**
channel	**string**	**Subscription channel**
is_new_sub	**bool**	**New resource trigger**
is_new_ch	**bool**	**New channel trigger**
is_new_contact	**bool**	**New contact trigger**

Parameter	Type	
is_status_changed	**bool**	**Subscription status cha**
trigger_statuses	**array(int)**	**Subscription status cha**
pixel_id	**int**	**Pixel ID**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "name": "New trigger mailing №1",
  "content": [
    {
      "subject": "Maybe very important message",
      "from_email": "",
      "from_name": "James",
      "replyto_email": "",
      "replyto_name": "",
      "message_id": 14
    }
  ],
  "trigger_type": "api_call",
  "send_channel": "email",
  "list_id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <name>New trigger mailing №1</name>
    <content array='true'>
        <from_name>James</from_name>
        <message_id>15</message_id>
        <subject>Maybe very important message</subject>
    </content>
    <trigger_type>api_call</trigger_type>
    <send_channel>email</send_channel>
    <list_id>1</list_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 95,
        "trigger_link": "http://trk.test.test.altcraft.com:8095/trg/65512f8ec196edd6ac71fa5d068d355f65512f8ec196edd62f3de6b87e2969e3/<email|md5
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>96</saved_id>
        <trigger_link>http://trk.test.test.altcraft.com:8095/trg/65512f8ec196edd6fc63524eba02bff065512f8ec196edd62f3de6b87e2969e3/
            <email|md5|xxh>
            </trigger_link>
        </data>
        <error>0</error>
        <error_text>Successful operation</error_text>
    </xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**New trigger maili**

Parameter	Type	
trigger_link	**string**	**trigger link. Only**

Update trigger mailing

Description

Updates a trigger mailing. Changes mailing settings.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/update/`

Request parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
format	**string**	**"csv"**
id	**int**	**1**
name	**string**	**"New Awesome Mailing"**
is_active	**bool**	**true**
content	**JSON array**	`[ { "from_email": "", "from_name": "", "message_id": 12, "replyto_email": "", "replyto_name": "", "subject": "Weather alert" } ]`
subid	**string**	**"Adv_1"**
trigger_type	**string**	**"api_call"**
skip_minutes	**int**	**0**
resource_id	**int**	**24**
policy_id	**int**	**1**
push_sending_mode	**string**	**"fanout" (default)** **"last"**
attributes	**object**	**"season":"summer_2024"**
Smart settings		
virtual_sender_id	**int**	**12**
is_random_trk_prefix	**bool**	**false**
custom_trk_domain	**JSON array**	`["trk.example.com", "trk2.example.com"]`
blind_carbon_copy	**JSON array**	`["john@example.com", "george@example.com"]`
unconditional_sending	**bool**	**true**

NOTE

You cannot change the channel in a saved mailing. When passing such parameters as send_channel, con

Trigger type parameters

Parameter	Type	
action	**string**	**Action based trigger typ**
update_query	**JSON array**	`{ “fieldname”: “custom_ID”, “fieldtype”: “int”, “operator”: “equal”, “value”: “2000” }`
link_name	**string**	**Message link name**
message_id	**int**	**Template ID**
list_id	**int**	**Database ID**
subscribe_resource_id	**int**	**Subscription trigger ID**
channel	**string**	**Subscription channel**
is_new_sub	**bool**	**New resource trigger**
is_new_ch	**bool**	**New channel trigger**
is_new_contact	**bool**	**New contact trigger**
is_status_changed	**bool**	**Subscription status cha**
trigger_statuses	**array(int)**	**Subscription status cha**
pixel_id	**int**	**Pixel ID**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 42,
  "name": "New trigger mailing update",
  "content": [
    {
      "subject": "Maybe not very important message",
      "from_email": "",
      "from_name": "Alex",
      "replyto_email": "",
      "replyto_name": "",
      "message_id": 8
    }
  ],
  "trigger_type": "api_call",
  "list_id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>42</id>
    <name>New trigger mailing 3</name>
    <content array='true'>
        <from_name>James</from_name>
        <message_id>15</message_id>
        <subject>Maybe very important message</subject>
    </content>
    <trigger_type>api_call</trigger_type>
    <list_id>1</list_id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "saved_id": 42,
    "trigger_link": "http://trk.example.com:8095/trg/3a345a07f01657e7966566e657a34d023a345a07f01802e7/<email|md5|xxh>"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <saved_id>42</saved_id>
        <trigger_link>http://trk.example.com:8095/trg/3a345a07f01657e7966566e657a34d023a345a07f01802e7/&lt;email|md5|xxh&gt;</trigger_link>
    </data>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
saved_id	**int**	**Trigger mailing ID**
trigger_link	**string**	**Trigger mailing link. Only for trigge**

Trigger launch (API call)

Top priority instantly launched trigger mailings.

> **TIP**
> **To launch a trigger at the precise time of a new profile import you can use Import + trigger request.**

Version 1.1

Description

It has several advantages compared with API V. 1.0:

- **Requests can change mailing content for each launch.**
- **Sends messages to an audience instead of single profiles.**
- **Advanced audience matching — profile ID, custom fields, external database queries.**
- **API token in message body for advanced security.**

Static request URL `/api/v1.1/campaigns/triggers/start/` **as compared to a dynamically generated link bound to**

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/start/`

Request parameters

Parameter	Type	Example
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
id	**string**	**5511**
content	**JSON object**	`{ "field": "value" }`
attach	**JSON array**	`[]`
expiration	**string**	**"2026-02-22T21:00:00Z"**
custom_data	**JSON object**	`{ "x": "y"}`
Profile matching mode		
matching	**string**	**"email"** **"phone"** **"profile_id"** **"custom"**
email	**string**	**"john@example.com"**
phone	**string**	**"+79000000000"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**
field_name	**string**	**"CRM_ID"**
field_value	**int/string**	**"123420"**
resource_id	**int**	**24**

custom_request	**JSON object**	`{`
		`  "email": {`
		`  "$in": [`
		`  "asd@dsfsd.com",`
		`  "zxcv@sdfsd.com"`
		`  ]`
		`  }`
		`}`

subscription_filter	**JSON object**	**Push example:**
		`"subscription_filter": {`
		`  "subscription_id": "CAREFULWITHTHATAXE",`
		`  "provider": "Chrome",`
		`  "not_strict": true`
		`}`
		Email example:
		`"subscription_filter": {`
		`  "email": "example@example.com",`
		`  "not_strict": true`
		`}`
		SMS example:
		`"subscription_filter": {`
		`  "phone": "+79106135133",`
		`  "not_strict": false`
		`}`

Request example

This example triggers a mailing for a single subscriber:

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 84,
  "matching": "profile_id",
  "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>84</id>
  <matching>profile_id</matching>
  <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
</xml>
```

Send messages to subscribers with `Custom_IDs` from a list `["IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiWFIt`

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 84,
  "custom_request":{"Custom_ID": {"$in": ["IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiWFItJhr"]}}
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>84</id>
    <custom_request>{"Custom_ID": {"$in": ["IpHicQquekzYkufl", "xzFKLXeaJkCYHuMb", "SRNOwHVZiWFItJhr"]}}</custom_request>
</xml>
```

Insert text `"New videos on your favorite channels"` to replace `{apicontent.email_title}` message content variable. Attach files “日本.txt” and US.txt".

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 96,
    "matching": "profile_id",
    "profile_id": "6606e08f10bcdd88297057a7",
    "attach": [
        {
            "data": "data:text/csv; base64,5LuK5pel44GvCg==",
            "name": "日本.txt"
        },
        {
            "data": "data:text/csv; base64,SEVMTE8K",
            "name": "US.txt"
        }
    ],
    "content": {
        "data_lines": [
            "Channel A",
            "Channel B"
        ],
        "email_title": "New videos on your favorite channels"
    }
}
```

XML

```
<xml>
<token>abcdefghijklmnqrstuvwxyz</token>
  <id>96</id>
  <matching>profile_id</matching>
  <profile_id>6606e08f10bcdd88297057a7</profile_id>
  <attach>
    <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
    <name>日本.txt</name>
  </attach>
  <attach>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <name>US.txt</name>
  </attach>
  <content>
    <data_lines>Channel A</data_lines>
    <data_lines>Channel B</data_lines>
    <email_title>New videos on your favorite channels</email_title>
  </content>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Version 1.0

Description

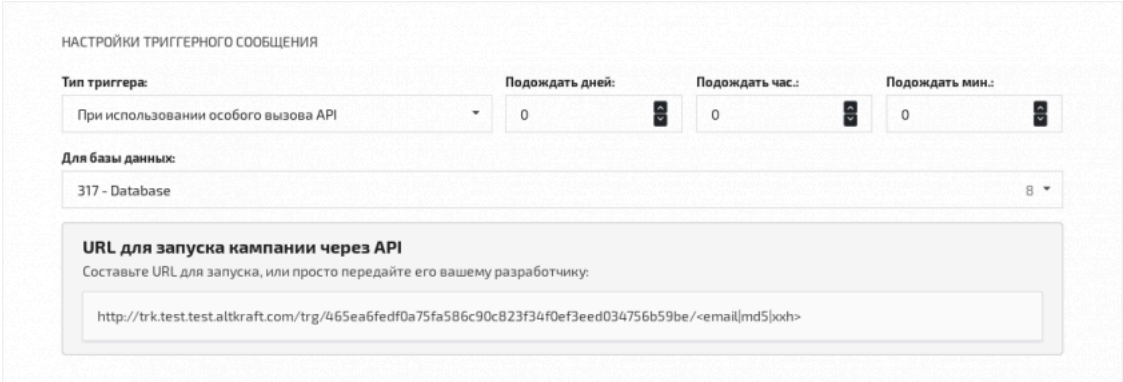
Methods: GET, POST

In this version, each API trigger has a unique link that is displayed after it is created.

`http://<your.tracking.domain>/trg/<base16_value_d1d93d9cca76a7c1742a2f23fbacd>/<email|md5|xxh>`

The link will change if you change the trigger database.

Using link 1.0 allows you to send a trigger to only one profile per request.



`"email"` **is a deprecated parameter.**

GET request

Sends a message with no optional parameters.

Matches profiles by email, md5 hash or XX hash. XX hash profile matching example:

`http://trk.example.org/trg/465ea6fedf0a75fa9ee0b5fb8f9d5fe89af34054909af8bb/dd78094ad0e36ca3`

POST JSON request

The same link is used as in the GET request.

`http://trk.example.org/trg/465ea6fedf0a75fa9ee0b5fb8f9d5fe89af34054909af8bb/dd78094ad0e36ca3`

With POST method you have an option to attach JSON data to message body.

You can override database field values for a mailing message or even update database fields.

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 96,
    "matching": "profile_id",
    "profile_id": "6606e08f10bcdd88297057a7",
    "fields": [
        {
            "name": "FirstName",
            "value": "Jason",
            "option": 0
        },
        {
            "name": "Order",
            "value": "123abc456-78",
            "option": 1
        }
    ]
}
```

XML

```
<xml>
  <token>2fefb577533d4cae919350c450755239</token>
  <id>96</id>
  <matching>profile_id</matching>
  <profile_id>6606e08f10bcdd88297057a4</profile_id>
  <fields>
    <name>FirstName</name>
    <value>Jason</value>
    <option>0</option>
  </fields>
  <fields>
    <name>Order</name>
    <value>123abc456-78</value>
    <option>1</option>
  </fields>
</xml>
```

- `"name"` **— profile database field name. If no field matches this name — it won't be used.**
- `"value"` **— profile database field value. Must match field data type.**
- `"option"` **— (0 | 1 | 2) determines data usage options.**
- `"option": 0` **— if a database field is not empty — request data will be ignored.**
- `"option": 1` **— request field value will be used in the message, but will not rewrite database value.**
- `"option": 2` **— request field value will be used in the message and will rewrite database value.**

Profile import + trigger mailing launch

Description

A new customer profile import followed by a trigger mailing launch.

URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/import_and_start`

Request parameters

Parameters	Type	Example	Required
Profile import			
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
data	**JSON object**	`{ "_fname": "John", "_lname": "Doe" }`	**Yes**
skip_triggers	**bool**	**true**	**No**
skip_invalid_subscriptions	**bool**	**true**	**No**
detect_geo	**bool**	**true**	**No**
Profile matching mode			
matching	**string**	**"email" "email_profile" "phone" "profile_id" "custom"**	**No, if matched by email from profile or subscriptio**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and subscrip "matching":"email_profile" - profile data only**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**
Trigger launch			
trigger_id	**int**	**13**	**Yes**
timeout	**JSON object**	`{ "days": 0, "hours": 0, "minutes": 5 }`	**No**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**
custom_data	**JSON object**	`{ "sending_time": "12:31:12" }`	**No**
content	**JSON object**	`{ "page": "registration_form" }`	**No**
attach	**JSON objects array**	`{ "data": "data:text/csv; base64,SEVMTE8K", "name": "Attach.txt" }`	**No**

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "matching": "email",
   "email": "example@example.com",
   "detect_geo": true,
   "data": {
      "_fname": "Olly",
      "_lname": "Lambert",
      "email": "example@example.com",
      "phones": [
         "+79000000000"
      ],
      "_bdate": "1990-02-22T21:00:00Z",
      "custom_field": "custom_value",
      "subscriptions": [
         {
            "channel": "email",
            "email": "example@example.com",
            "resource_id": 21
         },
         {
            "channel": "phone",
            "phone": "+79000000000",
            "resource_id": 21
         }
      ]
   },
   "trigger_id": 96,
   "timeout": {
         "days": 0,
         "hours": 0,
         "minutes": 5
   },
   "custom_data": {
      "sending_time": "12:31:12"
   },
     "attach": [
      {
         "data": "data:text/csv; base64,5LuK5pel44GvCg==",
         "name": "Attach.txt"
      }
   ]
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <matching>email</matching>
    <email>example@example.com</email>
    <detect_geo>true</detect_geo>
    <data>
        <_fname>Garry</_fname>
        <_lname>Sallivan</_lname>
        <_bdate>1990-02-22T21:00:00Z</_bdate>
        <email>example@example.com</email>
        <phones array='true'>+79000000027</phones>
        <custom_field>custom_value</custom_field>
        <subscriptions>
            <channel>email</channel>
            <email>testtriggers90@example.com</email>
            <resource_id>21</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>phone</channel>
            <phone>+79000000780</phone>
            <resource_id>21</resource_id>
        </subscriptions>
    </data>
    <trigger_id>96</trigger_id>
    <timeout>
        <days>0</days>
        <hours>0</hours>
        <minutes>5</minutes>
    </timeout>
    <custom_data>
        <sending_time>12:31:12</sending_time>
    </custom_data>
    <attach array='true'>
        <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
        <name>Attach.txt</name>
    </attach>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "profile_id": "54759eb3c090d83494e2d804"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>660feee02984cf021df478f3</profile_id>
</xml>
```

CAUTION

Subscription to the Resource that is used by the trigger mailing has to be sent within profile data object.

```
{
    "error": 400,
    "error_text": "No active subscriptions passed for trigger resource"
}
```

Response parameters

Parameter	
error	**int**
error_text	**string**
profile_id	**string**

Task for bulk trigger launch

Description

Launches a trigger mailing for multiple profiles.
This request is a batch version of the Trigger Launch (API call) request.
Returns a response with the task id, without waiting for completion.

> NOTE
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

URL request

Method: POST

`http://example.com/api/v1.1/campaigns/triggers/start_batch_task`

Request parameters

Parameter	Type	Example	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
format	**string**	**"json"**	**No**
id	**int**	**1**	**Yes**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**
attach	**JSON array of File object**	`[]`	**No**
content	**JSON object**	`{ "H1": "Weather alert", "H2": "Welcome" }`	**No**
custom_data	**JSON object**	`{ "field": "value" }`	**No**
data	**array**	`{ "_fname": "John", "_lname": "Doe" }`	**Yes**
Profile matching mode			
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email from profile or sub**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data a "matching":"email_profile" - profile data "matching":"email_sub" - subscription d**
phone	**string**	**"+79000000000"**	**"matching":"phone" - by phone number "matching":"phone_sub" - by phone nun**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**		**"matching":"custom"**
field_value			**"matching":"custom"**
custom_request	**JSON object**	`{ "email": { "$in": [ "asd@dsfsd.com", "zxcv@sdfsd.com" ] } }`	**When no matching specified**

Response parameters

Parameter
error
error_text
task_id

Request example (matching by email)

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 96,
    "format": "json",
    "matching": "email",
    "data": [
        {
            "email": "example1@example.com"
        },
        {
            "email": "example2@example.com"
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>96</id>
    <format>xml</format>
    <matching>email</matching>
    <data>
        <email>example1@example.com</email>
    </data>
    <data>
        <email>example2@altcraft.com</email>
    </data>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 96,
    "format": "json",
    "skip_triggers": false,
    "matching": "custom",
    "field_name": "Custom_ID",
    "data": [
        {
            "field_value": "001"
        },
        {
            "field_value": "002"
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>96</id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <data>
        <field_value>001</field_value>
    </data>
    <data>
        <field_value>002</field_value>
    </data>
</xml>
```

Response example

JSON

```
{
         "error": 0,
         "error_text": "Successful operation",
         "task_id": "30421978-7f6d-4f39-913f-fb6cad8e2477"
}
```


XML


```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <task_id>392b7e46-e4d9-4a07-959a-1e541138b684</task_id>
</xml>
```


Response example when a request constraint is triggered

JSON


```
{
    "error": 400,
    "error_debug": "http: request body too large",
    "error_text": "Invalid request"
}
```


XML


```
<xml>
  <error>400</error>
  <error_debug>http: request body too large</error_debug>
  <error_text>"Invalid request"</error_text>
</xml>
```

Task for bulk profiles import + trigger launch

Description

Add or update multiple profiles and then send a trigger mailing.
This request is a batch version of the Profile import + trigger mailing launch request.
Returns a response with the task id, without waiting for completion.

> NOTE
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

Request URL

Метод: POST

`http://example.com/api/v1.1/campaigns/triggers/import_and_start_batch_task`

Request parameters

Parameters	Type	Example	Required
Profile import			
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**

data	**JSON object**	<pre>[
		  {
		  "data": {
		  "_fname": "John",
		  "_lname": "Lambert",
		  "email": "profile4@example.com",
		  "phones": "+790000000000"
		  }
		  },
		  {
		  "data": {
		  "_fname": "Tom",
		  "_lname": "Hard",
		  "email": "profile3@example.com",
		  "phones": "+790000000001"
		  }
		  }
		]</pre>
			Yes

expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**
skip_triggers	**bool**	**true**	**No**
skip_invalid_subscriptions	**bool**	**true**	**No**
detect_geo	**bool**	**true**	**No**
Profile matching mode			
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email from profile or subscri**
email	**string**	**"john@example.com"**	**"matching":"email" - both profile data and sub** **"matching":"email_profile" - profile data only**
phone	**string**	**"+79000000000"**	**"matching":"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**"matching":"profile_id"**
field_name	**string**	**"CRM_ID"**	**"matching":"custom"**
field_value	**int/string**	**"123420"**	**"matching":"custom"**
resource_id	**int**	**24**	**No**
Trigger launch			
trigger_id	**int**	**13**	**Yes**

custom_data	**JSON object**	<pre>{
		  "sending_time": "12:31:12"
		}</pre>
			No

content	**JSON object**	<pre>{
		  "page": "registration_form"
		}</pre>
			No

attach	**JSON objects array**	<pre>{
		  "data": "data:text/csv; base64,SEVMTE8K",
		  "name": "Attach.txt"
		}</pre>
			No

Response parameters

Параметр	Тип
error_text	**string**
error	**int**
task_id	**string**

Request example (matching by email)

JSON


```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "matching": "email",
    "data": [
        {
            "data": {
                "_fname": "Victoria",
                "_lname": " Beckham",
                "email": "example1@example.com",
                "phones": "+79000000000",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example2@example.com",
                        "resource_id": 21
                    },
                    {
                        "channel": "phone",
                        "phone": "+79000005478",
                        "resource_id": 20
                    }
                ]
            }
        },
        {
            "data": {
                "_fname": "David",
                "_lname": " Beckham",
                "email": "example3@example.com",
                "phones": "+79000000001",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example14@example.com",
                        "resource_id": 21
                    },
                    {
                        "channel": "phone",
                        "phone": "+79000000000",
                        "resource_id": 20
                    }
                ]
            }
        }
    ],
    "trigger_id": 96,
    "timeout": {
        "days": 0,
        "hours": 0,
        "minutes": 5
    },
    "content": {
        "page": "registration_form"
    },
    "attach": [
        {
            "data": "data:text/csv; base64,5LuK5pel44GvCg==",
            "name": "Attach.txt"
        }
    ]
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <matching>email</matching>
    <data>
        <_fname>Brooklyn</_fname>
        <_lname>Beckham</_lname>
        <email>example3@example.com</email>
        <phones>+79000849376</phones>
        <subscriptions>
            <channel>email</channel>
            <email>example9@example.com</email>
            <resource_id>21</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>phone</channel>
            <phone>+79000005475</phone>
            <resource_id>20</resource_id>
        </subscriptions>
    </data>
    <data>
        <_fname>Romeo</_fname>
        <_lname>Beckham</_lname>
        <email>example32@example.com</email>
        <phones>+79000000651</phones>
        <subscriptions>
            <channel>email</channel>
            <email>@example.com</email>
            <resource_id>21</resource_id>
        </subscriptions>
        <subscriptions>
            <channel>phone</channel>
            <phone>+79000880000</phone>
            <resource_id>20</resource_id>
        </subscriptions>
    </data>
    <trigger_id>96</trigger_id>
    <timeout>
        <days>0</days>
        <hours>0</hours>
        <minutes>5</minutes>
    </timeout>
    <content>
        <page>registration_form</page>
    </content>
    <attach array='true'>
        <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
        <name>Attach.txt</name>
    </attach>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "trigger_id": 96,
    "format": "json",
    "skip_triggers": false,
    "detect_geo": true,
    "matching": "custom",
    "field_name": "Custom_id",
    "data": [
        {
            "data": {
                "_fname": "Tom",
                "_lname": "Cruise",
                "email": "example@example.com",
                "phones": "+79000000000",
                "Custom_id": "0002"
            }
        },
        {
            "data": {
                "_fname": "Lili",
                "_lname": "Hard",
                "email": "example@example.com",
                "phones": "+79000000001",
                "Custom_id": "0001"
            }
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <trigger_id>96</trigger_id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <detect_geo>true</detect_geo>
    <matching>custom</matching>
    <field_name>Custom_ID</field_name>
    <data>
        <_fname>Viola</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79000000000</phones>
        <Custom_ID>0002</Custom_ID>
    </data>
    <data>
        <_fname>Will</_fname>
        <_lname>Hard</_lname>
        <email>example@example.com</email>
        <phones>+79000000001</phones>
        <Custom_ID>0001</Custom_ID>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "task_id": "1c1a5861-6d4c-4d6f-bd6a-3ca4f46b217d"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <task_id>30421978-7f6d-4f39-913f-fb6cad8e2477</task_id>
</xml>
```

Response example when a request constraint is triggered

JSON

```
{
         "error":400,
         "error_debug":"http: request body too large",
         "error_text":"Invalid request"
}
```

XML

```
<xml>
    <error>400</error>
    <error_text>Invalid request</error_text>
</xml>
```

Checking task status

Request URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
task_id	**string**	**“task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "task_id": "task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <task_id>d521abe0-1abf-4e1d-b247-a790be66a424</task_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": {
            "error": 0,
            "error_text": "",
            "result": [
                {
                    "profile_id": "61e947333f62fc26d384452c"
                },
                {
                    "profile_id": "61e947333f62fc26d384452e"
                }
            ]
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <progress>100</progress>
        <result>
            <captured>0</captured>
            <status>ok</status>
        </result>
        <status>completed</status>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Bulk trigger launch

Description

Launches a trigger mailing for multiple profiles.
This request is a batch version of the Trigger Launch (API call) request.

> **NOTE**
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

URL request

Method: POST

`http://example.com/api/v1.1/campaigns/triggers/start_batch`

Request parameters

Parameter	Type	Example	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
format	**string**	**"json"**	**No**
trigger_id	**int**	**1**	**Yes**
expiration	**string**	**"2026-02-22T21:00:00Z"**	**No**
attach	**JSON array of File object**	`[]`	**No**
content	**JSON object**	`{ "H1": "Weather alert", "H2": "Welcome" }`	**No**
custom_data	**JSON object**	`{ "field": "value" }`	**No**
data	**array**	`[ {"email": "profile1@example.com"}, {"email": "profile2@example.com"} ]`	**Yes**
Profile matching mode			
matching	**string**	**"email" "email_profile" "email_sub" "phone" "phone_sub" "profile_id" "push_sub" "custom" "custom_sub" "email_phone" "email_phone_sub"**	**No, if matched by email from pr**

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**Profile IDs to wh**

Request example (matching by email)

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "trigger_id": 8,
    "format": "json",
    "matching": "email",
    "data": [
        {
            "email": "example1@example.com"
        },
        {
            "email": "example2@example.com"
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <trigger_id>85</trigger_id>
    <format>xml</format>
    <matching>email</matching>
    <data>
        <email>example1@example.com</email>
    </data>
    <data>
        <email>example2@example.com</email>
    </data>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "trigger_id": 96,
    "format": "json",
    "skip_triggers": false,
    "matching": "custom",
    "field_name": "Custom_ID",
    "data": [
        {
            "field_value": "003"
        },
        {
            "field_value": "002"
        }
    ]
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <trigger_id>85</trigger_id>
  <format>xml</format>
  <skip_triggers>false</skip_triggers>
  <matching>custom</matching>
  <field_name>Custom_ID</field_name>
  <data>
    <field_value>YY</field_value>
  </data>
    <data>
    <field_value>AA</field_value>
  </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "profile_id": "6606e08f10bcdd88297057a7"
        },
        {
            "profile_id": "66101437dc4594c2934da6da"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <result>
        <profile_id>66101437dc4594c2934da6da</profile_id>
    </result>
    <result>
        <profile_id>6606e08f10bcdd88297057a7</profile_id>
    </result>
</xml>
```

Response example when a request constraint is triggered

JSON

```
{
         "error":400,
         "error_debug":"http: request body too large",
         "error_text":"Invalid request"
}
```

XML

```
<xml>
   <error>400</error>
   <error_debug>http: request body too large</error_debug>
   <error_text>"Invalid request"</error_text>
</xml>
```

Bulk profiles import + trigger mailing launch

Description

Add or update multiple profiles and then send a trigger mailing.
This request is a batch version of the Profile import + trigger mailing launch request.

> **NOTE**
>
> **The request body size is limited by default to 10 Mb, exception for files — 100 Mb.**

Request URL

Method: POST

`http://example.com/api/v1.1/campaigns/triggers/import_and_start_batch`

Request parameters

Parameters	Type	Example
Profile import		
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
data	**JSON object**	`[ { "data": { "_fname": "John", "_lname": "Lambert", "email": "profile4@example.com", "phones": "+790000000000" } }, { "data": { "_fname": "Tom", "_lname": "Hard", "email": "profile3@example.com", "phones": "+790000000001" } } ]`
skip_triggers	**bool**	**true**
skip_invalid_subscriptions	**bool**	**true**
detect_geo	**bool**	**true**
subscription_filter	**JSON object**	**Push example:** `"subscription_filter": { "subscription_id": "CAREFULWITHTHATAXE", "provider": "Chrome", "not_strict": true }` **Email example:** `"subscription_filter": { "email": "example@example.com", "not_strict": true }` **SMS example:** `"subscription_filter": { "phone": "+79106135133", "not_strict": false }`
Profile matching mode		
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**
email	**string**	**"john@example.com"**
phone	**string**	**"+79000000000"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**
field_name	**string**	**"CRM_ID"**
field_value	**int/string**	**"123420"**
resource_id	**int**	**24**
Trigger launch		
trigger_id	**int**	**13**
timeout	**JSON object**	`{ "days": 0, "hours": 0, "minutes": 5 }`
expiration	**string**	**"2026-02-22T21:00:00Z"**
custom_data	**JSON object**	`{ "sending_time": "12:31:12" }`

Parameters	Type	Example

content	**JSON object**	<pre>{
		  "page": "registration_form"
		}</pre>

attach	**JSON objects array**	<pre>{
		  "data": "data:text/csv; base64,SEVMTE8K",
		  "name": "Attach.txt"
		}</pre>

Response parameters

Parameter	Type	
error_text	**string**	**Error**
error	**int**	**Error**
result	**array**	**IDs o**

Request example (matching by email)

JSON


```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "trigger_id": 96,
    "format": "json",
    "skip_triggers": false,
    "detect_geo": true,
    "matching": "email",
    "data": [
        {
            "data": {
                "_fname": "Lili",
                "_lname": "Collins",
                "email": "example@example.com",
                "phones": "+79000000000",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        },
        {
            "data": {
                "_fname": "Eva",
                "_lname": "Collins",
                "email": "example@example.com",
                "phones": "+79000000001",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        }
    ]
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <trigger_id>96</trigger_id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <detect_geo>true</detect_geo>
    <matching>email</matching>
<data>
    <_fname>Lili</_fname>
    <_lname>Collins</_lname>
    <email>example@example.com</email>
    <phones>+79000000000</phones>
        <subscriptions>
            <channel>email</channel>
            <email>example@example.com</email>
            <priority>0</priority>
            <resource_id>21</resource_id>
            <status>0</status>
        </subscriptions>
</data>
<data>
    <_fname>Eva</_fname>
    <_lname>Collins</_lname>
    <email>example@example.com</email>
    <phones>+79000000001</phones>
    <subscriptions>
        <channel>email</channel>
        <email>example@example.com</email>
        <priority>0</priority>
        <resource_id>21</resource_id>
        <status>0</status>
    </subscriptions>
</data>
</xml>
```

Request example (matching by custom field)

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "trigger_id": 85,
    "format": "json",
    "skip_triggers": false,
    "detect_geo": true,
    "matching": "custom",
    "field_name": "Custom_ID",
    "data": [
        {
            "data": {
                "_fname": "Adama",
                "_lname": "Fuller",
                "email": "example@example.com,
                "phones": "+79000000000",
                "Custom_ID": "345",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        },
        {
            "data": {
                "_fname": "Eva",
                "_lname": "Fuller",
                "email": "example@example.com",
                "phones": "+79000000001",
                "Custom_ID": "567",
                "subscriptions": [
                    {
                        "channel": "email",
                        "email": "example@example.com",
                        "priority": 0,
                        "resource_id": 21,
                        "status": 0
                    }
                ]
            }
        }
    ]
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <trigger_id>85</trigger_id>
    <format>xml</format>
    <skip_triggers>false</skip_triggers>
    <detect_geo>true</detect_geo>
    <matching>custom</matching>
    <field_name>custom_field</field_name>
    <data>
        <_fname>Irina</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+79000000000</phones>
        <custom_field>343</custom_field>
        <subscriptions array='true'>
            <channel>email</channel>
            <email>example@example.com</email>
            <priority>0</priority>
            <resource_id>21</resource_id>
            <status>0</status>
        </subscriptions>
    </data>
    <data>
        <_fname>Will</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+79000000001</phones>
        <custom_field>560</custom_field>
        <subscriptions array='true'>
            <channel>email</channel>
            <email>example@example.com</email>
            <priority>0</priority>
            <resource_id>21</resource_id>
            <status>0</status>
        </subscriptions>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "profile_id": "602103d65ae47a3bc65d5d83"
        },
        {
            "profile_id": "601ec71c5ae47a15d1c9b472"
        }
    ]
}
```

XML

```
<xml>
   <error>0</error>
   <error_text>Successful operation</error_text>
   <result>
      <element>
         <profile_id>602103d65ae47a3bc65d5d83</profile_id>
      </element>
      <element>
         <profile_id>601ec71c5ae47a15d1c9b472</profile_id>
      </element>
   </result>
</xml>
```

Response example when a request constraint is triggered

JSON

```
{
    "error": 400,
    "error_debug": "http: request body too large",
    "error_text": "Invalid request"
}
```


XML


```
<xml>
   <error>400</error>
   <error_debug>http: request body too large</error_debug>
   <error_text>"Invalid request"</error_text>
</xml>
```

Clone a trigger mailing

Description

Clones content and settings of a trigger mailing to a newly created mailing.

Request URL

Method: POST

`https://example.com/api/v1.1/campaigns/triggers/clone`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
name	**string**	**“Clone”**
is_active	**bool**	**false**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 96,
    "name": "Clone trigger mailing #1",
    "is_active": false
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <name>Clone trigger mailing #2</name>
    <id>95</id>
    <is_active>true</is_active>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 97
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>96</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Data array

Description

The `data` array is used in the following queries:

- **Task for bulk trigger launch**
- **Task for bulk profiles import + trigger launch**
- **Bulk trigger launch**
- **Bulk profiles import + trigger mailing launch**

The array consists of objects, each of which contains information about a separate profile. As the key and

Parameters

Parameter	Type	Example
email	**string**	**"john@example.com"**
phone	**string**	**"+79000000000"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**
field_value	**int/string**	**"12345"**
custom_request	**JSON object**	`{ "email": { "$in": [ "asd@dsfsd.com", "zxcv@sdfsd.com" ] } }`

Examples

email

JSON

```
 "data": [
                    {"email": "profile1@example.com"},
                    {"email": "profile2@example.com"}
          ]
```

XML

```
<data>
    <email>profile1@example.com</email>
</data>
<data>
    <email>profile2@example.com</email>
</data>
```

phone

JSON

```
 "data": [
                    {"phone": "+79000000000"},
                    {"phone": "+79000000001"}
          ]
```

XML

```
<data>
    <phone>+79000000000</phone>
</data>
<data>
    <phone>+79000000001</phone>
</data>
```

profile_id

JSON

```
 "data": [
                  {"profile_id": "60ae529b3f62fcdebdb159bb"},
                  {"profile_id": "843vfdkisfh934hfe834rhkk"}
         ]
```

XML

```
<data>
    <profile_id>+79000000000</profile_id>
</data>
<data>
    <profile_id>+79000000001</profile_id>
</data>
```

custom

JSON

```
"data": [
         {"field_value": "001"},
         {"field_value": "002"}
         ]
```

XML

```
<data>
    <field_value>001</field_value>
</data>
<data>
    <field_value>002</field_value>
</data>
```

Engage profile in scenario

Description

Engages a customer profile in scenario.

> **TIP**
>
> **When you launch the scenario using the API, the profiles will enter the “Default” start node.**

Request URL

Method: POST

`https://example.com/api/v1.1/workflows/start`

Request parameters

Parameter	Type	Example
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
format	**string**	**"csv"**
id	**int**	**12**
content	**JSON-object**	**{}**
attach	**JSON-массив**	**[]**
Profile matching mode		
matching	**string**	**"email"** **"phone"** **"profile_id"** **"custom"**
email	**string**	**"john@example.com"**
phone	**string**	**"+79000000000"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**
field_name	**string**	**"CRM_ID"**
field_value	**int/string**	**"123420"**
resource_id	**int**	**24**

subscription_filter	**JSON object**	**Push example:**
		`"subscription_filter": {`
		`"subscription_id": "CAREFULWITHTHATAXE",`
		`"provider": "Chrome",`
		`"not_strict": true`
		`}`
		Email example:
		`"subscription_filter": {`
		`"email": "example@example.com",`
		`"not_strict": true`
		`}`
		SMS example:
		`"subscription_filter": {`
		`"phone": "+79106135133",`
		`"not_strict": false`
		`}`

Request example 1

Engages a single profile matched by Profile ID.

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 11,
  "matching": "profile_id",
  "profile_id": "5a74876369d42613b31947b8"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>11</id>
  <matching>profile_id</matching>
  <profile_id>5a74876369d42613b31947b8</profile_id>
</xml>
```

Request example 2

Overwrites message subject by placing “New videos on your favorite channels” for “{apicontent.email_titl
Attaches “日本.txt” and “US.txt” files and a content object to be used in API content filter query.

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 12,
    "matching": "profile_id",
    "profile_id": "66102b984a5d12bd6cc0912f",
    "attach": [
        {
            "data": "data:text/csv; base64,5LuK5pel44GvCg==",
            "name": "日本.txt"
        },
        {
            "data": "data:text/csv; base64,SEVMTE8K",
            "name": "US.txt"
        }
    ],
    "content": {
        "data_lines": [
            "Channel A",
            "Channel B"
        ],
        "email_title": "New videos on your favorite channels"
    }
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>12</id>
  <matching>profile_id</matching>
  <profile_id>661027b04a5d12bd6cc09125</profile_id>
  <attach>
    <data>data:text/csv; base64,5LuK5pel44GvCg==</data>
    <name>日本.txt</name>
  </attach>
  <attach>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <name>US.txt</name>
  </attach>
  <content>
    <data_lines>Channel A</data_lines>
    <data_lines>Channel B</data_lines>
    <email_title>New videos on your favorite channels</email_title>
  </content>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Import and engage profile in scenario

Description

Use this request to import customer profile and engage it in scenario.

Request URL

Method: POST

`https://example.com/api/v1.1/workflows/import_and_start`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
content	**JSON-object**	`{}`
workflow_id	**Int**	**1**
matching	**String**	**“email”**
profile_id	**String**	**“5f19953a03e4227216278ba4”**
email	**String**	**“importStart9219@example.com”**
phone	**String**	**“+79001234567”**
field_name	**String**	**“CRM_ID”**
field_value	**Any**	**“12345”**
data	**Object**	`{ “_fname”: “John”, “_lname”: “Doe” }`
skip_triggers	**Bool**	**true**
skip_invalid_subscriptions	**Bool**	**true**
detect_geo	**Bool**	**true**
content	**Object**	`“content”:{ “one”: “My email = importStart9219@example.com”, “two”: “Timestamp: 1595512477”, “three”: “Three” }`
attach	**array of{name, flie_key, data, mimetype}**	

Request example

Import one customer profile and engage it in scenario.

JSON

```
{
  "token": "abcdefghkijlmnopqrstuv0123456789",
  "data": {
    "subscriptions": [
      {
        "email": "example001@example.com",
        "resource_id": 7
      }
    ]
  },
  "matching": "email",
  "email": "example001@example.com",
  "content": {
    "one": "My email = example001@example.com",
    "two": "Timestamp: 1595512477",
    "three": "Three"
  },
  "workflow_id": 12
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <data>
    <subscriptions array='true'>
      <email>example001@example.com</email>
      <resource_id>7</resource_id>
    </subscriptions>
  </data>
  <matching>email</matching>
  <email>example001@example.com</email>
  <content>
    <one>My email = example001@example.com</one>
    <two>Timestamp: 1595512477</two>
    <three>Three</three>
  </content>
  <workflow_id>12</workflow_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "5eeb54a2ce9448518e85fdad"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <profile_id>6613c13a4a5d12bd6cc09156</profile_id>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
profile_id	**string**

Service responses

Code	
0	**Operation is successful**
400,450	**Incorrect request**
401	**API token required**
402	**Tariff limit reached**
403	**No rights for this action**
404	**Object is not found in token vision zone**
409	**Other record with same unique attributes already exists**
413	**Database is not available in selected resource**
415	**Requested Content-Type is not supported**
435	**Ambiguous search, there are several objects with given attributes**
441	**Object belongs to another group**
500	**Internal service error**
501	**Method does not exist**

Batch import and engage profiles in a scenario

Description

The request allows you to import multiple profiles and immediately send them to a scenario. It is a batch

> **NOTE**
>
> **The request body is limited by default to 10MB, exception: files – 100MB**
> **It is not recommended to transfer more than 10,000 profiles in one request.**

URL

Method: POST

`https://example.com/api/v1.1/workflows/import_and_start_batch`

Request parameters

Parameter	Type	Example
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
format	**string**	**"json"**
workflow_id	**Int**	**1**
matching	**String**	**"email"**
profile_id	**String**	**"5f19953a03e4227216278ba4"**
email	**String**	**"importStart9219@example.com"**
phone	**String**	**"+79001234567"**
field_name	**String**	**"CRM_ID"**
field_value	**Any**	**"12345"**
data	**Object**	\{ "_fname": "John", "_lname": "Doe" }
skip_triggers	**Bool**	**true**
skip_invalid_subscriptions	**Bool**	**true**
detect_geo	**Bool**	**true**
content	**Object**	"content":\{ "one": "My email = importStart9219@example.com", "two": "Timestamp: 1595512477", "three": "Three" }
attach	**array of \{name, flie_key, data, mimetype}**	

Response parameters

Parameter	
error_text	**string**
error	**int**
task_id	**string**

Request example – “matching”:“email”

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "workflow_id": 12,
  "matching": "email",
  "data": [
    {
      "data": {
        "_fname": "Alisa",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79000000000"
      }
    },
    {
      "data": {
        "_fname": "Alex",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79000000000"
      }
    }
  ]
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <workflow_id>10</workflow_id>
  <matching>email</matching>
  <data>
    <_fname>John</_fname>
    <_lname>Lambert</_lname>
    <email>example@example.com</email>
  </data>
  <data>
    <_fname>Bert</_fname>
    <_lname>Lambert</_lname>
    <email>example@example.com</email>
  </data>
</xml>
```

Request example – “matching”:“custom”

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "workflow_id": 12,
  "matching": "custom",
  "field_name": "CRM_ID",
  "data": [
    {
      "data": {
        "_fname": "Alina",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79006574081",
        "CRM_ID": "00066"
      }
    },
    {
      "data": {
        "_fname": "Victor",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79000005422",
        "CRM_ID": "00076"
      }
    }
  ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <workflow_id>12</workflow_id>
    <matching>custom</matching>
    <field_name>New_field_7</field_name>
    <data>
        <_fname>Polina</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79006574082</phones>
        <New_field_7 string = "true" >00067</New_field_7>
    </data>
    <data>
        <_fname>Dariya</_fname>
        <_lname>Lambert</_lname>
        <email>example@example.com</email>
        <phones>+79000005423</phones>
        <New_field_7 string = "true">00077</New_field_7>
    </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation",
    "result": [
        {
            "profile_id": "6613c4754a5d12bd6cc0915f"
        },
        {
            "profile_id": "6613c4754a5d12bd6cc09163"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <result>
        <profile_id>6613c4bf4a5d12bd6cc09167</profile_id>
    </result>
    <result>
        <profile_id>6613c4bf4a5d12bd6cc0916b</profile_id>
    </result>
</xml>
```

Response example when a request limitation is triggered

JSON

```
{
  "error": 400,
  "error_debug": "http: request body too large",
  "error_text": "Invalid request"
}
```

XML

```
<xml>
   <error>400</error>
   <error_debug>http: request body too large</error_debug>
   <error_text>Invalid request</error_text>
</xml>
```

Task for batch import and engaging profiles in the scenario

Description

Adding or updating multiple profiles and then sending them to the scenario. Returns a response with the t

> **NOTE**
>
> **The request body is limited by default to 10mb, exception: files are 100mb.**
> **It is not recommended to transmit more than 10,000 profiles per request.**

URL

Method: POST

`https://example.com/api/v1.1/workflows/import_and_start_batch_task`

Request parameters

Parameter	Type	Example	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Yes**
format	**string**	**"json"**	**No**
workflow_id	**Int**	**1**	**Yes**
matching	**String**	**"email"**	**Yes**
profile_id	**String**	**"5f19953a03e4227216278ba4"**	**"matching"**
email	**String**	**"importStart9219@example.com"**	**"matching"** **"matching"**
phone	**String**	**"+79001234567"**	**"matching"**
field_name	**String**	**"CRM_ID"**	**"matching"**
field_value	**Any**	**"12345"**	**"matching"**
data	**Object**	\{ "_fname": "John", "_lname": "Doe" }	**No**
skip_triggers	**Bool**	**true**	**No**
skip_invalid_subscriptions	**Bool**	**true**	**No**
detect_geo	**Bool**	**true**	**No**
content	**Object**	"content":\{ "one": "My email = importStart9219@example.com", "two": "Timestamp: 1595512477", "three": "Three" }	**No**
custom_data	**JSON object**	"custom_data": \{ "field": "value", }	**No**
attach	**array of \{name, flie_key, data, mimetype}**		**No**

Returned parameters

Parameter	
error_text	**string**
error	**int**
task_id	**string**

Request example – “matching”:“email”

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "workflow_id": 12,
  "matching": "email",
  "data": [
    {
      "data": {
        "_fname": "Herbert",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79862014852"
      }
    },
    {
      "data": {
        "_fname": "Bert",
        "_lname": "Lambert",
        "email": "example@example.com",
        "phones": "+79862014853"
      }
    }
  ]
}
```

XML

```
<xml>
<token>abcdefghijklmnqrstuvwxyz</token>
  <workflow_id>12</workflow_id>
  <matching>email</matching>
  <data>
      <_fname>Herb</_fname>
      <_lname>Limbert</_lname>
      <email>example@example.com</email>
      <phones>+79847076617</phones>
  </data>
  <data>
      <_fname>Bert</_fname>
      <_lname>Limbert</_lname>
      <email>example@example.com</email>
      <phones>+79817076617</phones>
  </data>
</xml>
```

Request example – “matching”:“custom”

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "workflow_id": 12,
    "matching": "custom",
    "field_name": "New_field_7",
    "data": [
        {
            "data": {
                "_fname": "Kristina",
                "_lname": "Lambert",
                "email": "example@example.com",
                "phones": "+79568150395",
                "New_field_7": "00015"
            }
        },
        {
            "data": {
                "_fname": "Christopher",
                "_lname": "Lambert",
                "email": "example@example.com",
                "phones": "+79568150396",
                "New_field_7": "00016"
            }
        }
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <workflow_id>12</workflow_id>
    <matching>custom</matching>
    <field_name>CRM_ID</field_name>
    <data>
        <_fname>Naomy</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+798144890880</phones>
        <CRM_ID>7830</CRM_ID>
    </data>
    <data>
        <_fname>Nataly</_fname>
        <_lname>Depp</_lname>
        <email>example@example.com</email>
        <phones>+73755334351</phones>
        <CRM_ID>0934</CRM_ID>
    </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "task_id": "30421978-7f6d-4f39-913f-fb6cad8e2477"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <task_id>675231ce-63b4-4bca-a153-10d87b938d26</task_id>
</xml>
```

Example of a response when a request restriction is triggered

JSON

```
{
  "error": 400,
  "error_debug": "http: request body too large",
  "error_text": "Invalid request"
}
```

XML

```
<xml>
  <error>400</error>
  <error_debug>http: request body too large</error_debug>
  <error_text>Invalid request</error_text>
</xml>
```

Checking the status of the task

URL

Method: POST

`https://example.com/api/v1.1/tasks/status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnopqrstuvwxyz”**
task_id	**string**	**“task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "task_id": "task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5"
}
```


XML


```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <task_id>task_b2f9fk6a-2479-4a63-b592-f15d4e3b69c5</task_id>
</xml>
```


Response example

JSON


```
{
    "data": {
        "status": "completed",
        "progress": 100,
        "result": {
            "error": 0,
            "error_text": "",
            "result": [
                {
                    "profile_id": "61e947333f62fc26d384452c"
                },
                {
                    "profile_id": "61e947333f62fc26d384452e"
                }
            ]
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
    <data>
        <status>completed</status>
        <progress>100</progress>
        <result>
            <error>0</error>
            <error_text></error_text>
            <result>
                <profile_id>61e947333f62fc26d384452c</profile_id>
            </result>
            <result>
                <profile_id>61e947333f62fc26d384452e</profile_id>
            </result>
        </result>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Get scenarios list

Description

Retrieves all created scenarios list.

Request URL

Method: POST

`https://example.com/api/v1.1/workflows/list`

Request parameters

Parameter	Type	Exan
token	**string**	`"abcdefghijklmnqrstuvwxyz"`
format	**string**	`"json"`
from_id	**int**	`1`
limit	**int**	`1`
groups	**[]string**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`
ui_tags	**[]string**	`["tag1", "tag2"]`
attributes	**object**	`{"season": {"options": ["spring_2024", "summer_2024"]}}`

Request example

JSON

```
{
    "token": "2fefb577533d4cae919350c450755239",
    "from_id": 10,
    "limit": 1,
    "ui_tags": [
        "B2C"
    ]
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <from_id>10</from_id>
    <limit>1</limit>
    <ui_tags array='true'>B2C</ui_tags>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 10,
            "updated_user": "ivan_ivanov",
            "update_time": "2025-05-21T09:17:53.362Z",
            "name": "Government tip",
            "list_id": 1,
            "is_active": true,
            "ui_tags": [
                "B2C"
            ],
            "mode": "skip",
            "groups": [
                {
                    "ID": 2,
                    "title": "Sales and promotion"
                }
            ],
            "nodes": [
                {
                    "name": "DEFAULT_START_NODE_TITLE",
                    "type": "start",
                    "node_id": 0
                },
                {
                    "name": "ANY_IMPORT_START_NODE_TITLE",
                    "type": "start",
                    "node_id": 2
                },
                {
                    "templates": [
                        1
                    ],
                    "name": "Welcome letter",
                    "type": "mailing",
                    "node_id": 3,
                    "campaign_id": 17
                },
                {
                    "name": "1 d.0 h.0 m.",
                    "type": "delay",
                    "node_id": 4
                },
                {
                    "name": "Is the subscription confirmed?",
                    "type": "condition",
                    "node_id": 5
                },
                {
                    "name": "Got the SMS reminder?",
                    "type": "condition",
                    "node_id": 6
                },
                {
                    "templates": [
                        5
                    ],
                    "name": "SMS message",
                    "type": "mailing",
                    "node_id": 7,
                    "campaign_id": 30
                },
                {
                    "name": "1 d.0 h.0 m.",
                    "type": "delay",
                    "node_id": 8
                },
                {
                    "name": "Application to a call center or other product",
                    "type": "feedback",
                    "node_id": 9
                },
                {
                    "title": "Guidelines",
                    "type": "feedback",
                    "node_id": 10
                },
                {
                    "name": "Specify code",
```

```
                    "type": "promo_attach",
                    "node_id": 11
                },
                {
                    "templates": [
                        22
                    ],
                    "name": "Offer",
                    "type": "mailing",
                    "node_id": 12
                    "campaign_id": 39
                }
                {
                    "name": "Go to scenario #29",
                    "type": "workflow_jump",
                    "node_id": 13,
                    "next_workflow_id": 29,
                    "next_node_id": 0
                }
            ]
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": 11,
    "total_count": 20
}
```

XML

```
<xml>
    <data>
        <groups>
            <id>2</id>
            <updated_user>ivan_ivanov</updated_user>
            <update_time>2025-05-21T09:17:53.362Z</update_time>
            <name>Sales and promotion</name>
        </groups>
        <id>10</id>
        <is_active>true</is_active>
        <list_id>1</list_id>
        <mode>skip</mode>
        <name>Administrative chain</name>
        <nodes>
            <name>DEFAULT_START_NODE_TITLE</name>
            <node_id>0</node_id>
            <type>start</type>
        </nodes>
        <nodes>
            <name>ANY_IMPORT_START_NODE_TITLE</name>
            <node_id>2</node_id>
            <type>start</type>
        </nodes>
        <nodes>
            <campaign_id>17</campaign_id>
            <name>Welcome letter</name>
            <node_id>3</node_id>
            <templates>1</templates>
            <type>mailing</type>
        </nodes>
        <nodes>
            <name>1 д.0 ч.0 м.</name>
            <node_id>4</node_id>
            <type>delay</type>
        </nodes>
        <nodes>
            <name>Is the subscription confirmed?</name>
            <node_id>5</node_id>
            <type>condition</type>
        </nodes>
        <nodes>
            <name>Got the SMS reminder?</name>
            <node_id>6</node_id>
            <type>condition</type>
        </nodes>
        <nodes>
            <campaign_id>30</campaign_id>
            <name>SMS message</name>
            <node_id>7</node_id>
            <templates>5</templates>
            <type>mailing</type>
        </nodes>
        <nodes>
            <name>1 д.0 ч.0 м.</name>
            <node_id>8</node_id>
            <type>delay</type>
        </nodes>
        <nodes>
            <name>Application to a call center
            or other product</name>
            <node_id>9</node_id>
            <type>feedback</type>
        </nodes>
        <nodes>
            <name>Guidelines</name>
            <node_id>10</node_id>
            <type>feedback</type>
        </nodes>
        <nodes>
            <name>Specify code</name>
            <node_id>11</node_id>
            <type>promocode_attach</type>
        </nodes>
        <nodes>
            <campaign_id>39</campaign_id>
            <name>Offer</name>
            <node_id>12</node_id>
            <templates>22</templates>
            <type>mailing</type>
        </nodes>
```

```
        <nodes>
            <name>Go to scenario #29</name>
            <next_node_id>0</next_node_id>
            <next_workflow_id>29</next_workflow_id>
            <node_id>13</node_id>
            <type>workflow_jump</type>
        </nodes>
        <ui_tags>B2C</ui_tags>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>11</next_from_id>
    <total_count>20</total_count>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**
list_id	**int**
is_active	**bool**
mode	**string**
ui_tags	**array**
groups	**array of objects**
nodes	**array of objects**
total_count	**int**
next_from_id	**int**
node_id	**int**
campaign_id	**int**
templates	**int**
next_workflow_id	**int**
next_node_id	**int**
updated_user	**string**
update_time	**string**
attributes	**object**
scampaign_id	**int**

Activate scenario

Description

The request is used to activate a scenario.

URL

Method: POST

`https://example.com/api/v1.1/workflows/activate`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
skip_profiles	**bool**	**true**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "skip_profiles": true
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
    <skip_profiles>true</skip_profiles>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Deactivate scenario

Description

The request is used to deactivate a scenario.

URL

Method: POST

`https://example.com/api/v1.1/workflows/deactivate`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
skip_profiles	**bool**	**true**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "skip_profiles": true
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
    <skip_profiles>true</skip_profiles>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get profile tier in a loyalty program

Description

Retrieves the current tier of a loyalty program member within the requested tiers group.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/tiers`

Request parameters

Parameter	Type	Example	
loyalty_program_id	**int**	**123**	**Yes**
tiers_group_id	**int**	**10**	**Yes**
matching	**string**	**“email”**	**Yes**
field_name	**string**	**“user_id”**	**Yes**
field_value	**string / int**	**123456**	**Yes**
email	**string**	**“user@example.com”**	**Yes**
phone	**string**	**“+1234567890”**	**Yes**
profile_id	**string**	**“profile123”**	**Yes**
token	**string**	**“abc123def456”**	**Yes**

Request example

JSON

```
{
  "loyalty_program_id": 123,
  "tiers_group_id": 10,
  "matching": "email",
  "email": "user@example.com",
  "phone": "+1234567890",
  "profile_id": "profile123",
  "token": "abc123def456"
}
```

XML

```
<xml>
    <loyalty_program_id>123</loyalty_program_id>
    <tiers_group_id>10</tiers_group_id>
    <matching>email</matching>
    <email>user@example.com</email>
    <phone>+1234567890</phone>
    <profile_id>profile123</profile_id>
    <token>abc123def456</token>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": [
    {
      "tiers_group_type": "cumulative",
      "tier_eid": "silver1-2",
      "tier_id": 13,
      "tier_order": 1,
      "reached_at": "2024-06-15T09:12:34Z"
    },
    {
      "tiers_group_type": "cumulative",
      "tier_eid": "gold1-3",
      "tier_id": 14,
      "tier_order": 2,
      "reached_at": "2024-09-01T12:34:56Z"
    }
  ]
}
```


XML


```
<xml>
    <data>
        <tiers_group_type>cumulative</tiers_group_type>
        <tier_eid>silver1-2</tier_eid>
        <tier_id>13</tier_id>
        <tier_order>1</tier_order>
        <reached_at>2024-06-15T09:12:34Z</reached_at>
    </data>
    <data>
        <tiers_group_type>cumulative</tiers_group_type>
        <tier_eid>gold1-3</tier_eid>
        <tier_id>14</tier_id>
        <tier_order>2</tier_order>
        <reached_at>2024-09-01T12:34:56Z</reached_at>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
tiers_group_type	**string**
tier_eid	**string**
tier_id	**int**
tier_order	**int**
reached_at	**string**

Export points transactions

Description

Creates a task to generate a report of transactions for the selected points currency over a specified perio

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/transactions_export`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
points_currency_eid	**string**	**“bonus_points”**
points_currency_iid	**int**	**1**
from_id	**string**	**“a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8”**
date_from	**string (RFC3339)**	**“2023-09-01T00:00:00Z”**
date_to	**string (RFC3339)**	**“2023-09-30T23:59:59Z”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "points_currency_eid": "bonus_points",
  "date_from": "2023-09-01T00:00:00Z",
  "date_to": "2023-09-30T23:59:59Z",
  "from_id": "a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8"
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>bonus_points</points_currency_eid>
  <date_from>2023-09-01T00:00:00Z</date_from>
  <date_to>2023-09-30T23:59:59Z</date_to>
  <from_id>a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8</from_id>
</xml>
```

Response example

JSON

```
{
  "data": {
    "task_id": "task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>task_b2f9fd6a-2499-4a63-b592-f15d4e3b69c5</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
task_id	**string (UUID)**

Expiring points for a period

Description

Returns the number of points that are subject to expiration during the specified period for the selected po

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/expiring_points`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
points_currency_eid	**string**	**“bonus_points”**
points_currency_iid	**int**	**1**
date_from	**string (RFC3339)**	**“2023-09-01T00:00:00Z”**
date_to	**string (RFC3339)**	**“2023-09-30T23:59:59Z”**
matching	**string**	**“email”**
field_name	**string**	**“user_id”**
field_value	**string / int**	**“12345”**
email	**string**	**“user@example.com”**
phone	**string**	**“+1234567890”**
profile_id	**string**	**“profile_123”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "points_currency_eid": "bonus_points",
  "matching": "email",
  "email": "user@example.com",
  "date_from": "2023-09-01T00:00:00Z",
  "date_to": "2023-09-30T23:59:59Z"
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>bonus_points</points_currency_eid>
  <matching>email</matching>
  <email>user@example.com</email>
  <date_from>2023-09-01T00:00:00Z</date_from>
  <date_to>2023-09-30T23:59:59Z</date_to>
</xml>
```

Response example

JSON

```
{
  "data": {
    "expiring_points": "150.24"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <expiring_points>150.24</expiring_points>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
expiring_points	**string**	**Number of points that will exp**

Get profile account transactions

Description

Returns the transaction history for a loyalty program member’s points account. Supports pagination and f

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/transactions`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
points_currency_eid	**string**	**“bonus_points”**
points_currency_iid	**int**	**1**
from_id	**string**	**“a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8”**
limit	**int**	**20**
matching	**string**	**“email”**
field_name	**string**	**“user_id”**
field_value	**string / int**	**“12345”**
email	**string**	**“user@example.com”**
phone	**string**	**“+1234567890”**
profile_id	**string**	**“profile_123”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "points_currency_eid": "bonus_points",
  "matching": "email",
  "email": "user@example.com",
  "limit": 20
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>bonus_points</points_currency_eid>
  <matching>email</matching>
  <email>user@example.com</email>
  <limit>20</limit>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "transaction_id": "a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8",
      "transaction_eid": "trans_12345",
      "created_at": "2023-09-01T12:34:56Z",
      "transaction_type": "accrual",
      "amount": "100.1",
      "balance_before": "500.5",
      "balance_after": "600.6",
      "promo_iid": 42,
      "promo_eid": "promo_2023",
      "description": "Points accrual for a purchase",
      "activation_date": "2023-09-01T12:34:56Z",
      "expiration_date": "2023-12-31T23:59:59Z",
      "source_type": "order",
      "source_iid": 789,
      "source_sid": "order_123",
      "source_subiid": 1,
      "is_temp": false,
      "custom_json": {"key": "value"},
      "product_eid": "prod_123",
      "sku_eid": "sku_456",
      "canceled": false
    }
  ],
  "next_from_id": "b2c3d4e5-f6g7-8901-h2i3-j4k5l6m7n8o9",
  "total_count": 150,
  "error": 0,
  "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <transaction_id>a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8</transaction_id>
    <transaction_eid>trans_12345</transaction_eid>
    <created_at>2023-09-01T12:34:56Z</created_at>
    <transaction_type>accrual</transaction_type>
    <amount>100.1</amount>
    <balance_before>500.5</balance_before>
    <balance_after>600.6</balance_after>
    <promo_iid>42</promo_iid>
    <promo_eid>promo_2023</promo_eid>
    <description>Points accrual for a purchase</description>
    <activation_date>2023-09-01T12:34:56Z</activation_date>
    <expiration_date>2023-12-31T23:59:59Z</expiration_date>
    <source_type>order</source_type>
    <source_iid>789</source_iid>
    <source_sid>order_123</source_sid>
    <source_subiid>1</source_subiid>
    <is_temp>false</is_temp>
    <custom_json>{"key": "value"}</custom_json>
    <product_eid>prod_123</product_eid>
    <sku_eid>sku_456</sku_eid>
    <canceled>false</canceled>
  </data>
  <next_from_id>b2c3d4e5-f6g7-8901-h2i3-j4k5l6m7n8o9</next_from_id>
  <total_count>150</total_count>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
transaction_id	**string (UUID)**
transaction_eid	**string**
created_at	**string (RFC3339)**
transaction_type	**string**
amount	**string**
balance_before	**string**
balance_after	**string**

Parameter	Type
promo_iid	**int**
promo_eid	**string**
description	**string**
activation_date	**string (RFC3339)**
expiration_date	**string (RFC3339)**
source_type	**string**
product_eid	**string**
sku_eid	**string**
canceled	**boolean**
next_from_id	**string (UUID)**
total_count	**int**

Get trigger promotions list

Description

Returns a list of loyalty program trigger promotions. Pagination and filtering by active status are supporte

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/trigger_promotions`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
from_id	**int**	**0**
limit	**int**	**20**
is_active	**boolean**	**true**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "limit": 20,
  "is_active": true
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <limit>20</limit>
  <is_active>true</is_active>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "id": 5,
      "name": "Trigger promotion: Bonus for purchase",
      "description": "Bonus accrual for each purchase",
      "external_id": "trigger_promo_2023_q3",
      "start_date": "2023-09-01T00:00:00Z",
      "end_date": "2023-09-30T23:59:59Z",
      "is_active": true,
      "redeem_rules": [
        {
          "points_currency_iid": 1,
          "points_currency_eid": "curr1",
          "static_amount": "100.16",
          "min_amount": "50",
          "max_amount": "500"
        }
      ],
      "accrual_rules": [
        {
          "points_currency_iid": 1,
          "points_currency_eid": "curr1",
          "static_amount": "100",
          "min_amount": "50",
          "max_amount": "500"
        }
      ]
    }
  ],
  "next_from_id": 8,
  "total_count": 134,
  "error": 0,
  "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <id>5</id>
    <name>Trigger promotion: Bonus for purchase</name>
    <description>Bonus accrual for each purchase</description>
    <external_id>trigger_promo_2023_q3</external_id>
    <start_date>2023-09-01T00:00:00Z</start_date>
    <end_date>2023-09-30T23:59:59Z</end_date>
    <is_active>true</is_active>
  </data>
  <next_from_id>8</next_from_id>
  <total_count>134</total_count>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
id	**int**
name	**string**
description	**string**
external_id	**string**
start_date	**string (RFC3339)**
end_date	**string (RFC3339)**
is_active	**boolean**
redeem_rules	**array**
accrual_rules	**array**
next_from_id	**int**
total_count	**int**

Accrue points to a member

Description

Accrues points to a loyalty program member’s balance using a trigger-based promotion.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/accrue_points`

Request parameters

Parameter	Type	Example	
token	**string**	**“abc123def456”**	**Yes**
loyalty_program_id	**int**	**123**	**Yes**
promotion_id	**int**	**25**	**No**
points_currency_iid	**int**	**1**	**No**
points_currency_eid	**string**	**“stars”**	**No**
amount	**string**	**“100”**	**Yes**
is_temp	**boolean**	**false**	**No**
temp_timeout_sec	**int**	**123**	**No**
created_at	**string**	**“2024-09-01T12:34:56Z”**	**No**
activation_date	**string**	**“2024-09-01T12:34:56Z”**	**No**
expiration_date	**string**	**“2024-12-31T23:59:59Z”**	**No**
transaction_eid	**string**	**“uiuiuiiu-ouiihfs-sdfsdf”**	**No**
external_data	**object**	`{"order_id": "123"}`	**No**
matching	**string**	**“email”**	**Yes**
field_name	**string**	**“user_id”**	**Yes**
field_value	**string / int**	**123456**	**Yes**
email	**string**	**“user@example.com”**	**Yes**
phone	**string**	**“+1234567890”**	**Yes**
profile_id	**string**	**“profile123”**	**Yes**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "promotion_id": "PROMO123",
  "points_currency_eid": "stars",
  "amount": "100",
  "matching": "email",
  "email": "user@example.com",
  "is_temp": false,
  "transaction_eid": "uiuiuiiu-ouiihfs-sdfsdf",
  "external_data": {
    "order_id": "ORDER-123"
  }
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <promotion_id>PROMO123</promotion_id>
  <points_currency_eid>stars</points_currency_eid>
  <amount>100</amount>
  <matching>email</matching>
  <email>user@example.com</email>
  <is_temp>false</is_temp>
  <transaction_eid>uiuiuiiu-ouiihfs-sdfsdf</transaction_eid>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance_before": "500.00",
    "balance_after": "600.50"
  }
}
```


XML


```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance_before>500.00</balance_before>
    <balance_after>600.50</balance_after>
  </data>
</xml>
```


Response parameters

Parameter	
error	**int**
error_text	**string**
transaction_id	**string**
balance_before	**string**
balance_after	**string**

Redeem member points

Description

Redeems points from a loyalty program member’s account using a trigger-based promotion.

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/redeeme_points`

Request parameters

Parameter	Type	Example	
token	**string**	**“abc123def456”**	**Yes**
loyalty_program_id	**int**	**123**	**Yes**
promotion_id	**int**	**123**	**No**
points_currency_iid	**int**	**1**	**No**
points_currency_eid	**string**	**“stars”**	**No**
amount	**string**	**“100”**	**Yes**
is_temp	**boolean**	**false**	**No**
temp_timeout_sec	**int**	**123**	**No**
created_at	**string**	**“2024-09-01T12:34:56Z”**	**No**
transaction_eid	**string**	**“uiuiuiiu-ouiihfs-sdfsdf”**	**No**
external_data	**object**	`{"order_id": "123"}`	**No**
matching	**string**	**“email”**	**Yes**
field_name	**string**	**“user_id”**	**Yes**
field_value	**string / int**	**123456**	**Yes**
email	**string**	**“user@example.com”**	**Yes**
phone	**string**	**“+1234567890”**	**Yes**
profile_id	**string**	**“profile123”**	**Yes**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "promotion_id": "PROMO123",
  "points_currency_eid": "stars",
  "amount": "100",
  "matching": "email",
  "email": "user@example.com",
  "is_temp": false,
  "transaction_eid": "uiuiuiiu-ouiihfs-sdfsdf",
  "external_data": {
    "order_id": "ORDER-123"
  }
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <promotion_id>PROMO123</promotion_id>
  <points_currency_eid>stars</points_currency_eid>
  <amount>100</amount>
  <matching>email</matching>
  <email>user@example.com</email>
  <is_temp>false</is_temp>
  <transaction_eid>uiuiuiiu-ouiihfs-sdfsdf</transaction_eid>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance_before": "500.00",
    "balance_after": "399.50"
  }
}
```


XML


```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance_before>500.00</balance_before>
    <balance_after>399.50</balance_after>
  </data>
</xml>
```


Response parameters

Parameter	Ty
error	**int**
error_text	**string**
transaction_id	**string**
balance_before	**string**
balance_after	**string**

Commit temporary transaction

Description

Commits a temporary transaction created with the `is_temp` flag. If a temporary points accrual transaction

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/commit_transaction`

Request parameters

Parameter	Type	
transaction_id	**string**	**“TRANS12345”**
transaction_eid	**string**	**“TRANS12345”**

(At least one identifier must be provided.)

Request example

JSON

```
{
  "transaction_id": "TRANS12345"
}
```

XML

```
<xml>
  <transaction_id>TRANS12345</transaction_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance": "399.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance>399.50</balance>
  </data>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Erro**
error_text	**string**	**Erro**
transaction_id	**string**	**Con**
balance	**string**	**Cur**

Preliminary Order Calculation

Description

The `preorder` method allows you to perform a preliminary order calculation considering promotions and lo
The request result can be used to display final amounts, discounts, and write-offs before creating the fina

> **TIP**
> **This method does not create an order in the system; it returns the calculated state of the order with app**

URL

Method: POST

`https://example.com/api/v1.1/market/orders/preorder`

Request Parameters

Parameter	Type	Example
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
external_id	**string**	**"12"**
matching	**string**	**"email" "phone" "profile_id" "custom"**
email	**string**	**"example@example.com"**
phone	**string**	**"+79000000000"**
profile_id	**string**	**"696e304f547840a7286619e0"**
data	**JSON object**	`\{ "endpoint": "site", "status": "new", "currency": "RUB", "total_price": "139999.00", "lines": [...] \}`

> **TIP**
> **The `lines` field is required and must contain at least one order item.**

data object

Parameter	Type	Example
endpoint	**string**	**"bot", "website"**
status	**string**	**"new", "confirmed", "paid"**
currency	**string**	**"RUB", "USD", "EUR"**
total_price	**string**	**"2500.00"**
final_total_price	**string**	**"2150.00"**
delivery_cost	**string**	**"300.00"**
create_time	**string (ISO 8601)**	**"2026-02-18T13:01:05.141Z"**
region	**string**	**"region_moscow"**
lines	**array**	**See "lines array" section below**
promocodes	**array**	**See "promocodes array" section below**
applied_promotions	**array**	**See "applied_promotions array" section below**
redeem_points	**array**	**See "redeem_points array" section below**
custom_fields	**object**	`{ "field1": "value1", "field2": "value2" }`
tags	**array**	`[ "summer_sale", "first_order" ]`

lines array

The array contains information about the order items. Must contain at least 1 item.
Each item is a separate object within the array with the following parameters:

Parameter	Type	
external_id	**string**	**"line_1"**
product	**string**	**"iphone17pmb256"**
sku	**string**	**"IPHONE17-256-BLACK"**
line_number	**integer**	**1**
count	**integer**	**1**

base_price_per_item	**string**	**"139999.00"**
min_price_per_item	**string**	**"130000.00"**
final_price_per_item	**string**	**"139500.00"**
status	**string**	**"new"**
custom_fields	**object**	**\{"color": "black", "size": "XL"\}**
applied_promotions	**array**	**See "applied_promotions array" section below**

*** — one of the parameters is required:** `product` **or** `sku`

promocodes array

The array contains objects with information about promo codes.
A full description of the promo code structure can be found in this article.

Parameter	Type	
code	**string**	**"SUMMER20"**
pool_id	**integer**	**42**

redeem_points array

An array with information about loyalty point redemption. Each object describes a separate redemption.
A full description of the point redemption structure can be found in this article.

Parameter	Type	
points_currency_iid	**integer**	**1**
points_currency_eid	**string**	**"bonus1"**
name	**string**	**"Bonus po**
amount	**integer**	**499**

*** — one of the parameters is required:** `points_currency_iid` **or** `points_currency_eid`

applied_promotions array

> TIP
> **The** `applied_promotions` **array can be contained both in the** `data` **object and in the** `lines` **array within that o**

An array of promotions applied to the order. Each object in the array describes a separate promotion.

Parameter	Type	Example
type	**string**	**"discount", "accrue_points", "redeem_points", "code_issue", "message"**
promotion	**object**	`"promotion": { "external_id": "promo_1", "name": "Summer Sale" }`
amount	**string**	**"150.00"**
points_info	**object**	`"points_info": { "points_currency_iid": 1, "points_currency_eid": "bonus1", "name": "Bonus points", "amount": "50" }`

The structure of the `points_info` **object is as follows:**

Parameter	Type	
points_currency_iid	**integer**	1
points_currency_eid	**string**	"
name	**string**	"
amount	**string**	"
expiration_date	**string (ISO 8601)**	"

*** — one of the parameters is required:** `points_currency_iid` **or** `points_currency_eid`

Request Example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "external_id": "12",
  "matching": "email",
  "email": "example@example.com",
  "data": {
    "endpoint": "site",
    "status": "new",
    "delivery_cost": "0.00",
    "total_price": "139999.00",
    "currency": "RUB",
    "redeem_points": [
      {
        "points_currency_iid": 1,
        "amount": 499
      }
    ],
    "lines": [
      {
        "external_id": "line___ASTGRf",
        "product": "iphone17pmb256",
        "base_price_per_item": "139999.00",
        "min_price_per_item": "0.00",
        "final_price_per_item": "139999.00",
        "count": 1,
        "custom_fields": {},
        "currency": "RUB"
      }
    ]
  }
}
```


XML


```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <external_id>12</external_id>
  <matching>email</matching>
  <email>example@example.com</email>
  <data>
    <endpoint>site</endpoint>
    <status>new</status>
    <delivery_cost>0.00</delivery_cost>
    <total_price>139999.00</total_price>
    <currency>RUB</currency>
    <redeem_points>
      <points_currency_iid>1</points_currency_iid>
      <amount>499</amount>
    </redeem_points>
    <lines>
      <external_id>line___ASTGRf</external_id>
      <product>iphone17pmb256</product>
      <base_price_per_item>139999.00</base_price_per_item>
      <min_price_per_item>0.00</min_price_per_item>
      <final_price_per_item>139999.00</final_price_per_item>
      <count>1</count>
      <currency>RUB</currency>
    </lines>
  </data>
</xml>
```


Response Example

JSON

```
{
    "data": {
        "profile": {
            "profile_id": "696e304f547840a7286619e0"
        },
        "order": {
            "processing_target": "calculated",
            "total_price": "139999.00",
            "final_total_price": "139500.00",
            "delivery_cost": "0.00",
            "currency": "RUB",
            "lines": [
                {
                    "external_id": "line___ASTGRf",
                    "line_number": 0,
                    "count": 1,
                    "product": "iphone17pmb256",
                    "base_price_per_item": "139999.00",
                    "min_price_per_item": "0.00",
                    "final_price_per_item": "139500.00",
                    "custom_fields": {},
                    "applied_promotions": [
                        {
                            "type": "accrue_points",
                            "promotion": {
                                "external_id": "bonus",
                                "name": "Point Accrual"
                            },
                            "points_info": {
                                "points_currency_iid": 1,
                                "points_currency_eid": "bonus1",
                                "name": "Main Point",
                                "amount": "6999"
                            }
                        },
                        {
                            "type": "redeem_points",
                            "promotion": {
                                "external_id": "bonus",
                                "name": "Point Redemption"
                            },
                            "points_info": {
                                "points_currency_iid": 1,
                                "points_currency_eid": "bonus1",
                                "name": "Main Point",
                                "amount": "499"
                            }
                        }
                    ]
                }
            ],
            "custom_fields": {},
            "applied_promotions": [],
            "points_info": [
                {
                    "points_currency_iid": 1,
                    "points_currency_eid": "bonus1",
                    "name": "Main Point",
                    "status": "success",
                    "active_amount_for_current_order": "20999",
                    "accrue_points_amount_for_current_order": "6999",
                    "redeemed_points_amount_for_current_order": "499"
                }
            ],
            "promocodes_info": []
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
  <data>
    <profile>
      <profile_id>696e304f547840a7286619e0</profile_id>
    </profile>
    <order>
      <processing_target>calculated</processing_target>
      <total_price>139999.00</total_price>
      <final_total_price>139500.00</final_total_price>
      <delivery_cost>0.00</delivery_cost>
      <currency>RUB</currency>
      <lines>
        <external_id>line___ASTGRf</external_id>
        <line_number>0</line_number>
        <count>1</count>
        <product>iphone17pmb256</product>
        <base_price_per_item>139999.00</base_price_per_item>
        <min_price_per_item>0.00</min_price_per_item>
        <final_price_per_item>139500.00</final_price_per_item>
        <applied_promotions>
          <type>accrue_points</type>
          <promotion>
            <external_id>bonus</external_id>
            <name>Point Accrual</name>
          </promotion>
          <points_info>
            <points_currency_iid>1</points_currency_iid>
            <points_currency_eid>bonus1</points_currency_eid>
            <name>Main Point</name>
            <amount>6999</amount>
          </points_info>
        </applied_promotions>
        <applied_promotions>
          <type>redeem_points</type>
          <promotion>
            <external_id>bonus</external_id>
            <name>Point Redemption</name>
          </promotion>
          <points_info>
            <points_currency_iid>1</points_currency_iid>
            <points_currency_eid>bonus1</points_currency_eid>
            <name>Main Point</name>
            <amount>499</amount>
          </points_info>
        </applied_promotions>
      </lines>
      <points_info>
        <points_currency_iid>1</points_currency_iid>
        <points_currency_eid>bonus1</points_currency_eid>
        <name>Main Point</name>
        <status>success</status>
        <active_amount_for_current_order>20999</active_amount_for_current_order>
        <accrue_points_amount_for_current_order>6999</accrue_points_amount_for_current_order>
        <redeemed_points_amount_for_current_order>499</redeemed_points_amount_for_current_order>
      </points_info>
    </order>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response Parameters

Parameter	
data	**object**
error	**int**
error_text	**string**

The `data` object has the following structure.

Parameter	Type	
profile	**object**	**Member profile. Contains the**
order	**object**	**Calculated order. More details**

Order Confirmation

Description

The `preorder_commit` method performs the final order confirmation after a successful preliminary calculatio

URL

Method: POST

`https://example.com/api/v1.1/market/orders/preorder_commit`

Request Parameters

Parameter	Type	Example
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
external_id	**string**	**"order_12345"**
profile_id	**string**	**"696e304f547840a7286619e0"**
email	**string**	**"user@example.com"**
phone	**string**	**"+79991234567"**
skip_triggers	**boolean**	**false**

data	**object**	<pre>"data": {
		"endpoint": "bot",
		"status": "confirmed",
		"currency": "RUB",
		"total_price": "2500.00",
		"final_total_price": "2150.00",
		"delivery_cost": "300.00",
		"lines": [...],
		"promocodes": [...],
		"redeem_points": [...],
		"region": "region_moscow",
		"custom_fields": ,
		"tags": ["tag1", "tag2"]
		}</pre>

*** — to identify the profile, one of the parameters must be provided: `profile_id`, `email`, or `phone`**

data object

Parameter	Type	Example	
endpoint	**string**	**"bot", "website"**	**Ye**
status	**string**	**"new", "confirmed", "paid"**	**Ye**
currency	**string**	**"RUB", "USD", "EUR"**	**Ye**
total_price	**string**	**"2500.00"**	**Ye**
final_total_price	**string**	**"2150.00"**	**Ye**
delivery_cost	**string**	**"300.00"**	**No**
create_time	**string (ISO 8601)**	**"2026-02-18T13:01:05.141Z"**	**No**
region	**string**	**"region_moscow"**	**No**
lines	**array**	**See "lines array" section below**	**Ye**
promocodes	**array**	**See "promocodes array" section below**	**No**
requested_promotions	**array**	**See "requested_promotions array" section below**	**No**
redeem_points	**array**	**See "redeem_points array" section below**	**Ye**

custom_fields	**object**	<pre>{
		"field1": "value1",
		"field2": "value2"
		}</pre>
			No

tags	**array**	<pre>[
		"summer_sale",
		"first_order"
		]</pre>
			No

lines array

The array contains information about the order items. Must contain at least 1 item.
Each item is a separate object within the array with the following parameters:

Parameter	Type	Example
external_id	**string**	**"line_1"**
product	**string**	**"iphone17pmb256"**
sku	**string**	**"IPHONE17-256-BLACK"**
line_number	**integer**	**1**
count	**integer**	**1**
base_price_per_item	**string**	**"139999.00"**
min_price_per_item	**string**	**"130000.00"**
final_price_per_item	**string**	**"139500.00"**
status	**string**	**"new"**
custom_fields	**object**	**\{"color": "black", "size": "XL"\}**

requested_promotions	**array**	**See "requested_promotions array" section below**

*** — one of the parameters is required: `product` or `sku`**

promocodes array

The array contains objects with information about promo codes.
A full description of the promo code structure can be found in this article.

Parameter	Type	
code	**string**	**"SUMMER20"**
pool_id	**integer**	**42**

redeem_points array

An array with information about loyalty point redemption. Each object describes a separate redemption.
A full description of the point redemption structure can be found in this article.

Parameter	Type	
points_currency_iid	**integer**	**1**
points_currency_eid	**string**	**"bonus1"**
name	**string**	**"Bonus po**
amount	**integer**	**499**

*** — one of the parameters is required: `points_currency_iid` or `points_currency_eid`**

requested_promotions array

> TIP
> **The `requested_promotions` array can be contained both in the `data` object and in the `lines` array within that**

An array of promotions applied to the order. Each object in the array describes a separate promotion.

Parameter	Type	Example
type	**string**	**"discount", "accrue_points", "redeem_points", "code_issue", "message"**
promotion	**object**	`"promotion": { "external_id": "promo_1", "name": "Summer Sale" }`
amount	**string**	**"150.00"**
points_info	**object**	`"points_info": { "points_currency_iid": 1, "points_currency_eid": "bonus1", "name": "Bonus points", "amount": "50" }`

The structure of the `points_info` object is as follows:

Parameter	Type	
points_currency_iid	**integer**	1
points_currency_eid	**string**	"
name	**string**	"
amount	**string**	"
expiration_date	**string (ISO 8601)**	"

*** — one of the parameters is required: `points_currency_iid` or `points_currency_eid`**

Request Example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "external_id": "order_12345",
  "profile_id": "696e304f547840a7286619e0",
  "skip_triggers": false,
  "data": {
    "endpoint": "bot",
    "status": "confirmed",
    "currency": "RUB",
    "total_price": "2500.00",
    "final_total_price": "2150.00",
    "delivery_cost": "300.00",
    "create_time": "2026-02-18T13:01:05.141Z",
    "region": "region_moscow",
    "lines": [
      {
        "external_id": "line_1",
        "product": "popit",
        "base_price_per_item": "1500.00",
        "final_price_per_item": "1350.00",
        "count": 1,
        "requested_promotions": [
          {
            "type": "discount",
            "promotion": {
              "external_id": "promo_1",
              "name": "Summer Sale"
            },
            "amount": "150.00"
          },
          {
            "type": "accrue_points",
            "promotion": {
              "external_id": "bonus_promo",
              "name": "Point Accrual"
            },
            "points_info": {
              "points_currency_iid": 1,
              "points_currency_eid": "bonus1",
              "name": "Bonus points",
              "amount": "75"
            }
          }
        ]
      },
      {
        "external_id": "line_2",
        "sku": "skuf",
        "base_price_per_item": "700.00",
        "min_price_per_item": "650.00",
        "final_price_per_item": "650.00",
        "count": 2,
        "requested_promotions": [
          {
            "type": "redeem_points",
            "promotion": {
              "external_id": "redeem_promo",
              "name": "Point Redemption"
            },
            "points_info": {
              "points_currency_iid": 1,
              "points_currency_eid": "bonus1",
              "name": "Bonus points",
              "amount": "50"
            }
          }
        ]
      }
    ],
    "promocodes": [
      {
        "code": "HYDO-4728-DZGU",
        "pool_id": 42
      }
    ],
    "redeem_points": [
      {
        "points_currency_eid": "bonus1",
        "amount": 100
      }
```

```
    ],
    "requested_promotions": [
      {
        "type": "code_issue",
        "promotion": {
          "external_id": "welcome_promo",
          "name": "Welcome Gift"
        }
      }
    ],
    "custom_fields": {
      "source": "telegram_bot",
      "campaign": "spring_sale"
    },
    "tags": ["premium", "first_order"]
  }
}
```

XML

```xml
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <external_id>order_12345</external_id>
  <profile_id>696e304f547840a7286619e0</profile_id>
  <skip_triggers>false</skip_triggers>
  <data>
    <endpoint>bot</endpoint>
    <status>confirmed</status>
    <currency>RUB</currency>
    <total_price>2500.00</total_price>
    <final_total_price>2150.00</final_total_price>
    <delivery_cost>300.00</delivery_cost>
    <create_time>2026-02-18T13:01:05.141Z</create_time>
    <region>region_moscow</region>
    <lines>
      <external_id>line_1</external_id>
      <product>popit</product>
      <base_price_per_item>1500.00</base_price_per_item>
      <final_price_per_item>1350.00</final_price_per_item>
      <count>1</count>
      <requested_promotions>
        <type>discount</type>
        <promotion>
          <external_id>promo_1</external_id>
          <name>Summer Sale</name>
        </promotion>
        <amount>150.00</amount>
      </requested_promotions>
      <requested_promotions>
        <type>accrue_points</type>
        <promotion>
          <external_id>bonus_promo</external_id>
          <name>Point Accrual</name>
        </promotion>
        <points_info>
          <points_currency_iid>1</points_currency_iid>
          <points_currency_eid>bonus1</points_currency_eid>
          <name>Bonus points</name>
          <amount>75</amount>
        </points_info>
      </requested_promotions>
    </lines>
    <lines>
      <external_id>line_2</external_id>
      <sku>skuf</sku>
      <base_price_per_item>700.00</base_price_per_item>
      <min_price_per_item>650.00</min_price_per_item>
      <final_price_per_item>650.00</final_price_per_item>
      <count>2</count>
      <requested_promotions>
        <type>redeem_points</type>
        <promotion>
          <external_id>redeem_promo</external_id>
          <name>Point Redemption</name>
        </promotion>
        <points_info>
          <points_currency_iid>1</points_currency_iid>
          <points_currency_eid>bonus1</points_currency_eid>
          <name>Bonus points</name>
          <amount>50</amount>
        </points_info>
      </requested_promotions>
    </lines>
    <promocodes>
      <code>HYDO-4728-DZGU</code>
      <pool_id>42</pool_id>
    </promocodes>
    <redeem_points>
      <points_currency_eid>bonus1</points_currency_eid>
      <amount>100</amount>
    </redeem_points>
    <requested_promotions>
      <type>code_issue</type>
      <promotion>
        <external_id>welcome_promo</external_id>
        <name>Welcome Gift</name>
      </promotion>
    </requested_promotions>
  </data>
</xml>
```

Response Example

JSON


```
{
  "data": {
    "status": "success",
    "processing_target": "immediate",
    "external_id": "order_12345",
    "issued_promocodes": [
      {
        "codes": ["HYDO-4728-DZGU"],
        "pool": {
          "pool_id": 100,
          "name": "Welcome Promocodes"
        },
        "promotion": {
          "external_id": "welcome_promo",
          "name": "Welcome Gift"
        }
      }
    ],
    "points_changes": [
      {
        "points_info": {
          "points_currency_iid": 1,
          "points_currency_eid": "bonus1",
          "name": "Bonus points",
          "amount": "21500"
        },
        "accrue_points": 75,
        "redeem_points": 150
      }
    ],
    "contents": [
      {
        "promotion": {
          "external_id": "message_promo",
          "name": "Information Message"
        },
        "contents": [
          {
            "type": "text",
            "message": "Thank you for your purchase! You have received 75 bonus points."
          }
        ]
      }
    ]
  },
  "error": 0,
  "error_text": "Successful operation"
}
```


XML

```xml
<xml>
  <data>
    <status>success</status>
    <processing_target>immediate</processing_target>
    <external_id>order_12345</external_id>
    <issued_promocodes>
      <codes>HYDO-4728-DZGU</codes>
      <pool>
        <pool_id>100</pool_id>
        <name>Welcome Promocodes</name>
      </pool>
      <promotion>
        <external_id>welcome_promo</external_id>
        <name>Welcome Gift</name>
      </promotion>
    </issued_promocodes>
    <points_changes>
      <points_info>
        <points_currency_iid>1</points_currency_iid>
        <points_currency_eid>bonus1</points_currency_eid>
        <name>Bonus points</name>
        <amount>21500</amount>
      </points_info>
      <accrue_points>75</accrue_points>
      <redeem_points>150</redeem_points>
    </points_changes>
    <contents>
      <promotion>
        <external_id>message_promo</external_id>
        <name>Information Message</name>
      </promotion>
      <contents>
        <type>text</type>
        <message>Thank you for your purchase! You have received 75 bonus points.</message>
      </contents>
    </contents>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type
data	**object**
error	**int**
error_text	**string**

Parameters returned in the `data` object:

Parameter	Type	
status	**string**	**Processing status:** • `success` — **order successfully created** • `outdated` — **price has changed since the preliminary ca**
processing_target	**string**	**Only for `success` status. Value: `immediate` or `processed`**
external_id	**string**	**External order ID (from the request)**
issued_promocodes	**array**	**Array of issued promo codes (for `code_issue` type promot**
points_changes	**array**	**Array of point balance changes by point currency types**
contents	**array**	**Array of content for display (text messages from promot**

The `issued_promocodes` array consists of objects with information about applied promo codes:

Parameter	
codes	**array**
pool	**object**
pool.pool_id	**integer**
pool.name	**string**
promotion	**object**
promotion.external_id	**string**
promotion.name	**string**

The `points_changes` array consists of objects with information about point balance changes:

Parameter
points_info
points_info.points_currency_iid
points_info.points_currency_eid
points_info.name
points_info.amount

Parameter
accrue_points
redeem_points

Roll back temporary transaction

Description

Rolls back a temporary transaction created with the `is_temp` flag. After rollback, the active points balance

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/rollback_transaction`

Request parameters

Parameter	Type	
transaction_id	**string**	**“TRANS12345”**
transaction_eid	**string**	**“TRANS12345”**

At least one of the identifiers must be provided (`transaction_id` or `transaction_eid`).

Request example

JSON

```
{
  "transaction_id": "TRANS12345"
}
```

XML

```
<xml>
  <transaction_id>TRANS12345</transaction_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance": "399.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance>399.50</balance>
  </data>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
transaction_id	**string**	**Rolled-back transaction ID**
balance	**string**	**Current active points balance of**

Cancel points transaction

Description

Cancels any points transaction — temporary or permanent. When canceled, the current active points bala

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/cancel_transaction`

Request parameters

Parameter	Type	
transaction_id	**string**	**“TRANS1**
transaction_eid	**string**	**“TRANS1**

At least one identifier must be provided.

Request example

JSON

```
{
  "transaction_id": "TRANS12345"
}
```

XML

```
<xml>
  <transaction_id>TRANS12345</transaction_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "transaction_id": "TRANS12345",
    "balance": "399.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <transaction_id>TRANS12345</transaction_id>
    <balance>399.50</balance>
  </data>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
transaction_id	**string**
balance	**string**

Get points account balance

Description

Returns the current points account balance of a loyalty program member: the total number of points and t

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/account/balance`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
points_currency_iid	**int**	**1**
points_currency_eid	**string**	**“starts”**
matching	**string**	**“email”**
field_name	**string**	**“user_id”**
field_value	**string / int**	**123456**
email	**string**	**“user@example.com”**
phone	**string**	**“+1234567890”**
profile_id	**string**	**“profile123”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "points_currency_eid": "starts",
  "matching": "email",
  "email": "user@example.com"
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <points_currency_eid>starts</points_currency_eid>
  <matching>email</matching>
  <email>user@example.com</email>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": {
    "total_balance": "1000.00",
    "active_balance": "800.50"
  }
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <data>
    <total_balance>1000.00</total_balance>
    <active_balance>800.50</active_balance>
  </data>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
total_balance	**string**	**Total number of points in the po**
active_balance	**string**	**Number of active points in the p**

Register member in a loyalty program

Description

Registers a member in a loyalty program. Before registration, the profile is searched in the database using

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/register_member`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
matching	**string**	**“email”**
registration_date	**string**	**“2024-09-01T12:34:56Z”**
email	**string**	**“user@example.com”**
phone	**string**	**“+1234567890”**
profile_id	**string**	**“profile123”**
field_name	**string**	**“user_id”**
field_value	**string / int**	**“123456”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "matching": "email",
  "email": "user@example.com",
  "registration_date": "2024-09-01T12:34:56Z"
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <email>user@example.com</email>
  <registration_date>2024-09-01T12:34:56Z</registration_date>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "123456"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <profile_id>123456</profile_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
profile_id	**string**	**ID of the profile registered in the lo**

Batch adding participants to loyalty program

Description

Adds multiple participants to a loyalty program in a single request. Before adding, each profile is looked u

> **CAUTION**
>
> **In one request it is not recommended:**
>
> - **transfer more than 10,000 profiles;**
> - **transfer more than 10 MB of data. It is possible to transfer more data using multiple streams.**

URL

Method: POST

`https://example.com/api/v1.1/loyalty/register_member_batch`

Request parameters

Parameter	Type	E
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
matching	**string**	**“email”**
data	**array**	**[{“email”: “user@example.com”}]**

`data` array

Parameter	Type	
email	**string**	**“user@example.com”**
phone	**string**	**“+1234567890”**
profile_id	**string**	**“profile123”**
field_name	**string**	**“user_id”**
field_value	**string / int**	**“123456”**
registration_date	**string**	**“2024-09-01T12:34:56Z”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "matching": "email",
  "data": [
    {
      "email": "user1@example.com",
      "registration_date": "2024-09-01T12:34:56Z"
    },
    {
      "email": "user2@example.com"
    }
  ]
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <data>
    <item>
      <email>user1@example.com</email>
      <registration_date>2024-09-01T12:34:56Z</registration_date>
    </item>
    <item>
      <email>user2@example.com</email>
    </item>
  </data>
</xml>
```

Response example

JSON


```
{
  "error": 0,
  "error_text": "Successful operation",
  "result": [
    {
      "error": 0,
      "error_text": "Successful operation",
      "profile_id": "663a138cee44bb50a67115cd"
    },
    {
      "error": 409,
      "error_text": "Member already exists",
      "profile_id": "69df4fd5ff232800e2d1e2d6"
    },
    {
      "error": 404,
      "error_text": "Profile not found",
      "profile_id": null
    }
  ]
}
```


XML


```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <result>
    <item>
      <error>0</error>
      <error_text>Successful operation</error_text>
      <profile_id>663a138cee44bb50a67115cd</profile_id>
    </item>
    <item>
      <error>409</error>
      <error_text>Member already exists</error_text>
      <profile_id>69df4fd5ff232800e2d1e2d6</profile_id>
    </item>
    <item>
      <error>404</error>
      <error_text>Profile not found</error_text>
    </item>
  </result>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error co**
error_text	**string**	**Error de**
result	**array**	**Array of**

`result` element parameters

Parameter	Type	
error	**int**	**Error code for the specif**
error_text	**string**	**Error description for the**
profile_id	**string**	**Profile identifier (null on**

Error codes in `result`

Code	
0	**Successfully added**
404	**Profile not found**
409	**Member already exists in the loyalty program**

Task for batch add participants to loyalty program

Description

Starts an asynchronous task for bulk adding participants to a loyalty program. Returns a `task_id` that can

> **CAUTION**
>
> **A single request should not:**
>
> - **Exceed 10,000 profiles;**
> - **Exceed 10 MB of data. Larger volumes can be sent in multiple streams.**

URL

Method: POST

`https://example.com/api/v1.1/loyalty/register_member_batch_task`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
matching	**string**	**“email”**
data	**array**	**[{“email”: “user@example.com”}]**
task_id	**string**	**“199b9c75-c540-455e-9e38-3d54e51e1d7a”**
next_key	**string**	**“abc123”**

`data` array

Parameter	Type	
email	**string**	**“user@example.com”**
phone	**string**	**“+1234567890”**
profile_id	**string**	**“profile123”**
field_name	**string**	**“user_id”**
field_value	**string / int**	**“123456”**
registration_date	**string**	**“2024-09-01T12:34:56Z”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "matching": "email",
  "data": [
    {
      "email": "user1@example.com",
      "registration_date": "2024-09-01T12:34:56Z"
    },
    {
      "email": "user2@example.com"
    }
  ]
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <data>
    <item>
      <email>user1@example.com</email>
      <registration_date>2024-09-01T12:34:56Z</registration_date>
    </item>
    <item>
      <email>user2@example.com</email>
    </item>
  </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "task_id": "199b9c75-c540-455e-9e38-3d54e51e1d7a"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <task_id>199b9c75-c540-455e-9e38-3d54e51e1d7a</task_id>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error co**
error_text	**string**	**Error de**
task_id	**string**	**Asynchr**

Remove member from loyalty program

Description

Removes a member from a loyalty program. Before removal, the profile is searched using the selected ma

Request URL

Method: POST

`https://example.com/api/v1.1/loyalty/delete_member`

Request parameters

Parameter	Type	Example
token	**string**	**“abc123def456”**
loyalty_program_id	**int**	**123**
matching	**string**	**“email”**
email	**string**	**“user@example.com”**
phone	**string**	**“+1234567890”**
profile_id	**string**	**“profile123”**
field_name	**string**	**“user_id”**
field_value	**string / int**	**“123456”**

Request example

JSON

```
{
  "token": "abc123def456",
  "loyalty_program_id": 123,
  "matching": "email",
  "email": "user@example.com"
}
```

XML

```
<xml>
  <token>abc123def456</token>
  <loyalty_program_id>123</loyalty_program_id>
  <matching>email</matching>
  <email>user@example.com</email>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation",
  "profile_id": "123456"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <profile_id>123456</profile_id>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
profile_id	**string**

Get form information

Description

Allows you to retrieve information about a form: field structure, page settings, transition logic, styling, an

URL

Method: POST

`https://example.com/api/v1.1/forms/get`

Request Parameters

Parameter	Type	Example
token	**string**	**“a1b2c3d4e5f67890abcdef1234567890”**
id	**int**	**1**
shortname	**string**	**“form_6Ven2v7O”**
form	**boolean**	**true**
actions	**boolean**	**true**

Request Example

JSON

```
{
  "token": "a1b2c3d4e5f67890abcdef1234567890",
  "id": 1,
  "form": true,
  "actions": true
}
```

XML

```
<xml>
  <token>a1b2c3d4e5f67890abcdef1234567890</token>
  <id>1</id>
  <form>true</form>
  <actions>true</actions>
</xml>
```

Response Example

JSON

```
{
    "data": {
        "id": 1,
        "group_id": 0,
        "groups_id": [],
        "is_delete": false,
        "createtime": "2026-02-20T10:43:51.91Z",
        "updatetime": "2026-04-23T11:00:46.242Z",
        "name": "Example Form",
        "shortname": "form_6Ven2v7O",
        "descr": "",
        "type": "basic",
        "ui_tags": [],
        "is_nps": false,
        "is_published": true,
        "published_time": "2026-04-23T11:00:46.234Z",
        "is_temporary": false,
        "updated_user": {
            "name": "account_john",
            "fname": "John",
            "lname": "Doe",
            "avatar_date_update": ""
        },
        "tz": "",
        "actions": [
            {
                "action_id": "NrqKjgfX",
                "type": "profile_import",
                "profile_import": {
                    "db_id": 3,
                    "matching_mode": "email",
                    "mode_add": true,
                    "mode_update": true,
                    "is_update_utm": true,
                    "match_mapping": [
                        {
                            "ext_name": "field_4",
                            "ext_num": 0,
                            "int_name": "email",
                            "ignore_empty": true,
                            "are_nulls_empty": false,
                            "normalizers": [],
                            "action": "set",
                            "submatch": ""
                        }
                    ],
                    "column_mapping": [
                        {
                            "ext_name": "field_1",
                            "ext_num": 0,
                            "int_name": "_fname",
                            "ignore_empty": false,
                            "are_nulls_empty": false,
                            "normalizers": [],
                            "action": "set",
                            "submatch": ""
                        },
                        {
                            "ext_name": "field_4",
                            "ext_num": 0,
                            "int_name": "phones",
                            "ignore_empty": false,
                            "are_nulls_empty": false,
                            "normalizers": [],
                            "action": "set",
                            "submatch": ""
                        }
                    ],
                    "auto_subscribe": []
                }
            }
        ],
        "form": {
            "form_title": "Example Form",
            "last_page_id": 2,
            "language": "EN",
            "basic_form_pages": [
                {
                    "id": 1,
                    "title": "Page 1",
```

```
            "shortname": "page_1",
            "fields": [
                {
                    "type": "line",
                    "name": "field_1",
                    "comment": "",
                    "hidden": false,
                    "title": "Name",
                    "index_custom_option": -1,
                    "condition": {
                        "condition": "",
                        "value": null
                    },
                    "count_range": 5,
                    "min_range": 0,
                    "max_range": 20,
                    "options": [],
                    "default_value": "",
                    "validation_rules": [],
                    "right_answer": "",
                    "right_answer_score": 0,
                    "nps_field": null
                },
                {
                    "type": "email",
                    "name": "field_4",
                    "comment": "",
                    "hidden": false,
                    "title": "Email",
                    "index_custom_option": -1,
                    "condition": {
                        "condition": "",
                        "value": null
                    },
                    "count_range": 5,
                    "min_range": 0,
                    "max_range": 20,
                    "options": [],
                    "default_value": "",
                    "validation_rules": [
                        {
                            "type": "email",
                            "message": ""
                        }
                    ],
                    "right_answer": "",
                    "right_answer_score": 0,
                    "nps_field": null
                },
                {
                    "type": "rating_stars",
                    "name": "field_5",
                    "comment": "",
                    "hidden": false,
                    "title": "Rating",
                    "index_custom_option": -1,
                    "condition": {
                        "condition": "",
                        "value": null
                    },
                    "count_range": 5,
                    "min_range": 0,
                    "max_range": 20,
                    "options": [],
                    "default_value": 0,
                    "validation_rules": [],
                    "right_answer": "",
                    "right_answer_score": 0,
                    "nps_field": null
                }
            ]
        },
        {
            "id": 2,
            "title": "Page 2",
            "shortname": "page_2",
            "fields": [
                {
                    "type": "line",
                    "name": "field_6",
```

```
                    "comment": "",
                    "hidden": false,
                    "title": "Additional Question",
                    "index_custom_option": -1,
                    "condition": {
                        "condition": "",
                        "value": null
                    },
                    "count_range": 5,
                    "min_range": 0,
                    "max_range": 20,
                    "options": [],
                    "default_value": "",
                    "validation_rules": [],
                    "right_answer": "",
                    "right_answer_score": 0,
                    "nps_field": null
                },
                {
                    "type": "checkbox",
                    "name": "field_8",
                    "comment": "",
                    "hidden": false,
                    "title": "Checkbox",
                    "index_custom_option": -1,
                    "condition": {
                        "condition": "",
                        "value": null
                    },
                    "count_range": 5,
                    "min_range": 0,
                    "max_range": 20,
                    "options": [],
                    "default_value": "",
                    "validation_rules": [],
                    "right_answer": "",
                    "right_answer_score": 0,
                    "nps_field": null
                }
            ]
        }
    ],
    "final_page": {
        "title": "Thank you!",
        "message": "Your form has been submitted successfully.",
        "display_score": false,
        "required_score": 0,
        "redirect_pass_url": "",
        "redirect_fail_url": ""
    },
    "style": {
        "preset": "default",
        "background_color": "#6590e7",
        "background_image": "",
        "buttons_color": "#000000",
        "buttons_background_color": "#3678E2",
        "font": "Fira Sans",
        "text_align": "initial",
        "font_color": "#0D0F12",
        "font_size": 18,
        "border_radius": 12,
        "rating_stars_color": "#ffea78",
        "rating_stars_background_color": "#f1f3f9",
        "rate_star": {
            "figure_selected_color": "",
            "figure_unselected_color": "",
            "figure_size": 0
        },
        "rate_heart": {
            "figure_selected_color": "",
            "figure_unselected_color": "",
            "figure_size": 0
        },
        "rate_emoji": {
            "figure_selected_color": "",
            "figure_unselected_color": "",
            "figure_size": 0
        },
        "rate_number": {
            "buttons_selected_color": "",
```

```
            "buttons_unselected_color": "",
            "buttons_selected_background_color": "",
            "buttons_unselected_background_color": "",
            "number_size": 0
        }
    },
    "next_button": {
        "type": "next",
        "size": 14,
        "icon": "",
        "title": "Next Page"
    },
    "prev_button": {
        "type": "prev",
        "size": 14,
        "icon": "",
        "title": "Previous Page"
    },
    "send_button": {
        "type": "send",
        "size": 14,
        "icon": "paper plane",
        "title": "Send"
    },
    "final_button": {
        "type": "final",
        "size": 14,
        "icon": "",
        "title": "Go to the website"
    },
    "time_limit_sec": 900,
    "form_check_sum": "f742b0a68fb3fa17c3d845487ed4b5ed3f6405b7",
    "logic": [
        {
            "id": 1,
            "title": "",
            "page_id": 1,
            "next_node": 2,
            "type": "page",
            "condition": []
        },
        {
            "id": 2,
            "title": "",
            "page_id": 2,
            "type": "page",
            "condition": []
        }
    ],
    "start_logic_block_id": 1,
    "last_form_logic_block_id": 2,
    "is_logic_enabled": true,
    "flowchart_elements": [
        {
            "data": {
                "idOfLogicBlock": 1,
                "idOfPage": 1,
                "isFirstPage": true,
                "label": "Page 1"
            },
            "id": "jANe4JVI",
            "position": {
                "x": 18,
                "y": 54
            },
            "type": "page"
        },
        {
            "data": {
                "idOfLogicBlock": 2,
                "idOfPage": 2,
                "isFirstPage": false,
                "label": "Page 2"
            },
            "id": "sBFQLU8K",
            "position": {
                "x": 26,
                "y": 184
            },
            "type": "page"
```

```
                },
                {
                    "id": "EmaWmrar",
                    "source": "jANe4JVI",
                    "sourceHandle": null,
                    "target": "sBFQLU8K",
                    "targetHandle": null,
                    "type": "edge"
                }
            ],
            "is_nps": false,
            "restrict_refill": true,
            "resubmit_page": {
                "title": "Restriction!",
                "message": "You have already filled out this form.",
                "display_score": false,
                "required_score": 0,
                "redirect_pass_url": "",
                "redirect_fail_url": ""
            },
            "expired_days": 1
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <id>1</id>
        <group_id>0</group_id>
        <is_delete>false</is_delete>
        <createtime>2026-02-20T10:43:51.91Z</createtime>
        <updatetime>2026-04-23T11:00:46.242Z</updatetime>
        <name>Example Form</name>
        <shortname>form_6Ven2v7O</shortname>
        <type>basic</type>
        <is_nps>false</is_nps>
        <is_published>true</is_published>
        <published_time>2026-04-23T11:00:46.234Z</published_time>
        <is_temporary>false</is_temporary>
        <updated_user>
            <name>account_john</name>
            <fname>John</fname>
            <lname>Doe</lname>
        </updated_user>
        <actions>
            <action_id>NrqKjgfX</action_id>
            <type>profile_import</type>
            <profile_import>
                <db_id>3</db_id>
                <matching_mode>email</matching_mode>
                <mode_add>true</mode_add>
                <mode_update>true</mode_update>
                <is_update_utm>true</is_update_utm>
                <match_mapping>
                    <ext_name>field_4</ext_name>
                    <ext_num>0</ext_num>
                    <int_name>email</int_name>
                    <ignore_empty>true</ignore_empty>
                    <are_nulls_empty>false</are_nulls_empty>
                    <action>set</action>
                </match_mapping>
                <column_mapping>
                    <ext_name>field_1</ext_name>
                    <ext_num>0</ext_num>
                    <int_name>_fname</int_name>
                    <ignore_empty>false</ignore_empty>
                    <are_nulls_empty>false</are_nulls_empty>
                    <action>set</action>
                </column_mapping>
                <column_mapping>
                    <ext_name>field_4</ext_name>
                    <ext_num>0</ext_num>
                    <int_name>phones</int_name>
                    <ignore_empty>false</ignore_empty>
                    <are_nulls_empty>false</are_nulls_empty>
                    <action>set</action>
                </column_mapping>
            </profile_import>
        </actions>
        <form>
            <form_title>Example Form</form_title>
            <last_page_id>2</last_page_id>
            <language>EN</language>
            <basic_form_pages>
                <id>1</id>
                <title>Page 1</title>
                <shortname>page_1</shortname>
                <fields>
                    <type>line</type>
                    <name>field_1</name>
                    <hidden>false</hidden>
                    <title>Name</title>
                    <index_custom_option>-1</index_custom_option>
                    <condition>
                    </condition>
                    <count_range>5</count_range>
                    <min_range>0</min_range>
                    <max_range>20</max_range>
                    <right_answer_score>0</right_answer_score>
                </fields>
                <fields>
                    <type>email</type>
                    <name>field_4</name>
                    <hidden>false</hidden>
                    <title>Email</title>
```

```
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <validation_rules>
                <type>email</type>
            </validation_rules>
            <right_answer_score>0</right_answer_score>
        </fields>
        <fields>
            <type>rating_stars</type>
            <name>field_5</name>
            <hidden>false</hidden>
            <title>Rating</title>
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <default_value>0</default_value>
            <right_answer_score>0</right_answer_score>
        </fields>
    </basic_form_pages>
    <basic_form_pages>
        <id>2</id>
        <title>Page 2</title>
        <shortname>page_2</shortname>
        <fields>
            <type>line</type>
            <name>field_6</name>
            <hidden>false</hidden>
            <title>Additional Question</title>
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <right_answer_score>0</right_answer_score>
        </fields>
        <fields>
            <type>checkbox</type>
            <name>field_8</name>
            <hidden>false</hidden>
            <title>Checkbox</title>
            <index_custom_option>-1</index_custom_option>
            <condition>
            </condition>
            <count_range>5</count_range>
            <min_range>0</min_range>
            <max_range>20</max_range>
            <right_answer_score>0</right_answer_score>
        </fields>
    </basic_form_pages>
    <final_page>
        <title>Thank you!</title>
        <message>Your form has been submitted successfully.</message>
        <display_score>false</display_score>
        <required_score>0</required_score>
        <final_redirect>false</final_redirect>
    </final_page>
    <style>
        <preset>default</preset>
        <background_color>#6590e7</background_color>
        <buttons_color>#000000</buttons_color>
        <buttons_background_color>#3678E2</buttons_background_color>
        <font>Fira Sans</font>
        <text_align>initial</text_align>
        <font_color>#0D0F12</font_color>
        <font_size>18</font_size>
        <border_radius>12</border_radius>
        <rating_stars_color>#ffea78</rating_stars_color>
        <rating_stars_background_color>#f1f3f9</rating_stars_background_color>
        <rate_star>
            <figure_size>0</figure_size>
        </rate_star>
        <rate_heart>
```

```
            <figure_size>0</figure_size>
        </rate_heart>
        <rate_emoji>
            <figure_size>0</figure_size>
        </rate_emoji>
        <rate_number>
            <number_size>0</number_size>
        </rate_number>
    </style>
    <next_button>
        <type>next</type>
        <size>14</size>
        <title>Next Page</title>
    </next_button>
    <prev_button>
        <type>prev</type>
        <size>14</size>
        <title>Previous Page</title>
    </prev_button>
    <send_button>
        <type>send</type>
        <size>14</size>
        <icon>paper plane</icon>
        <title>Send</title>
    </send_button>
    <final_button>
        <type>final</type>
        <size>14</size>
        <title>Go to the website</title>
    </final_button>
    <time_limit_sec>900</time_limit_sec>
    <form_check_sum>f742b0a68fb3fa17c3d845487ed4b5ed3f6405b7</form_check_sum>
    <logic>
        <id>1</id>
        <page_id>1</page_id>
        <next_node>2</next_node>
        <type>page</type>
    </logic>
    <logic>
        <id>2</id>
        <page_id>2</page_id>
        <type>page</type>
    </logic>
    <start_logic_block_id>1</start_logic_block_id>
    <last_form_logic_block_id>2</last_form_logic_block_id>
    <is_logic_enabled>true</is_logic_enabled>
    <flowchart_elements>
        <data>
            <idOfLogicBlock>1</idOfLogicBlock>
            <idOfPage>1</idOfPage>
            <isFirstPage>true</isFirstPage>
            <label>Page 1</label>
        </data>
        <id>jANe4JVI</id>
        <position>
            <x>18</x>
            <y>54</y>
        </position>
        <type>page</type>
    </flowchart_elements>
    <flowchart_elements>
        <data>
            <idOfLogicBlock>2</idOfLogicBlock>
            <idOfPage>2</idOfPage>
            <isFirstPage>false</isFirstPage>
            <label>Page 2</label>
        </data>
        <id>sBFQLU8K</id>
        <position>
            <x>26</x>
            <y>184</y>
        </position>
        <type>page</type>
    </flowchart_elements>
    <flowchart_elements>
        <id>EmaWmrar</id>
        <source>jANe4JVI</source>
        <sourceHandle>null</sourceHandle>
        <target>sBFQLU8K</target>
        <targetHandle>null</targetHandle>
```

```
                <type>edge</type>
            </flowchart_elements>
            <is_nps>false</is_nps>
            <restrict_refill>true</restrict_refill>
            <resubmit_page>
                <title>Restriction!</title>
                <message>You have already filled out this form.</message>
                <display_score>false</display_score>
                <required_score>0</required_score>
            </resubmit_page>
            <expired_days>1</expired_days>
        </form>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	Type
error	**int**
error_text	**string**
data	**object**

`data` Object

Parameter	Type	
id	**int**	**System form ID.**
group_id	**int**	**Group ID.**
groups_id	**array[int]**	**List of group IDs.**
is_delete	**boolean**	**Deletion flag.**
is_published	**boolean**	**Publication flag.**
name	**string**	**Form name.**
descr	**string**	**Form description.**
createtime	**string**	**Creation date and time.**
updatetime	**string**	**Last update date and time.**
published_time	**string**	**Publication date and time.**
shortname	**string**	**Internal form ID.**
tz	**string**	**Time zone.**
ui_tags	**array[string]**	**Array of user tags.**
is_nps	**boolean**	**NPS survey flag.**
updated_user	**object**	**User who last updated the form. See upda**
actions	**array**	**Array of actions (present if actions=true i**
form	**object**	**Form settings object (present if form=true**
type	**string**	**Form type. Not used, always “basic”.**
is_temporary	**boolean**	**Unused parameter (always false).**

`updated_user` Object

Parameter	Type	
name	**string**	**Username.**
fname	**string**	**First name**
lname	**string**	**Last name**
avatar_date_update	**string**	**Last avata**
api	**string**	**Returns th**

`action` Object (element of actions array)

Parameter	Type
action_id	**string**
type	**string**
profile_import	**object**

`profile_import` Object

Parameter	Type	
db_id	**int**	**Database ID.**
matching_mode	**string**	**Matching mode (e.g., “email”). For more deta**
mode_add	**boolean**	**Allow adding new profiles.**
mode_update	**boolean**	**Allow updating existing profiles.**
is_update_utm	**boolean**	**Update UTM tags.**
skip_triggers	**boolean**	**Skip triggers.**
use_geo	**boolean**	**Use geodata.**
auto_subscribe	**array**	**Auto-subscription settings array.**
match_mapping	**array**	**Record matching mapping array. Each eleme**
column_mapping	**array**	**Form field to database field mapping array.**

`mapping` Object (element of match_mapping and column_mapping arrays)

Parameter	Type	
ext_name	**string**	**Form field name.**
int_name	**string**	**Field name in the target system.**
ignore_empty	**boolean**	**Ignore empty values.**
are_nulls_empty	**boolean**	**Whether to treat NULL as empty.**
action	**string**	**Action (usually “set”).**

Parameter	Type	
submatch	**string**	**Substitution expression. Used for custom fields in the**
ext_num	**int**	**External field number. Not used (default is “0”).**
normalizers	**array**	**Normalizers array. Not used, always an empty array.**

`form` Object

Parameter	Type	
form_title	**string**	**Form title.**
language	**string**	**Language code (e.g., “EN”).**
is_nps	**boolean**	**NPS survey flag (duplicated at**
restrict_refill	**boolean**	**Prevent repeat filling.**
last_page_id	**int**	**Last page ID.**
basic_form_pages	**array**	**Array of form pages. Each page**
final_page	**object**	**Final page. See final_page.**
resubmit_page	**object**	**Page shown when attempting t**
style	**object**	**Style settings. See style.**
next_button	**object**	**“Next” button. See button.**
prev_button	**object**	**“Previous” button. See button.**
send_button	**object**	**Submit button. See button.**
final_button	**object**	**“Go to website” button. See bu**
time_limit_sec	**int**	**Fill time limit in seconds.**
form_check_sum	**string**	**Form checksum.**
logic	**array**	**Page transition logic. See logic**
start_logic_block_id	**int**	**Starting logic block ID.**
last_form_logic_block_id	**int**	**Last logic block ID.**
is_logic_enabled	**boolean**	**Whether transition logic is enal**
flowchart_elements	**array**	**Flowchart elements (for the vis**
expired_days	**int**	**Days until the form becomes u**
start_date	**string**	**Form activation start date and**
end_date	**string**	**Form activation end date and ti**

`page` Object (element of basic_form_pages array)

Parameter	Type	
id	**int**	**Page ID.**
title	**string**	**Page title.**
shortname	**string**	**Page short name.**
fields	**array**	**Array of fields on the page. Each field is descril**

`field` Object (element of fields array)

Parameter	Type
type	**string**
name	**string**
title	**string**
comment	**string**
hidden	**boolean**
default_value	**string / number**
validation_rules	**array**
condition	**object**
options	**array**
right_answer	**string**
right_answer_score	**number**
nps_field	**string/null**

`final_page` Object

Parameter	
title	**string**
message	**string**
display_score	**boolean**
required_score	**int**
redirect_pass_url	**string**
redirect_fail_url	**string**

`resubmit_page` has a similar structure (title “Restriction!”, message about already filled).

`button` Object (for next_button, prev_button, send_button, final_button)

Parameter	Type	
type	**string**	**Button**
size	**int**	**Font siz**
icon	**string**	**Icon na**
title	**string**	**Button**

`style` Object

Parameter
preset
background_color
background_image
font
font_color
font_size
buttons_color
buttons_background_color
border_radius
text_align
rating_stars_color
rating_stars_background_color
rate_star, rate_heart, rate_emoji, rate_number

`logic` Object (element of logic array)

Parameter	Type	
id	**int**	**Logic block ID.**
title	**string**	**Title (may be empty).**
page_id	**int**	**ID of the page this block belongs to.**
next_node	**int**	**ID of the next node (only if there is a trans**
next_node_true	**int**	**ID of the next node when condition is met**
next_node_false	**int**	**ID of the next node when condition is not**
condition_type	**string**	**Logical operator for a group of conditions**
type	**string**	**Block type (page or condition).**
condition	**array**	**Transition conditions.**

For more details about form logic, see the article [Conditional Page-by-Page Logic in Forms and Surveys]

Get form list

Description

The method returns a list of forms with the ability to filter by tags, groups, publication status, and NPS ty

URL

Method: POST

`https://example.com/api/v1.1/forms/list`

Request Parameters

Parameter	Type	
token	**string**	**a1b2c3d4e5f67890abcdef1234567890**
from_id	**int**	**1**
limit	**int**	**50**
ui_tags	**array[string]**	**[“main”, “feedback”]**
groups	**array[string]**	**[“clients”, “partners”]**
groups_id	**array[int]**	**[5, 12]**
is_nps	**boolean**	**true**
is_published	**boolean**	**true**

Request Example

JSON

```
{
    "token": "a1b2c3d4e5f67890abcdef1234567890",
    "id": 1,
    "limit": 1,
    "is_nps": true
}
```

XML

```
<xml>
  <token>a1b2c3d4e5f67890abcdef1234567890</token>
  <id>1</id>
  <limit>2</limit>
  <is_nps>true</is_nps>
</xml>
```

Response Example

JSON

```
{
  "data": [
    {
      "id": 1,
      "group_id": 0,
      "groups_id": [],
      "is_delete": false,
      "createtime": "2026-02-20T10:43:51.91Z",
      "updatetime": "2026-03-26T13:19:23.961Z",
      "name": "Customer Survey",
      "shortname": "form_6Ven2v7O",
      "descr": "",
      "type": "basic",
      "ui_tags": [],
      "is_nps": false,
      "is_published": false,
      "is_temporary": false,
      "total_report": {
        "is_form_fill": false,
        "uloads": 0,
        "loads": 0,
        "uposts": 0,
        "posts": 0,
        "avg_fill_time_sec": 0
      },
      "updated_user": {
        "name": "userguide",
        "fname": "John",
        "lname": "Doe",
        "avatar_date_update": ""
      }
    }
  ],
  "error": 0,
  "error_text": "Successful operation",
  "total_count": 1
}
```


XML


```
<xml>
  <data>
    <id>1</id>
    <group_id>0</group_id>
    <is_delete>false</is_delete>
    <createtime>2026-02-20T10:43:51.91Z</createtime>
    <updatetime>2026-03-26T13:19:23.961Z</updatetime>
    <name>Customer Survey</name>
    <shortname>form_6Ven2v7O</shortname>
    <descr></descr>
    <type>basic</type>
    <is_nps>false</is_nps>
    <is_published>false</is_published>
    <is_temporary>false</is_temporary>
    <total_report>
      <is_form_fill>false</is_form_fill>
      <uloads>0</uloads>
      <loads>0</loads>
      <uposts>0</uposts>
      <posts>0</posts>
      <avg_fill_time_sec>0</avg_fill_time_sec>
    </total_report>
    <updated_user>
      <name>userguide</name>
      <fname>John</fname>
      <lname>Doe</lname>
      <avatar_date_update></avatar_date_update>
    </updated_user>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>1</total_count>
</xml>
```


Response Parameters

Parameter	
error	**int**

Parameter	
error_text	**string**
total_count	**int**
data	**array[object]**

`data` Array

Parameter	
id	**int**
group_id	**int**
groups_id	**array[int]**
is_delete	**boolean**
createtime	**string (date-time)**
updatetime	**string (date-time)**
name	**string**
shortname	**string**
descr	**string**
type	**string**
ui_tags	**array[string]**
is_nps	**boolean**
is_published	**boolean**
is_temporary	**boolean**
total_report	**object**
updated_user	**object**

`total_report` Object

Parameter	
is_form_fill	**boolean**
uloads	**int**
loads	**int**
uposts	**int**
posts	**int**
avg_fill_time_sec	**int**

`updated_user` Object

Parameter
name
fname
lname
avatar_date_update

Get form fills

Description

The method creates an export task for form fill data.

URL

Method: `POST`

`https://example.com/api/v1.1/forms/fills`

Request Parameters

Parameter	Type	Example
token	**string**	**a1b2c3d4e5f67890abcdef1234567890**
id	**int**	**1**
shortname	**string**	**form_6Ven2v7O**
date_from	**string**	**2026-04-01**
date_to	**string**	**2026-04-30**

Request Example

JSON

```
{
  "token": "a1b2c3d4e5f67890abcdef1234567890",
  "id": 1,
  "date_from": "2026-04-01",
  "date_to": "2026-04-30"
}
```

XML

```
<xml>
  <token>a1b2c3d4e5f67890abcdef1234567890</token>
  <id>1</id>
  <date_from>2026-04-01</date_from>
  <date_to>2026-04-30</date_to>
</xml>
```

Response Example

JSON

```
{
  "data": {
    "task_id": "550e8400-e29b-41d4-a716-446655440000"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <task_id>550e8400-e29b-41d4-a716-446655440000</task_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	
error	**int**
error_text	**string**
data	**object**

`data` Object

Parameter	Type	
task_id	**string**	**Async**

Publish form

Description

The method allows you to forcibly publish a form at the time of the request execution (run_now). While for

URL

Method: POST

`https://example.com/api/v1.1/forms/activate`

Request Parameters

Parameter	Type	
token	**string**	**“a1b2c3d4e5f67890abcdef1234567890”**
id	**int**	**1**
shortname	**string**	**“form_6Ven2v7O”**

Request Example

JSON

```
{
  "token": "a1b2c3d4e5f67890abcdef1234567890",
  "id": 1
}
```

XML

```
<xml>
<token>a1b2c3d4e5f67890abcdef1234567890</token>
<id>1</id>
</xml>
```

Response Example

JSON

```
{
  "data": {
    "url": "https://forms.example.com/form?token=a1b2c3d4e5f67890abcdef1234567890"
  },
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
<data>
<url>https://forms.example.com/form?token=a1b2c3d4e5f67890abcdef1234567890</url>
</data>
<error>0</error>
<error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter	
error	**int**
error_text	**string**
data	**object**

`data` Object

Parameter	Type
url	**string**

Unpublish form

Description

The method unpublishes a form. After this, the form becomes unavailable for users to fill out.

URL

Method: `POST`

`https://example.com/api/v1.1/forms/deactivate`

Request Parameters

Request body (application/json):

Parameter	Type	
token	**string**	**a1b2c3d4e5f67890abcdef1234567890**
id	**integer**	**1**
shortname	**string**	**form_6Ven2v7O**

Request Example

JSON

```
{
  "token": "a1b2c3d4e5f67890abcdef1234567890",
  "id": 1
}
```

XML

```
<xml>
  <token>a1b2c3d4e5f67890abcdef1234567890</token>
  <id>1</id>
</xml>
```

Response Example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter
error
error_text

Delete form

Description

The method deletes a form. If necessary, you can delete only the collected data while keeping the form it

URL

Method: `POST`

`https://example.com/api/v1.1/forms/delete`

Request Parameters

Request body (application/json):

Parameter	Type	Example	Req
token	**string**	**a1b2c3d4e5f67890abcdef1234567890**	**Yes**
id	**integer**	**1**	**Yes, if `shortname` is not specified**
shortname	**string**	**form_6Ven2v7O**	**Yes if `id` is not specified**
data_only	**boolean**	**false**	**No**

Request Example

JSON

```
{
  "token": "a1b2c3d4e5f67890abcdef1234567890",
  "id": 1,
  "data_only": false
}
```

XML

```
<xml>
  <token>a1b2c3d4e5f67890abcdef1234567890</token>
  <id>1</id>
  <data_only>false</data_only>
</xml>
```

Response Example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response Parameters

Parameter
error
error_text

Import promo codes

Description

Imports one or multiple promo codes into a loyalty program.

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/import`

Request example

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
codes	**JSON array**	`[“CODE-1”, “CODE-2”]`
loyalty_id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "codes": [
        "CODE-1",
        "CODE-2"
    ],
    "loyalty_id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <codes>CODE-1</codes>
    <codes>CODE-2</codes>
    <loyalty_id>6</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "code": "CODE-1",
            "error": 0,
            "error_text": ""
        },
        {
            "code": "CODE-2",
            "error": 0,
            "error_text": ""
        }
    ],
    "discarded_by_limit": 0,
    "duplicated": 0,
    "error": 0,
    "error_text": "Successful operation",
    "inserted": 2
}
```

XML

```
<xml>
    <data>
        <code>CODE-1</code>
        <error>0</error>
    </data>
    <data>
        <code>CODE-2</code>
        <error>0</error>
    </data>
    <discarded_by_limit>0</discarded_by_limit>
    <duplicated>0</duplicated>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <inserted>2</inserted>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
code	**string**
discarded_by_limit	**int**
duplicated	**int**
inserted	**int**

Get promo code information

Description

Retrieves a loyalty program promotional code information.

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**int**	**“json”**
code	**string**	**“CODE-12345”**
loyalty_id	**int**	**“1”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "code": "CODE-1",
    "loyalty_id": 5
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <code>CODE-1</code>
    <loyalty_id>5</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "created": "2024-04-01T12:39:34Z",
        "is_attached": false,
        "is_activated": false,
        "is_expired": false,
        "attached": "",
        "activated": "",
        "expired": "",
        "expire_date": "",
        "loyalty_id": 5,
        "code": "CODE-1"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <code>CODE-1</code>
        <created>2024-04-01T12:39:34Z</created>
        <is_activated>false</is_activated>
        <is_attached>false</is_attached>
        <is_expired>false</is_expired>
        <loyalty_id>5</loyalty_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Параметр	Тип	
error	**int**	**E**
error_text	**string**	**E**
activated	**datetime**	**P**
attached	**datetime**	**P**
code	**string**	**P**
created	**datetime**	**P**
expire_date	**datetime**	**P**
expired	**datetime**	**P**
is_activated	**bool**	**A**
is_attached	**bool**	**A**
is_expired	**bool**	**E**
loyalty_id	**int**	**L**
database_id	**int**	**D**
profile_id	**string**	**P**

Activate promo code

Description

Activates a promo code.

> **CAUTION**
>
> **You can only activate a promotional code when it is attached to a profile.**

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/activate`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
code	**string**	**“CODE-12345”**
format	**int**	**“json”**
loyalty_id	**int**	**“1”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "code": "Code-1",
    "loyalty_id": 5
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <code>Code-1</code>
    <loyalty_id>5</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Update promo code

Description

Updates promo code expiry date, activates or deactivates a promo code.

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/update`

Request parameters

Parameter	Type	Exampl
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**int**	**“json”**
activated	**bool**	**true**
code	**string**	**“CODE-12345”**
expire_date	**string**	**“2018-08-25T08:00:00Z”**
loyalty_id	**int**	**1**

CAUTION

At least one of request options is expected for a successful operation, expiry date change or activation/d

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "activated": true,
    "code": "CODE-12345",
    "expire_date": "2018-08-25T08:00:00Z",
    "loyalty_id": 1,
    "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <activated>true</activated>
    <code>Code-1</code>
    <expire_date>2024-08-10T08:00:00Z</expire_date>
    <loyalty_id>6</loyalty_id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data></data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error

Parameter
error_text

Attach promo code

Description

Attaches a promotion code to a profile.

If a `"code"` parameter is specified, it will be used for promo code matching.

- **If the code is already attached and its multiple usage is restricted, an error will occur.**
- **If a code is found unattached — it will be attached.**
- **If a code is not found — it will be imported and attached.**

With no `"code"` parameter loyalty program ID will be used for matching.

- **If there are promo codes available in a loyalty program — one will be attached.**

Request URL

Method: POST

`https://example.com/api/v1.1/promocodes/attach`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
code	**string**	**“CODE-12345”**
db_id	**int**	**1**
email	**string**	**“john@example.com”**
format	**string**	**“json”**
field_name	**string**	
field_value	**string**	
matching	**string**	**“profile_id”** **“email”** **“phone”** **“custom”**
loyalty_id	**int**	**1**
phone	**string**	**“+79000000000”**
profile_id	**string**	**“abcdefghijklmnqrstuvwxyz”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "loyalty_id": 5,
    "code": "CODE-1"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>1</db_id>
    <matching>profile_id</matching>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <loyalty_id>5</loyalty_id>
    <code>CODE-1</code>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data></data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Detach promo code

Description

Detaches a promotion code from a profile.

Use one of these parameters for profile matching:

- `email`
- `field_name` , `field_value`
- `phone`
- `profile_id`

If no `"code"` parameter is specified or if a specified code is attached to multiple profiles — an error will be

Request URL

method: POST

`https://example.com/api/v1.1/promocodes/detach`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
code	**string**	**“CODE-12345”**
db_id	**int**	**1**
email	**string**	**“john@example.com”**
format	**string**	**“json”**
field_name	**string**	
field_value	**string**	
matching	**string**	**“profile_id”** **“email”** **“phone”** **“custom”**
loyalty_id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "code": "Code-1",
    "db_id": 1,
    "loyalty_id": 4,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <code>Code-1</code>
    <db_id>1</db_id>
    <loyalty_id>4</loyalty_id>
    <matching>profile_id</matching>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data></data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get all promo codes

Description

Retrieves all promo codes from a loyalty program.

URL

Method: POST

`https://example.com/api/v1.1/promocodes/export_all`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**int**	**“json”**
loyalty_id	**int**	**1**

Request example

JSON

```
{
    "loyalty_id": 6,
    "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <loyalty_id>6</loyalty_id>
    <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "code": "Code-1"
        },
        {
            "code": "Code-2",
            "attached": "2024-04-01T14:02:05Z",
            "activated": "2024-04-01T14:04:12Z"
        },
        {
            "code": "Code-3"
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <code>Code-1</code>
    </data>
    <data>
        <activated>2024-04-01T14:04:12Z</activated>
        <attached>2024-04-01T14:02:05Z</attached>
        <code>Code-2</code>
    </data>
    <data>
        <code>Code-3</code>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
data	**Array**	**Promo codes list**
code	**string**	**Promo code**
attached	**string**	**Attachment date.**
activated	**string**	**Activation date. N**
expire_date	**string**	**Promo code expir**
expired	**string**	**Expiry date in cas**
error	**int**	**Error code**
error_text	**string**	**Error text**

Goals and goal values registration

Description

Records a goal achievement and optionally value for a customer profile. As part of event registration, you
You can use event data in the following mechanics:

- **segment profiles based on activity history;**
- **trigger a mailing or a scenario;**
- **add dynamic content into messages.**

Request URL

Method: POST

`https://example.com/api/v1.1/goals/register`

Request parameters

Main parameters

Parameter	Type	
token	**string**	**"abcdefghijklmnqrstuvwxyz"**
db_id	**int**	**1**
pixel_id	**int**	**1**
goal	**string**	**"card"**
goals	**JSON array**	`[{"goal": "card", "value": 10.0}]`
value	**float**	**-25.5**
date	**string**	**"2006-01-25T15:04:05+07:00"**

Matching parameters

These parameters are used to search for a profile in the database. Read more.

Parameter	Type	Example	Required
matching	**string**	**"email"** **"email_profile"** **"phone"** **"profile_id"** **"custom"**	**No, if matched by email**
email	**string**	**"john@example.com"**	**matching:"email"**
phone	**string**	**"+79000000000"**	**matching:"phone"**
profile_id	**string**	**"abcdefghijklmnqrstuvwxyz"**	**matching:"profile_id"**
field_name	**string**	**"CRM_ID"**	**matching:"custom"**
field_value	**int/string**	**"123420"**	**matching:"custom"**

Advanced parameters

Parameter	Type	Example

pixel_data	**JSON object**	`{`
		`  "key":"value",`
		`  "array":[1,2,3]`
		`}`

skip_triggers	**bool**	**false**
enrich_profile	**bool**	**true**
is_test	**bool**	**false**
vendor	**string**	**"altcraft"**
external_type	**string**	**"event_pixel_test"**
external_event_id	**string**	**"qwerty123"**
altcraft_client_id	**string**	**"9f11d32d-2935-43d0-b8df-f4c3568095b2"**

HTTP-parameters

Parameter	Type	Example
referer	**string**	**"api.example.com"**
user_agent	**string**	**"Mozilla/5.0 (Linux; Android 6.0.1; SM-G935S Build/MMB29K; wv)"**
accept_language	**string**	**"fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5"**
client_ip	**string**	**"10.0.0.0"**
client_ip_v4	**string**	**"198.16.74.224"**
client_ip_v6	**string**	**"fe80::5215:5556:d75f:806b"**

Parameter	Type	Example
browser	**string**	**"Chrome"**
os	**string**	**"Ubuntu"**
device	**string**	**"Desktop"**

TIP

The browser, os and device fields are automatically populated by the platform based on the contents of t
If you pass both user_agent and separate values for browser, os and device in the same request, the data
If you pass browser, os and device fields without user_agent, you must specify `"enrich_profile": true` to up

Geolocation parameters

The platform will automatically determine the values if `client_ip` is passed.

Parameter	Type	Example
lat	**float**	**55.244075**
lon	**float**	**55.244075**
country	**string**	**"RU"**
region	**string**	**"Ryazan"**
city	**string**	**"Ryazan"**
address	**string**	**"Pochtovaya street"**
zip	**string**	**"390000"**
time_zone	**string**	**"Europe/Moscow"**

TIP

The platform automatically determines the values of the country, region, city, zip, and time_zone fields b
take precedence and overwrite the manually passed values.
To update a profile using values from the country, region, city, zip and time_zone fields, the `"enrich_profile`

UTM-parameters

Parameter	Type
utm_content	**string**
utm_medium	**string**
utm_source	**string**
utm_campaign	**string**
utm_term	**string**

Network parameters

Parameter	Type	
net_ssid	**string**	**"10.200.0.0"**
net_mac_addr	**string**	**"28:87:ba:c1:0d:5e"**
net_gateway_ipv4	**string**	**"10.200.0.1"**
net_gateway_ipv6	**string**	**"::ff"**

Mailing parameters

Parameter	Type	Example	R
send_message_id	**string**	**"w4Wq877djSG1_2_a1_6t_2p_U_2_g. 2UCSmxz7zFnuCfCMK_5KmKH9"**	**No**

Loyalty programs parameters

Parameter	Type	
loyalty_id	**int**	**5**
promo_id	**string**	**"6285183c3f62fcd4ac5ed36d"**

Market parameters

Parameter	Type	Example	Required
endpoint_eid	**string**	**"example_endpoint"**	**No**
order_eid	**string**	**"12345"**	**No**
product_eid	**string**	**"example_product"**	**Yes, if sku_eid is passed**

Parameter	Type	Example	Required
sku_eid	**string**	**"example_sku"**	**No**
region_eid	**string**	**"region_eYFe2AE6s"**	**No**
count_items	**int**	**1**	**No**
categories	**array**	["category_1", "category_2"]	**No**

INFO

The maximum length of any string field is 128 characters.

Request examples

1 goal, no original request header data used:

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 1,
    "goal": "card",
    "pixel_id": 1,
    "referer": "api.example.com",
    "value": -25.5,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <token>2fefb577533d4cae919350c450755239</token>
    <db_id>1</db_id>
    <goal>card</goal>
    <pixel_id>1</pixel_id>
    <referer>api.example.com</referer>
    <value>-25.5</value>
    <matching>profile_id</matching>
    <profile_id>65d49bc74a6f1407ac656547</profile_id>
</xml>
```

2 goals, IP and referer from the original request head are used:

JSON

```
{
    "token": "91f1dfa81c264a938b475677c60ce967",
    "client_ip": "10.20.30.40",
    "db_id": 24,
    "goals": [
        {
            "goal": "cart",
            "value": 10.99
        },
        {
            "goal": "two",
            "value": -12.34
        }
    ],
    "pixel_id": 7,
    "matching": "profile_id",
    "profile_id": "abcdefghijklmnqrstuvwxyz",
    "referer": "api.req.from.my.site"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <client_ip>10.20.30.40</client_ip>
    <db_id>24</db_id>
    <goals>
        <goal>cart</goal>
        <value>10.99</value>
    </goals>
    <goals>
        <goal>two</goal>
        <value>-12.34</value>
    </goals>
    <pixel_id>7</pixel_id>
    <matching>profile_id</matching>
    <profile_id>abcdefghijklmnqrstuvwxyz</profile_id>
    <referer>api.req.from.my.site</referer>
</xml>
```

Example of a request to send additional information about an event:

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "db_id": 24,
    "goals": [
        {
            "goal": "cart",
            "value": 10.99
        },
        {
            "goal": "two",
            "value": -12.34
        }
    ],
    "client_ip": "192.168.0.0",
    "pixel_id": 7,
    "referer": "api.req.from.my.site",
    "accept_language": "fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5",
    "user_agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36",
    "matching": "profile_id",
    "profile_id": "651516181a5c52f94532fb1d",
    "pixel_data": {
        "hello": "hi",
        "arr": [
            1,
            2,
            3
        ],
        "int_val": 56893,
        "bool_val": true,
        "date_val": "2023-08-12"
    },
    "skip_triggers": false,
    "enrich_profile": true,
    "is_test": true,
    "vendor": "altcraft",
    "external_type": "event_pixel_test",
    "external_event_id": "qwerty123",
    "altcraft_client_id": "9f11d32d-2935-43d0-b8df-f4c3568095b2",
    "browser": "Chrome",
    "os": "Ubuntu",
    "device": "Desktop",
    "utm_content": "utm_content",
    "utm_medium": "utm_medium",
    "utm_source": "utm_source",
    "utm_campaign": "utm_campaign",
    "utm_term": "utm_term",
    "client_ip_v4": "198.16.74.224",
    "client_ip_v6": "fe80::5215:5556:d75f:806b",
    "net_ssid": "10.200.0.0",
    "net_mac_addr": "28:87:ba:c1:0d:5e",
    "net_gateway_ipv4": "10.200.0.1",
    "net_gateway_ipv6": "::ff",
    "loyalty_id": 5,
    "promo_id": "6285183c3f62fcd4ac5ed36d",
    "endpoint_eid": "online_store",
    "order_eid": "12345",
    "product_eid": "hd_450_bt_earphones",
    "sku_eid": "hd_450bt_earphones_black",
    "region_eid": "region_eYFe2AE6s",
    "count_items": 1,
    "categories": [
        "category_KBZ1qurFP"
    ],
    "lat": 55.244075,
    "lon": 19.379932,
    "country": "RU",
    "region": "Ryazan",
    "city": "Ryazan",
    "address": "Pochtovaya street",
    "zip": "390000",
    "time_zone": "Europe/Moscow",
    "send_message_id": "w4Wq877djSG1_2_a1_6t_2p_U_2_g.2UCSmxz7zFnuCfCMK_5KmKH9"
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <db_id>24</db_id>
    <goals>
        <goal>cart</goal>
        <value>10.99</value>
    </goals>
    <goals>
        <goal>two</goal>
        <value>-12.34</value>
    </goals>
    <client_ip>192.168.0.0</client_ip>
    <pixel_id>7</pixel_id>
    <referer>api.req.from.my.site</referer>
    <accept_language>fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5</accept_language>
    <user_agent>Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36</user_agent>
    <matching>profile_id</matching>
    <profile_id>651516181a5c52f94532fb1d</profile_id>
    <pixel_data>
        <hello>hi</hello>
        <arr>1</arr>
        <arr>2</arr>
        <arr>3</arr>
        <int_val>56893</int_val>
        <bool_val>true</bool_val>
        <date_val>2023-08-12</date_val>
    </pixel_data>
    <skip_triggers>false</skip_triggers>
    <enrich_profile>true</enrich_profile>
    <is_test>true</is_test>
    <vendor>altcraft</vendor>
    <external_type>event_pixel_test</external_type>
    <external_event_id>qwerty123</external_event_id>
    <altcraft_client_id>9f11d32d-2935-43d0-b8df-f4c3568095b2</altcraft_client_id>
    <browser>Chrome</browser>
    <os>Ubuntu</os>
    <device>Desktop</device>
    <utm_content>utm_content</utm_content>
    <utm_medium>utm_medium</utm_medium>
    <utm_source>utm_source</utm_source>
    <utm_campaign>utm_campaign</utm_campaign>
    <utm_term>utm_term</utm_term>
    <client_ip_v4>198.16.74.224</client_ip_v4>
    <client_ip_v6>fe80::5215:5556:d75f:806b</client_ip_v6>
    <net_ssid>10.200.0.0</net_ssid>
    <net_mac_addr>28:87:ba:c1:0d:5e</net_mac_addr>
    <net_gateway_ipv4>10.200.0.1</net_gateway_ipv4>
    <net_gateway_ipv6>::ff</net_gateway_ipv6>
    <loyalty_id>5</loyalty_id>
    <promo_id>6285183c3f62fcd4ac5ed36d</promo_id>
    <endpoint_eid>online_store</endpoint_eid>
    <order_eid>12345</order_eid>
    <product_eid>hd_450_bt_earphones</product_eid>
    <sku_eid>hd_450bt_earphones_black</sku_eid>
    <region_eid>region_eYFe2AE6s</region_eid>
    <count_items>1</count_items>
    <categories>category_KBZ1qurFP</categories>
    <lat>55.244075</lat>
    <lon>19.379932</lon>
    <country>RU</country>
    <region>Ryazan</region>
    <city>Ryazan</city>
    <address>Pochtovaya street</address>
    <zip>390000</zip>
    <time_zone>Europe/Moscow</time_zone>
    <send_message_id>w4Wq877djSG1_2_a1_6t_2p_U_2_g.2UCSmxz7zFnuCfCMK_5KmKH9</send_message_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Processing and adding a subscription

The API request is used to transfer the push token from the mobile application to the Altcraft Platform for
With this request, you can:

- **Add new subscriptions.**
- **Change the status of existing subscriptions.**

The profile search mode can be set using the `matching` field in the request (see specification below). If the
Moreover, if several databases are specified in the resource, then all of them will be processed separatel
If there are no databases in the resource, then an error will be returned.

Request URL

Method: POST

`https://<cookiesaver_host>/push_subscribe`

Request parameters

Parameter	Type	Example	Requ
id	**string**	**“abcdefghijklmnqrstuvwxyz”**	**No**
resource_token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
db_ids	**[ ] int**	**[2,3]**	**No**
matching	**string**	**“custom”**	**No**
subscriptions	**JSON array SubscriptionObject**	`[{ “provider” : “android-firebase”, “subscription_id” : “%subscriptionid%” }]`	**Yes**

SubscriptionObject

Parameter	Type	Example
provider	**string**	**“android-firebase”**
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**
unsubscribe	**bool**	**false**

Response parameters

Parameter	Type	
data	**JSON array DataObject**	**Information about the created subscriptions.** **The array can contain several objects in the following cases:** • **Several SubscriptionObject were passed in the request.** • **ID field was not used in the request, and the Altcraft MP resource uses several databases. In this case, a new**
error	**int**	**Error code**
error_text	**string**	**Error text**

DataObject

Parameter	Type	Example
subscription_id	**string**	**“abcdef-ghijklmn-qrstuvwxyz”**
unsubscribe	**bool**	**false**
profile_id	**string**	**“5fb62e815ae47a4aa04dfadd”**

Parameter	Type	Example
provider	**string**	**“android-firebase”**
db_id	**int**	**2**
resource_id	**int**	**1**
subscription_hash_id	**string**	**“92807e89”**
status	**string**	**“error”**
description	**string**	**“Duplicate push subscription”**

Examples

Request with matching by system profile ID

Request

```
{
  "id": "59db33d269d426509e9c4bf9",
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response

```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bf9",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

Request with matching by a custom profile field

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```

Response


```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfc",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```


Request with matching by push subscriptions

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "data": [
    {
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false,
      "profile_id": "59db33d269d426509e9c4bfa",
      "provider": "android-firebase",
      "db_id": 17,
      "resource_id": 5,
      "subscription_hash_id": "adsfdfdffdf",
      "status": "ok",
      "description": ""
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```


Request to a resource with several databases

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response

```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "provider": "android-firebase",
      "resource_id": 22,
      "status": "ok",
      "subscription_hash_id": "340ca49c",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    },
    {
      "db_id": 2,
      "description": "matching error: unknown field",
      "resource_id": 22,
      "status": "error"
    }
  ]
}
```


Request with matching by a custom profile field in a certain database

Request


```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "db_ids": [1],
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 123,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "error": 0,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 1,
      "description": "",
      "profile_id": "637f1a284fda7c5c08be62f",
      "provider": "android-firebase",
      "resource_id": 31,
      "status": "ok",
      "subscription_hash_id": "f8af057de",
      "subscription_id": "%subscriptionid%",
      "unsubscribe": false
    }
  ]
}
```


Request did not update or add any profile subscriptions

Request

```
{
  "resource_token": "3LXP2FRcRxJ-7b14db88ec937f14",
  "matching": "custom",
  "field_name": "CustomNumber1",
  "field_value": 12345,
  "subscriptions": [
    {
      "provider": "android-firebase",
      "subscription_id": "%subscriptionid%"
    }
  ]
}
```


Response


```
{
  "error": 200,
  "error_text": "Successful operation",
  "data": [
    {
      "db_id": 34,
      "description": "",
      "profile_id": "6156ed18eb7fec10a206e916",
      "resource_id": 22,
      "status": "ok"
    }
  ]
}
```

Add app push events

Description

Adds application push events.

Request URL

Method: POST

`https://example.com/api/v1.1/pushes/post_events`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
events	**int**	**1 - delivered** **2 - opened** **3 - delivered and opened**
launch_id	**int**	**1**
provider	**string**	**“android-firebase”**
subscription_id	**string**	**“subscriptionid”**
uid	**string**	**“uid”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "events": 1,
  "launch_id": "launch_id",
  "provider": "android-firebase",
  "subscription_id": "subscriptionid",
  "uid": "uid"
}
```

XML

```
<xml>
 <token>abcdefghijklmnqrstuvwxyz</token>
  <events>1</events>
  <launch_id>launch_id</launch_id>
  <provider>android-firebase</provider>
  <subscription_id>subscriptionid</subscription_id>
  <uid>uid</uid>
</xml>
```

Request example with uid in special format

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "events": 1,
  "uid": "e|2_4S_4HFn8RDhhmA|w4HFn8RPDFUi_2_4S_S_3___6.2NvyN8Hb67wmzJshM_6BuFTt|ios-apns"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <events>1</events>
    <uid>e|2_4S_4HFn8RDhhmA|w4HFn8RPDFUi_2_4S_S_3___6.2NvyN8Hb67wmzJshM_6BuFTt|ios-apns</uid>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Order data object

Description

Order data object is used in order import requests.

Parameter	Type	Example	Required	
external_id	string	“123456”	Yes	Order additional ID
matching	string	“email”	No	Subscriber search mode. By default – email For each type of matching, certai
endpoint	string	“website”	Yes	Touchpoint ID
tags	array	[“one”, “two”]	No	List of tags
status	string	“new” “delivered” “paid” “canceled” “part_refunded”	Yes	Order status ID
delivery_cost	string	“200”	No	Cost of delivery
total_price	string	“3000”	Yes	Total cost of the order. It consists of the cost of each ite
currency	string	“USD”	Yes	The currency that is used to indi Currencies are configured in the
create_time	string	“2006-01-25” “2006-01-25 15:04:05”	No	Order creation date (RFC 3339 fo
update_time	string	“2006-01-25” “2006-01-25 15:04:05”	No	Order update date (RFC 3339 for

custom fields	object	{
		“order_field_id_1”: “data”,
		“order_field_id_2”: 123
		}
			No	Custom order fields
				`"[field ID]": "[field value]"`

region	string	“region_external_id”	No	Region additional ID. You can create a region and get i

lines	array	[
		{
		“external_id”: “7777”,
		“product”: “product_external_id”,
		“base_price_per_item”: “600”,
		“min_price_per_item”: “300”,
		“final_price_per_item”: “500”,
		“count”: 5
		}
		]
			Yes	Order item data. At least one pos
				
				See below.

Object structure

JSON

```
[
  {
    "external_id": "123456",
    "matching": "email",
    "email": "johndoe@example.com",
    "endpoint": "endpoint_external_id",
    "tags": [
      "one",
      "two"
    ],
    "status": "new",
    "delivery_cost": "100",
    "total_price": "1000",
    "currency": "USD",
    "custom_fields": {
      "order_field_id_1": "data",
      "order_field_id_2": 123
    },
    "region": "region_external_id",
    "lines": [
      {
        "external_id": "1234000",
        "status": "new",
        "sku": "sku_external_id",
        "base_price_per_item": "900",
        "min_price_per_item": "900",
        "final_price_per_item": "900",
        "count": 5,
        "custom_fields": {
          "line_field_id_2": "data",
          "line_field_id_1": 123
        }
      }
    ]
  }
]
```


XML


```
<xml>
    <row>
        <external_id>123456</external_id>
        <matching>email</matching>
        <email>johndoe@example.com</email>
        <endpoint>endpoint_external_id</endpoint>
        <tags>one</tags>
        <tags>two</tags>
        <status>new</status>
        <delivery_cost>100</delivery_cost>
        <total_price>1000</total_price>
        <currency>USD</currency>
        <custom_fields>
            <order_field_id_1>data</order_field_id_1>
            <order_field_id_2>123</order_field_id_2>
        </custom_fields>
        <region>region_external_id</region>
        <lines>
            <external_id>1234000</external_id>
            <status>0001</status>
            <sku>sku_external_id</sku>
            <base_price_per_item>900</base_price_per_item>
            <min_price_per_item>900</min_price_per_item>
            <final_price_per_item>900</final_price_per_item>
            <count>5</count>
            <custom_fields>
                <line_field_id_2>data</line_field_id_2>
                <line_field_id_1>123</line_field_id_1>
            </custom_fields>
        </lines>
    </row>
</xml>
```


Lines object

Description

`lines` **is the array of objects. It contains objects with data about order items. One object stores one order**

Parameter	Type	Example	Required
external_id	**string**	**“1235678”**	**Yes**
status	**string**	**“new”**	**No**
sku	**string**	**IPhone_Red_128Gb**	**Do not pass it in the request if you pass the `product` parameter**
product	**string**	**MP3_player_BassPro**	**Do not pass it in the request if you pass the `sku` parameter**
base_price_per_item	**string**	**“600”**	**Yes**
min_price_per_item	**string**	**“300”**	**Yes**
final_price_per_item	**string**	**“500”**	**Yes**
count	**int**	**5**	**Yes**
custom fields	**object**	{ “line_field_id_2”: 123, “line_field_id_2”: “data” }	**No**

Object stucture

JSON

```
[
  {
    "external_id": "12345",
    "status": "new",
    "product": "MP3_player_BassPro",
    "base_price_per_item": "600",
    "min_price_per_item": "300",
    "final_price_per_item": "500",
    "count": 5,
    "custom_fields": {
      "lot_number": 123,
      "imported": true
    }
  }
]
```

Example of SKU:

```
[
  {
    "external_id": "76543",
    "status": "processing",
    "sku": "IPhone_Red_128Gb",
    "base_price_per_item": "1000",
    "min_price_per_item": "700",
    "final_price_per_item": "850",
    "count": 13,
    "custom_fields": {
      "lot_number": 93,
      "localization": "RU"
    }
  }
]
```

XML

```
<xml>
<external_id>1235678</external_id>
  <status>0001</status>
  <sku>sku_external_id</sku>
  <base_price_per_item>600</base_price_per_item>
  <min_price_per_item>300</min_price_per_item>
  <final_price_per_item>500</final_price_per_item>
  <count>5</count>
  <custom_fields>
    <line_field_id_2>data</line_field_id_2>
  </custom_fields>
</xml>
```

Product data object

Description

Product data object used in product import requests.

Parameter	Type	Example	Required	
name	**string**	**“Canon EOS Digital Camera”**	**Yes**	**Product’s name.**
external_id	**string**	**“12345”**	**Yes**	**Product ID that is**
price	**string**	**“490”**	**No**	**Current price of t**
old_price	**string**	**“520”**	**No**	**Old price of the p**
currency	**string**	**“USD”**	**No**	**The currency tha** **Currencies are c**
tags	**array**	[“camera”, “Canon”]	**No**	**List of tags.**
manufacturer_name	**string**	**“Canon”**	**No**	**Manufacturer’s n**
description	**string**	**“Test description”**	**No**	**Product descripti** **The length of the**
url	**string**	**“https://example.com”**	**No**	**URL of the produ**
pictures	**array**	[“https://example_1.com”, “https://example_2.com”]	**No**	**List of URL links**
count	**int**	**60**	**No**	**The number of pr**
expiration_time	**string**	**“730”**	**No**	**Shelf life or servi**
is_available	**bool**	**true**	**No**	**Whether the prod**
delivery	**bool**	**true**	**No**	**Possibility of cou**
barcode	**array**	[5235235, 347347, 335578769]	**No**	**List of product ba**
manufacturer_warranty	**bool**	**true**	**No**	**Manufacturer’s o**

custom_fields	**object**	{
		“color”: “black”,
		“weight”: 701,
		“dimensions”: “105*141*77”
		}
			No	**Custom product f**
				Learn more

regional_data	**array**	[
		{
		“external_id”: “region_YnCrJj8Y1”,
		“price”: “36000”,
		“old_price”: “38500”,
		“currency”: “RUB”,
		“custom_fields”:{
		“color”: “black”,
		“weight”: 701
		},
		“count”: 30,
		“is_available”: true,
		“delivery”: true
		}
		]
			No	**Product data that**
				`external_id` **is the**

categories	**array**	[“category_1”, “category_2”]	**No**	**Product category** **Learn more**

sku	**array**	[
		{
		“name”: “Canon EOS 90D Body”,
		“external_id”: “12345678”,
		}
		]
			No	**SKU data for this**
				Learn more

JSON

JSON

```
[
    {
        "name": "iPhone",
        "external_id": "123456",
        "price": "500",
        "old_price": "400",
        "currency": "USD",
        "tags": [
            "one",
            "two"
        ],
        "manufacturer_name": "Apple",
        "description": "Test description",
        "url": "https://example.com",
        "pictures": [
            "https://example.com",
            "https://example.com"
        ],
        "count": 60,
        "expiration_time": "730",
        "is_available": true,
        "delivery": true,
        "barcode": [
            5235235,
            347347,
            335578769
        ],
        "manufacturer_warranty": false,
        "custom_fields": {
            "product_field_id_1": 123,
            "product_field_id_2": "data"
        },
        "regional_data": [
            {
                "external_id": "region_YnCrJj8Y1",
                "price": "864825",
                "old_price": "400",
                "currency": "USD",
                "custom_fields": {
                    "product_field_id_1": 123,
                    "product_field_id_2": "data"
                },
                "count": 30,
                "is_available": true,
                "delivery": true
            }
        ],
        "categories": [
            "category_1",
            "category_2"
        ],
        "sku": [
            {
                "name": "iPhone",
                "external_id": "12345678",
            }
        ]
    }
]
```


XML

```xml
<xml>
  <row>
    <name>iPhone</name>
    <external_id>123456</external_id>
    <price>500</price>
    <old_price>400</old_price>
    <currency>USD</currency>
    <tags>one</tags>
    <tags>two</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <description>Test description</description>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>60</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>5235235</barcode>
    <barcode>347347</barcode>
    <barcode>335578769</barcode>
    <manufacturer_warranty>false</manufacturer_warranty>
    <custom_fields>
      <product_field_id_1>123</product_field_id_1>
      <product_field_id_2>data</product_field_id_2>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>864825</price>
      <old_price>400</old_price>
      <currency>USD</currency>
      <custom_fields>
        <product_field_id_1>123</product_field_id_1>
        <product_field_id_2>data</product_field_id_2>
      </custom_fields>
      <count>30</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
    <categories>category_1</categories>
    <categories>category_2</categories>
    <sku>
      <name>iPhone</name>
      <external_id>12345678</external_id>
    </sku>
  </row>
</xml>
```

SKU data object

Description

SKU data object used in SKU import requests.

Parameter	Type	Example
name	**string**	**“Canon EOS 90D Body”**
external_id	**string**	**“12345678”**
price	**string**	**“510”**
old_price	**string**	**“600”**
currency	**string**	**“USD”**
tags	**array**	[“camera”, “Canon”]
manufacturer_name	**string**	**“Canon”**
description	**string**	**“test description”**
product_external_id	**string**	**“12345”**
url	**string**	**“https://example.com”**
pictures	**array**	[“https://example_1.com”, “https://example_2.com”]
count	**int**	**10**
expiration_time	**string**	**“730”**
is_available	**bool**	**true**
delivery	**bool**	**false**
barcode	**array**	[5235235, 347347, 335578769]
manufacturer_warranty	**bool**	**true**

custom_fields	**object**	{
		“color”: “black”,
		“weight”: 730,
		“dimensions”: “105*139*79”
		}

regional_data	**array**	[
		{
		“external_id”: “region_YnCrJj8Y1”,
		“price”: “36000”,
		“old_price”: “38500”,
		“currency”: “RUB”,
		“custom_fields”:{
		“color”: “black”,
		“weight”: 701
		},
		“count”: 10,
		“is_available”: true,
		“delivery”: true
		}
		]

categories	**array**	[“category_1”, “category_2”]

JSON

JSON

```
[
  {
    "name": "iPhone",
    "external_id": "12345",
    "price": "235",
    "old_price": "400",
    "currency": "USD",
    "description": "",
    "product_external_id": "12345678",
    "categories": ["1234", "12345"],
    "tags": ["one", "two"],
    "manufacturer_name": "Apple",
    "url": "https://example.com",
    "pictures": ["https://example.com", "https://example.com"],
    "count": 50,
    "expiration_time": "730",
    "is_available": true,
    "delivery": true,
    "barcode": [23346346, 23546362, 76834566],
    "manufacturer_warranty": true,
    "custom_fields": {
      "sku_custom_field_1": 123,
      "sku_custom_field_2": "data"
    },
    "regional_data": [
      {
        "external_id": "region_YnCrJj8Y1",
        "price": "864825",
        "old_price": "400",
        "currency": "USD",
        "custom_fields": {
          "sku_custom_field_1": 123,
          "sku_custom_field_2": "data"
        },
        "count": 30,
        "is_available": true,
        "delivery": true
      }
    ]
  }
]
```


XML

```
<xml>
  <row>
    <name>iPhone</name>
    <external_id>12345</external_id>
    <price>235</price>
    <old_price>400</old_price>
    <currency>USD</currency>
    <description></description>
    <product_external_id>12345678</product_external_id>
    <categories>1234</categories>
    <categories>12345</categories>
    <tags>one</tags>
    <tags>two</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>50</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>23346346</barcode>
    <barcode>23546362</barcode>
    <barcode>76834566</barcode>
    <manufacturer_warranty>true</manufacturer_warranty>
    <custom_fields>
      <sku_custom_field_1>123</sku_custom_field_1>
      <sku_custom_field_2>data</sku_custom_field_2>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>864825</price>
      <old_price>400</old_price>
      <currency>USD</currency>
      <custom_fields>
        <sku_custom_field_1>123</sku_custom_field_1>
        <sku_custom_field_2>data</sku_custom_field_2>
      </custom_fields>
      <count>30</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
  </row>
</xml>
```

Categories array

The `categories` array is used to create categories and subcategories in the market. Each object creates or

- **Import products, SKUs and categories**
- **Import SKUs and categories**

Request parameters

Parameter	Type	E
external_id	**string**	**“1234”**
name	**string**	**“Cell phones and accessories”**
parent_external_id	**string**	**“123”**

JSON

JSON

```
[
  {
          "external_id": "123",
          "name": "Test category",
  },
  {
          "external_id": "12345",
          "name": "Test subcategory",
          "parent_external_id": "123"
  }
]
```

XML

```
<xml>
<external_id>123</external_id>
    <name>Test category</name>
  </row>
  <row>
    <external_id>12345</external_id>
    <name>Test subcategory</name>
    <parent_external_id>123</parent_external_id>
</xml>
```

Custom fields array

The `custom_fields` array is used to create custom fields for various market entities (order, order item, produ
The array is passed in the following API requests:

- **Import products, SKUs and categories**
- **Import SKUs and categories**
- **Import order and item statuses**

Request parameters

Parameter	Type	Example	
external_id	**string**	**“1234”**	**Yes**
name	**string**	**“color”**	**Yes**
entity_type	**string**	**“product”** **“sku”** **“order”** **“line_order”**	**Yes**
required	**bool**	**false**	**Yes**
value_type	**string**	**“unicode”** **“int”** **“float”** **“bool”** **“enum”** **“date”**	**Yes**
default_value	**string**	**“black”**	**Yes**
enum_values	**object**	`{"one": 1, "two": 2}`	**Yes, if val**

JSON

JSON

```
[
  {
        "external_id": "12",
        "name": "product_custom_field",
        "entity_type": "product",
        "required": true,
        "value_type": "unicode",
        "default_value": "data"
  },
  {
        "external_id": "123",
        "name": "sku_custom_field",
        "entity_type": "sku",
        "required": false,
        "value_type": "int",
        "default_value": 100
   },
   {
        "external_id": "1234",
        "name": "order_custom_field",
        "entity_type": "order",
        "required": false,
        "value_type": "enum",
        "default_value": {"one": 1, "two": 2}
   },
   {
        "external_id": "1234",
        "name": "order_custom_field",
        "entity_type": "line_order",
        "required": false,
        "value_type": "bool",
        "default_value": false
   }
]
```

XML

```
<xml>
  <row>
    <external_id>12</external_id>
    <name>product_custom_field</name>
    <entity_type>product</entity_type>
    <required>true</required>
    <value_type>unicode</value_type>
    <default_value>data</default_value>
  </row>
  <row>
    <external_id>123</external_id>
    <name>sku_custom_field</name>
    <entity_type>sku</entity_type>
    <required>false</required>
    <value_type>int</value_type>
    <default_value>100</default_value>
  </row>
  <row>
    <external_id>1234</external_id>
    <name>order_custom_field</name>
    <entity_type>order</entity_type>
    <required>false</required>
    <value_type>enum</value_type>
    <default_value>
      <one>1</one>
      <two>2</two>
    </default_value>
  </row>
  <row>
    <external_id>1234</external_id>
    <name>order_custom_field</name>
    <entity_type>line_order</entity_type>
    <required>false</required>
    <value_type>bool</value_type>
    <default_value>false</default_value>
  </row>
</xml>
```

Import order and item statuses

Description

Imports an order, payment method and statuses of order items into the market.

Request URL

Method: POST

`https://example.com/api/v1.1/market/orders/import`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
custom_fields	**array of objects**	`[ { “external_id”: “1234”, “name”: “1234”, “entity_type”: “line_order”, “required”: false, “value_type”: “unicode”, “default_value”: “data” } ]`
custom_line_statuses	**array of objects**	`[ { “external_id”: “1234”, “name”: “Arrived at the pickup point”, “type”: “in_process” } ]`
skip_triggers	**bool**	**true**
data	**array of objects**	`[{ “external_id”: “123456”, “matching”:“email”, “email”:“example@example.com” “endpoint”: “endpoint_external_id”, “status”: “new”, “total_price”: “3000”, “currency”: “USD”, “create_time”: “2022-01-02T15:04:05.141Z”, “lines”: [{ “external_id”: “7777”, “status”: “new”, “product”: “product_external_id”, “base_price_per_item”: “600”, “min_price_per_item”: “300”, “final_price_per_item”: “500”, “count”: 5 }] }]`

TIP

If you need to set a value for an existing additional field, pass an object in `custom_fields` where the key is

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "custom_fields": [
{
"external_id": "order_field_id_123",
"name": "Additional order field",
"entity_type": "order",
"required": false,
      "value_type": "unicode",
      "default_value": "value of the test field"
    },
    {
"external_id": "line_field_id_123",
"name": "Additional field of the order item",
"entity_type": "line_order",
"required": false,
      "value_type": "unicode",
      "default_value": "value of the test field"
    }
  ],
  "custom_line_statuses": [
{
"external_id": "status_id_123",
"name": "Delivered to pick-up point",
"type": "in_process"
    }
  ],
  "skip_triggers": true,
  "data": [
    {
      "external_id": "1234000",
      "matching": "email",
      "email": "johndoe@example.com",
      "endpoint": "website",
      "tags": ["one", "two"],
      "status": "delivered",
      "delivery_cost": "100",
      "total_price": "1000",
      "currency": "USD",
      "create_time": "2022-01-02T15:04:05.141Z",
      "custom_fields": {
        "order_field_id_123": "data"
      },
      "region": "region_YnCrJj8Y1",
      "lines": [
        {
          "external_id": "line_01",
          "status": "status_id_123",
          "product": "iphone_test_1",
          "base_price_per_item": "900",
          "min_price_per_item": "900",
          "final_price_per_item": "900",
          "count": 1,
          "custom_fields": {
            "line_field_id_123": "data"
          }
        }
      ]
    }
  ]
}
```

XML

```
<xml>
<token>abcdefghijklmnqrstuvwxyz</token>
  <custom_fields>
    <external_id>order_field_id_123</external_id>
    <name>Additional order field</name>
    <entity_type>order</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>value of the test field</default_value>
  </custom_fields>
  <custom_fields>
    <external_id>line_field_id_123</external_id>
    <name>Additional field of the order item</name>
    <entity_type>line_order</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>value of the test field</default_value>
  </custom_fields>
  <custom_line_statuses>
    <external_id>status_id_123</external_id>
    <name>Delivered to the pick-up point</name>
    <type>in_process</type>
  </custom_line_statuses>
  <skip_triggers>true</skip_triggers>
  <data>
    <external_id>1234000</external_id>
    <matching>email</matching>
    <email>johndoe@example.com</email>
    <endpoint>website</endpoint>
    <tags>one</tags>
    <tags>two</tags>
    <status>delivered</status>
    <delivery_cost>100</delivery_cost>
    <total_price>1000</total_price>
    <currency>USD</currency>
    <create_time>2022-01-02T15:04:05.141Z</create_time>
    <custom_fields>
      <order_field_id_123>data</order_field_id_123>
    </custom_fields>
    <region>region_YnCrJj8Y1</region>
    <lines>
      <external_id>line_01</external_id>
      <status>status_id_123</status>
      <product>iphone_test_1</product>
      <base_price_per_item>900</base_price_per_item>
      <min_price_per_item>900</min_price_per_item>
      <final_price_per_item>900</final_price_per_item>
      <count>1</count>
      <custom_fields>
        <line_field_id_123>data</line_field_id_123>
      </custom_fields>
    </lines>
  </data>
</xml>
```

INFO

`custom_fields` **at the root of the request creates new fields (an array), while in** `data` **it fills them with value**

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**Product IDs that were su**

Response example

JSON

```
{
  "error": 0,
  "error_text": "",
  "result": [
    {
      "error": 0,
      "external_id": "1234000"
    }
  ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <error>0</error>
        <external_id>1234000</external_id>
    </result>
</xml>
```

Get orders list

Description

Retrieves a list of all orders in an account or orders for a specific touchpoint.

Request URL

Method: POST

`https://example.com/api/v1.1/market/orders/list`

Request parameters

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
from_id	**string**	**“2oDantiaiPtY-b7b41276360564d4”**	**No**
limit	**int**	**3**	**No**
endpoint	**string**	**“website”**	**No**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "from_id": "",
  "limit": 1,
  "endpoint": "qwerty"
}
```

XML

```
<xml>
 <token>abcdefghijklmnqrstuvwxyz</token>
  <from_id></from_id>
  <limit>1</limit>
  <endpoint>qwerty</endpoint>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "next_from_id": "2p2FXnb7zzoh-26167c2af5162ca4",
    "result": [
        {
            "profile": {
                "profile_id": "61addfc13f62fc8440i8b013",
                "xxh": "636gcaf5614536asf"
            },
            "external_id": "12345678-0001",
            "endpoint": "website",
            "status": "new",
            "delivery_cost": "50000",
            "total_price": "4549000",
            "currency": "RUB",
            "custom_fields": {},
            "lines": [
                {
                    "external_id": "line_DwWuNkAYV",
                    "product": "xiaomi_11T",
                    "base_price_per_item": "4499000",
                    "min_price_per_item": "0",
                    "final_price_per_item": "4499000",
                    "count": 1,
                    "custom_fields": {},
                    "currency": "RUB"
                }
            ],
            "create_time": "2021-12-27T09:38:39Z",
            "update_time": "2021-12-27T09:38:39Z"

        }
    ]
}
```


XML


```
<xml>
    <next_from_id>2p2FXnb7zzoh-26167c2af5162ca4</next_from_id>
    <result>
        <profile>
            <profile_id>61addfc13f62fc8440i8b013</profile_id>
            <xxh>636gcaf5614536asf</xxh>
        </profile>
        <external_id>12345678-0001</external_id>
        <endpoint>website</endpoint>
        <status>new</status>
        <delivery_cost>50000</delivery_cost>
        <total_price>4549000</total_price>
        <currency>RUB</currency>
        <lines>
            <external_id>line_DwWuNkAYV</external_id>
            <product>xiaomi_11T</product>
            <base_price_per_item>4499000</base_price_per_item>
            <min_price_per_item>0</min_price_per_item>
            <final_price_per_item>4499000</final_price_per_item>
            <count>1</count>
            <currency>RUB</currency>
        </lines>
        <create_time>2021-12-27T09:38:39Z</create_time>
        <update_time>2021-12-27T09:38:39Z</update_time>
    </result>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**string**	**The pointer must be passed in the next request to get another piece of data until the output of the resu**
result	**array**	**Array of received data on orders**

Delete order

Description

Delete an order in the market, including from the profile history (the request completely deletes the order In the request, you need to use a token that is assigned a role with the appropriate access rules.

URL

Method: POST

`https://example.com/api/v1.1/market/orders/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
external_id	**string**	**“A598t32D”**

Request example

JSON

```
{
         "token": "abcdefghijklmnqrstuvwxyz",
         "id": 1
}

{
         "token": "abcdefghijklmnqrstuvwxyz",
         "external_id": "A598t32D"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
</xml>

<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <external_id>"A598t32D"</external_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": ""
}
```

XML

```
<xml>
   <error>0</error>
   <error_text></error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get order status

Description

Get information of order status.
In the request, you need to use a token that is assigned a role with the appropriate access rules.

URL

Method: POST

`https://example.com/api/v1.1/market/orders/get_status`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
external_id	**string**	**“A598t32D”**

Request example JSON

JSON

```
{
         "token": "abcdefghijklmnqrstuvwxyz",
         "id": 1
}

{
         "token": "abcdefghijklmnqrstuvwxyz",
         "external_id": "A598t32D"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
</xml>

<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <external_id>"A598t32D"</external_id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "result": {
        "endpoint": "marketplace",
        "status": "delivered"
    }
}
```

XML

```
<xml>
   <error>0</error>
   <error_text></error_text>
   <result>
      <endpoint>"marketplace"</endpoint>
      <status>"delivered"</status>
   </result>
</xml>
```

Response parameters

Parameter	Type
result	**array**
error	**int**
error_text	**string**

Update order line status

Description

The request updates the status of an order line in the Market.

Request URL

Method: POST

`https://example.com/api/v1.1/market/orders/update_order_line_status`

Request parameters

Parameter	Type	
token	**string**	**“abcdef**
format	**string**	**“json”**
order_external_id	**string**	**“order_1**
order_line_external_id	**string**	**“line_DB**
status_external_id	**string**	• d • p • n • r • c • d • <

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "format":"json",
    "order_external_id":"order_1",
    "order_line_external_id":"line_DBSrmlVYD",
    "status_external_id":"delivered"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <format>json</format>
  <order_external_id>order_1</order_external_id>
  <order_line_external_id>line_DBSrmlVYD</order_line_external_id>
  <status_external_id>delivered</status_external_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "order_external_id": "order_1",
        "order_line_external_id": "line_DBSrmlVYD",
        "status_external_id": "delivered"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <order_external_id>order_1</order_external_id>
        <order_line_external_id>line_DBSrmlVYD</order_line_external_id>
        <status_external_id>delivered</status_external_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
data	**object**

Import products, SKUs and categories

Description

Allows you to import 3 types of entities at the same time: products, SKUs and categories.

Request URL

Method: POST

`https://example.com/api/v1.1/market/products/import`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
custom_fields	**array of objects**	`[ { “external_id”: “1234”, “name”: “product_custom_field”, “entity_type”: “product”, “required”: false, “value_type”: “unicode”, “default_value”: “data” } ]`
categories	**array of objects**	`[ { “external_id”: “1234”, “name”: “Camera & Photo”, “parent_external_id”: “123” } ]`
data	**array of objects**	`[ { “name”: “iPhone”, “product_external_id”: “12345678”, “external_id”: “123456” } ]`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "custom_fields": [
{
"external_id": "test_field_string",
"name": "String field",
"entity_type": "sku",
"required": false,
      "value_type": "unicode",
      "default_value": "Default value of the sku test field"
    }
  ],
  "categories": [
{
"external_id": "test_category_2",
"name": "Test category 2"
}
],
  "data": [
    {
      "name": "Apple iPhone 13 256GB",
      "external_id": "sku_test_1",
      "price": "1000",
      "old_price": "1100",
      "currency": "USD",
"description": "SKU description",
"product_external_id": "iphone_test_1",
"categories": ["test_category_2"],
      "tags": ["phone", "Apple"],
      "manufacturer_name": "Apple",
      "url": "https://example.com",
      "pictures": ["https://example.com", "https://example.com"],
      "count": 10,
      "expiration_time": "730",
      "is_available": true,
      "delivery": true,
      "barcode": [23346346, 23546362, 76834566],
      "manufacturer_warranty": true,
      "custom_fields": {
"test_field_string": "Test field"
      },
      "regional_data": [
{
"external_id": "region_YnCrJj8Y1",
"price": "75000",
"old_price": "80000",
"currency": "RUB",
"custom_fields": {
"test_field_string": "Test field"
          },
          "count": 8,
          "is_available": true,
          "delivery": true
        }
      ]
    }
  ]
}
```


XML

```
<xml>
<token>abcdefghijklmnqrstuvwxyz</token>
  <custom_fields>
    <external_id>test_field_string</external_id>
    <name>String field</name>
    <entity_type>sku</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>Default value of the sku test field</default_value>
  </custom_fields>
  <categories>
    <external_id>test_category_2</external_id>
    <name>Test category 2</name>
  </categories>
  <data>
    <name>Apple iPhone 13 256GB</name>
    <external_id>sku_test_1</external_id>
    <price>1000</price>
    <old_price>1100</old_price>
    <currency>USD</currency>
    <description>SKU description</description>
    <product_external_id>iphone_test_1</product_external_id>
    <categories>test_category_2</categories>
    <tags>phone</tags>
    <tags>Apple</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>10</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>23346346</barcode>
    <barcode>23546362</barcode>
    <barcode>76834566</barcode>
    <manufacturer_warranty>true</manufacturer_warranty>
    <custom_fields>
      <test_field_string>Test field</test_field_string>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>75000</price>
      <old_price>80000</old_price>
      <currency>RUB</currency>
      <custom_fields>
        <test_field_string>Test field</test_field_string>
      </custom_fields>
      <count>8</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
  </data>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "result": [
        {
            "error": 0,
            "external_id": "sku_test_1"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <error>0</error>
        <external_id>sku_test_1</external_id>
    </result>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**Product IDs that were su**

Import SKUs and categories

Description

Imports 2 types of entities at the same time: SKUs and categories.

Request URL

Method: POST

`https://example.com/api/v1.1/market/sku/import`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
custom_fields	**array of objects**	`[ { “external_id”: “1234”, “name”: “1234”, “entity_type”: “sku”, “required”: false, “value_type”: “unicode”, “default_value”: “data” } ]`
categories	**array of objects**	`[ { “external_id”: “1234”, “name”: “Camera & Photo”, “parent_external_id”: “123” } ]`
data	**array of objects**	`[ { “name”: “iPhone”, “external_id”: “123456” } ]`

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "custom_fields": [
        {
          "external_id": "test_field_string",
          "name": "Custom field for SKU",
          "entity_type": "sku",
          "required": false,
          "value_type": "unicode",
          "default_value": "Default value of the SKU custom field"
        }
      ],
      "categories": [
        {
          "external_id": "test_category_2",
          "name": "Electronics"
        }
      ],
    "data": [
      {
        "name": "Apple iPhone 13 256GB",
        "external_id": "sku_test_1",
        "price": "1000",
        "old_price": "1100",
        "currency": "USD",
        "description":"SKU description",
        "product_external_id": "iphone_test_1",
        "categories": [
          "test_category_2"
        ],
        "tags": [
          "phone",
          "Apple"
        ],
        "manufacturer_name": "Apple",
        "url": "https://example.com",
        "pictures": [
          "https://example.com",
          "https://example.com"
        ],
        "count": 10,
        "expiration_time": "730",
        "is_available": true,
        "delivery": true,
        "barcode": [
          23346346,
          23546362,
          76834566
        ],
        "manufacturer_warranty": true,
        "custom_fields": {
          "test_field_string": "Test field value",
        },
        "regional_data": [
          {
            "external_id": "region_YnCrJj8Y1",
            "price": "75000",
            "old_price": "80000",
            "currency": "RUB",
            "custom_fields": {
                "test_field_string": "Test field value for this region",
            },
            "count": 8,
            "is_available": true,
            "delivery": true
          }
        ]
      }
    ]
  }
```

XML

```
<xml>
<token>abcdefghijklmnqrstuvwxyz</token>
  <custom_fields>
    <external_id>test_field_string</external_id>
    <name>Custom field for SKU</name>
    <entity_type>sku</entity_type>
    <required>false</required>
    <value_type>unicode</value_type>
    <default_value>Default value of the SKU custom field</default_value>
  </custom_fields>
  <categories>
    <external_id>test_category_2</external_id>
    <name>Electronics</name>
  </categories>
  <data>
    <name>Apple iPhone 13 256GB</name>
    <external_id>sku_test_1</external_id>
    <price>1000</price>
    <old_price>1100</old_price>
    <currency>USD</currency>
    <description>SKU description</description>
    <product_external_id>iphone_test_1</product_external_id>
    <categories>test_category_2</categories>
    <tags>phone</tags>
    <tags>Apple</tags>
    <manufacturer_name>Apple</manufacturer_name>
    <url>https://example.com</url>
    <pictures>https://example.com</pictures>
    <pictures>https://example.com</pictures>
    <count>10</count>
    <expiration_time>730</expiration_time>
    <is_available>true</is_available>
    <delivery>true</delivery>
    <barcode>23346346</barcode>
    <barcode>23546362</barcode>
    <barcode>76834566</barcode>
    <manufacturer_warranty>true</manufacturer_warranty>
    <custom_fields>
      <test_field_string>Test field value</test_field_string>
    </custom_fields>
    <regional_data>
      <external_id>region_YnCrJj8Y1</external_id>
      <price>75000</price>
      <old_price>80000</old_price>
      <currency>RUB</currency>
      <custom_fields>
        <test_field_string>Test field value for this region</test_field_string>
      </custom_fields>
      <count>8</count>
      <is_available>true</is_available>
      <delivery>true</delivery>
    </regional_data>
  </data>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "",
  "result": [
        {
          "error": 0,
          "external_id": "sku_test_1"
        }
  ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <error>0</error>
        <external_id>sku_test_1</external_id>
    </result>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
result	**array**	**IDs of SKUs that were suc**

Get products list

Description

Retrieves a list of products.

Request URL

Method: POST

`https://example.com/api/v1.1/market/products/list`

Request parameters

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
from_id	**string**	**“2oDantiaiPtY-b7b41276360564d4”**	**No**
limit	**int**	**3**	**No**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "from_id": "0014a2feb331ace2.20200124134127",
  "limit": 3
}
```

XML

```
<xml>
 <token>abcdefghijklmnqrstuvwxyz</token>
  <from_id>0014a2feb331ace2.20200124134127</from_id>
  <limit>3</limit>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "result": [
        {
            "name": "Xiaomi 11T",
            "external_id": "xiaomi_11T",
            "sku": [
                {
                    "name": "Xiaomi 11T 256ГБ",
                    "external_id": "xiaomi_11t_256gb",
                },
                {
                    "name": "Xiaomi 11T 128 ГБ",
                    "external_id": "xiaomi_11t_128gb",
                }
            ]
        },
        {
            "name": "Apple iPhone 13",
            "external_id": "iphone_13",
            "sku": [
                {
                    "name": "Apple iPhone 13 256GB",
                    "external_id": "iphone_13_256gb",
                },
                {
                    "name": "Apple iPhone 13 128GB",
                    "external_id": "iphone_13_128gb",
                }
            ]
        },
                  {
            "name": "Wireless headphones Xiaomi Mi True Pro",
            "external_id": "xiaomi_mi_true_pro",
        }
    ]
}
```


XML


```
<xml>
    <error>0</error>
    <error_text></error_text>
    <result>
        <name>Xiaomi 11T</name>
        <external_id>xiaomi_11T</external_id>
        <sku>
            <name>Xiaomi 11T 256ГБ</name>
            <external_id>xiaomi_11t_256gb</external_id>
        </sku>
        <sku>
            <name>Xiaomi 11T 128 ГБ</name>
            <external_id>xiaomi_11t_128gb</external_id>
        </sku>
    </result>
    <result>
        <name>Apple iPhone 13</name>
        <external_id>iphone_13</external_id>
        <sku>
            <name>Apple iPhone 13 256GB</name>
            <external_id>iphone_13_256gb</external_id>
        </sku>
        <sku>
            <name>Apple iPhone 13 128GB</name>
            <external_id>iphone_13_128gb</external_id>
        </sku>
    </result>
    <result>
        <name>Wireless headphones Xiaomi Mi True Pro</name>
        <external_id>xiaomi_mi_true_pro</external_id>
    </result>
</xml>
```


Response parameters

Parameter	Type	
error	**int**	**Error code**

Parameter	Type	
error_text	**string**	**Error text**
next_from_id	**string**	**The pointer must be passed in the next request to get another piece of data until the output of the resu**
result	**array**	**Array of received data on products and their SKUs**

Get SKUs list

Description

Retrieves a list of SKUs.

Request URL

Method: POST

`https://example.com/api/v1.1/market/sku/list`

Request parameters

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
from_id	**string**	**“2oDantiaiPtY-b7b41276360564d4”**	**No**
limit	**int**	**3**	**No**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "from_id": "",
  "limit": 3
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <from_id></from_id>
    <limit>3</limit>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "",
    "next_from_id": "L8PSdcsMmLq-6021b5621680598b",
    "result": [
        {
            "name": "Xiaomi 11T 256GB",
            "external_id": "xiaomi_11t_256gb"
        },
        {
            "name": "Xiaomi 11T 128GB",
            "external_id": "xiaomi_11t_128gb"
        },
        {
            "name": "Apple iPhone 13 256GB",
            "external_id": "iphone_13_256gb"
        }
    ]
}
```

XML

```
<xml>
    <error>0</error>
    <error_text></error_text>
    <next_from_id>L8PSdcsMmLq-6021b5621680598b</next_from_id>
    <result>
        <name>Xiaomi 11T 256GB</name>
        <external_id>xiaomi_11t_256gb</external_id>
    </result>
    <result>
        <name>Xiaomi 11T 128 GB</name>
        <external_id>xiaomi_11t_128gb</external_id>
    </result>
    <result>
        <name>Apple iPhone 13 256GB</name>
        <external_id>iphone_13_256gb</external_id>
    </result>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**string**	**The pointer must be passed in the next request to get another piece of data until the output of the resu**
result	**array**	**An array of received data on SKUs**

Delete products

Description

Delete products in the marketplace. The method supports both single deletion and batch deletion of multi
To delete, you must use a token for which a role with the appropriate access rights is assigned.

Request URL

Method: POST

`https://example.com/api/v1.1/market/products/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
ids	**array of integers**	**[1, 2, 3]**
external_ids	**array of strings**	**[“product_1”, “product_2”]**

Request example

Single JSON

Identification by parameter: `"ids"`:

```
{
        "token": "abcdefghijklmnqrstuvwxyz",
        "ids": [1]
}
```

Identification by parameter: `"external_ids"`:

```
{
        "token": "abcdefghijklmnqrstuvwxyz",
        "external_ids": ["product_123"]
}
```

Batch JSON

```
{
        "token": "abcdefghijklmnqrstuvwxyz",
        "ids": [1, 2],
        "external_ids": ["product_3"]
}
```

Single XML

Identification by parameter: `"ids"`:

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <ids>1</ids>
</xml>
```

Identification by parameter: `"external_ids"`:

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <external_ids>product_123</external_ids>
</xml>
```

Batch XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <ids>1</ids>
  <ids>2</ids>
  <external_ids>product_3</external_ids>
</xml>
```

Response example

JSON

```
{
    "deleted_count": 2,
    "error": 0,
    "error_text": ""
}
```

XML

```
<xml>
   <deleted_count>2</deleted_count>
   <error>0</error>
   <error_text></error_text>
</xml>
```

Response parameters

Parameter	Type
deleted_count	**int**
error	**int**
error_text	**string**

Delete SKU

Description

Delete SKUs (stock keeping units) in the marketplace. The method supports both single deletion and batc
To delete, you must use a token for which a role with the appropriate access rights is assigned.

Request URL

Method: POST

`https://example.com/api/v1.1/market/sku/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
ids	**array of integers**	**[1, 2, 3]**
external_ids	**array of strings**	**[“sku_1”, “sku_2”]**

Request example

Single JSON

Identification by parameter: `"ids"`:

```
{
        "token": "abcdefghijklmnqrstuvwxyz",
        "ids": [1]
}
```

Identification by parameter: `"external_ids"`:

```
{
        "token": "abcdefghijklmnqrstuvwxyz",
        "external_ids": ["sku_123"]
}
```

Batch JSON

```
{
        "token": "abcdefghijklmnqrstuvwxyz",
        "ids": [1, 2],
        "external_ids": ["sku_3"]
}
```

Single XML

Identification by parameter: `"ids"`:

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <ids>1</ids>
</xml>
```

Identification by parameter: `"external_ids"`:

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <external_ids>sku_123</external_ids>
</xml>
```

Batch XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <ids>1</ids>
  <ids>2</ids>
  <external_ids>sku_3</external_ids>
</xml>
```

Response example

JSON

```
{
    "deleted_count": 2,
    "error": 0,
    "error_text": ""
}
```

XML

```
<xml>
   <deleted_count>2</deleted_count>
   <error>0</error>
   <error_text></error_text>
</xml>
```

Response parameters

Parameter	Type
deleted_count	**int**
error	**int**
error_text	**string**

Get summary report

Description

Retrieves a summary report.

Request URL

Method: POST

`https://example.com/api/v1.1/reports/summary`

Request parameters

Parameter	Type	Example
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
resource_id	**int**	**1**
date_from	**string**	**“2018-08-10”**
date_to	**string**	**“2018-08-15”**
data_type	**string**	**“segment”**
data_id	*	**1**
data_group	**string**	**“day”**
show_deleted	**bool**	**false**
is_test_data	**bool**	**false**

Report sources:

`data_type`	Description
campaign	**Mailing**
list	**Database**
segment	**Segment**
message	**Message template**
sender	**Sender**
domaingroup	**Email domain group**
domain	**Email domain**
vendor	**Vendor**
day	**Daily report**
week	**Weekly report**
month	**Monthly report**
environment	**Mobile or desktop traffic**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "data_group": "vendor",
    "data_id": 7,
    "data_type": "list",
    "date_from": "2024-04-04",
    "date_to": "2024-05-05",
    "format": "json"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <data_group>vendor</data_group>
  <data_id>7</data_id>
  <data_type>list</data_type>
  <date_from>2024-04-04</date_from>
  <date_to>2024-05-05</date_to>
  <format>xml</format>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "clicks": 22,
            "complains": 2,
            "confirms": 7,
            "delivs": 22,
            "hbounces": 3,
            "id": "Vendor #2",
            "opens": 44,
            "reads": 2,
            "sbounces": 15,
            "sends": 41,
            "suppresses": 1,
            "uclicks": 15,
            "unsubs": 3,
            "uopens": 22
        },
        {
            "clicks": 34,
            "complains": 4,
            "confirms": 13,
            "delivs": 24,
            "hbounces": 7,
            "id": "Vendor #1",
            "opens": 48,
            "reads": 3,
            "sbounces": 21,
            "sends": 31,
            "suppresses": 3,
            "uclicks": 21,
            "unsubs": 6,
            "uopens": 24
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <clicks>22</clicks>
    <complains>2</complains>
    <confirms>7</confirms>
    <confirmsubs>0</confirmsubs>
    <delivs>22</delivs>
    <hbounces>3</hbounces>
    <id>Vendor #2</id>
    <opens>44</opens>
    <reads>2</reads>
    <sbounces>15</sbounces>
    <sends>41</sends>
    <suppresses>1</suppresses>
    <uclicks>15</uclicks>
    <unsubs>3</unsubs>
    <uopens>22</uopens>
  </data>
  <data>
    <clicks>34</clicks>
    <complains>4</complains>
    <confirms>13</confirms>
    <confirmsubs>0</confirmsubs>
    <delivs>24</delivs>
    <hbounces>7</hbounces>
    <id>Vendor #1</id>
    <opens>48</opens>
    <reads>3</reads>
    <sbounces>21</sbounces>
    <sends>31</sends>
    <suppresses>3</suppresses>
    <uclicks>21</uclicks>
    <unsubs>6</unsubs>
    <uopens>24</uopens>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
id	**string**
clicks	**int**
complains	**int**
opens	**int**
reads	**int**
replies	**int**
sends	**int**
confirms	**int**
delivs	**int**
undelivs	**int**
unsubs	**int**
uopens	**int**
ureads	**int**
offence	**int**
glanceds	**int**
hbounces	**int**
sbounces	**int**
suppresses	**int**
uclicks	**int**
ulclicks	**int**
sends_push	**int**
clicks_push	**int**
uclicks_push	**int**
delivs_push	**int**
ureads_push	**int**
opens_push	**int**
uopens_push	**int**
push_ctr	**int**
sbounce_push	**int**
hbounce_push	**int**
sends_sms	**int**
clicks_sms	**int**
uclicks_sms	**int**
delivs_sms	**int**
undelivs_sms	**int**
sms_ctr	**int**
unsubs_sms	**int**
open_rate	**int**
push_deliv_rate	**int**
u_read_rate	**int**
u_open_click_rate	**int**
sb_rate	**int**
email_read_rate	**int**
delivery_rate	**int**
email_unsub_rate	**int**
email_u_read_rate	**int**
open_click_rate	**int**
suppresses_rate	**int**
unsub_rate	**int**
u_click_rate	**int**
click_rate	**int**
sms_deliv_rate	**int**
hb_rate	**int**
push_open_rate	**int**
undelivs_push	**int**
complain_rate	**int**
u_open_rate	**int**
email_reply_rate	**int**
sends_total	**int**
delivs_total	**int**
undelivs_total	**int**
reads_total	**int**

Parameter	
clicks_total	**int**
uclicks_total	**int**
opens_total	**int**
ureads_total	**int**
uopens_total	**int**

Get soft bounces report

Description

Retrieves email soft bounces report.
Email providers return error codes from 400 to 499. This marks a soft bounce: a message cannot be temp

Request URL

Method: POST

`https://example.com/api/v1.1/reports/bounces`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
date_from	**string**	**“2018-08-10”**
date_to	**string**	**“2018-08-15”**
campaign_id	**int**	**1**
group_by	**string**	**“from_domain”**
top	**int**	**2**

`group_by` **parameter possible values:**

- `ip` **- group by sender IP address**
- `from_domain` **– group by from domain**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "campaign_id": 116,
    "date_from": "2024-04-26",
    "date_to": "2024-05-01",
    "format": "json",
    "top": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <campaign_id>116</campaign_id>
    <date_from>2024-04-26</date_from>
    <date_to>2024-05-01</date_to>
    <format>xml</format>
    <top>2</top>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "Your message was not delivered because the other user mailbox was not available",
            "to_domain": "example.com"
        },
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "This response is sent when the message simply failed. Often times this is not caused by you, but rather because of
            "to_domain": "jetwire.name"
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>Your message was not delivered because the other user mailbox was not available</bounce_text>
        <to_domain>example.com</to_domain>
    </data>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>This response is sent when the message simply failed. Often times this is not caused by you, but rather because of a far-e
        <to_domain>jetwire.name</to_domain>
    </data>
</xml>
```


Response parameters

Parameter	Type
error	**int**
error_text	**string**
bounce_text	**string**
count	**int**
data_id	**string**
to_domain	**string**

Get undeliveries report

Description

Retrieves undelivered messages report.

Request URL

Method: POST

`https://example.com/api/v1.1/reports/undelivered`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
date_from	**string**	**“2018-08-10”**
date_to	**string**	**“2018-08-15”**
campaign_id	**int**	**1**
group_by	**string**	**“from_domain”**
top	**int**	**2**

`group_by` **possible values:**

- `ip` **– group by sender IP address**
- `from_domain` **– group by from domain**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "campaign_id": 116,
    "date_from": "2024-04-26",
    "date_to": "2024-05-01",
    "format": "json",
    "top": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <campaign_id>116</campaign_id>
    <date_from>2024-04-26</date_from>
    <date_to>2024-05-01</date_to>
    <format>json</format>
    <top>2</top>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "This response is sent when the message simply failed. Often times this is not caused by you, but rather because of
            "to_domain": "example.com"
        },
        {
            "data_id": "",
            "count": 3,
            "bounce_text": "This response is sent when the message simply failed. Often times this is not caused by you, but rather because of
            "to_domain": "jetwire.name"
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>This response is sent when the message simply failed. Often times this is not caused by you, but rather because of a far-e
        <to_domain>example.com</to_domain>
    </data>
    <data>
        <data_id></data_id>
        <count>3</count>
        <bounce_text>This response is sent when the message simply failed. Often times this is not caused by you, but rather because of a far-e
        <to_domain>jetwire.name</to_domain>
    </data>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
bounce_text	**string**
count	**int**
data_id	**string**
to_domain	**string**

Get senders list

Description

Retrieves a list of senders available for the current account.

Request URL

Method: POST

`https://example.com/api/v1.1/senders/list`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
from_id	**int**	**2**
limit	**int**	**10**
format	**string**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "limit": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "channel": "email",
            "id": 1,
            "name": "AKMTA sender",
            "type": "akmta"
        },
        {
            "channel": "sms",
            "id": 2,
            "name": "Bulk SMS sender",
            "type": "bulksms_com"
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "next_from_id": 3,
    "total_count": 2
}
```

XML

```
<xml>
    <data>
        <channel>email</channel>
        <id>3</id>
        <name>Email sender</name>
        <type>akmta</type>
    </data>
    <data>
        <channel>sms</channel>
        <id>2</id>
        <name>Bulk SMS sender</name>
        <type>bulksms_com</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <next_from_id>5</next_from_id>
    <total_count>5</total_count>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Error code**
error_text	**string**	**Error text**
next_from_id	**int**	**The index must be passed in the next request to receive another piece of data, until the result outp**
id	**int**	**Sender ID**
name	**string**	**Sender name**
type	**string**	**Sender type**
channel	**string**	**Communication channel**

Get virtual senders list

Description

Retrieves a list of virtual senders.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/list`

Request parameters

Parameter	Type	Exar
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**2**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "limit": 2
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <limit>2</limit>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "email_channel": true,
            "id": 1,
            "name": "AKMTA virtual sender",
            "sms_channel": false
        },
        {
            "email_channel": false,
            "id": 2,
            "name": "Bulk SMS virtual sender",
            "sms_channel": true
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 2
}
```

XML

```
<xml>
<data>
    <email_channel>true</email_channel>
    <id>1</id>
    <name>AKMTA virtual sender</name>
    <sms_channel>false</sms_channel>
  </data>
  <data>
    <email_channel>false</email_channel>
    <id>2</id>
    <name>Bulk SMS virtual sender</name>
    <sms_channel>true</sms_channel>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
  <total_count>2</total_count>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
name	**string**
sms_channel	**bool**
email_channel	**bool**

Get virtual sender information

Description

Retrieves a virtual sender detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
format	**string**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>2</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 1,
        "name": "Virtual sender",
        "email_rules": [
            {
                "type": "default",
                "value": "",
                "senders": [
                    {
                        "from_email": "",
                        "replyto_email": "",
                        "from_name": "",
                        "sender_id": 3,
                        "percent": 100
                    }
                ]
            }
        ],
        "sms_rules": [],
        "custom_channels": {}
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
    <id>1</id>
    <name>Virtual sender</name>
    <email_rules>
      <type>default</type>
      <value></value>
      <senders>
        <from_email></from_email>
        <replyto_email></replyto_email>
        <from_name></from_name>
        <sender_id>3</sender_id>
        <percent>100</percent>
      </senders>
    </email_rules>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
id	**int**
name	**string**
email_rules	**JSON array**
sms_rules	**JSON array**

Clone virtual sender

Description

Creates a copy of an existing virtual sender.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/clone`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
name	**string**	**“Clone of ...”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "name": "Copy of virtual sender #99",
    "id": 3
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <name>Copy of virtual sender #99</name>
  <id>3</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 6
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <saved_id>7</saved_id>
    </data>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Create virtual sender

Description

Creates a new virtual sender.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
name	**string**	**“New virtual sender”**
email_rules	**JSON array**	
sms_rules	**JSON array**	

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "name": "New virtual sender",
    "email_rules": [
        {
            "senders": [
                {
                    "from_email": "",
                    "from_name": "",
                    "percent": 100,
                    "replyto_email": "",
                    "sender_id": 3
                }
            ],
            "type": "default",
            "value": ""
        }
    ],
    "sms_rules": []
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <name>New virtual sender</name>
    <email_rules array ='true'>
        <senders array = 'true'>
            <percent>100</percent>
            <sender_id>3</sender_id>
        </senders>
        <type>default</type>
    </email_rules>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 8
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <data>
        <saved_id>9</saved_id>
    </data>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Update virtual sender

Description

Updates a virtual sender, changes settings.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/update`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
name	**string**	**“New virtual sender”**
email_rules	**JSON array**	
sms_rules	**JSON array**	

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "name": "Virtual sender",
    "email_rules": [
        {
            "senders": [
                {
                    "from_email": "",
                    "from_name": "",
                    "percent": 100,
                    "replyto_email": "",
                    "sender_id": 3
                }
            ],
            "type": "default",
            "value": ""
        }
    ],
    "id": 9,
    "sms_rules": []
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <name>Virtual sender</name>
  <email_rules array='true'>
    <senders array='true'>
      <from_email></from_email>
      <from_name></from_name>
      <percent>100</percent>
      <replyto_email></replyto_email>
      <sender_id>3</sender_id>
    </senders>
    <type>default</type>
    <value></value>
  </email_rules>
  <id>9</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 9
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Delete virtual sender

Description

Deletes a virtual sender entry.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 12
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>11</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Add segmentation query

Description

Add segmentation query to external datatables.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/add/`

Request parameters

Parameter	Type	Example
token	string	"abcdefghijklmnqrstuvwxyz"
name	string	"Main query"
shortname	string	"query1"
description	string	"query to the DB"
connector_id	int	2
column	string	"author"
sql	string	"SELECT * FROM table"
cache_time_min	int	35
groups_id	\[\]int	\[0, 1 \]
parameters	array	[{ "name": "CITY", "type": "string", "title": "head", "options_type": "query" }]

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1,
   "column": "price"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
  <column>price</column>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Update segmentation query

Description

Update segmentation query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/update`

Request parameters

Parameter	Type	Example
token	string	"abcdefghijklmnqrstuvwxyz"
id	int	1
name	string	"Main query"
shortname	string	"query1"
description	string	"query to the DB"
connector_id	int	2
column	string	"author"
sql	string	"SELECT * FROM table"
cache_time_min	int	35
groups_id	\[\]int	\[0, 1 \]
parameters	object	{ "name": "CITY", "type": "string", "title": "head", "options_type": "query" }

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "id": 1,
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1,
   "column": "price"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
  <column>price</column>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Get segmentation query information

Description

Retrieves detailed segmentation query information.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/get`

Request parameters

Parameter	Type	Examp
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
shortname	**string**	**“query1”**

Request example

JSON

```
{
  "id": 2,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <id>2</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1,
        "parametrs": {
           "name": "variable1",
           "title": "var",
           "default_value": "low"
        },
        "cache_time_min": 35
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <cache_time_min>35</cache_time_min>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <parametrs>
         <default_value>low</default_value>
         <name>variable1</name>
         <title>var</title>
      </parametrs>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response example

Parameter	Type	
error	**int**	**Erro**
error_text	**string**	**Erro**
id	**int**	**Que**
name	**string**	**Que**
shortname	**string**	**Que**
description	**string**	**Que**
connector_id	**int**	**Exte**
column	**string**	**Colu**
sql	**string**	**SQL**
cache_time_min	**int**	**35**
groups_id	**[]int**	**List def**
parameters	**object**	**Para**

Get segmentation queries list

Description

Retrieves segmentation queries list.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/list`

Request parameters

Parameter	Type	
token	**string**	`"abcdefghijklmnqrstuvwxyz"`
format	**string**	`"json"`
from_id	**int**	`1`
groups	**[]String**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1,
        "parametrs": {
           "name": "variable1",
           "title": "var",
           "default_value": "low"
        },
        "cache_time_min": 35
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <cache_time_min>35</cache_time_min>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <parametrs>
         <default_value>low</default_value>
         <name>variable1</name>
         <title>var</title>
      </parametrs>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type	
error	**int**	**Erro**
error_text	**string**	**Erro**
id	**int**	**Que**
name	**string**	**Que**
shortname	**string**	**Que**
description	**string**	**Que**
connector_id	**int**	**Exte**
column	**string**	**Colu**
sql	**string**	**SQL**
cache_time_min	**int**	**35**
groups_id	**[]int**	**List defa**
parameters	**object**	**Para**

Delete segmentation query

Description

Delete segmentation query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_segments/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**

Request example

JSON

```
{
  "id": 2,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <id>2</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation",
}
```

XML

```
<xml>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter
error
error_text

Add template query

Description

Add template query to external datatables.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/add/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
name	**string**	**“Main query”**
shortname	**string**	**“query1”**
description	**string**	**“query to the DB”**
connector_id	**int**	**2**
sql	**string**	**“SELECT * FROM table”**
groups_id	**[]int**	**[0, 1]**

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	
error	**int**

Parameter	
error_text	**string**
saved_id	**int**

Update template query

Description

Update template query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/update`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
name	**string**	**“Main query”**
shortname	**string**	**“query1”**
description	**string**	**“query to the DB”**
connector_id	**int**	**2**
sql	**string**	**“SELECT * FROM table”**
groups_id	**[]int**	**[0, 1]**

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "id": 1,
   "name": "Main Query",
   "shortname": "query1",
   "sql": "SELECT * FROM table",
   "connector_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <name>Main Query</name>
  <shortname>query1uery</shortname>
  <sql>SELECT * FROM table</sql>
  <connector_id>1</connector_id>
</xml>
```

Response example

JSON

```
{
    "data": {
          "saved_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
  <data>
    <saved_id>1</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Get template query information

Description

Retrieves detailed template query information.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/get`

Request parameters

Parameter	Type	Examp
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**
shortname	**string**	**“query1”**

Request example

JSON

```
{
  "id": 2,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <id>2</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response example

Parameter	Type
error	**int**
error_text	**string**

Parameter	Type
id	**int**
name	**string**
shortname	**string**
description	**string**
connector_id	**int**
sql	**string**
groups_id	**[]int**

Get template queries list

Description

Retrieves template queries list.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/list`

Request parameters

Parameter	Type	
token	**string**	`"abcdefghijklmnqrstuvwxyz"`
format	**string**	`"json"`
from_id	**int**	`1`
groups	**[]String**	`[ "group1", "group2" ]`
groups_id	**[]int**	`[ 0, 1 ]`

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 2,
        "name": "QueryforextDB",
        "shortname": "query",
        "description": "External DB segment",
        "sql": "SELECT * FROM table",
        "connector_id": 1
     },
     "error": 0,
     "error_text": "Successful operation",
}
```

XML

```
<xml>
   <data>
      <connector_id>1</connector_id>
      <description>External DB segment</description>
      <id>2</id>
      <name>QueryforextDB</name>
      <shortname>query</shortname>
      <sql>SELECT * FROM table</sql>
   </data>
   <error>0</error>
   <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
error	**int**

Parameter	Type
error_text	**string**
id	**int**
name	**string**
shortname	**string**
description	**string**
connector_id	**int**
sql	**string**
groups_id	**[]int**

Delete template query

Description

Delete template query.

Request URL

Method: POST

`https://example.com/api/v1.1/datatables/query_templates/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**1**

Request example

JSON

```
{
  "id": 2,
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <id>2</id>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
  "data": null,
  "error": 0,
  "error_text": "Successful operation",
}
```

XML

```
<xml>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>t>
</xml>
```

Response parameters

Parameter
error
error_text

AKMTA object

JSON

JSON

```
{
    "dkim_id": 4,
    "dkim_selector": "dkim_selector",
    "ips": [
        {
            "ip": "24.24.24.34",
            "node": 10
        }
    ],
    "lock_rules_obj": {
        "common_rules": [],
        "isp_rules": [
            {
                "isps": [
                    6
                ],
                "rules": [
                    {
                        "patterns": [
                            1004,
                            1005,
                            497
                        ],
                        "rule": 1001
                    }
                ]
            }
        ]
    },
    "redirect_email": "user@example.com",
    "retry_rules_obj": {
        "common_rules": [
            {
                "patterns": [
                    1003
                ],
                "rule": 1002
            }
        ],
        "isp_rules": [
            {
                "isps": [
                    1007,
                    1008,
                    1009,
                    1010,
                    1011,
                    1012
                ],
                "rules": [
                    {
                        "patterns": [
                            1003
                        ],
                        "rule": 1002
                    }
                ]
            }
        ]
    },
    "sender_dkim_id": 8,
    "sender_dkim_selector": "sender_dkim_selector",
    "sending_rules_obj": {
        "common_rules": [
            {
                "isps": [
                    0
                ],
                "strategy": 1001
            }
        ],
        "ip_rules": [
            {
                "ips": [
                    "24.24.24.34"
                ],
                "rules": [
                    {
                        "isps": [
                            1007,
```

```
                        1009
                    ],
                    "strategy": 1002
                },
                {
                    "isps": [
                        1008,
                        1010
                    ],
                    "strategy": 1003
                }
            ]
        }
    ]
},
"users": [
    {
        "domain": "mx.localtest.ak",
        "pop3_pass": "nnr614g5",
        "redirect_email": "user1@example.com"
    },
    {
        "domain": "mx.localtest.ak",
        "pop3_pass": "nnr614g5",
        "redirect_email": "user2@example.com"
    }
]
}
```

XML

```
<xml>
    <dkim_id>4</dkim_id>
    <dkim_selector>dkim_selector</dkim_selector>
    <ips array='true'>
        <ip>24.24.24.34</ip>
        <node>10</node>
    </ips>
    <lock_rules_obj>
        <common_rules array='true'></common_rules>
        <isp_rules array='true'>
            <isps array='true'>6</isps>
            <rules array='true'>
                <patterns>1004</patterns>
                <patterns>1005</patterns>
                <patterns>497</patterns>
                <rule>1001</rule>
            </rules>
        </isp_rules>
    </lock_rules_obj>
    <redirect_email>user@example.com</redirect_email>
    <retry_rules_obj>
        <common_rules array='true'>
            <patterns array='true'>1003</patterns>
            <rule array='true'>1002</rule>
        </common_rules>
        <isp_rules array='true'>
            <isps>1007</isps>
            <isps>1008</isps>
            <isps>1009</isps>
            <isps>1010</isps>
            <isps>1011</isps>
            <isps>1012</isps>
            <rules array='true'>
                <patterns array='true'>1003</patterns>
                <rule>1002</rule>
            </rules>
        </isp_rules>
    </retry_rules_obj>
    <sender_dkim_id>8</sender_dkim_id>
    <sender_dkim_selector>sender_dkim_selector</sender_dkim_selector>
    <sending_rules_obj>
        <common_rules array='true'>
            <isps>0</isps>
            <strategy>1001</strategy>
        </common_rules>
        <ip_rules array='true'>
            <ips array='true'>24.24.24.34</ips>
            <rules array='true'>
                <isps>1007</isps>
                <isps>1009</isps>
                <strategy>1002</strategy>
            </rules>
            <rules array='true'>
                <isps>1008</isps>
                <isps>1010</isps>
                <strategy>1003</strategy>
            </rules>
        </ip_rules>
    </sending_rules_obj>
    <users>
        <domain>mx.localtest.ak</domain>
        <pop3_pass>nnr614g5</pop3_pass>
        <redirect_email>user1@example.com</redirect_email>
    </users>
    <users>
        <domain>mx.localtest.ak</domain>
        <pop3_pass>nnr614g5</pop3_pass>
        <redirect_email>user2@example.com</redirect_email>
    </users>
</xml>
```

Description

Field
ips
dkim_id

Field
dkim_selector
sender_dkim_id
sender_dkim_selector
redirect_email
sending_rules_obj
retry_rules_obj
lock_rules_obj
users

Content object

Content objects are used in mailing and scenarios queries.

JSON

JSON

```
{
  "subject": "Maybe very important message",
  "from_email": "",
  "from_name": "James",
  "replyto_email": "",
  "replyto_name": "",
  "message_id": 45
}
```

XML

```
<xml>
  <subject>Maybe very important message</subject>
  <from_email></from_email>
  <from_name>James</from_name>
  <replyto_email></replyto_email>
  <replyto_name></replyto_name>
  <message_id>45</message_id>
</xml>
```

Object parameters

Parameters	Type	
message_id	**int**	**12**
from_email	**string**	**“test@example.**
from_name	**string**	**“Jenny’s List”**
replyto_email	**string**	**“test@example.**
replyto_name	**string**	**“Jenny”**
subject	**string**	**“Weather alert”**

Email rule object

JSON

JSON

```
{
  "type": "from_domain",
  "value": "altcraft.com",
  "senders": [
    {
      "sender_id": 287,
      "percent": 100,
      "from_email": "no-reply@altcraft.com",
      "replyto_email": "no-reply@altcraft.com",
      "from_name": ""
    }
  ]
}
```

XML

```
<xml>
  <type>from_domain</type>
  <value>altcraft.com</value>
  <senders array='true'>
    <sender_id>287</sender_id>
    <percent>100</percent>
    <from_email>no-reply@altcraft.com</from_email>
    <replyto_email>no-reply@altcraft.com</replyto_email>
    <from_name></from_name>
  </senders>
</xml>
```

Object parameters

Parameter	Type
type	**string**
value	**string/int**
senders	**JSON array**

Rule types

Type	Value type
default	**string**
group_id	**int**
from_domain	**string**
to_domain	**string**

File object

JSON

JSON

```
{
    "data": "data:text/csv; base64,SEVMTE8K",
    "name": "US.txt"
}
```

XML

```
<xml>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <name>US.txt</name>
</xml>
```

Object parameters

Parameter	Type
name	**string**
data	**string**
file_key	**string**
mimetype	**string**

Profile data object

Profile data object is used in most profile import requests. It contains profile database field values and su
You can retrieve database fields information by making request to `api/v1.1/databases/fields_get/`.
You can also see information about databases and profile fields in User documentation.

JSON

JSON


```
{
    "data": {
        "_regcountry": "US",
        "_regcity": "New York",
        "_regip": "94.231.119.122",
        "_regurl": "www.example.com",
        "_regdate": "2019-03-14T22:00:00Z",
        "_bdate": "1990-02-22T21:00:00Z",
        "_browser": "Thunderbird",
        "_city": "Ryazan",
        "_country": "RU",
        "_fname": "Olly",
        "_ip": "94.231.119.122",
        "_lname": "Lambert",
        "_postal_code": "390000",
        "_region": "Ryazan Oblast",
        "_vendor": "form_#31",
        "_sex": 0,
        "_status": 0,
        "_tz": "Europe/Moscow",
        "email": "example@altcraft.com",
        "phones": [
            "+79000000000"
        ],
        "custom_field": "custom_value",
        "subscriptions": [
            {
                "channel": "email",
                "email": "boris.yudaev@altcraft.com",
                "priority": 0,
                "resource_id": 1,
                "status": 0
            }
        ]
    }
}
```


XML

```
<xml>
    <data>
        <regcountry>US</regcountry>
        <regcity>New York</regcity>
        <regip>94.231.119.122</regip>
        <regurl>www.example.com</regurl>
        <regdate>2019-03-14T22:00:00Z</regdate>
        <bdate>1990-02-22T21:00:00Z</bdate>
        <browser>Thunderbird</browser>
        <city>Ryazan</city>
        <country>RU</country>
        <fname>Olly</fname>
        <ip>94.231.119.122</ip>
        <lname>Lambert</lname>
        <postal_code>390000</postal_code>
        <region>Ryazan Oblast</region>
        <vendor>form_#31</vendor>
        <sex>0</sex>
        <status>0</status>
        <tz>Europe/Moscow</tz>
        <email>example@altcraft.com</email>
        <phones array='true'>+79000000000</phones>
        <custom_field>custom_value</custom_field>
        <subscriptions array='true'>
            <channel>email</channel>
            <email>boris.yudaev@altcraft.com</email>
            <priority>0</priority>
            <resource_id>1</resource_id>
            <status>0</status>
        </subscriptions>
    </data>
</xml>
```

Description

Parameter	Type	Example
_bdate	**string**	**1990-02-22T21:00:00Z**
_browser	**string**	**Firefox**
_city	**string**	**Ryazan**
_country	**string**	**RU**
_fname	**string**	**Olly**
_ip	**string**	**94.231.119.122**
_lname	**string**	**Lambert**
_postal_code	**int**	**390000**
_region	**string**	**Ryazan Oblast**
_status	**int**	**0**
_tz	**string**	**Europe/Moscow**
email	**string**	**example@altcraft.com**
phones	**JSON array**	**[“+790000000000”]**
subscriptions	**JSON array**	`[“”,””]`
_regcountry	**string**	**US**
_regcity	**string**	**New York**
_regip	**string**	**94.231.119.122**
_regurl	**string**	**www.example.org**
_regdate	**string**	**2018-02-22T21:00:00Z**
_sex	**int**	**0**
_vendor	**string**	**form#33**
custom_field	*	**custom_value**

INFO

*** If you send `"_ip"` parameter containing IPv4 value, Altcraft will automatically detect subscriber’s locat**

JSON

```
{
          "_ip": "94.231.119.122", // (records IP as well)
          "_lat": 54.6197, // (lattitude)
          "_lon": 39.74, // (longitude)
          "_postal_code": "390000",
          "_region": "Ryazanskaya Oblast'",
          "_country": "RU",
          "_city": "Ryazan",
          "_tz": "Europe/Moscow"
}
```

XML

```
<xml>
    <ip>94.231.119.122</ip>// уже добавляется само по себе
    <lat>54.6197</lat>// географическая широта
    <lon>39.74</lon>// географическая долгота
    <postal_code>390000</postal_code>
    <region>Ryazanskaya Oblast</region>
    <country>RU</country>
    <city>Ryazan</city>
    <tz>Europe/Moscow</tz>
</xml>
```

Sender object

JSON

JSON

```
{
  "sender_id": 287,
  "percent": 100,
  "from_email": "no-reply@altcraft.com",
  "replyto_email": "no-reply@altcraft.com",
  "from_name": "James",
}
```

XML

```
<xml>
  <sender_id>287</sender_id>
  <percent>100</percent>
  <from_email>no-reply@altcraft.com</from_email>
  <replyto_email>no-reply@altcraft.com</replyto_email>
  <from_name>James</from_name>
</xml>
```

Object parameters

Parameter	Type	
sender_id	**int**	`123`
percent	**int**	`100`
from_email	**string**	`"no-reply@altcraft.com"`
replyto_email	**string**	`"no-reply@altcraft.com"`
from_name	**string**	`"Altcraft"`

Sender types

Senders are created in Altcraft administrative panel to send emails and SMS.

Type	Cha
“akmta”	email
“dmta”	email
“eshark”	email
“webzo”	email
“mandrill”	email
“emaildirect”	email
“amazon_ses”	email
“smsc_ru”	sms
“bulksms_com”	sms

SMS rule object

JSON

JSON

```
{
    "senders": [
        {
            "from_email": "",
            "from_name": "Example",
            "percent": 100,
            "replyto_email": "",
            "sender_id": 798
        }
    ],
    "type": "default",
    "value": ""
}
```

XML

```
<xml>
  <senders array='true'>
    <from_email></from_email>
    <from_name>Example</from_name>
    <percent>100</percent>
    <replyto_email></replyto_email>
    <sender_id>798</sender_id>
  </senders>
  <type>default</type>
  <value></value>
</xml>
```

Object parameters

Parameter	Type
type	**string**
value	**string/int**
senders	**JSON array**

Rule types

Rule type	Value type
default	**string**
group_id	**int**
from_domain	**string**

Start schedule object

JSON

JSON

```
{
    "start_hours": [
        2,
        4
    ],
    "start_mdays": [
        1
    ],
    "start_mins": [
        0,
        10,
        20,
        30,
        40,
        50
    ],
    "start_months": [
        12
    ],
    "start_wdays": [],
    "start_years": [
        2016
    ]
}
```

XML

```
<xml>
<xml>
    <start_hours>2</start_hours>
    <start_hours>4</start_hours>
    <start_mdays array='true'>1</start_mdays>
    <start_mins>0</start_mins>
    <start_mins>10</start_mins>
    <start_mins>20</start_mins>
    <start_mins>30</start_mins>
    <start_mins>40</start_mins>
    <start_mins>50</start_mins>
    <start_months array='true'>12</start_months>
    <start_wdays></start_wdays>
    <start_years array='true'>2024</start_years>
</xml>
```

Object parameters

Parameter	Type	
start_mins	**JSON array**	**[1, 31]**
start_hours	**JSON array**	**[22]**
start_mdays	**JSON array**	**[]**
start_wdays	**JSON array**	**[]**
start_months	**JSON array**	**[]**
start_years	**JSON array**	**[]**
tz	**string**	**Europe/Mos**
profile	**bool**	**true**
losers	**bool**	**true**

Subscription object

Description

The `subscription` object contains data about a profile's subscription to communication channels. The struc

Basic Structure

Parameters	Type	Example	
"channel"	**string**	**"email"**	**Channel type ("email" / "sms" / "push" / "notify" / "viber" / "w**
"db_id"	**int**	**1**	**Database identifier**
"email"	**string**	**"john@example.com"**	**Email address**
"phone"	**string**	**"+79000000000"**	**Phone number **(For SMS subscriptions)****
"provider"	**string**	**"Firefox"**	**Provider type **(For Push subscriptions)****
"subscription_id"	**string**	**"abcdefghijklmnqrstuvwxyz"**	**Subscription identifier **(For Push subscriptions)****
"cc_data"	**object**	\{ "id": 1 \}	**Chat ID **in Telegram bot****
		\{ "phone": "+79000000000" \}	**Profile phone in **WhatsApp*/ Viber™ / Notify**
"priority"	**int**	**1**	**Subscription priority (optional field). If the subscription priority is passed in the re**
"profile_id"	**string**	**"54759eb3c090d83494e2d804"**	**Profile identifier**
"resource_id"	**int**	**1**	**Resource identifier**
"custom_fields"	**object**	**"subscribed"**	**Standard and additional subscription fields (o You can get information about available subs**
"status"	**string**	**"subscribed"**	**Subscription status [Learn more](/en/developer-guide/objects/Subs**
"cats"	**array**	["category_1", "category_2"]	**Resource categories to which the profile is su**

Extended structure when querying the profile through the handler `/api/v1.1/profiles/get/`

When querying the profile through the `/api/v1.1/profiles/get/` handler, the `subscription` object contains addi

- **Channel data is stored in separate objects:**
  - **`"email_data"` - for Email subscriptions**
  - **`"phone_data"` - for SMS/Viber/WhatsApp**
  - **`"cc_data"` - for Telegram**
- **Technical fields**
  - **`"hash_id"` - unique subscription hash**
  - **`"is_delete"` - delete flag (soft delete)**
- **History:**
  - **`"reg_info"` - subscription registration date**

Example structure

JSON

```
{
  "resource_id": 4,
  "channel": "email",
  "email_data": {
        "email": "user@example.com",
        "domain": "example.com",
        "md5": "d922b1da12193df553aad2bee61da46b"
  },
  "status": "subscribed",
  "reg_info": {
    "date": "2025-07-23T11:27:05.487Z"
  },
  "hash_id": "7641c5c5",
  "is_delete": false
}
```

XML

```
<xml>
<subscription>
    <resource_id>4</resource_id>
    <channel>email</channel>
    <email_data>
        <email>user@example.com</email>
        <domain>example.com</domain>
        <md5>d922b1da12193df553aad2bee61da46b</md5>
    </email_data>
    <status>subscribed</status>
    <reg_info>
        <date>2025-07-23T11:27:05.487Z</date>
    </reg_info>
    <hash_id>7641c5c5</hash_id>
    <is_delete>false</is_delete>
</subscription>
</xml>
```

Object example

JSON

```
{
  "subscriptions": [
    {
      "resource_id": 1,
      "channel": "email",
      "email": "example@example.com"
    },
    {
      "resource_id": 2,
      "channel": "sms",
      "phone": "+79000000000"
    },
    {
      "resource_id": 1,
      "channel": "push",
      "subscription_id": "a81c264a938b475",
      "provider": "android-firebase",
      "custom_fields": {
        "_browser_name": "Firefox",
        "_device_type": "web",
        "custom_field_1": "test value"
      },
      "cats": [
        "category_1",
        "category_2"
      ]
    },
    {
      "resource_id": 1,
      "channel": "whatsapp",
      "cc_data": {
        "phone": "+79000000000"
      }
    },
    {
      "resource_id": 1,
      "channel": "viber",
      "cc_data": {
        "phone": "+79000000000"
      }
    },
    {
      "resource_id": 1,
      "channel": "telegram_bot",
      "cc_data": {
        "id": 12345
      }
    },
    {
      "resource_id": 1,
      "channel": "notify",
      "cc_data": {
        "phone": "+79000000000"
      }
    }
  ]
}
```

XML

```
<xml>
<subscriptions>
  <subscription>
    <resource_id>1</resource_id>
    <channel>email</channel>
    <email>example@example.com</email>
  </subscription>

  <subscription>
    <resource_id>2</resource_id>
    <channel>sms</channel>
    <phone>+79000000000</phone>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>push</channel>
    <subscription_id>a81c264a938b475</subscription_id>
    <provider>android-firebase</provider>
    <custom_fields>
      <_browser_name>Firefox</_browser_name>
      <_device_type>web</_device_type>
      <custom_field_1>test value</custom_field_1>
    </custom_fields>
    <cats>
      <category>category_1</category>
      <category>category_2</category>
    </cats>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>whatsapp</channel>
    <cc_data>
      <phone>+79000000000</phone>
    </cc_data>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>viber</channel>
    <cc_data>
      <phone>+79000000000</phone>
    </cc_data>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>telegram_bot</channel>
    <cc_data>
      <id>12345</id>
    </cc_data>
  </subscription>

  <subscription>
    <resource_id>1</resource_id>
    <channel>notify</channel>
    <cc_data>
      <phone>+79000000000</phone>
    </cc_data>
  </subscription>
</subscriptions>
```

Subscription status

Description

The status parameter in the subscription object overwrites the subscription status. It is recommended to Both string and numeric status representations can be used.

Word value	
subscribed	**Ready for messaging**
unsubscribed	**Unsubscribed**
complained	**Reported spam**
hardbounced	**Hard bounce, email probably invalid**
unconfirmed	**Subscription unconfirmed**

Word value	
suspended	**Temporarily unsubscribed**
invalid	**Invalid for a variety of reasons**

Meta organization, which owns Instagram, Facebook, and WhatsApp products, is recognized as extremis

***** Viber™ is a trademark of Rakuten Group, Inc. Blocked in the Russian Federation.***

Trigger types

For more details see the corresponding [User documentation] article.

Type	
api_call	**API call only**
api_import	**API profile import**
action	**Subscribers message actions**
update	**Profile fields update**
pixel	**Pixel activation**

Upload file

Description

Uploads a file.

Request URL

Method: POST

`https://example.com/api/v1.1/file_upload`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
header	**string**	**“emails.csv”**
data	**string**	**“data:text/csv; base64,YQpi...”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "data": "data:text/csv; base64,SEVMTE8K",
    "header": "US.csv"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <data>data:text/csv; base64,SEVMTE8K</data>
    <header>US.csv</header>
</xml>
```

Response example

JSON

```
{
    "data": {
        "file": "fd7006e5-5c42-4eba-aa6a-f6cb7f5cb6f2"
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <file>fd7006e5-5c42-4eba-aa6a-f6cb7f5cb6f2</file>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**str**
file	**str**

Get message web version

Description

Retrieves an email message web version.

Request URL

Method: POST

`https://example.com/api/v1.1/webver/get`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvw**
format	**string**	**“json”**
send_message_id	**string**	**“abcdefghijklmnqrstuvw**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "send_message_id": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <send_message_id>abcdefghijklmnqrstuvwxyz</send_message_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "createtime": "2018-08-24T09:26:23.292Z",
        "email": {
            "from_email": "example@example.com",
            "from_name": "example.com",
            "html": "HTML content ...",
            "replyto_email": "example@example.com",
            "replyto_name": "example.com",
            "subject": "Welcome to Altcraft",
            "text": "Text version also important\n\n--\nUnsubscribe: http://trk.gmail.com/37cfe6oTGQF4LYqk6oMi8eCtZ?dummy=1"
        }
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <createtime>2018-08-24T09:26:23.292Z</createtime>
    <email>
      <from_email>example@example.com</from_email>
      <from_name>example.com</from_name>
      <html>HTML content ...</html>
      <replyto_email>example@example.com</replyto_email>
      <replyto_name>example.com</replyto_name>
      <subject>Welcome to Altcraft</subject>
      <text>Text version also important --
Unsubscribe: http://trk.gmail.com/37cfe6oTGQF4LYqk6oMi8eCtZ?dummy=1</text>
    </email>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	
error	**int**
error_text	**string**
createtime	**date**
email	**JSON object**

Push providers

Push provider	Web / App
Google Chrome (deprecated)	**Web**
Mozilla Firefox	**Web**
Safari	**Web**
Android Firebase	**App**
IOS Firebase	**App**
Firefox Firebase	**Web**
Chrome Firebase	**Web**
Opera Firebase	**Web**
Yandex Firebase	**Web**
Samsung Firebase	**Web**
IOS APNs	**App**
AppMetrica with iOS device id	**App**
AppMetrica with Android device id	**App**
AppMetrica with iOS IFA	**App**
AppMetrica with Google AID	**App**
AppMetrica with Android push token	**App**
AppMetrica with iOS push token	**App**
MailRu (устаревший)	**App**
Android Huawei Mobile Services	**App**
IOS Huawei Mobile Services	**Web**
Android RuStore	**App**

Deduplication of requests

Description

Repeat requests are API requests that contain the same headers and the same data as the original reques
The platform will not accept a repeated request if it changes the data in order to avoid duplicate events. T
The platform performs deduplication according to the following algorithm:

- **Checking for the presence of Request-ID header**
- **If Request-ID is not specified, the checksum of request body and headers is checked**

It should be noted that not all requests support deduplication with checksum, and if no Request-ID is spec

> **INFO**
>
> **Different request sending clients may add their own system headers or other information to the request.**

How do you know if the request was taken as a duplicate one?

The response to the repeated request will be the same as the response to the original one. However, the r

```
Duplicate-Key: NeOScRqBgsqj67CR5dYcqvcmPZA=
Duplicate-Request: true
```

The Duplicate-Key header consists of the Request-ID of the request and the sender token. If Request-ID is

How to send a request so that it is not taken as a duplicate one?

It is recommended to mark each request with a unique identifier. In this way when you change the identifi
There are 2 ways to do it:

- **Change the data in the request body or URL, for example, &request_id=12345**
- **Add the Request-ID header, for example Request-ID: a0fe37f2-b4e0-4e28-a328-4ec7461604fc**

API requests that have deduplication

All requests that modify data or activation / launch of entities are checked for repetitions:

Category	
Profiles (deprecated methods)	**/v1.0/lists/:list_id/le /v1.0/lists/:list_id/le /v1.0/lists/:list_id/le /v1.0/lists/:list_id/le /v1.0/lists/:list_id/le /v1.0/lists/:list_id/le /v1.0/lists/:list_id/su /v1.0/lists/:list_id/un /v1.0/lists/:list_id/in**
Integrations	**/integrations/lpgen /integrations/tilda/ /integrations/any/p**
Suppression lists	**/v1.1/suppresses/a /v1.1/suppresses/u /v1.1/suppresses/d /v1.1/suppresses/d /v1.1/suppresses/d /v1.1/suppresses/d /v1.1/suppresses/e /v1.1/suppresses/e /v1.1/suppresses/e /v1.1/suppresses/e /v1.1/suppresses/pl /v1.1/suppresses/pl /v1.1/suppresses/pl /v1.1/suppresses/pl**
Subscriptions	**/v1.1/subscriptions /v1.1/subscriptions /v1.1/subscriptions /v1.1/subscribers/ir /v1.1/subscribers/u**
Segments	**/v1.1/segments/add /v1.1/segments/upd /v1.1/segments/ref**
Mailings	**/v1.1/campaigns/de /v1.1/campaigns/ac /v1.1/campaigns/cl**
Trigger mailings	**/v1.1/campaigns/tri /v1.1/campaigns/tri /v1.1/campaigns/tri /v1.1/campaigns/tri /v1.1/campaigns/tri /v1.1/triggers/start /v1.1/triggers/start /v1.1/triggers/impo /v1.1/triggers/impo**
Broadcasts	**/v1.1/campaigns/br /v1.1/campaigns/br /v1.1/campaigns/br**

Category	
Regular mailings	**/v1.1/campaigns/re** **/v1.1/campaigns/re** **/v1.1/campaigns/re**
Senders	**/v1.1/admin/sender** **/v1.1/admin/sender** **/v1.1/admin/sender** **/v1.1/admin/sender** **/v1.1/admin/sender** **/v1.1/admin/sender** **/v1.1/admin/vsende** **/v1.1/admin/vsende** **/v1.1/admin/vsende**
Nodes	**/v1.1/admin/nodes/** **/v1.1/admin/nodes/**
Static segments	**/v1.1/profiles/segst** **/v1.1/profiles/segst**
Profiles	**/v1.1/profiles/add** **/v1.1/profiles/updat** **/v1.1/profiles/impor** **/v1.1/profiles/add_b** **/v1.1/profiles/updat** **/v1.1/profiles/impor**
Promocodes	**/v1.1/promocodes/a** **/v1.1/promocodes/a** **/v1.1/promocodes/i** **/v1.1/promocodes/c** **/v1.1/promocodes/u**
Relations	**/v1.1/relations/atta** **/v1.1/relations/deta** **/v1.1/relations/edit** **/v1.1/relations/stre**
Profiles databases	**/v1.1/databases/wi** **/v1.1/databases/ref**
Virtual senders	**/v1.1/vsenders/add** **/v1.1/vsenders/upd** **/v1.1/vsenders/clon**
Scenarios	**/v1.1/workflows/sta** **/v1.1/workflows/im** **/v1.1/workflows/im** **/v1.1/workflows/im**
Push event registration	**/v1.1/pushes/post_** **/v1.1/pushes/crypto**
Message templates	**/v1.1/templates/ad** **/v1.1/templates/up** **/v1.1/templates/del**
Resources	**/v1.1/resources/pus** **/v1.1/resources/ref**
Data tables	**/v1.1/datatables/qu** **/v1.1/datatables/qu** **/v1.1/datatables/qu** **/v1.1/datatables/qu** **/v1.1/datatables/qu** **/v1.1/datatables/qu**

How to send API request with RabbitMQ

Description

In this article, you will learn how to send API requests using RabbitMQ.

> **INFO**
>
> **A unique parameter `request_id` is sent to identify requests.**

Queues

When working with API in RabbitMQ, two queues are used:

1. **api_req — queue for incoming requests;**
2. **api_resp — the queue that receives responses to requests (only if request_id is specified).**

Structure of the sent JSON message

Parameter	Type	Example
request_id	**string**	**“abcd1234”**
request	**string**	**api/v1.1/campaigns/triggers/import_and_start_batch**

body	**JSON object**	“body”:{
		“data”:
		{
		“_fname”: “Fname1”,
		“_lname”: “Lname1”,
		“email”: “profile1@example.com”
		},
		“email”: “profile1@example.com”,
		“db_id”: 1,
		“detect_geo”: true,
		“token”: “abcdefghijklmnopqrstuvwxyzABCD”
		}

Structure of the received JSON message

Parameter	Type	Example

body	**JSON объект**	“body”:
		{
		“error”:0,
		“error_text”:“Successful
		“profile_id”:“60f039e830b
		}

request_id	**string**	**“abcd1234”**

How to send API requests in Go

You can send an API request using a script that calls RabbitMQ:

- **Write and execute a script**

Example

```
package main

        "encoding/json"
        "log"

        "github.com/streadway/amqp"
)

const accID = 1
const resourceID = 3
const dbID = 23
const msgID = 17
const segmentID = 85
const amountOfPushMsgs = 1
const emailDomain = "example.com"

const req = `{
        "account_id": 1,
        "request_id": "db1894e4-1a2c-4021-8233-9cca0b96b79e",
        "request": "api/v1.1/campaigns/triggers/import_and_start_batch",
    "body": {
                "token": "abcdefghijklmnopqrstuvwxyzABC"
                "format": "json",
                "trigger_id": 240,
                "skip_triggers": false,
                "detect_geo": true,
                "matching": "custom",
                "field_name":"CustomF",

                "custom_data":{
                        "some":"some0"
                },
                "content":{
                        "someCont":"someCont0"
                },
                "data": [
                        {
                                "data":{
                                        "_fname":"NUMBER13",
                                        "_lname":"Lambert",
                                        "phones":"790000000013",
                                        "email": "profile1@example.com",
                                        "CustomF":18
                                },
                                "custom_data":{
                                        "some":"some1"
                                },
                                "content":{
                                        "someCont":"someCont1"
                                }
                        },
                        {
                                "data":{
                                        "_fname":"NUMBER14",
                                        "_lname":"Hard",
                                        "phones":"790000000014",
                                        "email": "profile2@example.com",
                                        "CustomF":14
                                }
                        }
                ]
        }
}`

func failOnError(err error, msg string) {
        if err != nil {
                log.Fatalf("%s: %s", msg, err)
        }
}

func main() {
        var err error

        var conn *amqp.Connection
        conn, err = amqp.Dial("amqp://example:abcdefghijklmnopqrstuvwxyz127.0.0.1:5672/")
        failOnError(err, "Failed to connect to RabbitMQ")
        defer conn.Close()

        var ch *amqp.Channel
        ch, err = conn.Channel()
```

```
        failOnError(err, "Failed to open RabbitMQ channel")
        defer ch.Close()

        qUeueImport, err := ch.QueueDeclare(
                "api_req", // name
                true,      // durable
                false,     // delete when unused
                false,     // exclusive
                false,     // no-wait
                nil,       // arguments
        )
        failOnError(err, "Failed to declare RabbitMQ queue")

        for i := 0; i < amountOfPushMsgs; i++ {
                var bodyMap = make(map[string]interface{})
                err = json.Unmarshal([]byte(req), &bodyMap)
                failOnError(err, "Failed json.Unmarshal to bodyMap")

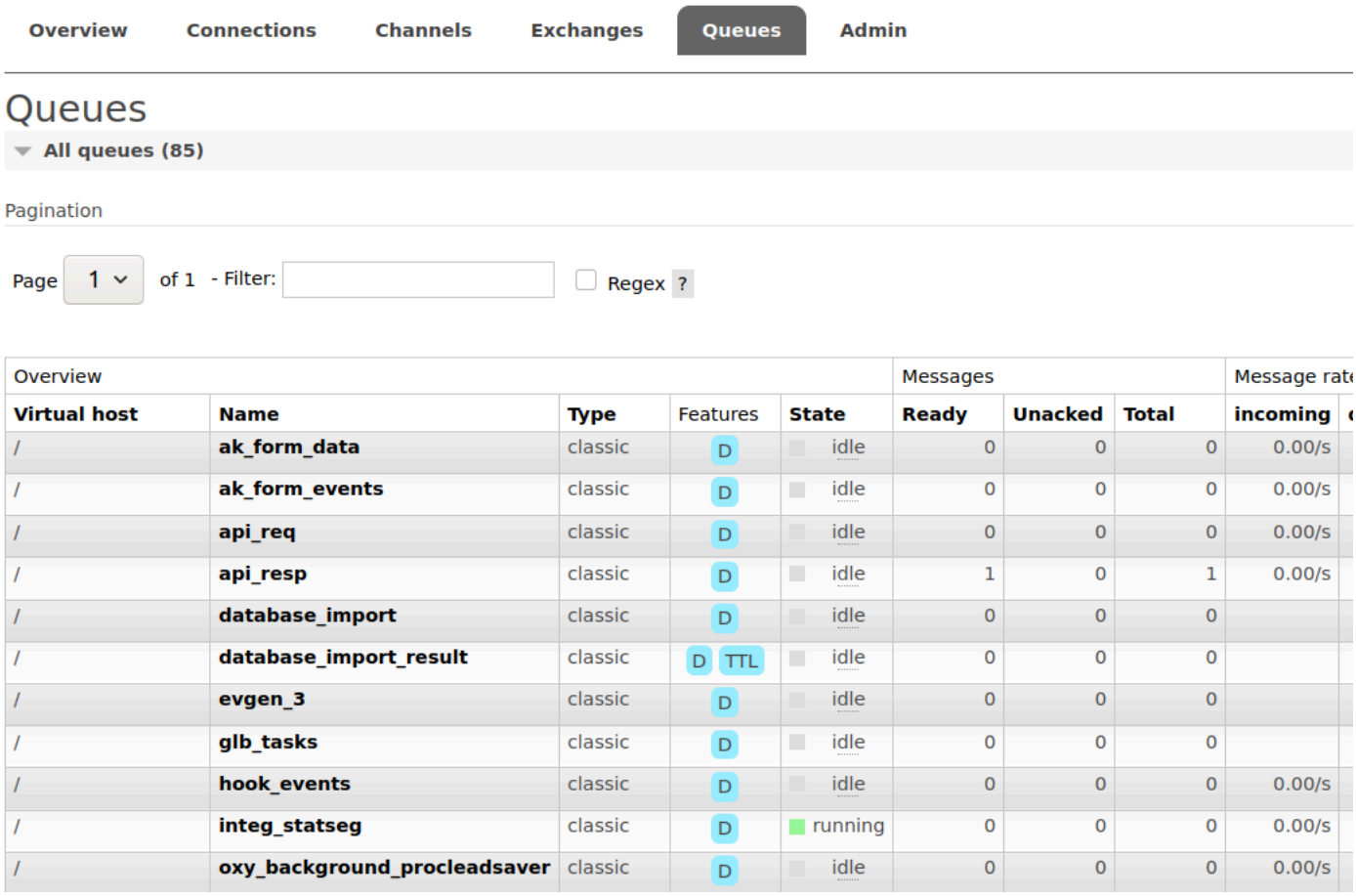
                for k, v := range bodyMap {
                        log.Println(k, v)
                }

                body, err := json.Marshal(bodyMap)
                failOnError(err, "Failed to json.Marshal bodyMap")

                err = ch.Publish(
                        "",               // exchange
                        qUeueImport.Name, // routing key
                        false,            // mandatory
                        false,            // immediate
                        amqp.Publishing{
                                ContentType: "text/plain",
                                Body:        []byte(body),
                        })
                failOnError(err, "Failed to publish a message")
        }

        log.Printf("[v] Sended %d requests to %s", amountOfPushMsgs, qUeueImport.Name)
}
```

- **After executing the script, go to RabbitMQ —> Queues**

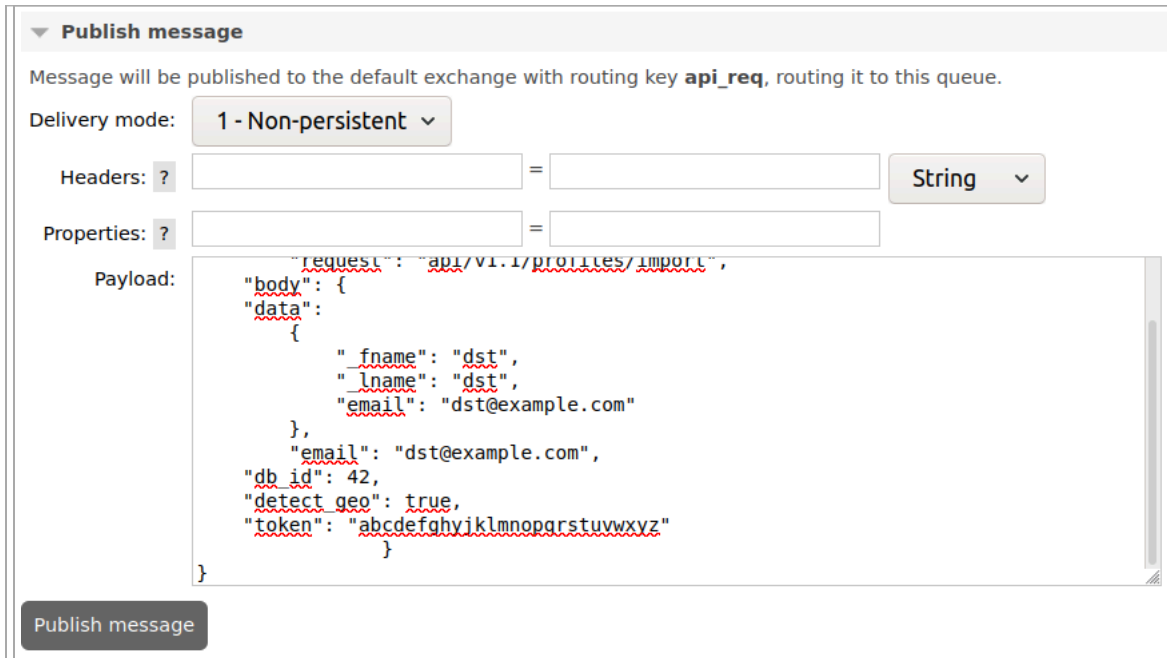


- **Choose api_resp and get a response to the API request**

How to send API requests via RabbitMQ Management Plugin

You can send an API request directly to RabbitMQ via api_req:

- **Go to RabbitMQ —> the Queues tab. Then select api_req and Publish message.**



- **After the request is sent, you can see its result in the same way as in the first method.**

List of gender identifications

In API requests, various reports and system parameters, gender can be displayed in different ways.
Male gender can be indicated on the platform by one of the parameters of the object below:

```
{
        "id": 1,
        "title": "Male",
        "values": ["male", "1", "Male", "MASCULINE",
            "M",
            "MAN",
            "♂",
            "М",
            "МУЖ",
            "МУЖСК",
            "МУЖСКОЙ",
            "МУЖЧИНА"
        ]
}
```

Any of the following code parameters is responsible for female gender:

```
{
        "id": 2,
        "title": "Female",
        "values": ["female", "2", "Female", "FEM",
            "FEMININE",
            "F",
            "WOMAN",
            "W",
            "♀",
            "Ж",
            "ЖЕН",
            "ЖЕНСК",
            "ЖЕНСКИЙ",
            "ЖЕНЩИНА"
        ]
}
```

TIP

If you need to add other designations, you can do so by adding the required values to the gender identifie

Obtain valid values for fields: browsers, devices, tz, oses, languages

Description

The API method allows you to retrieve valid values for the following fields:

- `browser`
- `device`
- `tz`
- `os`
- `language`

These fields are used in the following requests:

- `/v1.1/resources/fields_get` (**obtaining resource subscription fields**)
- `/api/v1.1/goals/register` (**goals and goal values registration**)

URL

Method: POST

Request URL: `https://example.com/api/v1.1/dictionary`

Request parameters

Parameter	Type	Example	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
name	**string**	**“browsers”** **“devices”** **“tz”** **“oses”** **“languages”**	**Yes**

Request example

JSON

```
{
   "token": "abcdefghijklmnqrstuvwxyz",
   "name": "devices"
}
```

XML

```
<xml>
  <token>1</token>
  <name>devices</name>
</xml>
```

Response example

JSON

```
{
  "data": [
    {
      "title": "Desktop",
      "value": "web"
    },
    {
      "title": "Other",
      "value": "other"
    },
    {
      "title": "Mobile",
      "value": "mob"
    }
  ],
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <value>web</value>
    <title>Desktop</title>
  </data>
  <data>
    <value>other</value>
    <title>Other</title>
  </data>
  <data>
    <value>mob</value>
    <title>Mobile</title>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

mSDK

Description

Altcraft mSDK is a mobile SDK for reliable push notification handling and interaction with the Altcraft plat users on the same device.

The SDK can use JWT tokens or rToken, which enables flexible security configuration and profile matchin

Minimum requirements:

- **iOS: Minimum Deployment Target — 13**
- **Android: minimum API level (minSdkVersion) — 25**

Features

- **Compatible with Kotlin, Java (Android), Swift, Objective-C (iOS).**
- **Push subscription management: subscribe, pause, unsubscribe.**
- **Automatic notification display with content configured in the platform, with UI customization options.**
- **Automatic push token refresh when it changes.**
- **Automatic delivery and open event reporting.**
- **Support for anonymous and registered users, including multiple users on the same device (with JWT).**
- **Mobile event registration.**
- **Automatic retry of requests via WorkManager for maximum delivery.**
- **Push provider support: Firebase, Huawei, RuStore (Android), APNS (iOS).**
- **Use of JWT tokens for security and unrestricted profile matching variants.**
- **Ability to work via rToken with restricted access tied to a push subscription identifier.**
- **Ability to override the push notification format and access all message data fields.**

GitHub

Altcraft mSDK repositories on GitHub:

- **iOS**
- **Android**

Dependency Managers

Cocoapods:

```
pod 'Altcraft', '1.0.3'
```

Swift Package Manager:

```
1.0.3
```

Maven Central:

```
com.altcraft:android-sdk:1.0.3
```

Authorization Methods

JWT token

This method uses a JWT token passed to the SDK by the application. The token is added to the header of

JWT (JSON Web Token) is a JSON-formatted string containing claims that are signed to verify authenticity

The token is generated and signed on the client's backend (encryption keys are not stored in the applicati

Advantages:

- **Higher security of API requests.**
- **Ability to match profiles by any identifiers (email, phone, custom ID).**
- **Support for multiple users on the same device.**
- **Restoring access to a profile after reinstalling the app.**
- **Identification of the same profile across multiple devices.**

rToken

An alternative authorization method uses a role token (rToken) passed in the SDK configuration. With this

Characteristics:

- **Profile lookup is only possible via the device's push token (e.g., FCM).**
- **If the push token changes and is not reported to the server (e.g., after removing and reinstalling the a**

Limitations:

- **Loss of profile linkage if a push token change is not registered on the Altcraft server.**
- **No support for using multiple profiles on the same device.**
- **No ability for a single user to log in on another device.**

Managing JWT and Role Token

Authorization Options

JWT Token

This authorization type uses a JWT token that the application passes to the SDK. The token is added to th
JWT (JSON Web Token) is a JSON-formatted string containing claims signed to verify authenticity and inte
The token is generated and signed with an encryption key on the client server side (encryption keys are n
Advantages:

- **Increased security of API requests.**
- **Ability to search profiles using any identifiers (email, phone number, custom ID).**
- **Support for multiple users on one device.**
- **Restoring access to a profile after reinstalling the application.**
- **Identification of a specific profile across different devices.**

rToken

An alternative authorization method uses a role token (rToken) passed in the SDK configuration paramete
Characteristics:

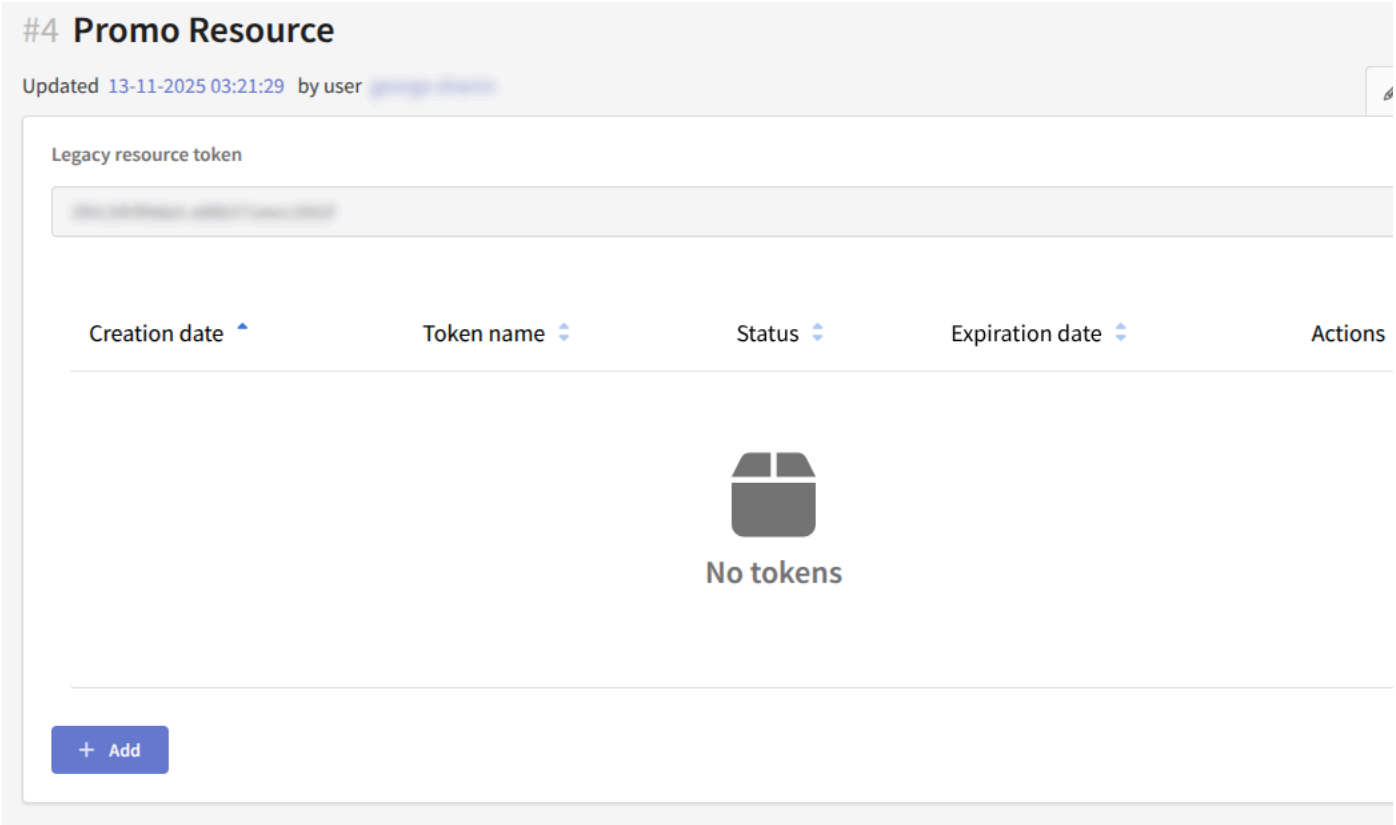
- **Profile search is only possible using the device push token (e.g. FCM).**
- **If the push token changes and is not sent to the server (for example, after app deletion and reinstallat**

Limitations:

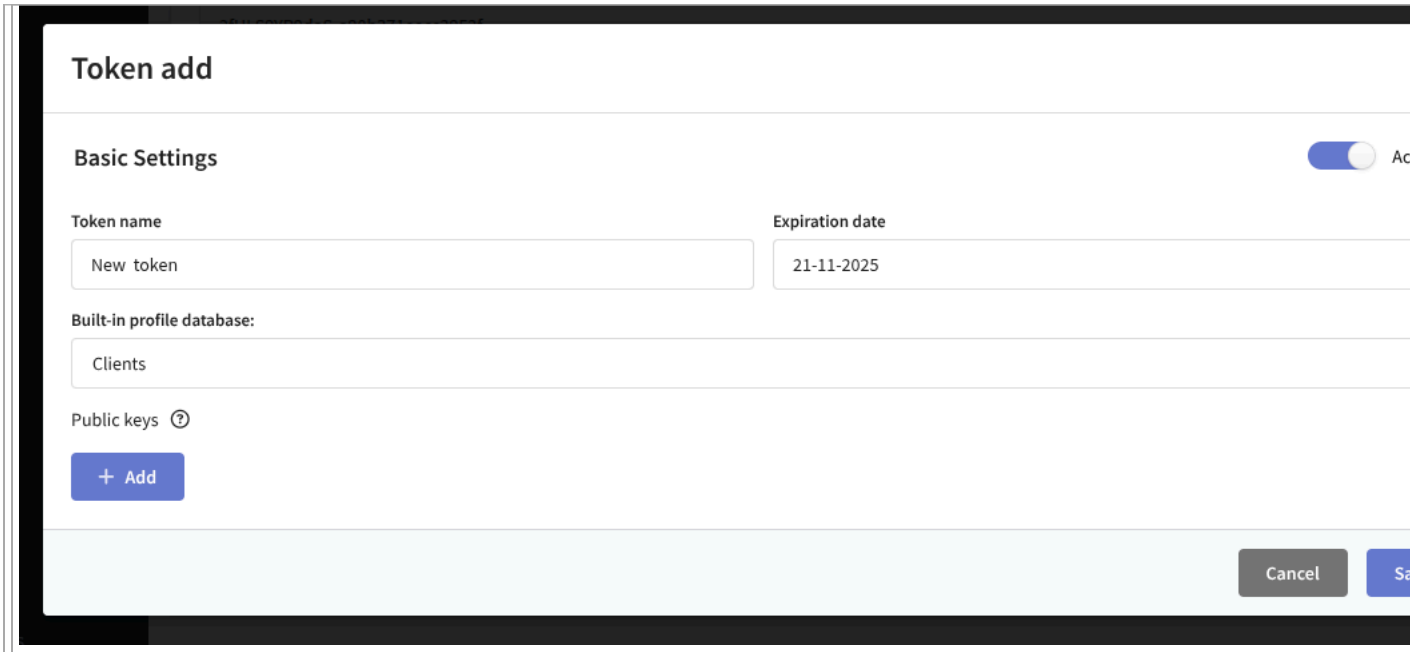
- **Loss of profile linkage if the push token changes and is not registered on the Altcraft server.**
- **No ability to use the application for different profiles on the same device.**
- **No ability to register the same user on another device.**

Configuring the Role Token and JWT Issuing Service

After creating a resource, the token management section becomes available in its settings:



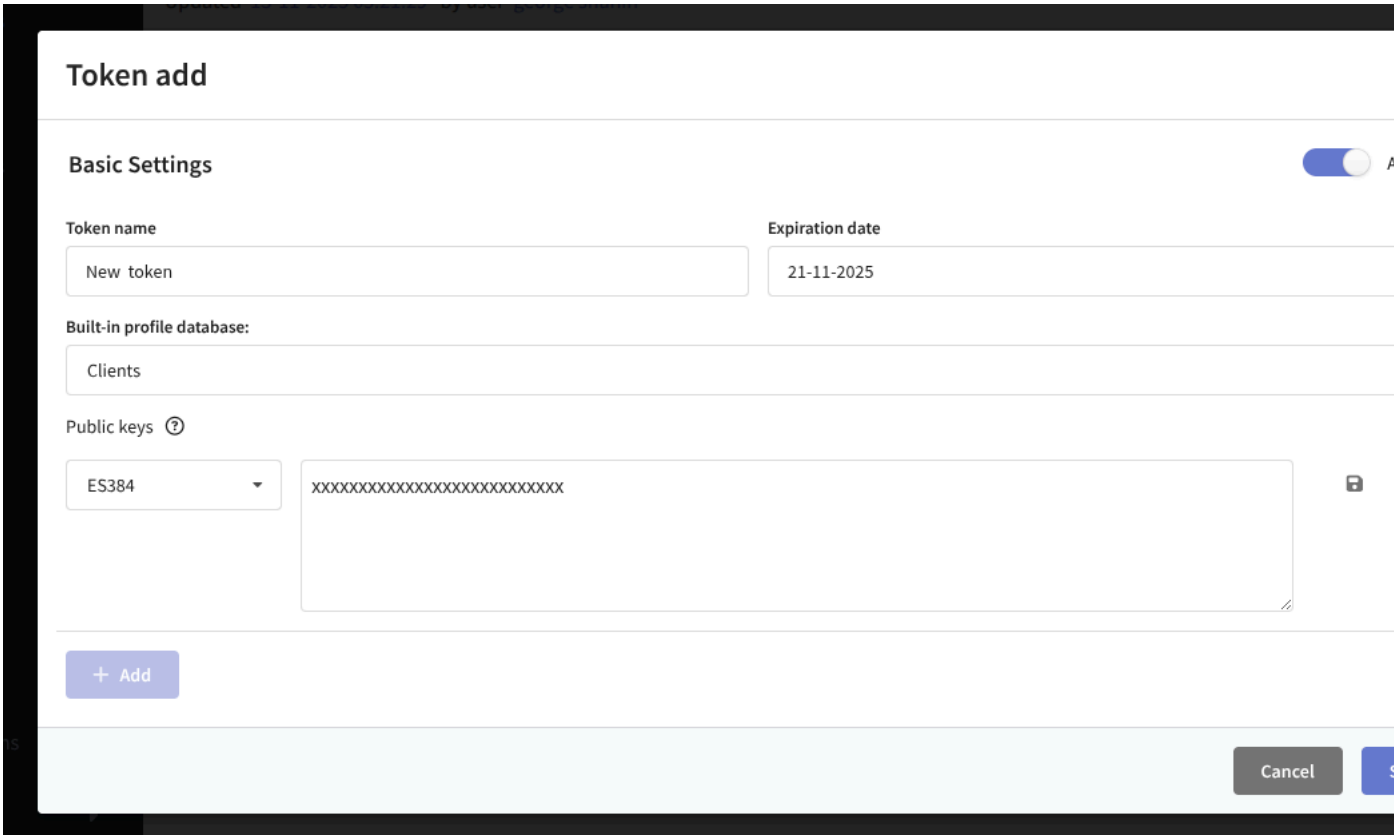
To create a role token, you do not need to provide a public key. Simply specify its name, expiration date, a



The role token acts as an access key for mSDK to the specific “resource–database” link. With this authori

- **Event registration**
- **Profile field updates**
- **Profile import**

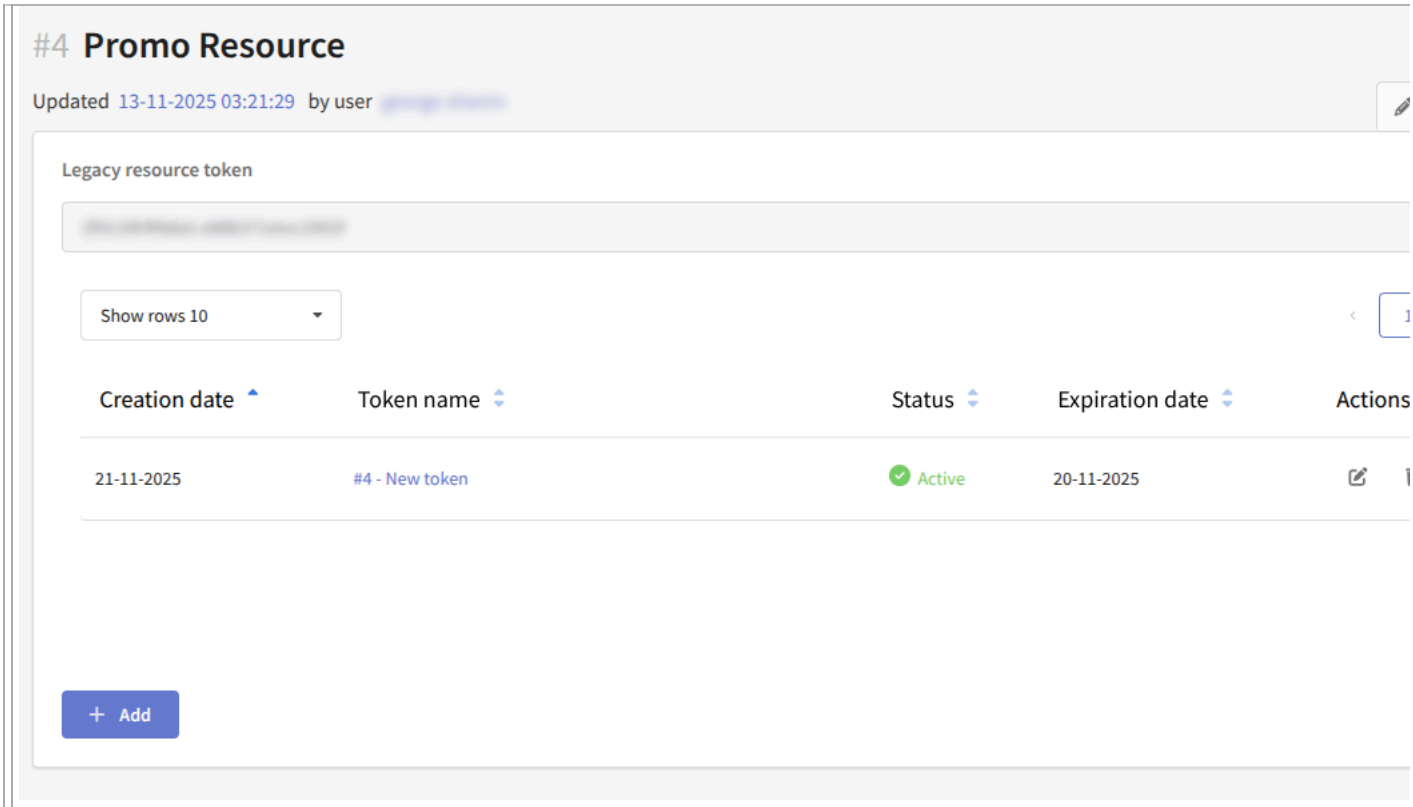
To create a JWT token, a public key is required. The platform supports the ES384 algorithm (ECDSA as the



TIP: Key generation

Example of generating a key using ES384: `openssl ecparam -name secp384r1 -genkey -noout -out private.ec.key` `ope`

After the token is created, it can be copied and used further:



Authorizing mSDK Requests with a Role Token

When used for authorization, rToken grants access to operations within a specific resource. Due to its ope
With a role token, profiles are searched using query parameters. In the current implementation, the backe

- **provider — mobile push provider**
- **subscription_id — device token provided by the provider**

The search for a profile is done by subscription identifier within the databases linked to the resource.

Characteristics

- **Profile search is possible only by the device push token (e.g. FCM).**
- **When importing a profile via push subscription, it is marked as “temporary”.**
- **When registering mobile events, linking events to a profile (writing to the event history) is not possibl**
- **If the push token changes and is not sent to the server, the link to the profile is lost and a new profile**

Usage

When using the `Altcraft SDK`, the token is passed as the `rToken` parameter of the `AltcraftConfiguration` class.
The library later includes it in the authorization header.

Authorizing mSDK Requests with a JWT Token

For secure authorization in the platform, it is recommended to use a JWT token. The token must be signed

Characteristics

- **Increased security of API requests. JWT acts as a protective wrapper for the role token, preventing u**
- **Ability to search profiles by any identifiers (email, phone number, custom ID), according to the matchi**
- **Support for multiple users on one device.**
- **Identification of a specific profile on different devices.**
- **Restoration of access to a profile after reinstalling the application.**

Usage

The platform expects the following JWT payload structure:

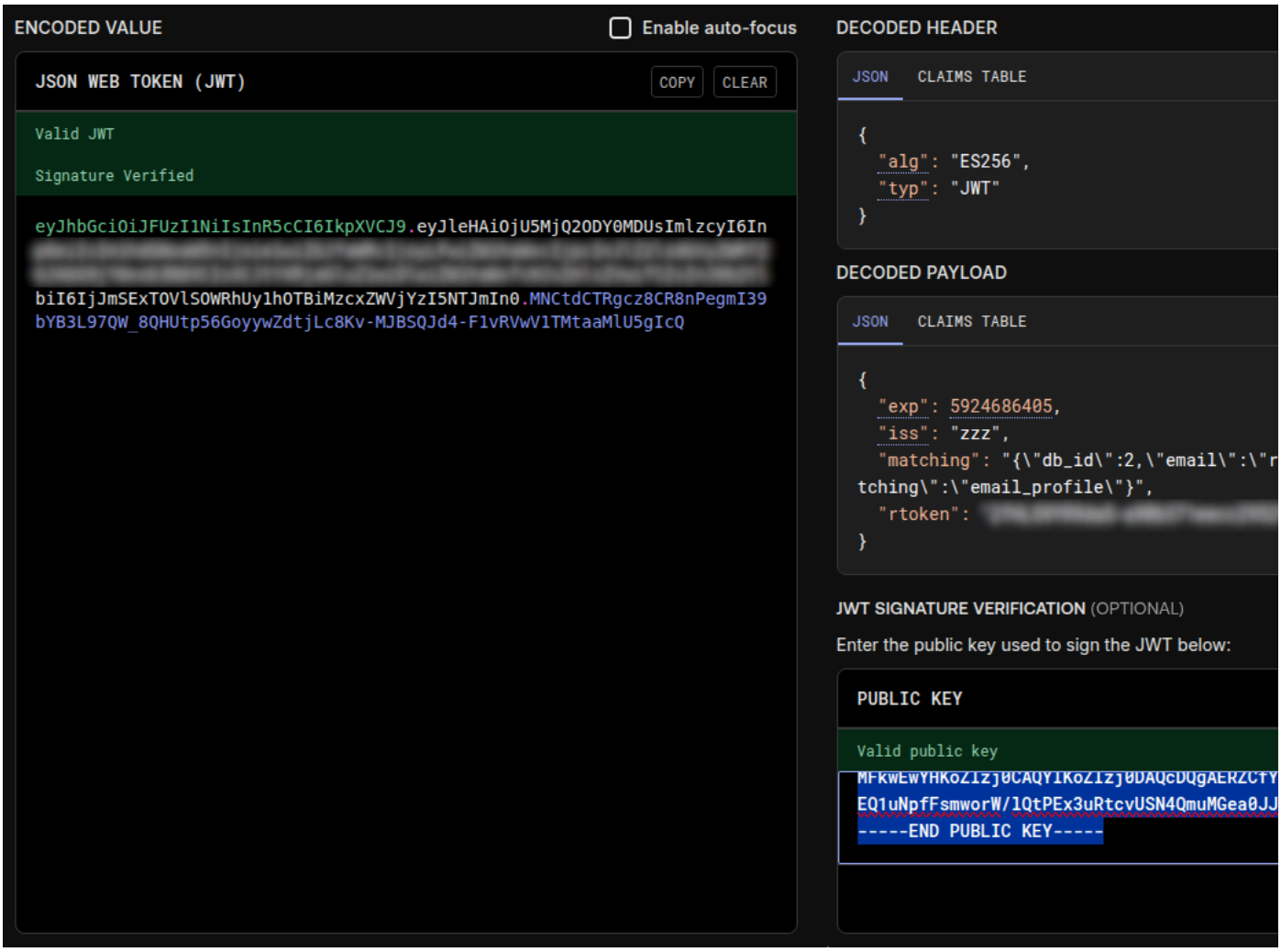
```
{
  "iss": "<App Name>",
  "exp": <UnixTimeUTC>,
  "rtoken": "<RoleToken>",
  "matching": "JSONString"
}
```

- **`iss` — issuer — unique identifier of the token issuer**
- **`exp` — expiration time — UNIX timestamp in seconds**

- `rtoken` — **the role token obtained during resource configuration**
- `matching` — `JSONString` — **a serialized object composed according to the documentation, for example:** `{`

When using the `Altcraft SDK`**, the token is passed as an implementation of the** `JWTInterface`**. The library late**

Example of a final JWT:



Quick Start

This guide demonstrates how to quickly integrate Altcraft mSDK with Firebase Cloud Messaging (FCM). A
To use Huawei Mobile Services (HMS) or RuStore, add their corresponding SDKs to your project and imple

Step 0. Prerequisites

- **Push provider SDKs are integrated into your app project (see the corresponding integration guides);**
- **The app includes a class extending `Application()`;**
- **All required entities (resource, mailing, template, profile database) are pre-created in the Altcraft plat**
- **On Android 13 (Tiramisu) and later, explicit user permission is required to send push notifications.**

Step 1. Connect Altcraft mSDK to the app

Add the library dependency to your app-level `build.gradle.kts` file:

```
dependencies {
        implementation("com.altcraft:android-sdk:1.0.3")
}
```

Then sync your Gradle project.

Step 2. Implementing JWTInterface

Implement the SDK interface that provides the JWT token. Create a class in your app module that implem

```
class JWTProvider(private val context: Context /** add context if required */) : JWTInterface {
    override fun getJWT(): String? {
        // your code returning the current JWT token
    }
}
```

> **TIP: Note**
>
> - **`getJWT()` is a synchronous function. The SDK execution thread will be blocked until the JWT is obtain**
> - **It's recommended that `getJWT()` return the token immediately from cache (`in-memory`, `SharedPreferences`**
> - **Ideally, prepare and cache the JWT as early as possible (e.g., during app startup) so that the SDK ca**

Example implementation using a suspend function

```
class JWTProvider(private val context: Context /** add context if required */) : JWTInterface {
    private suspend fun fetchJwt(context: Context): String? {
        // your code returning the current JWT token
        return jwt
    }

    override fun getJWT(): String? = runBlocking {
        fetchJwt(context)
    }
}
```

Step 3. Registering the JWT provider

Register the JWT provider in `Application.onCreate()`:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.setJWTProvider(JWTProvider(/** add context if required */))
    }
}
```

Step 4. Implementing the FCM interface

Create a class in your app module that implements `FCMInterface`:

Example implementation

```
/**
 * FCM implementation for managing Firebase Cloud Messaging tokens.
 *
 * Provides methods to retrieve and delete FCM tokens via the Firebase SDK.
 */
class FCMProvider : FCMInterface {

    /** Retrieves the current FCM token. Returns null if retrieval fails. */
    override suspend fun getToken(): String? {
        return try {
            Firebase.messaging.token.await()
        } catch (e: Exception) {
            null
        }
    }

    /** Deletes the current FCM token. */
    override suspend fun deleteToken(completion: (Boolean) -> Unit) {
        try {
            FirebaseMessaging.getInstance().deleteToken().addOnCompleteListener {
                completion(it.isSuccessful)
            }
        } catch (e: Exception) {
            completion(false)
        }
    }
}
```

Step 5. Creating a service extending `FirebaseMessagingService()`

In your app module, create a class that extends `FirebaseMessagingService` and override `onMessageReceived()` to p

Example service implementation

```
/**
 * FCM service for handling push tokens and messages.
 */
class FCMService : FirebaseMessagingService() {

    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)
        AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
    }
}
```

Step 6. Registering the FCM service in AndroidManifest

Register your `FirebaseMessagingService` in the app's `AndroidManifest.xml`:

```
<!-- FCM service -->
<service
    android:name="<your_package_name>.FCMService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

Step 7. Registering the FCM provider in `Application.onCreate()`

Register the Firebase Cloud Messaging provider in `Application.onCreate()`:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
    }
}
```

Step 8. Initializing the SDK

In `Application.onCreate()`, create a configuration object and initialize the SDK using `initialization()`. Make su

Initialization details

```
AltcraftSDK
 // SDK initialization and configuration setup
  └── fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? = null): Unit
```

`fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? = null): Un`

Example initialization:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())

        val config = AltcraftConfiguration.Builder(
            apiUrl = "https://pxl-your-instance.altcraft.com",
            icon = R.drawable.your_icon
        ).build()

        // set additional configuration parameters (see Android mSDK README)

        AltcraftSDK.initialization(context = this, configuration = config)
    }
}
```

Step 9. Subscribing to push notifications

Use the `pushSubscribe()` function to subscribe to push notifications at any point in your code — for example

```
AltcraftSDK
└──  val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
    // Push notifications subscription (status = SUBSCRIBED)
    ├─ fun pushSubscribe(
         context: Context,
         sync: Boolean = true,
         profileFields: Map<String, Any?>? = null,
         customFields: Map<String, Any?>? = null,
         cats: List<DataClasses.CategoryData>? = null,
         replace: Boolean? = null,
         skipTriggers: Boolean? = null
       ): Unit
```

Example subscription call:

```
class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        NotificationPermissionHandler.isGranted(
            activity = this,
            onGranted = {
                if (isUnsubscribed()) {
                   AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(context = this)
                }
            }
        )
    }
}
```

SDK Functionality

TIP

Before proceeding, you need to configure the SDK for your application. The detailed setup guide is availa

Working with subscription statuses

Changing a subscription status

```
AltcraftSDK
└─ val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
   // Subscribe (status = SUBSCRIBED)
   ├─ fun pushSubscribe(
   │     context: Context,
   │     sync: Boolean = true,
   │     profileFields: Map<String, Any?>? = null,
   │     customFields: Map<String, Any?>? = null,
   │     cats: List<DataClasses.CategoryData>? = null,
   │     replace: Boolean? = null,
   │     skipTriggers: Boolean? = null
   │   ): Unit
   // Suspend (status = SUSPENDED)
   ├─ fun pushSuspend(
   │     context: Context,
   │     sync: Boolean = true,
   │     profileFields: Map<String, Any?>? = null,
   │     customFields: Map<String, Any?>? = null,
   │     cats: List<DataClasses.CategoryData>? = null,
   │     replace: Boolean? = null,
   │     skipTriggers: Boolean? = null
   │   ): Unit
   // Unsubscribe (status = UNSUBSCRIBED)
   └─ fun pushUnSubscribe(
         context: Context,
         sync: Boolean = true,
         profileFields: Map<String, Any?>? = null,
         customFields: Map<String, Any?>? = null,
         cats: List<DataClasses.CategoryData>? = null,
         replace: Boolean? = null,
         skipTriggers: Boolean? = null
      ): Unit
```

Subscription status change functions:

- `fun pushSubscribe()` **— subscribes to push notifications;**
- `fun pushUnSubscribe()` **— cancels the push subscription;**
- `fun pushSuspend()` **— suspends the push subscription (no notifications are delivered, and an unsubscribe**
- `fun unSuspendPushSubscription()` **— used to implement** `LogIn` / `LogOut` **transitions.**

These functions share the same signature with the following parameters:

---

context: `Context`

Required: Yes Description: Android Context.

---

sync: `Boolean`

Default: `true` **Required: No Description: Flag that sets synchronous execution of the request.**

Successful request execution:

On success, this group of functions emits an event with code 230, 231 or 232 that contains `event.value`**, wh**

If `sync == true`

```
ResponseWithHttpCode
  ├ httpCode: 200
  └ response
     ├ error: 0
     ├ errorText: ""
     └ profile
        ├ id: "your id"
        ├ status: "subscribed"
        ├ isTest: false
        └ subscription
           ├ subscriptionId: "your subscriptionId"
           ├ hashId: "7f31a9c4"
           ├ provider: "android-firebase"
           ├ status: "subscribed"
           ├ fields
           │  ├ _os_ver: {"raw":"14","ver":"[\"14.0\", \"0.0\", \"0.0\"]"}
           │  ├ _device_type: "mob"
           │  ├ _ad_track: true
           │  ├ _device_name: "Pixel"
           │  ├ _os_language: "en"
           │  ├ _os_tz: "+0100"
           │  ├ _os: "Android"
           │  ├ _device_model: "Pixel 7"
           │  └ _app_ver: "1.0.0"
           └ cats
              └ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

For synchronous requests, `event.value["response_with_http_code"]` exposes:

- **httpCode – transport status code;**
- **Response – a data class containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty if no errors)*,**
  - `profile: ProfileData?` **— profile data if the request is successful:**
    - **profile info (ProfileData)**
    - **subscription (SubscriptionData)**
    - **subscription categories (CategoryData)**
    - **if the request fails, `profile = null` will be returned only**

▶ **Data structures**

If `sync == false`

```
ResponseWithHttpCode
  ├ httpCode: Int?
  └ response: Response?
     ├ error: Int?
     ├ errorText: String?
     └ profile: ProfileData? = null
```

For asynchronous requests, `event.value["response_with_http_code"]` exposes:

- **httpCode – transport status code;**
- **Response – a data class containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty if no errors)*,**
  - `profile: ProfileData?` **— profile data, always `null` for async requests.**

Failed request execution:

If a request from this group fails, an event with one of the following codes will be emitted:

Operations without automatic retry on the SDK side:

- **430 – notification subscription;**
- **431 — subscription suspension;**
- **432 — unsubscription.**

Operations with automatic retry on the SDK side:

- **530 – notification subscription;**
- **531 — subscription suspension;**
- **532 — unsubscription.**

Event contents:

- **only `httpCode` if the Altcraft server was unavailable;**
- **`error` and `errorText` if the server returned an error.**

Reading event values

```
AltcraftSDK.eventSDKFunctions.subscribe { event ->
    val codes = setOf(
        230, 231, 232,
        430, 431, 432,
        531, 532, 533
    )

    val code = event.eventCode ?: return@subscribe
    if (code !in codes) return@subscribe

    (event.eventValue?.get("response_with_http_code") as? DataClasses.ResponseWithHttpCode)?.let { responseWithHttp ->
        val httpCode = responseWithHttp.httpCode

        val response = responseWithHttp.response
        val error = response?.error
        val errorText = response?.errorText

        val profile = response?.profile
        val profileId = profile?.id
        val profileStatus = profile?.status
        val profileIsTest = profile?.isTest

        val subscription = profile?.subscription
        val subscriptionId = subscription?.subscriptionId
        val hashId = subscription?.hashId
        val provider = subscription?.provider
        val subscriptionStatus = subscription?.status

        val fields = subscription?.fields

        val cats = subscription?.cats
        val firstCat = cats?.firstOrNull()
        val catName = firstCat?.name
        val catTitle = firstCat?.title
        val catSteady = firstCat?.steady
        val catActive = firstCat?.active
    }
}
```

---

profileFields: `Map<String, Any?>?`
Default: `null`
Required: No
Description: A map containing profile fields.
The parameter may include system fields (e.g., `_fname` — first name or `_lname` — last name) as well as opti

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Int**
  - **Long**
  - **Float**
  - **Double**
  - `null`
- **Objects:** `Map<String, *>`
- **Lists:** `List<*>`
- **Array of maps:** `Array<Map<String, *>>`

If an invalid optional field is passed, the request will fail with:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: with field "field_name": Incorrect field
```

---

customFields: `Map<String, Any?>?`
Default: `null` **Required: No Description: A map containing subscription fields.**
May include system fields (e.g., `_device_model` — device model or `_os` — operating system) as well as optio

- **String**
- **Boolean**
- **Int**
- **Long**
- **Float**

- **Double**
- `null`

If an invalid optional field is passed, the request will fail with:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "field_name" is not valid: failed convert custom field
```

> **TIP: Note**
>
> **Most subscription system fields are collected automatically by the SDK and added to push requests. The**

---

cats: `listOf(CategoryData)`

Default: `null` **Required: No Description: Subscription categories.**

Category structure:

```
data class CategoryData(
    val name: String?,
    val title: String? = null,
    val steady: Boolean? = null,
    val active: Boolean?
)
```

When sending a push request with categories, use only the `name` **(category name) and** `active` **(category ac**

Example:

```
val cats = listOf(
    DataClasses.CategoryData(name = "football", active = true),
    DataClasses.CategoryData(name = "hockey",   active = true)
)
```

Categories used in the request must be created beforehand and added to the resource in the Altcraft platf

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "subscriptions.cats" is not valid: category not found in resource
```

---

replace: `Boolean?`

Default: `null` **Required: No Description: When enabled, all subscriptions of other profiles with the same pu**

---

skipTriggers: `Boolean?`

Default: `null` **Required: No Description: When enabled, the profile containing this subscription will be igno**

Request implementation examples

Example: subscribing to push notifications

Minimal working setup:

```
AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(context)
```

Passing all available parameters:

```
AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(
    context = this,
    sync = true,
    profileFields = mapOf("_fname" to "Andrey", "_lname" to "Pogodin"),
    customFields  = mapOf("developer" to true),
    cats = listOf(DataClasses.CategoryData(name = "developer_news", active = true)),
    replace = false,
    skipTriggers = false
)
```

> **TIP: Note**
>
> **For** `pushSubscribe`**,** `pushSuspend`**, and** `pushUnSubscribe`**, the SDK automatically retries the request if the HTTP s**

Function `unSuspendPushSubscription()`

The function `fun unSuspendPushSubscription()` **is intended to implement** `LogIn` / `LogOut` **transitions. It works as fo**

- **searches for subscriptions with the same push token as the current one that do not belong to the prof**
- **changes the status of the found subscriptions from** `subscribed` **to** `suspended`**;**
- **changes the status of subscriptions in the profile referenced by the current JWT from** `suspended` **to** `sub`
- **returns a** `ResponseWithHttpCode?` **data class in which** `response.profile` **is the current profile referenced by t**

Recommended `LogIn` / `LogOut` implementation

`LogIn` **transition:**

- **An anonymous user enters the app. The user is assigned** `JWT_1`**, pointing to database #1Anonymous.**

- **A push subscription is created; the profile is created in #1Anonymous.**
- **The user registers and is assigned `JWT_2`, pointing to database #2Registered.**
- **Call `unSuspendPushSubscription()` — the anonymous user's subscription in #1Anonymous is suspended.**
- **Search the profile in #2Registered to restore the subscription.**
- **Since a subscription with this push token does not exist in #2Registered, `unSuspendPushSubscription()` ret**
- **After receiving `null`, call `pushSubscribe()` to create a new profile in #2Registered.**

`LogOut` transition:

- **The user logs out in the app (`LogOut`).**
- **The user is assigned `JTW_1`, pointing to database #1Anonymous.**
- **Call `unSuspendPushSubscription()`, which will suspend the subscription in #2Registered and set the subscr**
- **The request returns `#1Anonymous != null` — the subscription already exists; no new one is required.**

Example:

```
private suspend fun unSuspend(context: Context, logIn: Boolean) {

    // Change JWT before the request
    setAuth(context, logIn)

    AltcraftSDK.pushSubscriptionFunctions
        .unSuspendPushSubscription(context)
        ?.let { result ->
            if (result.httpCode == 200 && result.response?.profile?.subscription == null) {
                AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(
                    context = context
                    // Specify required params
                )
            }
        }
}

fun logIn(context: Context)  = CoroutineScope(Dispatchers.IO).launch { unSuspend(context, true) }
fun logOut(context: Context) = CoroutineScope(Dispatchers.IO).launch { unSuspend(context, false) }
```

Getting a subscription status

```
AltcraftSDK
└── val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
    // Status of the profile's latest subscription
    ├── suspend fun getStatusOfLatestSubscription(
    │       context: Context
    │   ): DataClasses.ResponseWithHttpCode?
    // Status for the current token/provider
    ├── suspend fun getStatusForCurrentSubscription(
    │       context: Context
    │   ): DataClasses.ResponseWithHttpCode?
    // Status of the latest subscription for the given provider (if null – current provider is used)
    └── suspend fun getStatusOfLatestSubscriptionForProvider(
            context: Context,
            provider: String? = null
        ): DataClasses.ResponseWithHttpCode?
```

Subscription status request functions:

- **`fun getStatusOfLatestSubscription()` — status of the profile's latest subscription;**
- **`fun getStatusOfLatestSubscriptionForProvider()` — status for the current token/provider;**
- **`fun getStatusForCurrentSubscription()` — status of the latest subscription by provider. If `null` is passed, the**

`suspend fun getStatusOfLatestSubscription(context: Context): DataClasses.ResponseWithHttpCode?`

Retrieves the status of the profile's latest subscription. Returns a `ResponseWithHttp?` object containing `respo`

Usage example:

```
// Status of the profile's latest subscription
AltcraftSDK.pushSubscriptionFunctions.getStatusOfLatestSubscription(context)
```

`suspend fun getStatusForCurrentSubscription(context: Context): DataClasses.ResponseWithHttpCode?`

Retrieves the status for the current token/provider. Returns a `ResponseWithHttp?` object containing `response?.`

Usage example:

```
// Status matching the current token/provider
AltcraftSDK.pushSubscriptionFunctions.getStatusForCurrentSubscription(context)
```

`suspend fun getStatusOfLatestSubscriptionForProvider(context: Context, provider: String? = null): DataClasses.ResponseWith`

Retrieves the status of the latest subscription for a provider. Returns a `ResponseWithHttp?` object containing subscription exists, `null` is returned.

```
// Latest subscription status for a provider (uses current if null)
AltcraftSDK.pushSubscriptionFunctions.getStatusOfLatestSubscriptionForProvider(context, provider = null)
```

Below is an example of extracting profile, subscription, and category data from a status response. This a

Data from status functions

```
CoroutineScope(Dispatchers.IO).launch {
    AltcraftSDK.pushSubscriptionFunctions
        .getStatusForCurrentSubscription(this@App)
        ?.let { it ->
            val httpCode = it.httpCode
            val response = it.response
            val error = response?.error
            val errorText = response?.errorText
            val profile = response?.profile
            val subscription = profile?.subscription
            val cats = subscription?.cats
        }
}
```

Managing provider push tokens

```
AltcraftSDK
└── val pushTokenFunctions: PublicPushTokenFunctions
    |
    // Save a token manually (onNewToken)
    ├── fun setPushToken(context: Context, provider: String, token: String): Unit
    |
    // Get current device token data
    ├── suspend fun getPushToken(context: Context): DataClasses.TokenData?
    |
    // Set Firebase Cloud Messaging provider (null — remove)
    ├── fun setFCMTokenProvider(provider: FCMInterface?): Unit
    |
    // Set Huawei Mobile Services provider (null — remove)
    ├── fun setHMSTokenProvider(provider: HMSInterface?): Unit
    |
    // Set RuStore provider (null — remove)
    ├── fun setRuStoreTokenProvider(provider: RustoreInterface?): Unit
    |
    // Delete the push token for the specified provider
    ├── suspend fun deleteDeviceToken(context: Context, provider: String, complete: () -> Unit): Unit
    |
    // Force token refresh (delete –> refresh)
    ├── fun forcedTokenUpdate(context: Context, complete: () -> Unit): Unit
    |
    // Change provider priority list and trigger token refresh
    └── suspend fun changePushProviderPriorityList(context: Context, priorityList: List<String>): Unit
```

SDK functions for working with a provider token:

- `fun setPushToken()` **— manually set the device push token and provider in** `UserDefaults`**;**
- `fun getPushToken()` **— get the current push token;**
- `fun setFCMTokenProvider()` **— set or clear the FCM push token;**
- `fun setHMSTokenProvider()` **— set or clear the HMS push token;**
- `fun setRuStoreTokenProvider()` **— set or clear the RuStore push token;**
- `fun changePushProviderPriorityList()` **— dynamically switch provider and refresh the subscription's push to**
- `fun deleteDeviceToken()` **— delete the push token of the specified provider;**
- `fun forcedTokenUpdate()` **— delete the current push token and then refresh it.**

`fun setPushToken(context: Context, provider: String, token: String): Unit`

Used to manually set the device push token and provider. Should be called in the push provider service's

Using this function to pass the token is not recommended. The recommended approach is to implement `F`

Usage example:

```
// Manually save the token (onNewToken)
AltcraftSDK.pushTokenFunctions.setPushToken(context, provider, token)
```

Passing a token in `FCMService.onNewToken()`**:**

```
class FCMService : FirebaseMessagingService() {
    override fun onNewToken(token: String) {
        super.onNewToken(token)

        // Pass the new token manually
        AltcraftSDK.pushTokenFunctions.setPushToken(this, FCM_PROVIDER, token)
    }

    override fun onDeletedMessages() {}

    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)
        AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
    }
}
```

`suspend fun getPushToken(context: Context): DataClasses.TokenData?`

The function returns the current device push-token data and its provider as the data class `TokenData(val pro`

Usage example:

```
// Get current push token data
AltcraftSDK.pushTokenFunctions.getPushToken(context)
```

Example of a push token data request:

```
CoroutineScope(Dispatchers.IO).launch {
    AltcraftSDK.pushTokenFunctions.getPushToken(context).let {
        val provider = it?.provider
        val token = it?.token
    }
}
```

`fun setFCMTokenProvider(provider: FCMInterface?): Unit`

Sets or removes the Firebase Cloud Messaging push-token provider. To disable the provider, pass `null`**.**

Usage example:

```
// Set Firebase Cloud Messaging provider (null – remove)
AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
```

> **TIP: Important**
>
> **Call** `setFCMTokenProvider()` **in** `Application.onCreate()` **before calling** `AltcraftSDK.initialization()`**. This guarantees**

`fun setHMSTokenProvider(provider: HMSInterface?): Unit`

Sets or removes the Huawei Mobile Services push-token provider. To disable the provider, pass `null`**.**

Usage example:

```
// Set Huawei Mobile Services provider (null – remove)
AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
```

> **TIP: Important**
>
> **Call** `setHMSTokenProvider()` **in** `Application.onCreate()` **before calling** `AltcraftSDK.initialization()`**. This guarantees**

`fun setRuStoreTokenProvider(provider: RustoreInterface?): Unit`

Sets or removes the RuStore push-token provider. To disable the provider, pass `null`**.**

Usage example:

```
// Set RuStore provider (null – remove)
AltcraftSDK.pushTokenFunctions.setRustoreTokenProvider(RustoreProvider())
```

> **TIP: Important**
>
> **Call** `setRustoreTokenProvider()` **in** `Application.onCreate()` **before calling** `AltcraftSDK.initialization()`**. This guarant**

`suspend fun changePushProviderPriorityList(context: Context, priorityList: List<String>): Unit`

Allows dynamic switching of the push provider with subscription token refresh. Pass a new list with a diff

```
// Apply a new provider priority list and trigger token refresh
AltcraftSDK.pushTokenFunctions.changePushProviderPriorityList(context, listOf(HMS_PROVIDER, RUSTORE_PROVIDER, FCM_PROVIDER))
```

`suspend fun deleteDeviceToken(context: Context, provider: String, complete: () -> Unit): Unit`

Deletes the push token for the specified provider. The token is invalidated and removed from the device's

```
// Delete the push token for the given provider (invalidate locally and on the server)
AltcraftSDK.pushTokenFunctions.deleteDeviceToken(context, provider) {
    // callback after deletion
}
```

`fun forcedTokenUpdate(context: Context, complete: () -> Unit): Unit`

Deletes the current push token and then refreshes it.

Usage example:

```
// Force token refresh (delete -> update)
AltcraftSDK.pushTokenFunctions.forcedTokenUpdate(context) {
    // callback after update
}
```

Example of provider registration

We do not recommend using the `setPushToken` function to set a push token manually. Instead, configure tok

Recommended provider registration method

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        // initialize RuStore provider
        RuStorePushClient.init(this, "rustore project id")

        // set JWT provider
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))

        // set FCM provider
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())

        // set HMS provider
        AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())

        // set RuStore provider
        AltcraftSDK.pushTokenFunctions.setRuStoreTokenProvider(RuStoreProvider())

        // create AltcraftConfiguration
        val config = AltcraftConfiguration.Builder(
            apiUrl = "your api url",
            R.drawable.ic_altcraft_label
        ).build()

        // initialize SDK
        AltcraftSDK.initialization(context = this@App, configuration = config)
    }
}
```

Passing push notifications to the SDK

`PushReceiver` **is a public class that contains the function that accepts notifications.**

```
AltcraftSDK
└── open class PushReceiver
    // Incoming push handling
    ├── open fun pushHandler(
    │       context: Context,
    │       message: Map<String, String>
    │   ): Unit
    // SDK entry point for push delivery
    └── companion object
        └── fun takePush(
                context: Context,
                message: Map<String, String>
            ): Unit
```

Functions for passing push notifications to the SDK:

- `fun takePush()` **— accepts notifications in the push providers' service for further processing on the SDK**
- `fun pushHandler()` **— launches the standard Altcraft push-notification handling mechanism in the SDK.**

Receiving a notification

`fun takePush(context: Context, message: Map<String, String>): Unit`

Accepts push notifications for further processing on the SDK side.

Usage example:

```
  override fun onMessageReceived(message: RemoteMessage) {
       super.onMessageReceived(message)

      AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
   }
   // message.data — message payload
```

Handling a notification

`open fun pushHandler(context: Context, message: Map<String, String>): Unit`

Starts the standard Altcraft push-notification handling mechanism in the SDK.

Receiving notifications in any app package (optional)

To receive notifications in any application package, do the following:

Step 1. Create an `AltcraftPushReceiver` class extending `PushReceiver`. Override the `pushHandler()` function in th

```
@Keep
class AltcraftPushReceiver : AltcraftSDK.PushReceiver() {
    override fun pushHandler(context: Context, message: Map<String, String>) {
        // standard processing of push messages and displaying notifications
        super.pushHandler(context, message)
    }
}
```

> TIP: Note
>
> **The class must be named AltcraftPushReceiver. If you name it differently, the SDK will not be able to find**

Step 2. Depending on your business goals, configure the logic of the `AltcraftPushReceiver` class:

- **If you want the SDK to handle and display notifications:**
  - **use `super.pushHandler(context, message)`.**
- **If you want to handle notifications yourself:**
  - **do not call `super.pushHandler()`;**
  - **manually register open events using the `openEvent()` function; otherwise, it will not be registered b**

"deliveryEvent" delivery events are registered automatically. Creating `AltcraftPushReceiver` classes does no

> TIP: Note
>
> **If you created multiple `AltcraftPushReceiver` classes, calling `super.pushHandler()` in each of them will display**

Step 3. Add the package names that contain `AltcraftPushReceiver` classes to the `pushReceiverModules` configura

If the application code will be obfuscated, the class must be annotated with `@Keep` or added to the `R8/ProGu`

Example of adding a package to the `pushReceiverModules` parameter:

```
pushReceiverModules = listOf(
   context.packageName, // application package
   "com.altcraft.altcraftmobile.test"
)
```

Mobile Events

```
// Function of the PublicMobileEventFunctions object

AltcraftSDK
└── val mobileEventFunction: PublicMobileEventFunctions

    // Send a mobile event to the server
    └── fun mobileEvent(
        context: Context,
        sid: String,
        eventName: String,
        sendMessageId: String? = null,
        payload: Map<String, Any?>? = null,
        matching: Map<String, Any?>? = null,
        matchingType: String? = null,
        profileFields: Map<String, Any?>? = null,
        subscription: DataClasses.Subscription? = null,
        utm: DataClasses.UTM? = null
    ): Unit
```

Case: Transferring advertising campaign information

Information about the app advertising campaign that led to the installation can be sent to the platform as string, you need to call the `mobileEvent()` function:

Transfer via payload

```
AltcraftSDK.mobileEventFunctions.mobileEvent(
    this,
    sid = "your sid",
    eventName = "app_install",
    payload = mapOf("utm" to "your utm tag")
)
```

Transfer to UTM fields

```
AltcraftSDK.mobileEventFunctions.mobileEvent(
    this,
    sid = "your sid",
    eventName = "app_install",
    utm = DataClasses.UTM(
        campaign = "your campaign utm",
        content = "your content utm",
        keyword = "your keyword utm",
        medium = "your medium utm",
        source = "your source utm",
        temp = "your temp utm"
    )
)
```

Transfer to a custom profile field

```
// transfer a UTM tag to a profile field
// profile fields must be created in the platform in advance
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    profileFields: ["utm": "your utm tag"]
)
```

Data sent using any of these methods can be used to create a segment in the platform.
To register a mobile event, use the `mobileEvent()` function. It has the following parameters:

---

context: `Context`
Required: Yes
Description: Android Context.

---

sid: `String`
Required: Yes
Description: String identifier of the pixel to which mobile events are linked.

---

eventName: `String`
Required: Yes
Description: The name of the mobile event.

---

sendMessageId: `String?`
Required: No
Description: SMID identifier of the sent message (if the event is associated with a specific mailing).

---

payload: `String`
Required: No
Description: Event data — a map with string keys and values, where only scalar data types are allowed:

- **String**
- **Boolean**
- **Int**
- **Long**
- **Float**

- **Double**
- `null`

---

matching: `Map<String, Any?>?`
Required: No
Description: A map for passing values with matching types and identifiers.

---

matchingType: `String?`
Required: No
Description: The type of matching.

---

profileFields: `Map<String, Any?>?`
Required: No
Description: Profile fields — a map with string keys and JSON-compatible values:

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Int**
  - **Long**
  - **Float**
  - **Double**
  - `null`
- **Objects:** `Map<String, *>`
- **Lists:** `List<*>`
- **Arrays of maps:** `Array<Map<String, *>>`

> TIP: Note
> **This parameter is used only when working with JWT authorization.**

---

utm: `DataClasses.UTM?`
Required: No
Description: UTM tags. Added using the `UTM` data class, where each UTM type is a separate parameter.

```
data class UTM (
    val mailing: String? = null,
    val content: String? = null,
    val keyword: String? = null,
    val medium: String? = null,
    val source: String? = null,
    val temp: String? = null
)
```

---

subscription: `DataClasses.Subscription?`
Required: No
Description: Parameter for adding a subscription for a selected channel.
Possible values are implementations (subtypes) of the sealed interface `Subscription`:

- `EmailSubscription` — **email subscription**
- `SmsSubscription` — **SMS subscription**
- `PushSubscription` — **push subscription**
- `CcDataSubscription` — **subscription in Telegram, WhatsApp, Viber, or Notify.**

> TIP: Note
> **Used only when working with JWT authorization.**

Implementations of the `Subscription` interface

General model: `Subscription`
Purpose — base (sealed) interface for all types of subscriptions. Serialization is polymorphic, with the dis
Common fields (for all implementations):

Field	Type	
`resource_id`	`Int`	Ye
`status`	`String?`	N
`priority`	`Int?`	N
`custom_fields`	`Map<String, Any?>?`	N

Field	Type	
`cats`	`List<String>?`	N
`channel`	`String`	Y

Subscription types:

EmailSubscription (`channel = "email"`)

Required fields

Field	Type
`resourceId`	`Int`
`email`	`String`

Optional fields

Field	Type
`status`	`String`
`priority`	`Int`
`customFields`	`Map<String, Any?>`
`cats`	`List<String>`

SmsSubscription (`channel = "sms"`)

Required fields

Field	Type	
`resourceId`	`Int`	**Yes**
`phone`	`String`	**Yes**

Optional fields

Field	Type
`status`	`String`
`priority`	`Int`
`customFields`	`Map<String, Any?>`
`cats`	`List<String>`

PushSubscription (`channel = "push"`)

Required fields

Field	Type	Req
`resourceId`	`Int`	**Yes**
`provider`	`String`	**Yes**
`subscriptionId`	`String`	**Yes**

Optional fields

Field	Type
`status`	`String`
`priority`	`Int`
`customFields`	`Map<String, Any?>`
`cats`	`List<String>`

CcDataSubscription (`channel ∈ {"telegram_bot","whatsapp","viber","notify"}`)

Required fields

Field	Type	Required
`resourceId`	`Int`	**Yes**
`channel`	`String`	**Yes**
`ccData`	`JsonObject`	**Yes**

Optional fields

Field	Type
`status`	`String`
`priority`	`Int`
`customFields`	`Map<String, Any?>`
`cats`	`List<String>`

Profile Fields Update

INFO: Important

Updating profile fields with the `updateProfileFields()` function works only when using JWT authorization fo

```
AltcraftSDK
└── val profileFunctions: PublicProfileFunctions
    // Updates Altcraft profile fields
    └── fun updateProfileFields(
        context: Context,
        profileFields: Map<String, Any?>? = null,
        skipTriggers: Boolean? = null
    ): Unit
```

To update profile fields, use the `updateProfileFields()` function. It has the following parameters:

---

context: `Context`

Required: Yes
Description: Android Context.

---

profileFields: `Map<String, Any?>?`

Required: No
Description: Profile fields — a map containing the values for the fields that need to be updated (JSON-com

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Int**
  - **Long**
  - **Float**
  - **Double**
  - `null`
- **Objects:** `Map<String, *>`
- **Lists:** `List<*>`
- **Array of maps:** `Array<Map<String, *>>`

> INFO: Note
>
> **For the update to succeed, the fields must have been previously added to the subscriber's profile.**

---

skipTriggers: `Boolean?`

Default: `null`
Required: No
Description: When enabled, the profile containing this subscription will be ignored in mailing triggers and

Receiving SDK Events in the Application

```
// Functions of the Events object

AltcraftSDK
└── val eventSDKFunctions: Events
    // Subscribe to SDK events (replaces the existing subscriber)
    ├── fun subscribe(
    │       newSubscriber: (DataClasses.Event) -> Unit
    │   ): Unit
    // Unsubscribe from events
    └── fun unsubscribe(): Unit
```

The application can have only one active subscriber to SDK events.

Event Subscription

`fun subscribe(newSubscriber: (DataClasses.Event) -> Unit): Unit`

Used to subscribe and receive SDK events in the application. When an SDK event occurs, it calls the `callb`
Usage example:

```
AltcraftSDK.eventSDKFunctions.subscribe { event ->
    // Handle event
}
```

All events provided by the SDK are instances of `Event` or its subclasses:

- **Event — general event (info, successful requests);**
- **Error — error event;**
- **RetryError — error event for a request that will be automatically retried by the SDK.**

Each event contains the following fields:

- **function — the name of the function that triggered the event;**
- **eventCode — internal SDK event code (see "SDK Events");**

- **eventMessage — event message;**
- **eventValue — arbitrary `[String: Any?]?` data passed as event payload;**
- **date — timestamp of the event.**

SDK Event classes

```
open class Event(
    val function: String,
    val eventCode: Int? = null,
    val eventMessage: String? = null,
    val eventValue: Map<String, Any?>? = null,
    val date: Date = Date(),
)

open class Error(
    function: String,
    eventCode: Int? = 0,
    eventMessage: String? = null,
    eventValue: Map<String, Any?>? = null,
    date: Date = Date(),
) : Event(function, eventCode, eventMessage, eventValue, date)

class RetryError(
    function: String,
    eventCode: Int? = 0,
    eventMessage: String? = null,
    eventValue: Map<String, Any?>? = null,
    date: Date = Date(),
) : Error(function, eventCode, eventMessage, eventValue, date)
```

Example of a successful push subscription event

```
├─ function: processResponseprocessResponse
├─ eventCode: 230
├─ eventMessage: "successful request: push/subscribe"
├─ eventValue
│  ├─ http code: 200
│  └─ response
│     ├─ error: 0
│     ├─ errorText: ""
│     └─ profile
│        ├─ id: "your id"
│        ├─ status: "subscribed"
│        ├─ isTest: false
│        └─ subscription
│           ├─ subscriptionId: "your subscriptionId"
│           ├─ hashId: "c52b28d2"
│           ├─ provider: "android-firebase"
│           ├─ status: "subscribed"
│           ├─ fields
│           │  ├─ _device_name: "Pixel 7"
│           │  ├─ _device_model: "Google Pixel 7"
│           │  ├─ _os_tz: "+0300"
│           │  ├─ _os_language: "ru"
│           │  ├─ _os_ver: {"raw":"14","ver":[14]}
│           │  ├─ _ad_track: true
│           │  ├─ _os: "Android"
│           │  └─ _device_type: "mob"
│           └─ cats
│              └─ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
└─ date: 2025-09-03 09:01:44 +0000
```

Unsubscribing from events

`fun unsubscribe(): Unit`

This function unsubscribes from SDK events. It stops event delivery while keeping the callback assigned.

Usage example:

```
// Unsubscribe from SDK events
AltcraftSDK.eventSDKFunctions.unsubscribe()
```

List of all SDK events

Event list

Code	
200	**SDK configuration is installed**
201	**Push provider set**
202	**Received a notification unrelated to the Altcraft Platform**

Code	
203	Received Altcraft push notification
204	Push is posted
205	SDK data has been cleared
206	Waiting for SDK initialization to complete
220	Receiver override event
230	Subscribe request succeeded
231	Token update request succeeded
232	Unsuspend request succeeded
233	Status request succeeded
234	Push event delivered successfully
235	Mobile event delivered successfully
401	Configuration is not set
402	userTag is null — unable to identify the user
403	Unsupported entity type
404	SDK initialization timeout expired
405	Mobile event parts are null
422	Unsuspend request data is null
423	Profile request data is null
430	Subscribe request failed
431	Token update request failed
432	Unsuspend request failed
433	Status request failed
434	Push event delivery failed
435	Mobile event request failed
450	Push data is null
451	UID in push data is null or empty — unable to send push event to server
452	Error uploading notification image
453	Couldn’t create notification
454	Foreground info is null
455	Notification channel has not been created
471	Invalid provider (available: android-firebase, android-huawei, android-rustore)
472	Invalid customFields — not all values are primitives
480	Subscribe retry limit reached (request removed from DB)
484	Push event retry limit reached (request removed from DB)
485	Mobile event retry limit reached (request removed from DB)
501	Config data is null
502	Current push token is null
503	userTag is null — unable to identify the user
504	No permission to send notifications
505	No internet connection — retry when restored
506	failed to update the push token. The subscription request was rejected
520	Push subscribe request data is null
521	Token update request data is null
524	Push event request data is null
525	Mobile event request data is null
529	Common data is null
530	Subscribe request failed (retryable)
531	Token update request failed (retryable)
534	Push event delivery failed (retryable)
535	Mobile event delivery failed (retryable)
540	JWT token is null
541	Matching mode is null
542	JWT payload exceeds allowed size (16 KB limit): input rejected to prevent DoS
543	JWT does not contain a payload
544	Auth data is null
545	Matching claim does not contain a matching ID
560	Response data is null

Additional SDK Functions

Manual Push Event Registration

TIP: Note

These functions are only used if you implement your own notification handling logic and do not pass ther

```
// Functions of the PublicPushEventFunctions object

AltcraftSDK
└── val pushEventFunctions: PublicPushEventFunctions

    // Register Altcraft push delivery (triggers delivery event)
    ├── fun deliveryEvent(
    │       context: Context,
    │       message: Map<String, String>? = null,
    │       uid: String? = null
    │   ): Unit

    // Register Altcraft push open (triggers open event)
    └── fun openEvent(
            context: Context,
            message: Map<String, String>? = null,
            uid: String? = null
        ): Unit
```

Manual event registration functions:

- `fun deliveryEvent()` — **registers push delivery;**
- `fun openEvent()` — **registers push open.**

`fun deliveryEvent(context: Context, message: Map<String, String>? = null, uid: String? = null): Unit`

Registers a push delivery event manually on the Altcraft platform. To log the event on the server, pass the

Usage example:

```
// Register Altcraft push delivery (triggers delivery event)
AltcraftSDK.pushEventFunctions.deliveryEvent(context, message, uid)
```

`fun openEvent(context: Context, message: Map<String, String>? = null, uid: String? = null): Unit`

Registers a push open event manually on the Altcraft platform. To log the event on the server, pass the no

Usage example:

```
// Register Altcraft push open (triggers open event)
AltcraftSDK.PublicPushEventFunctions.openEvent(context, message, uid)
```

Passing custom data to Intent extras when opening a push notification

The SDK supports passing arbitrary data from a push notification to the application using the special `"_ex`

When a user taps the notification, the SDK automatically:

1. **Extracts the `_extra` value from the push payload.**
2. **Adds it to the `Intent`.**
3. **Passes it to the Activity that handles the notification open event or deep link.**

As a result, the contents of `_extra` become a regular `Intent` extra and are available in the application via:

```
val extraJson = intent?.getStringExtra("_extra")
```

The `_extra` field is intended for passing additional parameters that should not be included in the URL but a

The `_extra` value can be added using Custom JSON in the push notification template.

SDK Data Clearing

`fun clear(context: Context, onComplete: (() -> Unit)? = null)`

This function clears all SDK data and cancels all pending background tasks. It cancels `WorkManager` jobs, d

Usage example:

```
AltcraftSDK
// Full SDK data cleanup (DB, SharedPreferences, background tasks)
└── fun clear(
        context: Context,
        onComplete: (() -> Unit)? = null
    ): Unit
```

The function includes an optional `completion` parameter, which is called after cleanup and task cancellatio

Functional Profile Field Update

```
AltcraftSDK
└── val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
    └── fun actionField(key: String): ActionFieldBuilder
```

`fun actionField(key: String): ActionFieldBuilder`

A helper function for simplifying the functional profile field update process. It supports the following opera

```
    .set(value)
    .unset(value)
    .incr(value)
    .add(value)
    .delete(value)
    .upsert(value)
```

Usage example:

```
AltcraftSDK.pushSubscriptionFunctions.pushSubscribe(
    context = context,
    // "_fname" — field to be changed
    // .set("Andrey") — command that sets the value "Andrey" for this field
    profileFields = AltcraftSDK.pushSubscriptionFunctions
        .actionField("_fname").set("Andrey")
)
```

Resetting retry-operation restrictions during SDK initialization

> **TIP: Note**
>
> **When the SDK is initialized, background tasks are launched to control and retry requests related to push**
> **In some cases, you may need to reset this restriction. Use the following function:**

`fun reinitializeRetryControlInThisSession(): Unit`

This function resets the internal flag that prevents background retry-control tasks from running again wit

Requesting Notification Permission

> **TIP: Note**
>
> **Starting from Android 13 (API 33, Tiramisu), apps must explicitly request the `POST_NOTIFICATIONS` permissio**

`fun requestNotificationPermission(context: Context, activity: ComponentActivity)`

Checks the current notification permission status and, if necessary, displays the system permission requ

---

SDK Configuration

Prerequisites

- **The app includes a class that extends Application.**
- **The project is connected to the Maven Central repository.**
- **Add the library dependency to the app-level `build.gradle.kts` file:**

```
dependencies {
    implementation("com.altcraft:android-sdk:1.0.3")
}
```

App Preparation

Configuring the JWT interface (optional)

JWTInterface is an interface for requesting a JWT token. It provides the current JWT token from the app c
JWT authentication is required if you use a matching type other than push data from subscription (for exa
SDK interface:

```
interface JWTInterface {
    fun getJWT(): String?
}
```

App-side implementation

```
class JWTProvider(): JWTInterface {
    override fun getJWT(): String? {
        // code that returns a JWT token
    }
}
```

Registering the provider in `Application.onCreate()`

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))
    }
}
```

> **TIP: Note**
>
> **`getJWT()` is a synchronous function. The SDK execution thread will be paused until the JWT is obtained. I**
> **possible (at app start) and store it in cache so the token is available to the SDK without delays. If no valu**

Preparing to work with push providers

To prepare the app for working with push providers, you need to:

- **Integrate the push providers' SDKs into the project.**
- **Implement the SDK interfaces for passing and deleting the push token.**
- **Extend the push providers' services.**
- **Register the push provider service in `AndroidManifest.xml`.**

Firebase Cloud Messaging

Integrating the FCM SDK into the project

The article on integrating FCM is available here.

Altcraft SDK interface implementation

FCMInterface is an interface for requesting and deleting the FCM push token:

- **Passes the FCM token to Altcraft SDK**
- **Deletes the FCM token at the request of Altcraft SDK**

SDK interface:

```
interface FCMInterface {
    suspend fun getToken(): String?
    fun deleteToken(completion: (Boolean) -> Unit)
}
```

▶ **Recommended app-side implementation**

Realization the provider service

Recommended `FCMService` implementation that forwards the notification to the `takePush()` function:

```
/**
 * FCM service for handling push tokens and messages.
 */
class FCMService : FirebaseMessagingService() {

    /**
     * Called when a new FCM token is generated.
     *
     * @param token The new FCM token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)

       AltcraftSDK.PushReceiver.takePush(this@FCMService, message.data)
    }
}
```

Registering the push provider service in `AndroidManifest.xml`

```
<service
    android:name="<your_package_name>.FCMService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

Huawei Mobile Services

Integrating the HMS SDK into the project

The article on integrating HMS is available here.

Altcraft SDK interface implementation

HMSInterface is an interface for requesting and deleting the HMS push token.

SDK interface:

```
interface HMSInterface {
    suspend fun getToken(context: Context): String?
    fun deleteToken(context: Context, complete: (Boolean) -> Unit)
}
```

▶ Recommended app-side implementation

Realization the provider service

Recommended `HMSService` implementation that forwards the notification to the `takePush()` function:

```
/**
 * HMS service for handling push tokens and incoming notifications.
 *
 * Extends [HmsMessageService] and overrides key HMS callback methods.
 */
class HMSService : HmsMessageService() {
    /**
     * Called when a new HMS token is generated.
     *
     * @param token The new HMS token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received from HMS.
     *
     * Forwards the message with additional metadata to all receivers.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        AltcraftSDK.PushReceiver.takePush(this@HMSService, message.dataOfMap)
    }
}
```

Registering the push provider service in `AndroidManifest.xml`

```
<service
    android:name="<your_package_name>.HMSService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

RuStore

Integrating the RuStore SDK into the project

The article on integrating RuStore is available here.

Altcraft SDK interface implementation

RustoreInterface is an interface for requesting and deleting the RuStore push token.

SDK interface:

```
interface RustoreInterface {
    suspend fun getToken(): String?
    fun deleteToken(complete: (Boolean) -> Unit)
}
```

▶ Recommended app-side implementation

Realization the provider service

Recommended `RuStoreService` implementation that forwards the notification to the `takePush()` function:

```
/**
 * RuStore service for handling push notifications.
 *
 * Extends [RuStoreMessagingService] and overrides key callbacks.
 */
class RuStoreService : RuStoreMessagingService() {
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * Forwards the message to all receivers with added metadata.
     *
     * @param message The received RemoteMessage.
     */
    override fun onMessageReceived(message: RemoteMessage) {
        AltcraftSDK.PushReceiver.takePush(this, message.data)
    }
}
```

Registering the push provider service in `AndroidManifest.xml`

```
<service
    android:name="<your_package_name>.RuStoreService"
    android:exported="true"
    tools:ignore="ExportedService">
    <intent-filter>
    </intent-filter>
</service>
```

SDK Initialization

Initialization parameters

Use the `AltcraftConfiguration` class to pass configuration parameters:

```
class AltcraftConfiguration private constructor(
    private val apiUrl: String,
    private val icon: Int? = null,
    private val rToken: String? = null,
    private val appInfo: DataClasses.AppInfo? = null,
    private val providerPriorityList: List<String>? = null,
    private val pushReceiverModules: List<String>? = null,
    private val pushChannelName: String? = null,
    private val pushChannelDescription: String? = null,
    private val enableLogging: Boolean? = null
)
```

Parameter descriptions:

apiUrl `String`

Required: Yes
Description: URL of the Altcraft API endpoint.

---

icon `Int?`

Default: `null`
Required: No
Description: Resource ID of the `drawable` used as the notification icon.

---

rToken `String?`

Default: `null`
Required: No
Description: Altcraft role token (identifies a resource, database, or account). Used when the only matchin

---

appInfo `DataClasses.AppInfo?`

Default: `null`
Required: No
Description: Basic app metadata for Firebase Analytics. To set it, use the public SDK data class `AppInfo` (p

```
data class AppInfo(
    /** Firebase app_id */
    val appID: String,
    /** Firebase app_instance_id */
    val appIID: String,
    /** Firebase app_version */
    val appVer: String
)
```

---

enableLogging `Boolean`

Default: `null`
Required: No
Description: Enables/disables logging. Logs can be found by logCat tag `altcraft_lib`.

---

providerPriorityList `List<String>?`

Default: `null`
Required: No
Description: Provider priority list. Used to automatically refresh the subscription’s push token if a higher-p

Usage example:

```
providerPriorityList = listOf(
    "android-firebase",
    "android-huawei",
    "android-rustore"
)
```

With this setup:

- **The SDK first requests FCM; if FCM is unavailable—HMS; if HMS is unavailable—RuStore.**

- **Works provided the corresponding provider interfaces are implemented in the app.**

For convenience, the SDK exposes public constants:

```
const val FCM_PROVIDER: String = "android-firebase"
const val HMS_PROVIDER: String = "android-huawei"
const val RUS_PROVIDER: String = "android-rustore"
```

If not set, the default provider priority is:

```
FCM_PROVIDER -> HMS_PROVIDER -> RUS_PROVIDER
```

The value may contain a single element—in this case only that provider will be used, regardless of push-t

You can omit this parameter if:

- **the project uses only one provider;**
- **the default priority meets your requirements.**

> TIP: Use cases
>
> **Case 1. Priority `RUS_PROVIDER -> FCM_PROVIDER`. A user removes RuStore—RuStore notifications become unava**
> **2. Parameter not set; defaults `FCM_PROVIDER -> HMS_PROVIDER -> RUS_PROVIDER` apply. On a Huawei device without**

---

pushReceiverModules `List<String>?`

Default: `null` Required: No Description: List of package names containing implementations of `AltcraftPushH`

If set, the SDK discovers these classes and forwards incoming notifications to them via `pushHandler(context:`

Example of creating an `AltcraftPushReceiver` class

```
package com.altcraft.altcraftmobile.test

@Keep
class AltcraftPushReceiver: AltcraftSDK.PushReceiver() {
    override fun pushHandler(context: Context, message: Map<String, String>) {

        // handle the notification using SDK functions
        super.pushHandler(context, message)

        // additional notification handling logic
    }
}
```

> TIP: Use case
>
> **Case: The app contains multiple modules that need Altcraft incoming push data. In each module you can**
> **Note the class must be named exactly `AltcraftPushReceiver`.**
> **If the project has only one `AltcraftPushReceiver`, call `super.pushHandler(context, message)` inside `pushHandler` so t**
> **manual control of push handling, you can omit `super.pushHandler`.**

---

pushChannelName `String?`

Default: `null` Required: No Description: Base name of the push-notification channel (visible in Android set

- **`allSignal` — sound and vibration enabled**
- **`soundless` — silent channel**
- **`onlySound` — sound only (no vibration)**

Example: with `pushChannelName = "Altcraft"` and mode `allSignal`, the visible channel name is `"Altcraft_allSigna`

---

pushChannelDescription `String?`

Default: `null` Required: No Description: Description of the push channel (visible in Android settings). The

- `Vibration and sound enabled`
- `Vibration and sound disabled`
- `Sound enabled, vibration disabled`

Example: `"Altcraft notification channel. (vibration and sound enabled)"`. If not set, the SDK uses these defaults:

- **icon — `altcraft-sdk/src/main/res/drawable/icon.xml` (if icon = null)**
- **providerPriorityList — `FCM_PROVIDER -> HMS_PROVIDER -> RUS_PROVIDER` (if null)**
- **pushChannelName — `"allSignal", "soundless", "onlySound"` (if null)**
- **pushChannelDescription — `"Vibration and sound enabled", "Vibration and sound disabled", "Sound enabled, vibrat`**

Running initialization

> TIP: **Note**
>
> **Call `AltcraftSDK.initialization(...)` when needed, but only after registering all providers (the JWT provider**

Use `fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? = null`

```
// Initialize the SDK and set configuration

AltcraftSDK.initialization(
    context = context,
    configuration = config,
    complete = null // optional
)
```

Example of correct initialization order in `Application.onCreate()` (after registering providers)

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        // Initialize the RuStore SDK provider (if used)
        RuStorePushClient.init(this, "rustore-project-id-1234")

        // Register providers before initializing the SDK
        AltcraftSDK.setJWTProvider(JWTProvider(applicationContext))
        AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
        AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
        AltcraftSDK.pushTokenFunctions.setRuStoreTokenProvider(RuStoreProvider())

        // SDK configuration
        val config = AltcraftConfiguration.Builder(
            apiUrl = "https://pxl-example.altcraft.com",
            icon   = R.drawable.ic_notification
        ).build()

        // Initialization
        AltcraftSDK.initialization(this, config)
    }
}
```

Example of minimal working setup

```
val config = AltcraftConfiguration.Builder(
    apiUrl = "https://pxl-example.altcraft.com"
).build()

AltcraftSDK.initialization(context, config)
```

Example of full parameter configuration

```
val config = AltcraftConfiguration.Builder(
    apiUrl = "https://pxl-example.altcraft.com",
    icon   = R.drawable.ic_notification,
    rToken = null,
    appInfo = DataClasses.AppInfo(
        appID  = "com.example.app",
        appIID = "8b91f3a0-1111-2222-3333-c1a2c1a2c1a2",
        appVer = "1.0.0"
    ),
    providerPriorityList = listOf(
        FCM_PROVIDER, // "android-firebase"
        HMS_PROVIDER, // "android-huawei"
        RUS_PROVIDER  // "android-rustore"
    ),
    pushReceiverModules = listOf(
        context.packageName,
        "com.example.push_receiver",
        "com.example.feature.test"
    ),
    pushChannelName        = "Altcraft",
    pushChannelDescription = "Altcraft notifications channel"
).build()

AltcraftSDK.initialization(context, config)
```

Example of initialization with completion callback

```
AltcraftSDK.initialization(context, config) { result ->
    when {
        result.isSuccess -> {
            // actions on successful initialization
        }
        result.isFailure -> {
            // handle initialization errors
        }
    }
}
```

Public SDK API

object AltcraftSDK

```
AltcraftSDK
// SDK initialization and configuration setup
├─ fun initialization(context: Context, configuration: AltcraftConfiguration, complete: ((Result<Unit>) -> Unit)? = null): Unit
// Full cleanup of SDK data (DB, SP, background tasks)
├─ fun clear(context: Context, onComplete: (() -> Unit)? = null): Unit
// Register a JWT provider
├─ fun setJWTProvider(provider: JWTInterface?): Unit
// Allow push-module reinitialization in the current session
├─ fun unlockInitialOperationsInThisSession(): Unit
// Base Altcraft push receiver (can be overridden)
├─ open class PushReceiver
│  // Handle incoming push message
│  ├─ open fun pushHandler(context: Context, message: Map<String, String>): Unit
│  // Entry point for delivering push messages to the SDK
│  └─ companion object
│     └─ fun takePush(context: Context, message: Map<String, String>): Unit
// Public subscription functions
├─ val pushSubscriptionFunctions: PublicPushSubscriptionFunctions
│  // Subscribe to push notifications (status = SUBSCRIBED)
│  ├─ fun pushSubscribe(
│  │     context: Context,
│  │     sync: Boolean = true,
│  │     profileFields: Map<String, Any?>? = null,
│  │     customFields: Map<String, Any?>? = null,
│  │     cats: List<DataClasses.CategoryData>? = null,
│  │     replace: Boolean? = null,
│  │     skipTriggers: Boolean? = null
│  │   ): Unit
│  // Suspend push notifications (status = SUSPENDED)
│  ├─ fun pushSuspend(...): Unit
│  // Unsubscribe from push notifications (status = UNSUBSCRIBED)
│  ├─ fun pushUnSubscribe(...): Unit
│  // Change subscription from suspended to subscribed for the one in JWT; others with this token switch to suspended
│  ├─ suspend fun unSuspendPushSubscription(context: Context): DataClasses.ResponseWithHttpCode?
│  // Status of the latest profile subscription
│  ├─ suspend fun getStatusOfLatestSubscription(context: Context): DataClasses.ResponseWithHttpCode?
│  // Status of latest subscription for a specific push provider
│  ├─ suspend fun getStatusOfLatestSubscriptionForProvider(context: Context, provider: String? = null): DataClasses.ResponseWithHttpCode?
│  // Status of subscription with current device token
│  ├─ suspend fun getStatusForCurrentSubscription(context: Context): DataClasses.ResponseWithHttpCode?
│  // Functional profile field (set/incr/…)
│  └─ fun actionField(key: String): ActionFieldBuilder
// Token management functions
├─ val pushTokenFunctions: PublicPushTokenFunctions
│  // Manually save provider token (onNewToken)
│  ├─ fun setPushToken(context: Context, provider: String, token: String): Unit
│  // Get current device push token
│  ├─ suspend fun getPushToken(context: Context): DataClasses.TokenData?
│  // Register FCM provider
│  ├─ fun setFCMTokenProvider(provider: FCMInterface?): Unit
│  // Register HMS provider
│  ├─ fun setHMSTokenProvider(provider: HMSInterface?): Unit
│  // Register RuStore provider
│  ├─ fun setRuStoreTokenProvider(provider: RustoreInterface?): Unit
│  // Delete provider device token
│  ├─ suspend fun deleteDeviceToken(context: Context, provider: String, complete: () -> Unit): Unit
│  // Forced token update (delete → refresh)
│  ├─ fun forcedTokenUpdate(context: Context, complete: () -> Unit): Unit
│  // Change provider priority list and update token
│  └─ suspend fun changePushProviderPriorityList(context: Context, priorityList: List<String>): Unit
// Push event tracking functions
├─ val pushEventFunction: PublicPushEventFunctions
│  // Track delivery (delivery event)
│  ├─ fun deliveryEvent(context: Context, message: Map<String, String>? = null, uid: String? = null): Unit
│  // Track open (open event)
│  └─ fun openEvent(context: Context, message: Map<String, String>? = null, uid: String? = null): Unit
│  // Registering mobile events
├─ val mobileEventFunctions: PublicMobileEventFunctions
│  // Registers a mobile event for Altcraft
│  └─ fun mobileEvent(
│        context: Context,
│        sid: String,
│        eventName: String,
│        sendMessageId: String? = null,
│        payload: Map<String, Any?>? = null,
│        matching: Map<String, Any?>? = null,
│        matchingType: String? = null,
│        profileFields: Map<String, Any?>? = null,
│        subscription: DataClasses.Subscription? = null,
│        utm: DataClasses.UTM? = null
```

```
    ): Unit
  // Profile field update
├─ val profileFunctions: PublicProfileFunctions
  // Updates profile fields for Altcraft
  └─ fun updateProfileFields(
        context: Context,
        profileFields: Map<String, Any?>? = null,
        skipTriggers: Boolean? = null
    ): Unit
  // SDK events (single subscriber)
└─ val eventSDKFunctions: Events
   ├─ fun subscribe(newSubscriber: (DataClasses.Event) -> Unit): Unit
   └─ fun unsubscribe(): Unit
```

class AltcraftConfiguration

```
com.altcraft.sdk.config
└─ class AltcraftConfiguration private constructor(...)
   // SDK configuration class:
   // API URL, rToken, app info, foreground service flag,
   // push channel settings, receiver modules, provider priority.

   ├─ class Builder(
   │     apiUrl: String,
   │     icon: Int? = null,
   │     rToken: String? = null,
   │     appInfo: DataClasses.AppInfo? = null,
   │     enableLogging: Boolean = false,
   │     providerPriorityList: List<String>? = null,
   │     pushReceiverModules: List<String>? = null,
   │     pushChannelName: String? = null,
   │     pushChannelDescription: String? = null
   │   )
   │    └─ fun build(): AltcraftConfiguration
   │
   ├─ fun getApiUrl(): String
   ├─ fun getIcon(): Int?
   ├─ fun getRToken(): String?
   ├─ fun getAppInfo(): DataClasses.AppInfo?
   ├─ fun getEnableLogging(): Boolean
   ├─ fun getProviderPriorityList(): List<String>?
   ├─ fun getPushReceiverModules(): List<String>?
   ├─ fun getPushChannelName(): String?
   └─ fun getPushChannelDescription(): String?
```

Event Classes

```
Event
function: String
eventCode: Int?
eventMessage: String?
eventValue: Map<String, Any?>?
date: Date

Error
function: String
eventCode: Int?
eventMessage: String?
eventValue: Map<String, Any?>?
date: Date

RetryError
function: String
eventCode: Int?
eventMessage: String?
eventValue: Map<String, Any?>?
date: Date
```

Data Classes

```
data class TokenData(
    provider: String,
    token: String
)

data class AppInfo(
    appID: String,
    appIID: String,
    appVer: String
)
asMap(): Map<String, String>

data class ResponseWithHttpCode(
    httpCode: Int?,
    response: Response?
)

data class Response(
    error: Int? = null,
    @SerialName("error_text") errorText: String? = null,
    profile: ProfileData? = null
)

data class ProfileData(
    id: String? = null,
    status: String? = null,
    @SerialName("is_test") isTest: Boolean? = null,
    subscription: SubscriptionData? = null
)

data class SubscriptionData(
    @SerialName("subscription_id") subscriptionId: String? = null,
    @SerialName("hash_id") hashId: String? = null,
    provider: String? = null,
    status: String? = null,
    fields: Map<String, JsonElement>? = null,
    cats: List<CategoryData>? = null
)

data class CategoryData(
    name: String?,
    title: String? = null,
    steady: Boolean? = null,
    active: Boolean?
)

data class EmailSubscription(...)

data class SmsSubscription(...)

data class PushSubscription(...)

data class CcDataSubscription(...)

data class UTM(
    mailing: String? = null,
    content: String? = null,
    keyword: String? = null,
    medium: String? = null,
    source: String? = null,
    temp: String? = null
)
```

Interfaces

JWTInterface

```
fun getJWT(): String?
```

FCMInterface

```
suspend fun getToken(): String?
suspend fun deleteToken(completion: (Boolean) -> Unit)
```

HMSInterface

```
suspend fun getToken(context: Context): String?
suspend fun deleteToken(context: Context, complete: (Boolean) -> Unit)
```

RuStoreInterface

```
suspend fun getToken(): String?
suspend fun deleteToken(complete: (Boolean) -> Unit)
```

Subscription (sealed interface)

```
val resourceId: Int
val status: String?
val priority: Int?
val customFields: Map<String, Any?>?
val cats: List<String>?
val channel: String
```

Firebase Cloud Messaging

Firebase Cloud Messaging Project Setup

Create a [Firebase] project:
Add your app to the project:
Register the app by filling in the required details:
Download the `google-services.json` file:

Configuring the Altcraft Resource

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existing
Then, scroll down to the Firebase settings section and specify your Project ID (`messagingSenderId`) and Send
You can generate this Firebase file in Project settings —> Service account in the Firebase console.
After making changes, don't forget to save the resource.

Integrating FCM into the App Project

Manual Integration

Move the downloaded `google-services.json` file to the root directory of your app module (app):
Then add the necessary plugins and dependencies to your Gradle files:

- `build.gradle.kts (project level)`

Add the `google-services` plugin to the project-level `build.gradle.kts` file:

```
plugins {
    id("com.google.gms.google-services") version "4.4.3" apply false
}
```

> TIP: Note
>
> **Information about the latest plugin version is available here.**

- `build.gradle.kts (app level)`

Add the `google-services` plugin to the app-level `build.gradle.kts` file:

```
plugins {
    id("com.google.gms.google-services")
}
```

Also add dependencies:

Using firebase-bom (automatically manages Firebase library versions)

```
dependencies {
    implementation(platform("com.google.firebase:firebase-bom:34.0.0"))
    implementation("com.google.firebase:firebase-messaging")
}
```

Without firebase-bom

```
dependencies {
    implementation("com.google.firebase:firebase-messaging:25.0.0")
}
```

After adding the plugins and dependencies, sync your Gradle changes.

Automatic Setup in Android Studio

Navigate to Tools -> Firebase -> Cloud Messaging -> Set up Firebase Cloud Messaging:

Creating a Class Extending `FirebaseMessagingService()`

Create a class that extends `FirebaseMessagingService()`:

```
/**
 * FCM service for handling push tokens and messages.
 */
class FCMService : FirebaseMessagingService() {

    /**
     * Called when a new FCM token is generated.
     *
     * @param token The new FCM token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * Forwards the message to all receivers with additional metadata.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        super.onMessageReceived(message)

        // Message
    }
}
```

Updating the App Manifest

Add the `FCMService` declaration to your app's `AndroidManifest.xml` file:

```
        <service
            android:name="com.altcraft.fcm.FCMService"
            android:exported="false">
            <intent-filter>
            </intent-filter>
        </service>
```

Replace `com.altcraft.fcm.FCMService` with your own package and class name.

Huawei Mobile Services

Huawei Mobile Services Project Setup

Create an [AppGallery Connect] project. Go to Development and Services —> Add project:
Add a new app to the project:
Provide your app details:
Download the `agconnect-services.json` file:
Return to the project settings and add the SHA-256 certificate fingerprint. You can obtain it by running `./`
In the left sidebar, select Push Kit under Growth:
Click Enable —> Select data storage location:

Configuring the Altcraft Resource

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existing
Scroll down to the Huawei settings and specify the App ID, OAuth 2.0 client ID, and OAuth 2.0 client secre
After making changes, don't forget to save the resource.

Integrating Huawei Push Kit into the App Project

Move the downloaded `agconnect-services.json` file to the root directory of your app module (app):
Add repositories and dependencies to your Gradle files:

`settings.gradle.kts (project settings)`

```
dependencyResolutionManagement {
    repositories {
        // Other repositories..
        maven("https://developer.huawei.com/repo/")
    }
}
```

`build.gradle.kts (project level)`

```
buildscript {
    repositories {
        maven("https://developer.huawei.com/repo/")
    }
    dependencies {
        classpath("com.android.tools.build:gradle:8.10.1") // Important: use the AGP version your project uses.
        classpath("com.huawei.agconnect:agcp:1.9.1.301")
    }
}
```

`build.gradle.kts (app level)`

```
plugins {
    id("com.huawei.agconnect")
}

dependencies {
    implementation("com.huawei.hms:push:6.11.0.300")
    implementation("com.huawei.agconnect:agconnect-core:1.7.3.302")
}
```

> TIP: Note
>
> **Information about the latest dependency versions is available on Huawei Developer pages:**
>
> - **[AppGallery Connect]**
> - **[HMS Push Core]**

After adding the repository and dependencies, sync your Gradle changes.

Creating a Class Extending `HmsMessageService()`

Create a class that extends `HmsMessageService()`:

```
/**
 * HMS service for handling push tokens and incoming notifications.
 *
 * Extends [HmsMessageService] and overrides key HMS callback methods.
 */
class HMSService : HmsMessageService() {
    /**
     * Called when a new HMS token is generated.
     *
     * @param token The new HMS token.
     */
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received from HMS.
     *
     * Forwards the message with additional metadata to all receivers.
     *
     * @param message The received [RemoteMessage].
     */
    override fun onMessageReceived(message: RemoteMessage) {
        // Message
    }
}
```

Updating the App Manifest

Add the `HMSService` declaration to your app's `AndroidManifest.xml`:

```
<service
    android:name="com.altcraft.hms.HMSService"
    android:exported="false">
    <intent-filter>
    </intent-filter>
</service>
```

Replace `com.altcraft.hms.HMSService` with your own package and class name.

RuStore

> INFO: Note
>
> **The RuStore app must be installed on the device.**

RuStore Developer Console Project Setup

Go to the [RuStore Developer Console]:
Add an app under the Application section:
Go to the Tools section in the developer console:
Select your created app and open Push Notifications:
Create a new project:
In the project setup, specify:

- **Project name**
- **App package name**
- **SHA-256 signature fingerprint (you can get it by running the `./gradlew signingReport` command in the roo**

Inside the created project, generate a service token:

Configuring the Altcraft Resource

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existing
Scroll down to the RuStore settings and specify the Project ID and Password:
After making changes, don't forget to save the resource.

Integrating RuStore Push Messaging into the App Project

Add repositories and dependencies to your Gradle files:

`settings.gradle.kts (project settings)`

```
dependencyResolutionManagement {
    repositories {
        // Other repositories..
        maven("https://artifactory-external.vkpartner.ru/artifactory/maven")
    }
}
```

`build.gradle.kts (app level)`

```
dependencies {
    implementation("ru.rustore.sdk:pushclient:6.10.0")
}
```

> TIP: Note
>
> **Information about the latest dependency versions is available here.**

After adding the repository and dependencies, sync your Gradle changes.

Initializing `RuStorePushClient`

Initialize `RuStorePushClient` in `Application.onCreate()`:

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()
        RuStorePushClient.init(this, "project_ID")
    }
}
```

Creating a Class Extending `RuStoreMessagingService()`

Create a class that extends `RuStoreMessagingService()`:

```
/**
 * RuStore service for handling push notifications.
 *
 * Extends [RuStoreMessagingService] and overrides key callbacks.
 */
class RuStoreService : RuStoreMessagingService() {
    override fun onNewToken(token: String) {
        super.onNewToken(token)
    }

    /**
     * Called when a push message is received.
     *
     * Forwards the message to all receivers with added metadata.
     *
     * @param message The received RemoteMessage.
     */
    override fun onMessageReceived(message: RemoteMessage) {
        // Message
    }
}
```

Updating the App Manifest

Add the `RuStoreService` declaration to your app's `AndroidManifest.xml`:

```
<service
    android:name="com.altcraft.rustore.RuStoreService"
    android:exported="true"
    tools:ignore="ExportedService">
    <intent-filter>
    </intent-filter>
</service>
```

Replace `com.altcraft.rustore.RuStoreService` with your package and class name. Note that the manifest must i

> **TIP: Conditions for Proper RuStore Notification Functionality**
>
> - **The RuStore app is installed on the device.**
> - **The RuStore app has notification permissions.**
> - **The RuStore app is excluded from battery optimization.**

Quick Start

This article provides a rapid integration example of Altcraft mSDK with Apple Push Notification System (A
To use Huawei Mobile Services, add the corresponding SDKs to the project and implement their interface

Step 0. Prerequisites

- **APNS access obtained on the website;**
- **The app is configured in a Firebase project, and `GoogleServiceInfo.plist` is downloaded.**

Step 1. Altcraft SDK integration into the app

Add packages from repositories using Swift Package Manager:

- **Repository Altcraft SDK;**
- **Repository Firebase.**

Step 2. Add `GoogleServiceInfo` to the app

Add `GoogleServiceInfo.plist` to your app folder:

Step 3. App preparation

Configure the following app target settings:

- **General:**
  - **Ensure the Altcraft library is added under Frameworks, Libraries, and Embedded Content of the a**

```
* Ensure the **FirebaseMessaging** library is added under **Frameworks, Libraries, and Embedded Content** of the app target:
```

- **Signing & Capabilities:**
  - **PushNotifications;**
  - **AppGroups — specify a group identifier (adding an App Group identifier is required for data excha**
  - **Background Modes — enable Background fetch, Remote notification, Background processing.**
- **Info:**
  - **Add the following keys and values:**

Key
Permitted background task scheduler identifiers
FirebaseAppDelegateProxyEnabled

The first key registers a `bgTask` that retries failed server requests in background mode. The second key di

Step 4. JWTInterface realization

Implement the SDK protocol for providing a JWT token. Create a class that implements `JWTInterface` and o

```
class JWTProvider: JWTInterface {
    func getToken() -> String? {
        // your code returning JWT
      }
}
```

> **TIP: Note**
>
> **`getJWT()` is a synchronous function. The SDK execution thread will be paused until JWT is obtained. It's r**
> **start) and store it in cache so the token is available without delays. Returning `nil` is acceptable if no va**

Step 5. FCMInterface realization

Implement the SDK protocol for providing and deleting the FCM token. Create a class that implements `FCM`

```
class FCMProvider: FCMInterface {

    /// Returns the current FCM token
    func getToken(completion: @escaping (String?) -> Void) {

        /// APNs token retrieved from UserDefaults
        let apnsToken = getAPNsTokenDataFromUserDefaults()

        /// set the APNs token for FCM before requesting the FCM token
        Messaging.messaging().apnsToken = apnsToken

        /// request the FCM token
        Messaging.messaging().token { token, error in
            if error != nil {
                completion(nil)
            } else {
                completion(token)
            }
        }
    }

    /// Deletes the current FCM token
    func deleteToken(completion: @escaping (Bool) -> Void) {
        Messaging.messaging().deleteToken { error in
            if error != nil {
                completion(false)
            } else {
                completion(true)
            }
        }
    }
}
```

Step 6. APNSInterface realization

Implement the SDK protocol for providing the APNS token. Create a class that implements `APNSInterface` ar

```
class APNSProvider: APNSInterface {
    /// Retrieves the current APNs token from local storage
    func getToken(completion: @escaping (String?) -> Void) {
        let token = getAPNsTokenFromUserDefault()  // pass the APNs token
        completion(token)
    }
}
```

Step 7. Configure Altcraft SDK and Firebase SDK in `AppDelegate`

Altcraft SDK:

```
- pass the `AppGroups` identifier to Altcraft SDK;
- register Altcraft `backgroundTasks`;
- set JWT, APNS, and FCM providers;
- register with the notification center.
```

Firebase SDK:

- **apply the configuration;**

Example of correct Altcraft SDK and Firebase SDK setup in `AppDelegate`

```
class AppDelegate: UIResponder, UIApplicationDelegate,  UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        // Firebase SDK configuration
        FirebaseApp.configure()

        // Altcraft SDK
        /// set App Group identifier
        AltcraftSDK.shared.setAppGroup(groupName: "group.your_company.name")
        /// register BGTask
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
        /// set JWTProvider
        AltcraftSDK.shared.setJWTProvider(provider: JWTProvider())
        /// set APNSProvider
        AltcraftSDK.shared.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())
        /// set FCMProvider
        AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
        /// register with notification center
        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        return true
    }

    func application(
        _ application: UIApplication,
        didRegisterForRemoteNotificationsWithDeviceToken apnsToken: Data
    ) {
        // save APNs token in UserDefaults
        setAPNsTokenInUserDefault(apnsToken)
    }
}
```

- **set the `delegate`;**
- **pass the APNS token to Firebase in `AppDelegate.application(_:didFinishLaunchingWithOptions:)`. This ensures c use the saved value (e.g., in `UserDefaults`).**

Step 8. Create a Notification Service Extension

Create a Notification Service Extension:

- **File — New — Target — Notification Service Extension;**
- **Choose a Product Name for the extension target;**
- **Activate it.**

In the Notification Service Extension:

- **General:**
  - **Set Minimum Deployments — the Xcode build setting that defines the minimum OS version suppor**
  - **Add the Altcraft library under Frameworks, Libraries and Embedded Content.**
- **Signing & Capabilities:**
  - **AppGroups — specify the `AppGroup` identifier.**

Then configure the SDK inside your `UNNotificationServiceExtension`. Replace the entire automatically generat

Example implementation of `UNNotificationServiceExtension`

```
class NotificationService: UNNotificationServiceExtension {

    /// - important! Set your App Group identifier.
    let appGroupID = "group.your.id"

    /// - important! Set the provider to JWT if you are using JWT authentication.
    let jwtProvider = JWTProvider()

    override func didReceive(
        _ request: UNNotificationRequest,
        withContentHandler contentHandler: @escaping (UNNotificationContent) -> Void
    ) {

        AltcraftNSE.shared.setAppGroup(groupName: appGroupID)
        AltcraftNSE.shared.setJWTProvider(provider: jwtProvider)

        contentHandler(request.content)

        if AltcraftNSE.shared.isAltcraftPush(request) {
            AltcraftNSE.shared.handleNotificationRequest(
                request: request,
                contentHandler: contentHandler
            )
        } else {
            contentHandler(request.content)
        }
    }

    override func serviceExtensionTimeWillExpire() {
        AltcraftNSE.shared.serviceExtensionTimeWillExpire()
    }
}
```

TIP: Note

If you are not using JWT authentication, remove the call to `AltcraftSDK.shared.setJWTProvider(provider: jwtProvi`

Step 9. SDK initialization

Create a configuration variable and initialize using `initialization()`, choosing `ios-firebase` as the preferred

```
let config = AltcraftConfiguration.Builder()
.setApiUrl("your_api_url")
.setProviderPriorityList([Constants.ProviderName.firebase])
.build()

AltcraftSDK.shared.initialization(configuration: config)
```

In this example, `ios-firebase` is set as the preferred provider. You can also set APNS as the preferred provi

```
let config = AltcraftConfiguration.Builder()
.setApiUrl("your_api_url")
.setProviderPriorityList([Constants.ProviderName.apns])
.build()

AltcraftSDK.shared.initialization(configuration: config)
```

You may omit `setProviderPriorityList`; the default priority will be used — `ios-apns -> ios-firebase -> ios-huawei`:

```
let config = AltcraftConfiguration.Builder()
.setApiUrl("your_api_url")
.build()

AltcraftSDK.shared.initialization(configuration: config)
```

Step 10. Subscribe to push notifications

Subscribe to push notifications using `pushSubscribe()` and pass the required parameters:

```
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
 profileFields: ["_fname": "user_first_name", "_lname": "user_last_name"]
)
```

SDK Functionality

> **TIP**
>
> **Before proceeding, you need to configure the SDK for your application. The detailed setup guide is availa**

Working with subscription statuses

Changing subscription status

```
AltcraftSDK
//   Singleton, the entry point to the SDK
└── static let shared: AltcraftSDK
    // Push subscription managment
    └── pushSubscriptionFunctions: PublicPushSubscriptionFunctions
        // Subscribe (status = SUBSCRIBED)
        ├── pushSubscribe(
        │       sync: Bool = true,
        │       profileFields: [String: Any?]? = nil,
        │       customFields: [String: Any?]? = nil,
        │       cats: [CategoryData]? = nil,
        │       replace: Bool? = nil,
        │       skipTriggers: Bool? = nil
        │   ): Void
        // Unsubscribe (status = UNSUBSCRIBED)
        ├── pushUnSubscribe(
        │       sync: Bool = true,
        │       profileFields: [String: Any?]? = nil,
        │       customFields: [String: Any?]? = nil,
        │       cats: [CategoryData]? = nil,
        │       replace: Bool? = nil,
        │       skipTriggers: Bool? = nil
        │   ): Void
        // Suspend (status = SUSPENDED)
        └── pushSuspend(
                sync: Bool = true,
                profileFields: [String: Any?]? = nil,
                customFields: [String: Any?]? = nil,
                cats: [CategoryData]? = nil,
                replace: Bool? = nil,
                skipTriggers: Bool? = nil
            ): Void
```

Subscription status change functions:

- `func pushSubscribe()` **— subscribes to push notifications;**
- `func pushUnSubscribe()` **— unsubscribes from push notifications;**
- `func pushSuspend()` **— suspends the push subscription (no notifications are delivered, but no unsubscribe**
- `func unSuspendPushSubscription()` **— used to implement** `LogIn` **/** `LogOut` **transitions.**

All functions in this group have the same signature and parameters:

---

sync: `Bool`
Default: `true`
Required: No
Description: A flag that sets whether the request is executed synchronously.

Successful request execution:

On success, these functions produce an event with code 230, 231 or 232, containing an `event.value` **define**

If `sync == true`

```
ResponseWithHttpCode
  ├ httpCode: 200
  └ response
     ├ error: 0
     ├ errorText: ""
     └ profile
        ├ id: "your id"
        ├ status: "subscribed"
        ├ isTest: false
        └ subscription
           ├ subscriptionId: "your subscriptionId"
           ├ hashId: "c52b28d2"
           ├ provider: "ios-apns"
           ├ status: "subscribed"
           ├ fields
           │  ├ _os_ver: {"raw":"18.6.2","ver":"[\"18.0\", \"6.0\", \"2.0\"]"}
           │  ├ _device_type: "mob"
           │  ├ _ad_track: false
           │  ├ _device_name: "iPhone"
           │  ├ _os_language: "en"
           │  ├ _os_tz: "+0300"
           │  ├ _os: "IOS"
           │  └ _device_model: "iPhone14,7"
           └ cats
              └ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

For synchronous requests, the `event.value["response_with_http_code"]` contains:

- **httpCode – transport-level status code;**
- **Response – a public struct containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData?` **— profile data if the request is successful:**
    - **profile info (ProfileData)**
    - **subscription (SubscriptionData)**
    - **subscription categories (CategoryData)**
    - **if the request fails,** `profile = nil`

▶ **Data structures**

If `sync == false`

```
ResponseWithHttpCode
  ├ httpCode: Int?
  └ response: Response?
     ├ error: Int?
     ├ errorText: String?
     └ profile: ProfileData? = nil
```

For asynchronous requests, `event.value["response_with_http_code"]` **contains:**

- **httpCode – transport-level status code;**
- **Response – a public struct containing:**
  - `error: Int?` **— internal server error code *(0 if no errors)*,**
  - `errorText: String?` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData?` **— profile data, always** `nil` **for async requests.**

Failed request execution:

If a request in this group fails, an event with one of the following codes is created:

Operations without automatic retry on the SDK side:

- **430 – notification subscription;**
- **431 — subscription suspension;**
- **432 — unsubscription.**

Operations with automatic retry on the SDK side:

- **530 – notification subscription;**
- **531 — subscription suspension;**
- **532 — unsubscription.**

Event contents:

- **only** `httpCode` **if the Altcraft server was unavailable;**
- `error` **and** `errorText` **if the server returned an error.**

Retrieving event values

```
AltcraftSDK.shared.eventSDKFunctions.subscribe { event in
    let codes: Set<Int> = [
        230, 231, 232,
        430, 431, 432,
        531, 532, 533
    ]

    guard let code = event.eventCode, codes.contains(code) else { return }
    guard let responseWithHttp = event.value?["response_with_http_code"] as? ResponseWithHttp else { return }

    let httpCode = responseWithHttp.httpCode

    let response = responseWithHttp.response
    let error = response?.error
    let errorText = response?.errorText

    let profile = response?.profile
    let profileId = profile?.id
    let profileStatus = profile?.status
    let profileIsTest = profile?.isTest

    let subscription = profile?.subscription
    let subscriptionId = subscription?.subscriptionId
    let hashId = subscription?.hashId
    let provider = subscription?.provider
    let subscriptionStatus = subscription?.status

    let fields = subscription?.fields

    let cats = subscription?.cats
    let firstCat = cats?.first
    let catName = firstCat?.name
    let catTitle = firstCat?.title
    let catSteady = firstCat?.steady
    let catActive = firstCat?.active
}
```

The `fields` inside `subscription` may include your custom subscription fields. They are of type `[String: JSONVa`

```
public var stringValue: String? {
    if case let .string(value) = self { return value }
    return nil
}

/// Returns the numeric value if case is `.number`, otherwise `nil`.
public var numberValue: Double? {
    if case let .number(value) = self { return value }
    return nil
}

/// Returns the bool value if case is `.bool`, otherwise `nil`.
public var boolValue: Bool? {
    if case let .bool(value) = self { return value }
    return nil
}

/// Returns the object dictionary if case is `.object`, otherwise `nil`.
public var objectValue: [String: JSONValue]? {
    if case let .object(value) = self { return value }
    return nil
}

/// Returns the array if case is `.array`, otherwise `nil`.
public var arrayValue: [JSONValue]? {
    if case let .array(value) = self { return value }
    return nil
}
```

Example of extracting the field `"_device_name"`:

```
AltcraftSDK.shared.eventSDKFunctions.subscribe { event ->
    if event.eventCode == 230 {
        if let responseWithHttp = event.value?["response_with_http_code"] as? ResponseWithHttp {
            let response = responseWithHttp.response
            let profile = response?.profile
            let subscription = profile?.subscription

            // extract fields?["_device_name"] as a string value
            if let deviceName = subscription?.fields?["_device_name"]?.stringValue {
                print(deviceName)
            }
        }
    }
}
```

---

profileFields: `[String: Any?]?`
Default: `nil`
Required: No
Description: A dictionary containing profile fields.
The parameter may include system fields (e.g., `_fname` — first name, `_lname` — last name) and optional fiel

- **String**
- **Bool**
- **Numbers**
- `nil`
- **Objects** `[String: Any]`
- **Arrays**

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: with field "field_name": Incorrect field
```

---

customFields: `[String: Any?]?`
Default: `nil`
Required: No
Description: A dictionary containing subscription fields.
The parameter may include system fields (e.g., `_device_model` — device model, `_os` — OS), as well as option

- **String**
- **Bool**
- **Int**
- **Float**
- **Double**
- `nil`

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "field_name" is not valid: failed convert custom field
```

Nested objects, arrays, and collections are not allowed.

> TIP: Note
> **Most system subscription fields are automatically collected by the SDK and added to push requests. The**

---

cats: `[CategoryData]?`
Default: `nil`
Required: No
Description: Subscription categories.
Category structure:

```
public struct CategoryData: Codable {
    public var name: String?
    public var title: String?
    public var steady: Bool?
    public var active: Bool?
}
```

When sending a push request with categories, use only `name` (category name) and `active` (category active
Request example:

```
let cats: [CategoryData] = [CategoryData(name: "football", title: nil, steady: nil, active: true)]
```

Response example:

```
[ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

Categories used in the request must be created beforehand and added to the resource in the Altcraft plat

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "subscriptions.cats" is not valid: category not found in resource
```

---

replace: `Bool?`

Default: `nil`

Required: No

Description: When enabled, all subscriptions of other profiles with the same push token in the current dat

---

skipTriggers: `Bool?`

Default: `nil`

Required: No

Description: When enabled, the profile containing this subscription will be ignored by mailing and scenari

---

Request examples

Example: subscribing to push notifications

Minimal working call:

```
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe()
```

Passing all available parameters:

```
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    sync: true,
    profileFields: ["_fname":"Andrey", "_lname":"Pogodin"],
    customFields:  ["developer":true],
    cats: [CategoryData(name: "developer_news", active: true)],
    replace: false,
    skipTriggers:  false
)
```

> TIP: Note
>
> **For `pushSubscribe`, `pushSuspend`, and `pushUnSubscribe`, the SDK automatically retries the request if the HTTP s**

Function `unSuspendPushSubscription()`

`func unSuspendPushSubscription()` is designed for `LogIn` / `LogOut` transitions. It works as follows:

- **searches for subscriptions with the same push token as the current one that do not belong to the prof**
- **changes the status of the found subscriptions from `subscribed` to `suspended`;**
- **changes the status of the subscriptions in the profile referenced by the current JWT from `suspended` to**
- **returns a `ResponseWithHttpCode?` where `response.profile` is the current profile referenced by the JWT (if th**

Example:

```
AltcraftSDK.shared.pushSubscriptionFunctions.unSuspendPushSubscription(completion: @escaping (ResponseWithHttp?) -> Void)
```

Recommended `LogIn` / `LogOut` flow

`LogIn` flow:

- **An anonymous user opens the app. The user has `JWT_1`, pointing to database #1Anonymous;**
- **The user subscribes to push; a profile is created in #1Anonymous;**
- **The user registers and receives `JWT_2`, pointing to #2Registered;**
- **Call `unSuspendPushSubscription()` — the anonymous subscription in #1Anonymous is suspended;**
- **The SDK searches for a profile in #2Registered to restore the subscription;**
- **Since a subscription with this push token does not exist in #2Registered, `unSuspendPushSubscription()` ret**
- **After receiving `nil`, call `pushSubscribe()` to create a new profile in #2Registered.**

`LogOut` flow:

- **The user logs out in the app (`LogOut`);**
- **The user receives `JTW_1`, pointing to #1Anonymous;**
- **Call `unSuspendPushSubscription()`, which suspends the subscription in #2Registered and sets the status in**
- **The request returns `#1Anonymous != nil` — the subscription already exists, no new one is required.**

Example implementation:

```
func logIn() {
    JWTManager.shared.setRegJWT()
    // JWT is set for an authorized user
    AltcraftSDK.shared.pushSubscriptionFunctions.unSuspendPushSubscription { result in
        if result?.httpCode == 200, result?.response?.profile?.subscription == nil {
            AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
                // specify the required parameters
            )
        }
    }
}

func logOut() {
    JWTManager.shared.setAnonJWT()
    // JWT is set for an anonymous user
    AltcraftSDK.shared.pushSubscriptionFunctions.unSuspendPushSubscription { result in
        if result?.httpCode == 200, result?.response?.profile?.subscription == nil {
            AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
                // specify the required parameters
            )
        }
    }
}
```

Requesting subscription status

```
AltcraftSDK
//   Singleton, the entry point to the SDK
└── static let shared: AltcraftSDK
    // Subscription managment
    └── pushSubscriptionFunctions: PublicPushSubscriptionFunctions
        // Last profile subscription status
        ├── getStatusOfLatestSubscription(
        │       completion: @escaping (ResponseWithHttp?) -> Void
        │   ): Void
        // Subscription status corresponding to the current token/provider
        ├── getStatusForCurrentSubscription(
        │       completion: @escaping (ResponseWithHttp?) -> Void
        │   ): Void
        // Status of the latest subscription for the provider (if nil – the current one is used)
        └── getStatusOfLatestSubscriptionForProvider(
                provider: String? = nil,
                completion: @escaping (ResponseWithHttp?) -> Void
            ): Void
```

Subscription status functions:

- `func getStatusOfLatestSubscription()` — the last subscription status in the profile;
- `func getStatusOfLatestSubscriptionForProvider()` — the subscription status for the current token/provider;
- `func getStatusForCurrentSubscription()` — the latest subscription status by provider. If `nil` is specified, the

---

`public func getStatusOfLatestSubscription(completion: @escaping (ResponseWithHttp?) -> Void): Void`

Retrieves the status of the profile's latest subscription. The `completion` receives a `ResponseWithHttp?` contain

Example:

```
// Last profile subscription status
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusOfLatestSubscription(completion: @escaping (ResponseWithHttp?) -> Void)
```

`public func getStatusForCurrentSubscription(completion: @escaping (ResponseWithHttp?) -> Void): Void`

Retrieves the subscription status for the current token/provider. The `completion` receives a `ResponseWithHttp?`

Example:

```
// Subscription status corresponding to the current token/provider
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusForCurrentSubscription(completion: @escaping (ResponseWithHttp?) -> Void)
```

`public func getStatusOfLatestSubscriptionForProvider(provider: String? = nil, completion: @escaping (ResponseWithHttp?) ->`

Retrieves the latest subscription status by provider. The `completion` receives a `ResponseWithHttp?` containing `nil` is passed.

Example:

```
// Status of the latest subscription for the provider (if nil – the current one is used)
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusOfLatestSubscriptionForProvider(provider: String? = nil, completion: @escaping (ResponseW
```

Below is an example of extracting profile, subscription, and category data from the status response. This

Data from status-retrieval functions

```
AltcraftSDK.shared.pushSubscriptionFunctions.getStatusForCurrentSubscription{ responseWithHttp in
    // HTTP code
    let httpCode = responseWithHttp?.httpCode

    // Response
    let response = responseWithHttp?.response
    let error = response?.error
    let errorText = response?.errorText

    // Profile
    let profile = response?.profile
    let profileId = profile?.id
    let profileStatus = profile?.status
    let profileIsTest = profile?.isTest

    // Subscription
    let subscription = profile?.subscription
    let subscriptionId = subscription?.subscriptionId
    let hashId = subscription?.hashId
    let provider = subscription?.provider
    let subscriptionStatus = subscription?.status

    // Fields (dictionary [String: JSONValue])
    let fields = subscription?.fields

    // Cats (array of CategoryData)
    let cats = subscription?.cats

    // CategoryData (every element of cats array)
    let firstCat = cats?.first
    let catName = firstCat?.name
    let catTitle = firstCat?.title
    let catSteady = firstCat?.steady
    let catActive = firstCat?.active
}
```

On a successful request, an event with code 234 is created. On error — an event with code 434.

Provider push token management

```
AltcraftSDK
//   Singleton, the entry point to the SDK
└── static let shared: AltcraftSDK
    // Managing push tokens and providers (FCM / HMS / APNs)
    └── pushTokenFunctions: PublicPushTokenFunctions
        // Manually save the token (String for FCM/HMS, Data for APNs)
        ├── setPushToken(
        │       provider: String,
        │       pushToken: Any?
        │   ): Void
        // Asynchronously retrieve the current push token (completion is optional)
        ├── getPushToken(
        │       completion: ((TokenData?) -> Void)? = nil
        │   ): Void
        // Set the Firebase Cloud Messaging provider (nil — remove)
        ├── setFCMTokenProvider(
        │       _ provider: FCMInterface?
        │   ): Void
        // Set the Huawei Mobile Services provider (nil — remove)
        ├── setHMSTokenProvider(
        │       _ provider: HMSInterface?
        │   ): Void
        // Set the Apple Push Notification service provider (nil — remove)
        ├── setAPNSTokenProvider(
        │       _ provider: APNSInterface?
        │   ): Void
        // Apply a new provider priority list and initiate token update
        ├── changePushProviderPriorityList(
        │       _ list: [String]
        │   ): Void
        // Delete the push token of the specified provider and call completion
        ├── deleteDeviceToken(
        │       provider: String,
        │       completion: @escaping () -> Void
        │   ): Void
        // Force token update
        └── forcedTokenUpdate(
                completion: (() -> Void)? = nil
            ): Void
```

SDK functions for working with a provider's token:

- `func setPushToken()` — **manually sets the device push token and provider in** `UserDefaults`**;**
- `func getPushToken()` — **retrieves the current push token;**
- `func setAPNSTokenProvider()` — **sets or removes the APNs push-token provider;**
- `func setFCMTokenProvider()` — **sets or removes the FCM push-token provider;**
- `func setHMSTokenProvider()` — **sets or removes the HMS push-token provider;**
- `func changePushProviderPriorityList()` — **dynamically switches the provider and updates the subscription's**
- `func deleteDeviceToken()` — **deletes the push token for the specified provider;**
- `func forcedTokenUpdate()` — **deletes the current push token and then refreshes it.**

`public func setPushToken(provider: String, pushToken: Any?): Void`

Used to manually set the device push token and provider in `UserDefaults`**. This serves as a simplified way t**
Not recommended for passing the token. The preferred approach is to implement `FCMInterface`**,** `HMSInterface`
Example:

```
// Manually save the token (String for FCM/HMS, Data for APNs)
AltcraftSDK.shared.pushTokenFunctions.setPushToken(provider: String, pushToken: Any?)
```

`public func getPushToken(completion: ((TokenData?) -> Void)? = nil): Void`

Passes a `TokenData(provider: String, token: String)` **object to** `completion`**, containing the current device push to**
Example:

```
// Asynchronously retrieve the current push token (completion is optional)
AltcraftSDK.shared.pushTokenFunctions.getPushToken(completion: ((TokenData?) -> Void)? = nil)
```

Example of requesting push-token data:

```
AltcraftSDK.shared.pushTokenFunctions.getPushToken { data in
    let provider = data?.provider
    let token = data?.token
}
```

`public func setFCMTokenProvider(_ provider: FCMInterface?): Void`

Sets or removes the Firebase Cloud Messaging provider. Pass `nil` **to disable the provider.**
Example:

```
// Set the Firebase Cloud Messaging provider (nil – remove)
AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(_ provider: FCMInterface?)
```

> TIP: Important
>
> **Call** `setFCMTokenProvider()` **in** `AppDelegate.application(_:didFinishLaunchingWithOptions:)` **before** `AltcraftSDK.initializa`

`public func setHMSTokenProvider(_ provider: HMSInterface?): Void`

Sets or removes the Huawei Mobile Services provider. Pass `nil` **to disable the provider.**

```
// Set the Huawei Mobile Services provider (nil – remove)
AltcraftSDK.shared.pushTokenFunctions.setHMSTokenProvider(_ provider: HMSInterface?)
```

> TIP: Important
>
> **Call** `setHMSTokenProvider()` **in** `AppDelegate.application(_:didFinishLaunchingWithOptions:)` **before** `AltcraftSDK.initializa`

`public func setAPNSTokenProvider(_ provider: APNSInterface?): Void`

Sets or removes the Apple Push Notification service provider. Pass `nil` **to disable the provider.**

```
// Set the Apple Push Notification service provider (nil – remove)
AltcraftSDK.shared.pushTokenFunctions.setHMSTokenProvider(_ provider: APNSInterface?)
```

> TIP: Important
>
> **Call** `setAPNSTokenProvider()` **in** `AppDelegate.application(_:didFinishLaunchingWithOptions:)` **before** `AltcraftSDK.initializ`

`public func changePushProviderPriorityList(_ list: [String]): Void`

Enables dynamic switching of the push provider with token refresh. Pass a new array with a different prov
Example:

```
// Apply a new provider priority list and initiate token update
AltcraftSDK.shared.pushTokenFunctions.changePushProviderPriorityList(_ list: [String])
```

`public func deleteDeviceToken(provider: String, completion: @escaping () -> Void): Void`

Deletes the push token for the specified provider. The token is invalidated and removed from the local cac
Example:

```
// Delete the push token of the specified provider (invalidated locally and on the server)
AltcraftSDK.shared.pushTokenFunctions.deleteDeviceToken(provider: String, completion: @escaping () -> Void)
```

`public func forcedTokenUpdate(completion: (() -> Void)? = nil): Void`

Deletes the current push token and then refreshes it.

Example:

```
// Force token update (determine provider -> delete (except APNs) -> request new -> update)
AltcraftSDK.shared.pushTokenFunctions.forcedTokenUpdate(completion: (() -> Void)? = nil)
```

Example of registering providers

We do not recommend using `setPushToken` to set the push token. Instead, configure token retrieval for each

Recommended provider registration in `AppDelegate`

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        let appGroup = "your appGroup id"

        AltcraftSDK.shared.setAppGroup(groupName: appGroup)
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
        AltcraftSDK.shared.setJWTProvider(provider: JWTProvider())

        // apns provider
        AltcraftSDK.shared.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())

        // fcm provider
        AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(FCMProvider())

        // hms provider
        AltcraftSDK.shared.pushTokenFunctions.setHMSTokenProvider(HMSProvider())

        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        let config = AltcraftConfiguration.Builder().setApiUrl("your api url").build()
        AltcraftSDK.shared.initialization(configuration: config)

        // other functions

        return true
    }
}
```

Managing Push Notifications

Push Notification Initialization

To initialize push notifications, call the following method in `application(_:didFinishLaunchingWithOptions:)`:

`AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)`

This method assigns `UNUserNotificationCenter.current().delegate` to the Altcraft SDK `NotificationManager`, reques

> TIP: Important
>
> **The system permission dialog is shown only on the first request (or if the status has not yet been determ**

Passing Push Notifications to the SDK

The SDK includes the `AltcraftNSE` class, intended for use inside a Notification Service Extension. It provide

```
// Used inside Notification Service Extension (NSE) — facade for handling Altcraft push notifications,
// detecting Altcraft-specific requests, and registering mobile events without direct dependency on UIApplication
AltcraftNSE
// Shared singleton instance for centralized NSE logic
├─ public static let shared: AltcraftNSE
// Provides access to mobile event registration API
├─ public let mobileEventFunctions: PublicMobileEventFunctions
// Check whether the notification request belongs to Altcraft
├─ public func isAltcraftPush(_ request: UNNotificationRequest) -> Bool
// Handle an incoming Altcraft notification request and deliver the modified content
├─ public func handleNotificationRequest(
│     request: UNNotificationRequest,
│     contentHandler: @escaping (UNNotificationContent) -> Void
│  ): Void
// Called when NSE is about to time out — returns the best available notification content
├─ public func serviceExtensionTimeWillExpire(): Void
// Set the App Group identifier and initialize Core Data in the shared container
├─ public func setAppGroup(groupName: String): Void
// Register a JWT provider for secure token management in Altcraft SDK
└─ public func setJWTProvider(provider: JWTInterface): Void
```

`AltcraftNSE` **class methods:**

- `func isAltcraftPush()` **— checks the notification source;**
- `func handleNotificationRequest()` **— accepts a** `UNNotificationRequest` **from the Notification Service Extension**
- `func serviceExtensionTimeWillExpire()` **— called when the Notification Service Extension execution time ex**

`public func isAltcraftPush(_ request: UNNotificationRequest) -> Bool`

An SDK method that checks whether the notification source is Altcraft based on a marker in `userInfo`**.**

Usage example:

```
AltcraftNSE.shared.isAltcraftPush(_ request: UNNotificationRequest) -> Bool
```

`public func handleNotificationRequest(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNo`

An SDK method that accepts a `UNNotificationRequest` **from the Notification Service Extension, processes it i**

Usage example:

```
AltcraftNSE.shared.handleNotificationRequest(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotificationCont
```

`public func serviceExtensionTimeWillExpire()`

A method called by the system when the Notification Service Extension execution time expires (~30 seco

Usage example:

```
AltcraftNSE.shared.serviceExtensionTimeWillExpire()
```

`public func setAppGroup(groupName: String): Void`

A method for setting the `AppGroup` **identifier (required).**

Usage example:

```
AltcraftNSE.shared.setAppGroup(groupName: "group.your.app.id")
```

`public func setJWTProvider(provider: JWTInterface): Void`

A method for setting a JWT provider:

```
AltcraftNSE.shared.setJWTProvider(provider: jwtProvider)
```

Receiving Foreground Notifications in the App (Optional)

By default, this functionality does not require configuration. If push notification handling is delegated to tl

- **is set as** `UNUserNotificationCenterDelegate`**,**
- **processes incoming notifications,**
- **displays notifications in the foreground using standard parameters (banner/alert + badge/sound).**

The setup described below is required only if:

- **access to notification content in the foreground is needed (for example, analytics or a custom UI),**
- **or you want to manually control notification presentation (show/hide, change presentation options).**

To configure foreground push notification behavior, use `NotificationManager`**:**

```
// Access to the SDK push notification manager
let manager = AltcraftSDK.shared.notificationManager

// Control of foreground push notification handling.
// true  — handling and presentation choice are delegated to the app.
// false — the SDK uses the standard notification presentation logic.
manager.customPushProcessing = false

// Callback for foreground push notification handling.
// Called when a notification is received while the app is active.
// completion must be called only when customPushProcessing = true.
manager.onForegroundNotification = { notification, complete in
    // Custom notification handling logic

    // Important: call complete(...) only if manager.customPushProcessing = true
    if #available(iOS 14.0, *) {
        complete([.banner, .badge, .sound])
    } else {
        complete([.alert, .badge, .sound])
    }

    // or [] to suppress presentation
}
```

If `customPushProcessing = false`, the SDK applies standard notification presentation parameters, and the callb

Handling Notification Click Events in the App (Optional)

By default, this functionality does not require additional configuration. If push notification handling is dele

- **handles notification taps or actions,**
- **sends a push event of type `open` to Altcraft,**
- **executes the standard notification action handling logic.**

The setup described below is required only if:

- **the app needs access to notification data on click,**
- **custom navigation or action handling logic is required,**
- **or notification click handling must be fully intercepted by the app.**

To delegate click handling to the app, use `NotificationManager`:

```
let manager = AltcraftSDK.shared.notificationManager

// Control of notification click handling.
// true  — click handling is fully delegated to the app.
// false — the SDK uses the standard click handling logic.
manager.customClickProcessing = true

// Notification click handling callback.
// Called when the user taps a notification or performs an action.
// completion must be called when customClickProcessing = true.
manager.onNotificationClick = { response, complete in
    // Notification data
    let userInfo = response.notification.request.content.userInfo

    // Custom click handling logic
    print("push click")

    // Complete handling
    complete()
}
```

If `customClickProcessing = false`, the SDK handles notification clicks on its own, sends the corresponding eve

If `customClickProcessing = true`, the app must call `complete()`. Without calling `complete()`, iOS will treat the clic

Mobile events

```
// PublicMobileEventFunctions object

AltcraftSDK.shared
└── let mobileEventFunctions: PublicMobileEventFunctions

    // Send a mobile event to the server
    └── func mobileEvent(
            sid: String,
            eventName: String,
            sendMessageId: String? = nil,
            payload: [String: Any?]? = nil,
            matching: [String: Any?]? = nil,
            matchingType: String? = nil,
            profileFields: [String: Any?]? = nil,
            subscription: (any Subscription)? = nil
            utm: UTM? = nil
        )
```

Case: Transferring advertising campaign information

Information about the app advertising campaign that led to the installation can be sent to the platform as string, you need to call the `mobileEvent()` function:

Transfer via payload

```
AltcraftSDK.shared.mobileEventFunctions.mobileEvent(
    sid: "your sid",
    eventName: "app_install",
    payload: ["utm": "your utm tag"]
)
```

Transfer to UTM fields

```
AltcraftSDK.shared.mobileEventFunctions.mobileEvent(
    sid: "your sid",
    eventName: "app_install",
    utm: UTM(
        campaign: "your campaign utm",
        content: "your content utm",
        keyword: "your keyword utm",
        medium: "your medium utm",
        source: "your source utm",
        temp: "your temp utm"
    )
)
```

Transfer to a custom profile field

```
// transfer a UTM tag to a profile field
// profile fields must be created in the platform in advance
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    profileFields: ["utm": "your utm tag"]
)
```

Data sent using any of these methods can be used to create a segment in the platform.

Please note that when setting up on iOS, the application must use AppsFlyer, otherwise the UTM tag will

Use `mobileEvent()` from `PublicMobileEventFunctions` to register a mobile event. Parameters:

---

sid: `String`

Required: Yes

Description: String pixel identifier to which mobile events are linked.

---

eventName: `String`

Required: Yes
Description: Mobile event name.

---

sendMessageId: `String?`
Required: No
Description: SMID of the sent message (when the event is tied to a specific mailing).

---

payload: `[String: Any?]?`
Required: No
Description: Event data — a map with string keys; only scalar value types are allowed:

- `String`
- `Bool`
- `Int`
- `Int64` / `UInt64` **(or** `NSNumber` **equivalents)**
- `Float`
- `Double`
- `nil` / `NSNull`

> TIP: Note
> **Non-serializable objects (e.g.,** `Date` **without conversion, custom classes) will cause encoding errors.**

---

matching: `[String: Any?]?`
Required: No Description: A map for passing values with matching types/identifiers (e.g., `["email" : "user@ex`

---

matchingType: `String?`
Required: No
Description: Matching type.

---

profileFields: `[String: Any?]?`
Required: No
Description: Profile fields — a map with string keys and values (JSON-compatible types):

- **Scalar values:**
  - `String`
  - `Bool`
  - `Int`
  - `Int64` / `UInt64` **(or** `NSNumber` **equivalents)**
  - `Float`
  - `Double`
  - `nil` / `NSNull`
- **Objects:** `[String: Any?]`
- **Lists:** `[Any?]`
- **Array of maps:** `[[String: Any?]]`

> TIP: Note
> `profileFields` **is used only with JWT authorization.**

---

utm: `UTM?`
Required: No
Description: UTM tags. Added via the `UTM` **struct, where each UTM type is a property:**

```
public struct UTM{
    /// UTM Mailing
    public var campaign: String?
    /// UTM Content
    public var content: String?
    /// UTM Keyword/Term
    public var keyword: String?
    /// UTM Medium
    public var medium: String?
    /// UTM Source
    public var source: String?
    /// UTM Temp
}
```

---

subscription: `(any Subscription)?`

Required: No Description: A subscription to be added to the profile, one of: `EmailSubscription`, `SmsSubscription`

> **TIP: Note**
>
> **Used only with JWT authorization.**

`Subscription` **interface implementations**

Purpose — a common protocol for all subscription types. Serialization — via `Codable`**, discriminator field —**

Common fields (for all implementations):

Field	Type
`resource_id`	`Int`
`status`	`String?`
`priority`	`Int?`
`custom_fields`	`[String: JSONValue]?`
`cats`	`[String]?`
`channel`	`String`

> **TIP: Note**
>
> `JSONValue` **is a serializable enum/type of JSON values in your SDK. All values must be JSON-compatible.**

Subscription types:

EmailSubscription (`channel = "email"` **)**

Required fields

Field	Type
`resourceId`	`Int`
`email`	`String`

Optional fields

Field	Type
`status`	`String?`
`priority`	`Int?`
`customFields`	`[String: JSONValue]?`
`cats`	`[String]?`

SmsSubscription (`channel = "sms"` **)**

Required fields

Field	Type	
`resourceId`	`Int`	**Yes**
`phone`	`String`	**Yes**

Optional fields

Field	Type
`status`	`String?`
`priority`	`Int?`
`customFields`	`[String: JSONValue]?`
`cats`	`[String]?`

PushSubscription (`channel = "push"` **)**

Required fields

Field	Type	Required
`resourceId`	`Int`	**Yes**
`provider`	`String`	**Yes**
`subscriptionId`	`String`	**Yes**

Optional fields

Field	Type
`status`	`String?`
`priority`	`Int?`
`customFields`	`[String: JSONValue]?`
`cats`	`[String]?`

CcDataSubscription (`channel ∈ {"telegram_bot","whatsapp","viber","notify"}` **)**

Required fields

Field	Type	Required
`resourceId`	`Int`	**Yes**
`channel`	`String`	**Yes**
`ccData`	`[String: JSONValue]`	**Yes**

Optional fields

Field	Type
`status`	`String?`
`priority`	`Int?`
`customFields`	`[String: JSONValue]?`
`cats`	`[String]?`

Profile Fields Update

INFO: Important

Updating profile fields using the `updateProfileFields()` **function works only when using JWT authorization f**

```
AltcraftSDK.shared
└── let profileFunctions: PublicProfileFunctions
    // Updates Altcraft profile fields
    └── func updateProfileFields(
        profileFields: [String: Any?]? = nil,
        skipTriggers: Bool? = nil
    ): Void
```

To update profile fields, use the `updateProfileFields()` **function. It has the following parameters:**

---

profileFields: `[String: Any?]?`

Required: No

Description: Profile fields — a map containing the values for the fields that need to be updated (JSON-com

- **Scalar values:**
  - `String`
  - `Bool`
  - `Int`
  - `Int64` / `UInt64` **(or equivalent** `NSNumber` **)**
  - `Float`
  - `Double`
  - `nil` / `NSNull`
- **Objects:** `[String: Any?]`
- **Lists:** `[Any?]`
- **Array of maps:** `[[String: Any?]]`

INFO: Note

For the update to succeed, the fields must have been previously added to the subscriber's profile.

---

skipTriggers: `Bool?`

Default: `nil`

Required: No

Description: When enabled, the profile containing this subscription will be ignored in mailing triggers and

Receiving SDK events in the app

```
// Events object functions

AltcraftSDK
// Singleton, SDK entry point
└── static let shared: AltcraftSDK
    // SDK event stream (one active subscriber)
    └── eventSDKFunctions: SDKEvents
        // Subscribe to SDK events (replaces an existing subscriber)
        ├── subscribe(
        │       callback: @escaping (Event) -> Void
        │   ): Void
        // Unsubscribe from events (callback remains assigned, delivery stops)
        └── unsubscribe(): Void
```

Only one active subscriber to SDK events can exist in the app.

Subscribing to events

`public func subscribe(callback: @escaping (Event) -> Void): Void`

Subscribes to and receives SDK events in the app. When an SDK event occurs, it invokes `callback` and pas

Example:

```
AltcraftSDK.shared.eventSDKFunctions.subscribe { event in
  // handle event
}
```

All events provided by the SDK are instances of `Event` or its subclasses:

- **Event — general event (information, successful requests);**
- **Error — error event;**
- **RetryError — error event for a request that the SDK will automatically retry.**

Each event contains:

- **function — the name of the function that produced the event;**
- **eventCode — internal SDK event code (see “SDK events”);**
- **eventMessage — event message;**
- **eventValue — arbitrary payload `[String: Any?]?` added to some events;**
- **date — event timestamp.**

SDK events classes

```
open class Event: Hashable {
    public let id = UUID()
    public let function: String
    public let message: String?
    public let eventCode: Int?
    public let value: [String: Any?]?
    public let date: Date
}

open class ErrorEvent: Event {
    public override init(
        function: String,
        message: String? = nil,
        eventCode: Int? = nil,
        value: [String: Any?]? = nil,
        date: Date = Date()
    ) {
        super.init(
            function: function,
            message: message,
            eventCode: eventCode,
            value: value,
            date: date
        )
    }
}

public class RetryEvent: ErrorEvent {
    public override init(
        function: String,
        message: String? = nil,
        eventCode: Int? = 0,
        value: [String: Any?]? = nil,
        date: Date = Date()
    ) {
        super.init(
            function: function,
            message: message,
            eventCode: eventCode,
            value: value,
            date: date
        )
    }
}
```

Example contents of a successful push subscription event

```
├ function: processResponse
├ eventCode: 230
├ message: "successful request: push/subscribe"
├ value
│  ├ http code: 200
│  └ response
│     ├ error: 0
│     ├ errorText: ""
│     └ profile
│        ├ id: // profile ID
│        ├ status: subscribed
│        ├ isTest: false
│        └ subscription
│           ├ subscriptionId: // push token of the subscription
│           ├ hashId: // hash ID
│           ├ provider: "ios-apns"
│           ├ status: subscribed
│           ├ fields
│           │  ├ _ad_id: // google service ad id
│           │  ├ _ad_track: false
│           │  ├ _app_id: "AltcraftMobile"
│           │  ├ _app_iid: "1.0.0"
│           │  ├ _app_ver: {"raw":"1.0.0","ver":[1,0]}
│           │  ├ _device_model: "iPhone14,7"
│           │  ├ _device_name: "iPhone"
│           │  ├ _device_type: "mob"
│           │  ├ _os: "Android"
│           │  ├ _os_language: "ru"
│           │  ├ _os_tz: "+0300"
│           │  └ _os_ver: {"raw":"18.6.2","ver":"[\"18.0\", \"6.0\", \"2.0\"]"}
│           └ cats
│              └ [ { name: "cats_1", title: "cats_1", steady: false, active: false } ]
└ date: Tue Aug 12 15:49:20 GMT+03:00 2025
```

Unsubscribing from events

`public func unsubscribe(): Void`

Unsubscribes from SDK events. It stops event delivery while keeping the `callback` assigned.

Example:

```
// unsubscribe from SDK events
AltcraftSDK.shared.eventSDKFunctions.unsubscribe()
```

List of all SDK events

Event list

Code	
200	**SDK configuration is installed**
201	**push provider set**
203	**received Altcraft push notification**
204	**push is posted**
205	**SDK data has been cleared**
206	**SDK background task is registered**
207	**SDK background task completed**
208	**SDK background task is registered**
209	**SDK background task completed**
230	**subscribe request succeeded**
231	**token update request succeeded**
232	**unsuspend request succeeded**
233	**status request succeeded**
234	**push event delivered successfully**
235	**mobile event delivered successfully**
401	**the configuration is not set**
402	**userTag is null. It is impossible to identify the user**
403	**error in CoreData. Re-initialize the SDK**
404	**failed to load model from framework**
405	**App Group name was not set**
406	**forcing a push token update is not possible: the operation is not supported for APNs**
407	**the CoreData entity name is not valid**
408	**entity not found by ID**
409	**SDK initialization timeout has expire**
422	**unsuspend request data is nil**
423	**profile request data is nil**

Code	
430	**subscribe request failed**
431	**token update request failed**
432	**unsuspend request failed**
433	**status request failed**
434	**push event delivery failed**
435	**mobile event delivery failed**
450	**invalid userInfo format**
451	**uid in the push data is null; it is impossible to send a push event to the server**
452	**could not download image data from URL**
453	**'media' key is missing or URL string is invalid**
454	**'buttons' key is missing or is invalid**
455	**invalid button identifier (see event value)**
456	**out of range for identifier (see event value)**
457	**unknown button identifier (see event value)**
471	**invalid apiUrl value — empty or null**
472	**invalid resource token value — resource token is empty**
473	**invalid provider. Available — ios-apns, ios-firebase, ios-huawei**
474	**non-JSON object has been provided**
475	**invalid values: not all values are primitives**
476	**unsupported subscription type. Available: EmailSubscription, SmsSubscription, PushSubscription, CcDataSubscription**
501	**config data is nil**
502	**current token is nil**
503	**userTag is null. It is impossible to identify the user**
504	**notification permission denied**
505	**SDK background task expired**
520	**subscribe request data is nil**
521	**update request data is nil**
524	**push event request data is nil**
525	**mobile event request data is nil**
529	**common data is nil**
530	**subscribe request failed (retryable)**
531	**token update request failed (retryable)**
534	**push event delivery failed (retryable)**
535	**mobile event delivery failed (retryable)**
540	**JWT token is nil**
541	**JWT parsing error**
542	**invalid matching**
543	**auth data is nil**
561	**failed to create request**
562	**failed processing the response**
571	**invalid request URL**
572	**invalid response format**
573	**invalid push subscribe request data**
574	**invalid push event request data**

Additional functions

Functional profile field updates

`public func actionField(key: String) -> ActionFieldBuilder`

A helper for the functional profile field update process. Supports:

```
.set(value)
.unset(value)
.incr(value)
.add(value)
.delete(value)
.upsert(value)
```

Example:

```
// "_fname" — field to update
// .set("Andrey") — sets the field to "Andrey"
AltcraftSDK.shared.pushSubscriptionFunctions.pushSubscribe(
    profileFields: AltcraftSDK.shared.pushSubscriptionFunctions.actionField(key: "_fname").set(value: "Andrey")
)
```

Clearing SDK data

`public func clear(completion: (() -> Void)? = nil): Void`

Clears SDK data and cancels all pending background tasks. Removes `CoreData` records and clears `UserDefa`

Example:

```
// Full SDK reset (cache/database/settings), then call completion
AltcraftSDK.shared. clear(completion: (() -> Void)? = nil)
```

The function accepts an optional `completion` called after cleanup and task cancellation.

Manual Push Event Registration

> **TIP: Note**
>
> **Use these only if you implement your own notification handling and do not pass them to Altcraft SDK. By**

```
// PublicPushEventFunctions

AltcraftSDK
//   Singleton, SDK entry point
└── public static let shared: AltcraftSDK
    // Manual sending of push events (delivery / open)
    └── public let pushEventFunctions: PublicPushEventFunctions
        // Record Altcraft push delivery (fires a delivery event)
        ├── public func deliveryEvent(from request: UNNotificationRequest): Void
        // Record Altcraft push open (fires an open event)
        └── public func openEvent(from request: UNNotificationRequest): Void
```

Manual event registration functions:

- `func deliveryEvent()` **— registers a delivery event;**
- `func openEvent()` **— registers an open event.**

`public func deliveryEvent(from request: UNNotificationRequest): Void`

Registers a delivery event in the Altcraft platform. Pass `UNNotificationRequest` in the `request` parameter.

Example:

```
// Record Altcraft push delivery (fires a delivery event)
AltcraftSDK.shared.pushEventFunctions.deliveryEvent(from: UNNotificationRequest)
```

`public func openEvent(from request: UNNotificationRequest): Void`

Registers an open event in the Altcraft platform. Pass `UNNotificationRequest` in the `request` parameter.

```
// Record Altcraft push open (fires an open event)
AltcraftSDK.shared.pushEventFunctions.openEvent(from: UNNotificationRequest)
```

SDK Configuration

Prerequisites

- **Push provider SDKs are integrated into the app project (see the push provider integration guides).**
- **The Altcraft package is added via Swift Package Manager.**

App preparation

Target settings

Configure the following app target settings:

- ☑ **General**

When using Swift Package Manager, verify that the Altcraft library is added to the app target under Frame

- ☑ **Signing & Capabilities**

Add these capabilities:

- **Push Notifications**
- **App Groups**
- **Background Modes (Background fetch, Background processing)**
- ☑ **Info**

Add the key Permitted background task scheduler identifiers with the value "lib.Altcraft.bgTask.systemCo

- ☑ **Optional**

If you use Firebase Cloud Messaging, add the key `FirebaseAppDelegateProxyEnabled` (Boolean) with the value `NO`

AppDelegate settings

In `AppDelegate.application(_:didFinishLaunchingWithOptions:)`, do the following:

- **pass the `AppGroup` identifier to the SDK**
- **register the SDK `BGTask` (background tasks)**
- **(optional) initialize `UNUserNotificationCenter` features on the SDK side**

Setting the AppGroup identifier

Adding an `AppGroup` identifier is required to exchange data with the Notification Service Extension and for

```
// Configure the App Group and initialize Core Data for the shared container
AltcraftSDK.shared.setAppGroup(groupName: String)
```

Registering the BGTask

Registering SDK background tasks enables retrying failed requests in the background. Use `registerBackgrou`

```
// Register the SDK background task and schedule periodic execution (about every 3 hours)
AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
```

Initializing UNUserNotificationCenter features (optional)

The SDK includes the `NotificationManager` class, which:

- **configures `UNUserNotificationCenter` and requests notification permissions (alert/sound/badge);**
- **calls `UIApplication.registerForRemoteNotifications()` and ensures correct APNs registration;**
- **displays notifications in foreground;**
- **handles notification taps and action buttons — navigates to a specified URL or opens the app;**
- **registers the push open event ("open").**

To use the SDK `NotificationManager`, call this function in `AppDelegate.application(_:didFinishLaunchingWithOptions:)`:

```
func registerForPushNotifications(for application: UIApplication, completion: ((_ granted: Bool, _ error: Error?) -> Void)? = nil): Void
```

This function sets the delegate, requests the user's permission, and registers the app for push notificatio

Example of a correct `AppDelegate.application` setup

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        let appGroup = "group.your.id"

        //ID App Group Setup
        AltcraftSDK.shared.setAppGroup(groupName: appGroup)

        //Backgroung SDK task registration
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()

        //SDK UNUserNotificationCenter functions initialization
        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        //Other functions

        return true
    }
    //Other functions AppDelegate
}
```

Using this class is optional; you can replace it with your own implementation. To do so:

- **set the notification center delegate:** `UNUserNotificationCenter.current().delegate = self`**;**
- **request notification permissions and call** `UIApplication.shared.registerForRemoteNotifications()` **strictly on th**
- **implement** `userNotificationCenter(_:willPresent:withCompletionHandler:)` **to display notifications in foreground**
- **call** `AltcraftSDK.shared.pushEventFunctions.openEvent(from:)` **in** `userNotificationCenter(_:didReceive:withCompletionHan`
- **implement custom logic for handling notification taps (URL/Deep Link/buttons), and ensure a fallback**

JWT protocol setup (optional)

JWTInterface is the protocol for requesting a JWT token. It provides a current JWT token from the app up
JWT authentication is required if the matching type differs from push data from the subscription (for exam
SDK protocol:

```
public protocol JWTInterface {
    func getToken() -> String?
}
```

App-side implementation

```
class JWTProvider: JWTInterface {
    func getToken() -> String? {
         // your code that returns a JWT
      }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

In `AppDelegate`**, specify:**

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
  func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Other code

    AltcraftSDK.setJWTProvider(provider: JWTProvider())
  }
  // Other functions
}
```

> TIP: Note
>
> `getJWT()` **is a synchronous function. The SDK execution thread will pause until JWT is obtained. It's recor and store it in cache so the token is available without delays. Returning** `nil` **is acceptable if no value is**

Provider push token request and deletion protocols

These token request/deletion protocol implementations ensure the current token is used and allow dynam
Register providers in `application(_:didFinishLaunchingWithOptions:)` **(AppDelegate). This registration point guara**

Apple Push Notification service

APNSInterface — the protocol for requesting an APNs push token. This protocol has no token deletion fun
SDK protocol:

```
public protocol APNSInterface {

    func getToken(completion: @escaping (String?) -> Void)
}
```

Recommended app-side implementation

```
class APNSProvider: APNSInterface {
    func getToken(completion: @escaping (String?) -> Void) {
        // your code that returns the APNs token
    }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
  func application(
  _ application: UIApplication,
  didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // other code

    AltcraftSDK.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())
  }
  // other functions
}
```

Firebase Cloud Messaging

FCMInterface — the protocol for requesting and deleting a Firebase Cloud Messaging push token.

SDK protocol:

```
public protocol FCMInterface {
    func getToken(completion: @escaping (String?) -> Void)

    func deleteToken(completion: @escaping (Bool) -> Void)
}
```

> **TIP: Note**

Recommended app-side implementation

```
class FCMProvider: FCMInterface {

    /// Retrieves the current FCM token
   func getToken(completion: @escaping (String?) -> Void) {

        /// APNs token (as Data), retrieved from UserDefaults
        let apnsToken = getAPNsTokenDataFromUserDefaults()

        /// Set the APNs token for FCM before requesting the FCM token
        Messaging.messaging().apnsToken = apnsToken

        /// Request the FCM token
        Messaging.messaging().token { token, error in
            if error != nil {
                completion(nil)
            } else {
                completion(token)
            }
        }
    }

    /// Deletes the current FCM token
    func deleteToken(completion: @escaping (Bool) -> Void) {
        Messaging.messaging().deleteToken { error in
            if error != nil {
                completion(false)
            } else {
                completion(true)
            }
        }
    }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
  func application(
  _ application: UIApplication,
  didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Other code

    AltcraftSDK.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
  }
  // Other functions
}
```

Huawei Mobile Services

HMSInterface — the protocol for requesting and deleting a Huawei Mobile Services push token.

SDK protocol:

```
public protocol HMSInterface {

    func getToken(completion: @escaping (String?) -> Void)

    func deleteToken(completion: @escaping (Bool) -> Void)
}
```

Recommended app-side implementation

```
class HMSProvider: HMSInterface {

    func getToken(completion: @escaping (String?) -> Void) {

        // variable containing the APNs token (as a String), which is later
        // passed to HmsInstanceId.getInstance().getToken(apnsToken)
        guard let apnsToken = getAPNsTokenFromUserDefault() else {
            completion(nil)
            return
        }

        let token = HmsInstanceId.getInstance().getToken(apnsToken)
        completion(token)
    }


    func deleteToken(completion: @escaping (Bool) -> Void) {
        HmsInstanceId.getInstance().deleteToken()
        completion(true)
    }
}
```

Registering the provider in `application(_:didFinishLaunchingWithOptions:)`

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
  func application(
  _ application: UIApplication,
  didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Other code

    AltcraftSDK.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
  }
  // Other functions
}
```

Preparing the Notification Service Extension

Adding an NSE is required for Rich Push support and guaranteed delivery event registration.

Create a Notification Service Extension:

- **File — New — Target — Notification Service Extension;**
- **Choose a Product Name for the extension target;**
- **Activate it.**

In the Notification Service Extension:

- ☑ **General**
- **Set Minimum Deployments — this Xcode build setting defines the minimum OS version required to run**
- **Add the Altcraft library under Frameworks, Libraries and Embedded Content.**
- ☑ **Signing & Capabilities**
- **Specify the `AppGroup` identifier.**

Then configure the SDK inside `UNNotificationServiceExtension`:

- **Import the `Altcraft` library.**
- **In the method `didReceive(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotifica`**
  - **Pass the `appGroup` identifier to `AltcraftNSE.shared.setAppGroup(groupName: appGroupsName)`. The identifier mu**
  - **Set the JWT provider (if JWT authentication is used): `AltcraftNSE.shared.setJWTProvider(provider: jwtProv`**
  - **Check whether the incoming notification belongs to Altcraft using `AltcraftNSE.shared.isAltcraftPush(r`**
  - **If the notification is from Altcraft, pass the `UNNotificationRequest` and the `contentHandler` to `AltcraftNS`**

The example below can be used as a ready-to-use `UNNotificationServiceExtension` class. Replace all autogene

Example implementation of `UNNotificationServiceExtension`

```
class NotificationService: UNNotificationServiceExtension {

    /// - important! Set app groups identifier.
    let appGroupID = "group.your.id"

    /// - important! Set the provider to JWT if you are using JWT authentication.
    let jwtProvider = JWTProvider()

    override func didReceive(_ request: UNNotificationRequest, withContentHandler contentHandler: @escaping (UNNotificationContent) -> Void) {

        AltcraftNSE.shared.setAppGroup(groupName: appGroupID)
        AltcraftNSE.shared.setJWTProvider(provider: jwtProvider)

        if AltcraftNSE.shared.isAltcraftPush(request) {
            AltcraftNSE.shared.handleNotificationRequest(request: request, contentHandler: contentHandler)
        } else {
            contentHandler(request.content)
        }
    }

    override func serviceExtensionTimeWillExpire() {
        AltcraftNSE.shared.serviceExtensionTimeWillExpire()
    }
}
```

> **TIP: Note**
>
> **The app target and the Notification Service Extension target run in separate processes and do not share `withContentHandler contentHandler: @escaping (UNNotificationContent) -> Void)`. This ensures the SDK has access to**

SDK initialization

Initialization parameters

Use the `AltcraftConfiguration` class to pass configuration parameters:

```
public final class AltcraftConfiguration {
    private let apiUrl: String
    private let rToken: String?
    private var appInfo: AppInfo?
    private var providerPriorityList: [String]?
}
```

The configuration object is built after calling the `build()` function of `AltcraftConfiguration.Builder(): Altcraft`

Parameter descriptions:

apiUrl `String`

Required: Yes

Description: Altcraft API endpoint URL

---

rToken `String?`

Default: `nil`

Required: No

Description: Altcraft role token (identifies a resource, DB, or account). Used when the only matching type

---

appInfo `AppInfo?`

Default: `nil`

Required: No

Description: Basic app metadata for Firebase Analytics. To set this parameter, use the public SDK struct

```
public struct AppInfo: Codable {

    /// Unique App ID (Firebase `app_id`)
    public var appID: String

    /// Unique app instance ID (Firebase `app_instance_id`)
    public var appIID: String

    /// App version (Firebase `app_version`)
    public var appVer: String
}
```

---

enableLogging `Bool?`
Default: `nil`
Required: No
Description: Enables/disables logging. Logs can be found by `[Altcraft SDK]` **prefix.**

---

providerPriorityList `[String]?`
Default: `nil`
Required: No
Description: Provider priority list. Used to automatically refresh the subscription's push token if the highe
Usage example:

```
providerPriorityList = [
    Constants.ProviderName.apns,
    Constants.ProviderName.firebase,
    Constants.ProviderName.huawei
]
```

With this configuration:

- **The SDK first requests the APNs token; if APNs is unavailable — FCM; if FCM is unavailable — HMS;**
- **Works provided the corresponding provider protocols are implemented in the app.**

For convenient assignment, the SDK exposes public constants in `enum Constants`**:**

```
public enum Constants {
    public enum ProviderName {
        /// Provider name for Firebase
        public static let firebase = "ios-firebase"

        /// Provider name for APNs
        public static let apns = "ios-apns"

        /// Provider name for HMS
        public static let huawei = "ios-huawei"
    }
}
```

If not specified, the default provider priority is:

```
apns -> firebase -> huawei
```

The parameter value may contain a single element — in this case, only that provider will be used regardle
You can omit this parameter if:

- **only one provider is used in the project;**
- **the default priority fits your needs.**

Performing initialization

> **TIP: Note**
>
> **Call** `AltcraftSDK.shared.initialization()` **when needed, but only after setting the** `AppGroup` **identifier and regis**

Use `public func initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil)` **to initialize**

```
// SDK initialization and configuration
AltcraftSDK.shared.initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil)
```

The `initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil)` **function expects an** `A`

```
 let config = AltcraftConfiguration.Builder()
        .setApiUrl("your apiUrl")
        .setRToken("your rToken")
        .setAppInfo(AppInfo(appID: "your appID", appIID: "your appIID", appVer: "your appVer"))
        .setProviderPriorityList([Constants.ProviderName.firebase])
        .build()
```

Example of the correct initialization order in `AppDelegate.application(_:didFinishLaunchingWithOptions:)` **(after setting the** `AppGroup` **ID and registering provide**

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        // Firebase configuration can be placed here if Firebase is used in the project

        let appGroup = "group.altcraft.apns.example"

        AltcraftSDK.shared.setAppGroup(groupName: appGroup)
        AltcraftSDK.shared.backgroundTasks.registerBackgroundTask()
        AltcraftSDK.shared.setJWTProvider(provider: JWTProvider())
        AltcraftSDK.shared.pushTokenFunctions.setAPNSTokenProvider(APNSProvider())
        AltcraftSDK.shared.pushTokenFunctions.setFCMTokenProvider(FCMProvider())
        AltcraftSDK.shared.pushTokenFunctions.setHMSTokenProvider(HMSProvider())
        AltcraftSDK.shared.notificationManager.registerForPushNotifications(for: application)

        let config = AltcraftConfiguration.Builder()
            .setApiUrl("your apiUrl")
            .setRToken("your rToken")
            .setAppInfo(AppInfo(appID: "your appID", appIID: "your appIID", appVer: "your appVer"))
            .setProviderPriorityList([Constants.ProviderName.firebase])
            .build()

        AltcraftSDK.shared.initialization(configuration: config) { success in
            if success {
               // actions after successful initialization
            }
        }

        // Other functions

        return true
    }
    // Other AppDelegate functions
}
```

Public SDK API

class AltcraftSDK

```
AltcraftSDK
// Singleton entry point to the SDK
├ public static let shared: AltcraftSDK
// SDK initialization with configuration (completion is called on main)
├ public func initialization(configuration: AltcraftConfiguration?, completion: ((Bool) -> Void)? = nil): Void
// Sets App Group and initializes Core Data for the shared container
├ public func setAppGroup(groupName: String): Void
// Registers a JWT provider for token retrieval
├ public func setJWTProvider(provider: JWTInterface): Void
// Completely clears SDK data (cache / DB / settings), then calls completion
├ public func clear(completion: (() -> Void)? = nil): Void
|
// SDK event stream (single active subscriber)
├ public let eventSDKFunctions: SDKEvents
|  // Subscribe to SDK events (replaces the existing subscriber)
|  ├ func subscribe(callback: @escaping (Event) -> Void): Void
|  // Unsubscribe from events (callback remains set, delivery stops)
|  ├ func unsubscribe(): Void
|  // Event model (base type for all events and errors)
|  ├ Event
|  |   // Base event model
|  |   id: UUID
|  |   function: String
|  |   message: String?
|  |   eventCode: Int?
|  |   value: [String: Any?]?
|  |   date: Date
|  |   // Initializer (normalizes function, filters nil values)
|  |   init(function: String, message: String? = nil, eventCode: Int? = nil, value: [String: Any?]? = nil, date: Date = Date())
|  |   // Equality by unique id
|  |   static func ==(lhs: Event, rhs: Event) -> Bool
|  |   // Hashing by id
|  |   func hash(into hasher: inout Hasher): Void
|  |
|  |   // Non-retryable errors (4xx class)
|  |   ErrorEvent : Event
|  |   init(function: String, message: String? = nil, eventCode: Int? = nil, value: [String: Any?]? = nil, date: Date = Date())
|  |
|  |   // Retryable errors (typically 5xx)
|  |   RetryEvent : ErrorEvent
|  |   init(function: String, message: String? = nil, eventCode: Int? = 0, value: [String: Any?]? = nil, date: Date = Date())
|
// Push token and provider management (FCM / HMS / APNs)
├  public let pushTokenFunction: PublicPushTokenFunctions
|  // Set Firebase Cloud Messaging provider (nil to remove)
|  ├ public func setFCMTokenProvider(_ provider: FCMInterface?): Void
|  // Set Huawei Mobile Services provider (nil to remove)
|  ├ public func setHMSTokenProvider(_ provider: HMSInterface?): Void
|  // Set Apple Push Notification service provider (nil to remove)
|  ├ public func setAPNSTokenProvider(_ provider: APNSInterface?): Void
|  // Asynchronously get the current push token (completion optional)
|  ├ public func getPushToken(completion: ((TokenData?) -> Void)? = nil): Void
|  // Manually save a token (String for FCM/HMS, Data for APNs); provider: "ios-firebase" | "ios-huawei" | "ios-apns"
|  ├ public func setPushToken(provider: String, pushToken: Any?): Void
|  // Apply a new provider priority list and trigger token update
|  ├ public func changePushProviderPriorityList(_ list: [String]): Void
|  // Delete a token for the selected provider and call completion
|  ├ public func deleteDeviceToken(provider: String, completion: @escaping () -> Void): Void
|  // Force token update (resolve provider → delete token (except APNs) → request a new one)
|  └ public func forcedTokenUpdate(completion: (() -> Void)? = nil): Void
|
// Push subscription management (status changes, status requests)
├  public let pushSubscriptionFunctions: PublicPushSubscriptionFunctions
|  // Subscribe (status = SUBSCRIBED)
|  ├ public func pushSubscribe(
|  |     sync: Bool = true,
|  |     profileFields: [String: Any?]? = nil,
|  |     customFields: [String: Any?]? = nil,
|  |     cats: [CategoryData]? = nil,
|  |     replace: Bool? = nil,
|  |     skipTriggers: Bool? = nil
|  |   ): Void
|  // Unsubscribe (status = UNSUBSCRIBED)
|  ├ public func pushUnSubscribe(
|  |     sync: Bool = true,
|  |     profileFields: [String: Any?]? = nil,
|  |     customFields: [String: Any?]? = nil,
|  |     cats: [CategoryData]? = nil,
|  |     replace: Bool? = nil,
|  |     skipTriggers: Bool? = nil
```

```
    ): Void
  // Suspend (status = SUSPENDED)
  ├ public func pushSuspend(
      sync: Bool = true,
      profileFields: [String: Any?]? = nil,
      customFields: [String: Any?]? = nil,
      cats: [CategoryData]? = nil,
      replace: Bool? = nil,
      skipTriggers: Bool? = nil
    ): Void
  // UnSuspend subscription (single request, no retries / persistence)
  ├ public func unSuspendPushSubscription(
      completion: @escaping (ResponseWithHttp?) -> Void
    ): Void
  // Status of the latest profile subscription
  ├ public func getStatusOfLatestSubscription(
      completion: @escaping (ResponseWithHttp?) -> Void
    ): Void
  // Subscription status for the current token / provider
  ├ public func getStatusForCurrentSubscription(
      completion: @escaping (ResponseWithHttp?) -> Void
    ): Void
  // Latest subscription status for a provider (current one if nil)
  └ public func getStatusOfLatestSubscriptionForProvider(
       provider: String? = nil,
       completion: @escaping (ResponseWithHttp?) -> Void
     ): Void
   // add a functional profile field (set / incr / ...)
  └ public func actionField(key: String) -> ActionFieldBuilder
// Registering Mobile Events
├ public let mobileEventFunctions: PublicMobileEventFunctions
  // Registers an Altcraft mobile event
  └ public func mobileEvent(
        sid: String,
        altcraftClientID: String = "",
        eventName: String,
        sendMessageId: String? = nil,
        payload: [String: Any?]? = nil,
        matching: [String: Any?]? = nil,
        matchingType: String? = nil,
        profileFields: [String: Any?]? = nil,
        subscription: (any Subscription)? = nil,
        utm: UTM? = nil
    ): Void
|
// Updating Profile Fields
├ public let profileFunctions: PublicProfileFunctions
  // Updates Altcraft profile fields
  └ public func updateProfileFields(
        profileFields: [String: Any?]? = nil,
        skipTriggers: Bool? = nil
    ): Void
|
// Manual push event sending (delivery / open)
├  public let  pushEventFunctions: PublicPushEventFunctions
  // Register delivery of an Altcraft push (fires delivery event)
  ├ public func deliveryEvent(from request: UNNotificationRequest): Void
  // Register opening of an Altcraft push (fires open event)
  └ public func openEvent(from request: UNNotificationRequest): Void
|
// Periodic background task registration (BGAppRefreshTask)
├ public let backgroundTasks: BackgroundTasks
  // Register a task and schedule periodic execution (~every 3 hours)
  └ public func registerBackgroundTask(): Void
|
// In-app notification management (notification center delegate)
├  public let notificationManager: NotificationManager
   // NotificationManager singleton (entry point to the notification manager)
   ├ public static let shared: NotificationManager
   // ObjC-friendly accessor (equivalent to shared)
   ├ public class func sharedInstance() -> NotificationManager
   //
   // NotificationCenter events (SDK posts these)
   ├ public extension Notification.Name
   │  // Fired when a notification is about to be presented in foreground.
   │  // userInfo: ["notification": UNNotification]
   │  ├ static let altcraftPushWillPresent: Notification.Name
   │  // Fired when user taps on a delivered notification or performs an action.
   │  // userInfo: ["response": UNNotificationResponse]
   │  └ static let altcraftPushDidReceive: Notification.Name
```

```
//
// App-level delegation modes
├─ public var customPushProcessing: Bool
├─ public var customClickProcessing: Bool
//
// Callbacks for custom handling
│  // Called when a foreground push arrives.
│  // App must call completion when customPushProcessing = true.
├─ public var onForegroundNotification: ((UNNotification, @escaping (UNNotificationPresentationOptions) -> Void) -> Void)?
│  // Called when the user taps a notification (or performs an action).
│  // App must call completion when customClickProcessing = true.
├─ public var onNotificationClick: ((UNNotificationResponse, @escaping () -> Void) -> Void)?
//
// Push registration (delegate + requestAuthorization + registerForRemoteNotifications)
├─ func registerForPushNotifications(for application: UIApplication, completion: ((_ granted: Bool, _ error: Error?) -> Void)? = nil): Void
//
// UNUserNotificationCenterDelegate
│  // Posts altcraftPushWillPresent; if customPushProcessing is enabled,
│  // delegates the decision to the app,
│  // otherwise shows banner/alert (+ badge/sound)
├─ public func userNotificationCenter(
│     _ center: UNUserNotificationCenter,
│     willPresent notification: UNNotification,
│     withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void
│   ): Void
│  // Posts altcraftPushDidReceive; if customClickProcessing is enabled,
│  // delegates handling to the app,
│  // otherwise parses userInfo, sends an open event and performs the action,
│  // then calls completionHandler
└─ public func userNotificationCenter(
      _ center: UNUserNotificationCenter,
      didReceive response: UNNotificationResponse,
      withCompletionHandler completionHandler: @escaping () -> Void
    ): Void
```

class AltcraftNSE

```
// Used inside the Notification Service Extension (NSE) — handles processing Altcraft push notifications,
// determining Altcraft notifications, and registering mobile events without relying on UIApplication
AltcraftNSE
// Shared (singleton) instance for centralized NSE logic
├─ public static let shared: AltcraftNSE
// Provides access to the mobile event registration API
├─ public let mobileEventFunctions: PublicMobileEventFunctions
// Registers an Altcraft mobile event
├─ public func mobileEvent(
│      sid: String,
│      altcraftClientID: String = "",
│      eventName: String,
│      sendMessageId: String? = nil,
│      payload: [String: Any?]? = nil,
│      matching: [String: Any?]? = nil,
│      matchingType: String? = nil,
│      profileFields: [String: Any?]? = nil,
│      subscription: (any Subscription)? = nil,
│      utm: UTM? = nil
│  ): Void
// Updates Altcraft profile fields
├─ public func updateProfileFields(
│      profileFields: [String: Any?]? = nil,
│      skipTriggers: Bool? = nil
│  ): Void
// Checks if the notification is related to Altcraft
├─ public func isAltcraftPush(_ request: UNNotificationRequest) -> Bool
// Processes the incoming Altcraft notification and passes the modified content to the contentHandler
├─ public func handleNotificationRequest(
│      request: UNNotificationRequest,
│      contentHandler: @escaping (UNNotificationContent) -> Void
│  ): Void
// Called by the system when the NSE execution time expires
├─ public func serviceExtensionTimeWillExpire(): Void
// Sets the App Group and initializes the shared data container
├─ public func setAppGroup(groupName: String): Void
// Registers a JWT provider for secure token handling
└─ public func setJWTProvider(provider: JWTInterface): Void
```

class AltcraftConfiguration

```
AltcraftConfiguration
// Altcraft SDK configuration class.
// Stores the base API URL and optional parameters (rToken, AppInfo, providerPriorityList, enableLogging).
// Created only via Builder. ObjC-compatible: AltcraftConfiguration / AltcraftConfiguration_Builder.
│
├─ @objc(AltcraftConfiguration_Builder) public final class Builder
│  // Builder for creating a configuration with validation.
│  // Allows setting required and optional fields and assembling the configuration object.
│  // Builder constructor
│  ├─ public override init()
│  // Set API URL (required)
│  ├─ @discardableResult public func setApiUrl(_ url: String) -> Builder
│  // Set resource token (optional)
│  ├─ @discardableResult public func setRToken(_ rToken: String?) -> Builder
│  // Set application metadata (Swift AppInfo model) (optional)
│  ├─ @discardableResult public func setAppInfo(_ info: AppInfo?) -> Builder
│  // Set application metadata (ObjC DTO AppInfoObjC) (optional, not available in Swift)
│  ├─ @available(swift, obsoleted: 1) @objc(setAppInfo:) @discardableResult public func setAppInfo(_ dto: AppInfoObjC?) -> Builder
│  // Set push provider priority list (optional)
│  ├─ @discardableResult public func setProviderPriorityList(_ list: [String]?) -> Builder
│  // SDK internal logging control (optional)
│  // true — enable, false — disable, nil — keep default behavior
│  ├─ @discardableResult public func setEnableLogging(_ enabled: Bool?) -> Builder
│  // Build a valid configuration (nil if validation fails)
│  // Validation: apiUrl is not empty; providerPriorityList is valid; appInfo resolves (Swift AppInfo has priority, otherwise AppInfoObjC -> Ap
│  └─ public func build() -> AltcraftConfiguration?
// Access to configuration parameters
├─ public func getApiUrl() -> String
├─ public func getRToken() -> String?
├─ public func getAppInfo() -> AppInfo?
├─ public func getEnableLogging() -> Bool?
└─ public func getProviderPriorityList() -> [String]?
```

Structures

```
/// Stores a push token and provider name.
public struct TokenData: Codable {
    public let provider: String
    public let token: String
}

/// Application metadata for Firebase Analytics (if used).
public struct AppInfo: Codable {
    public var appID: String
    public var appIID: String
    public var appVer: String
}

/// API response wrapper with HTTP status code.
public struct ResponseWithHttp {
    public let httpCode: Int?
    public let response: Response?
}

/// Synchronous subscription request response.
public struct Response: Codable {
    public let error: Int?
    public let errorText: String?
    public let profile: ProfileData?
}

/// User profile data.
public struct ProfileData: Codable {
    public let id: String?
    public let status: String?
    public let isTest: Bool?
    public let subscription: SubscriptionData?
}

/// Current profile subscription.
public struct SubscriptionData: Codable {
    public let subscriptionId: String?
    public let hashId: String?
    public let provider: String?
    public let status: String?
    public let fields: [String: JSONValue]?
    public let cats: [CategoryData]?
}

/// Subscription category (name / title / flags).
public struct CategoryData: Codable {
    public var name: String?
    public var title: String?
    public var steady: Bool?
    public var active: Bool?
}

/// Email channel subscription (mobile events).
public struct EmailSubscription: Subscription, Codable {
    public let resourceId: Int
    public let email: String
    public let status: String?
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
    public let channel: String = "email"
}

/// SMS channel subscription (mobile events).
public struct SmsSubscription: Subscription, Codable {
    public let resourceId: Int
    public let phone: String
    public let status: String?
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
    public let channel: String = "sms"
}

/// Push channel subscription (mobile events).
public struct PushSubscription: Subscription, Codable {
    public let resourceId: Int
    public let provider: String
    public let subscriptionId: String
    public let status: String?
```

```
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
    public let channel: String = "push"
}

/// Subscription with cc_data (Telegram, WhatsApp, Viber, Notify).
public struct CcDataSubscription: Subscription, Codable {
    public let resourceId: Int
    public let channel: String
    public let ccData: [String: JSONValue]
    public let status: String?
    public let priority: Int?
    public let customFields: [String: JSONValue]?
    public let cats: [String]?
}

/// UTM parameters for mobile events.
public struct UTM: Codable {
    public let campaign: String?
    public let content: String?
    public let keyword: String?
    public let medium: String?
    public let source: String?
    public let temp: String?
}
```

Protocols

JWTInterface:

```
JWTInterface
// Access to the current JWT
└─ func getToken() -> String?
   // Returns the current JWT or nil
```

APNSInterface:

```
APNSInterface
// APNs token retrieval contract
└─ func getToken(completion: @escaping (String?) -> Void)
   // Returns the APNs token via completion or nil
```

FCMInterface:

```
FCMInterface
// Contract for FCM token operations (retrieve / revoke)
├─ func getToken(completion: @escaping (String?) -> Void)
│  // Returns the FCM token via completion or nil
└─ func deleteToken(completion: @escaping (Bool) -> Void)
   // Deletes the FCM token; completion(true|false)
```

HMSInterface:

```
HMSInterface
// Contract for HMS token operations
├─ func getToken(completion: @escaping (String?) -> Void)
│  // Returns the HMS token via completion or nil
└─ func deleteToken(completion: @escaping (Bool) -> Void)
   // Deletes the HMS token; completion(true|false)
```

Subscription:

```
Subscription (protocol)
// Base protocol for all subscription types passed with a mobile event
```

Apple Push Notification Service

You can configure APNS using either an authorization key or a certificate. The setup method depends on t

Option 1: using an Auth Key

Step 1. APNS configuration

Go to the Apple Developer website, section Account. Navigate to Certificates, Identifiers & Profiles — Key

Select Apple Push Notification service:

Configure APNs. Click Configure and select the Environment based on your setup:

Register the key:

Apple Developer allows the created key to be downloaded only once. Make sure to store it securely. After

Step 2. Altcraft resource configuration

Configure the resource in the platform. Add a Push channel and enable Apple Push Notification service:

Enable Use APNs token and fill in the following fields:

- **apns-topic — the app Bundle ID**
- **kid — the key ID created in the previous step. It is available in Certificates, Identifiers & Profiles — Ke**
- **iss — the Team ID from the Apple Developer website**
- **Signing key — the created Auth Key. To add it, open the downloaded p8 file in a text editor and copy it**

Save the resource. Provider configuration is complete. Next, follow the SDK initialization guide.

Option 2: using a certificate

Step 1. APNS configuration

Request a certificate from the Keychain Access certificate authority:

Fill in the required fields and save the file to disk:

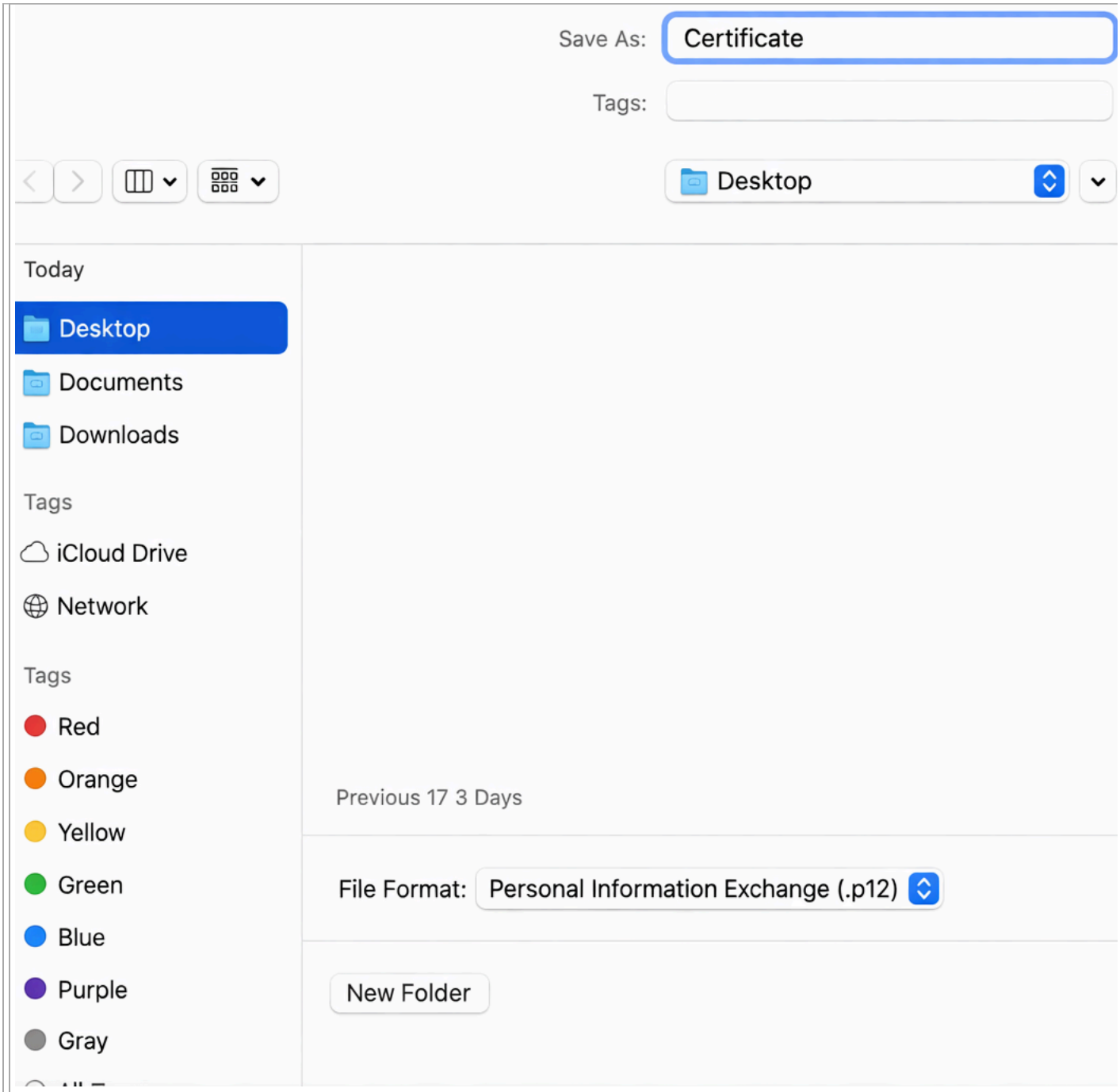
The saved .certSigningRequest file will be needed at the next step.
Go to the Apple Developer website, section Account. Navigate to Certificates, Identifiers & Profiles — Cer

In Services, select Apple Push Notification service SSL (Sandbox) or (Sandbox and Production) depending

Upload the previously created Certificate Signing Request (.certSigningRequest file):

Download the created certificate. Open the downloaded file in Keychain Access. Double-click the certifica

Right-click the certificate and export it as a .p12 file. Set a password if required:



Using Terminal, navigate to the folder containing the exported certificate. Create a pem key using the cor
Without closing Terminal or changing directories, create a pem certificate using `openssl pkcs12 -legacy -in f`
Open both files in Terminal using nano. Their contents will be required in the platform fields at the next st

UW PICO 5.09
Bag Attributes
friendlyName:
localKeyID:
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
^G Get Help
^X Exit
^O WriteOut
^J Justify
^R Read File
^W Where is

Step 2. Altcraft resource configuration

Configure the resource in the platform. Add a Push channel and enable Apple Push Notification service:

Enable Use APNs certificate and fill in the following fields:

- **apns-topic — the app Bundle ID**
- **Certificate — the certificate file contents**
- **Private key — the certificate private key value**

Save the resource. Provider configuration is complete. Next, follow the SDK initialization guide.

Firebase Cloud Messaging

Step 1. Firebase Cloud Messaging project setup

- **Create a [Firebase] project.**
- **Register a new app in the project:**
- **Download the `GoogleSrvice-Info.plist` file.**
- **Go to Project Settings —> Cloud Messaging —> Apple app configuration.**
- **Add the required APNs Authentication Key data (the key must first be created in [Apple Developer]):**

Enter the following data:

- **upload the `.p8` key file**
- **specify the `key ID`**
- **specify the `team ID`**

Upload development APNs au

APNs auth key ⓘ

Drag file here to preview
Supports P8

Key ID (required) ⓘ

Enter key ID

Team ID (required) ⓘ

Enter team ID

Can

Step 2. Altcraft resource configuration

- **specify the** `messagingSenderId`
- **specify the** `projectId`
- **upload the Firebase private key file (.json). The file is created in Project settings —> Service account.**

Step 3. Integrating Firebase Cloud Messaging into the app project

- **Move the downloaded** `GoogleSrvice-Info.plist` **file into the application folder (app).**
- **Add the Firebase package using Swift Package Manager — repository [Firebase](). The Firebase package**
- **General:**
  - **Ensure that the FirebaseMessaging library is added under Frameworks, Libraries, and Embedded**
- **Signing & Capabilities:**
  - **PushNotifications**

Option 1: FirebaseAppDelegateProxyEnabled — YES (Default for Firebase Messaging)

1. **Add the** `MessagingDelegate` **protocol to** `AppDelegate`**:**

```
class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate {}
```

This allows receiving a callback with the current FCM token through the `messaging(_:didReceiveRegistrationTok`

2. **Initialize Firebase in** `didFinishLaunchingWithOptions`**:**

```
FirebaseApp.configure()
Messaging.messaging().delegate = self
application.registerForRemoteNotifications()
```

3. **Implement the delegate method to receive the token:**

```
func messaging(_ messaging: Messaging, didReceiveRegistrationToken fcmToken: String?) {
    guard let token = fcmToken else { return }
    print(" FCM Token: \(token)")
}
```

Full example of Firebase initialization and FCM token retrieval

```
@main
struct IOSQuickStartApp: App {
   @UIApplicationDelegateAdaptor(AppDelegate.self) var delegate

   var body: some Scene {
       WindowGroup {
           ContentView()
       }
   }
}

class AppDelegate: UIResponder, UIApplicationDelegate, MessagingDelegate, UNUserNotificationCenterDelegate {
   func application(
       _ application: UIApplication,
       didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
   ) -> Bool {
       /// Initialize Firebase
       FirebaseApp.configure()
       /// Set delegate
       Messaging.messaging().delegate = self
       application.registerForRemoteNotifications()

       return true
   }

   func messaging(_ messaging: Messaging, didReceiveRegistrationToken fcmToken: String?) {
       if let token = fcmToken {
          //FCM token
       }
   }
}
```

Option 2: FirebaseAppDelegateProxyEnabled - NO

1. **In the app target settings:**
   - **Info:**
     - **Add the key** `FirebaseAppDelegateProxyEnabled (Boolean)` **with the value** `NO` **in the app's Info.**
2. **The** `MessagingDelegate` **protocol is not required in this case. When** `FirebaseAppDelegateProxyEnabled` **is disable**
3. **Initialize Firebase in** `didFinishLaunchingWithOptions`**:**

```
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        FirebaseApp.configure()
        return true
    }
```

4. Retrieve and save the APNs token in `didRegisterForRemoteNotificationsWithDeviceToken`:

```
class AppDelegate: UIResponder, UIApplicationDelegate, UNUserNotificationCenterDelegate {
    func application(
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
    ) -> Bool {
        FirebaseApp.configure()
        return true
    }

    func application(
        _ application: UIApplication,
        didRegisterForRemoteNotificationsWithDeviceToken apnsToken: Data
    ) {
        // Save the token for later use when requesting the FCM token
        setAPNsTokenInUserDefault(apnsToken)
    }
}
```

Step 4. Manually setting the APNs token in Firebase Messaging before requesting the FCM token

Example inside SDK provider

```
class FCMProvider: FCMInterface {

    /// Retrieves the current FCM token
    func getToken(completion: @escaping (String?) -> Void) {

        /// APNs token retrieved from UserDefaults
        let apnsToken = getAPNsTokenDataFromUserDefaults()

        /// Manually set the APNs token before requesting the FCM token
        Messaging.messaging().apnsToken = apnsToken

        /// Request FCM token
        Messaging.messaging().token { token, error in
            if error != nil {
                completion(nil)
            } else {
                completion(token)
            }
        }
    }

    /// Deletes the current FCM token
    func deleteToken(completion: @escaping (Bool) -> Void) {
        Messaging.messaging().deleteToken { error in
            if error != nil {
                completion(false)
            } else {
                completion(true)
            }
        }
    }
}
```

Huawei Mobile Services

Step 1. Huawei Mobile Services project setup

Create an [AppGallery Connect] project. Go to Develop and Services —> Add project:
Add a new app to the project:
Provide your app information:
Download the `agconnect-services.plist` file:
In the left sidebar, under Grow, select Push Kit:
Click Enable —> Select data storage location:
Then go to Push Kit —> Settings —> App —> iOS configuration —> Enable. In the opened iOS configuration

```
- **APNs Authentication Key** — the contents of the private .p8 key file opened in a text editor;
- **Key ID** — the `keyID` of the private .p8 key file (usually matches the file name — the value after `AuthKey_`);
- **Developer ID** — the `teamID` from Apple Developer.
```

Step 2. Altcraft resource configuration

In the Altcraft platform, create a resource or add a new communication channel — Push — to an existing r
Scroll down to Huawei settings and enter the App ID, OAuth 2.0 client ID, and OAuth 2.0 client secret:
Note that the App ID is the Client ID from the App information section.
After making changes, don't forget to save the resource.

Step 3. Integrating Huawei Push Kit into the app project

Move the downloaded `agconnect-services.plist` file to the root directory of the app module (app). Then add tl
Configure the app target:

- **General — Frameworks, Libraries, and Embedded Content**

Make sure the required frameworks are added and set to Do Not Embed:

- `AGConnectCore.xcframework`
- `AGConnectCredential.xcframework`
- `HiAnalytics.xcframework`
- `HMFoundation.xcframework`
- **Signing & Capabilities**

Add Push Notifications.

Step 4. Obtaining the HMS token

Get the HMS token using the `HmsInstanceId.getInstance().getToken(apnsToken)` function, passing the APNs token a

Example of obtaining and deleting an HMS token in an SDK interface implementation

```
// Be sure to import HmsPushSdk in the files where you use HMS functions

class HMSProvider: HMSInterface {

    /// Retrieves the current HMS token using the APNs token
    func getToken(completion: @escaping (String?) -> Void) {
        guard let apnsToken = getAPNsTokenFromUserDefault() else {
            completion(nil)
            return
        }

        let token = HmsInstanceId.getInstance().getToken(apnsToken)
        completion(token)
    }


    func deleteToken(completion: @escaping (Bool) -> Void) {
        HmsInstanceId.getInstance().deleteToken()
        completion(true)
    }
}
```

Quick Start

This article provides an example of quick integration of Altcraft mSDK for React Native.

Step 0. Prerequisites

- **A React Native project is created and runs on iOS/Android.**
- **Native push providers are integrated into the project:**
  - **Android: FCM, HMS, RuStore**
  - **iOS: APNS, FCM, HMS**
- **Altcraft mSDK is configured on the native side:**
  - **providers are registered (JWT / APNS / FCM / HMS / RuStore);**
  - **interception of incoming push notifications and their forwarding to the SDK is configured.**

Step 1. Package installation

```
npm i @altcraft/react-native-sdk
# or
yarn add @altcraft/react-native-sdk
```

Step 2. SDK configuration and initialization

Import the SDK and initialize it at app startup (for example, in `App.tsx`):

```
AltcraftSDK.initialize({
  apiUrl: 'https://pxl-your-instance.altcraft.com',
  rToken: null,
  appInfo: {
    appID: 'your-app-id',
    appIID: 'your-installation-id',
    appVer: '1.0.0',
  },
  providerPriorityList: null,
  enableLogging: true,

  // -------- ANDROID-ONLY (optional) --------
  android_icon: null,
  android_pushReceiverModules: null,
  android_pushChannelName: null,
  android_pushChannelDescription: null,
});
```

Step 3. Subscribing to push notifications

Push subscription is performed by calling `pushSubscribe()`:

```
AltcraftSDK.pushSubscribe(
  true,
  { _fname: 'user_first_name', _lname: 'user_last_name' }, // profileFields
  null,                                                    // customFields
  null,                                                    // cats
  null,                                                    // replace
  null                                                     // skipTriggers
);
```

- **`sync: true` — performs the request synchronously (the result can be obtained via an SDK event if supp**
- **`sync: false` — performs the request asynchronously.**

(Optional) Subscribing to SDK events

To receive call results and internal SDK events, use `subscribeToEvents()`:

```
AltcraftSDK.subscribeToEvents((event: SdkEvent) => {
  // event.function — name of the SDK function that triggered the event
  // event.type — 'event' | 'error' | 'retryError'
  // event.code / event.message — event code and message
  // event.value — additional data (if any)
  console.log('[AltcraftSDK event]', event);
});
```

To unsubscribe:

```
AltcraftSDK.unsubscribeFromEvent();
```

SDK Configuration

Prerequisites

- **Push provider SDKs are integraated into the app project (see the push provider integration guides).**
- **The npm package is installed from the repository npm using one of the following commands:**

```
yarn add altcraft-react-native-sdk
# or
npm i altcraft-react-native-sdk
```

App preparation

Passing a JWT token to the SDK (optional)

JWT implementation is required if API requests use JWT authentication. JWT confirms that user identifier
JWT authentication is mandatory if the matching type differs from push data from the subscription (for ex
JWT setup follows the same approach as in the native SDKs (Android / iOS).

Preparing to work with push providers

For iOS, preparation for working with push providers is done natively. Detailed instructions are available i

SDK initialization

Initialization parameters

Use the `AltcraftConfig` object to pass configuration parameters:

```
  apiUrl: string;

  rToken?: string | null;
  appInfo?: {
    appID: string;
    appIID: string;
    appVer: string;
  } | null;

  providerPriorityList?: string[] | null;
  enableLogging?: boolean | null;

  // -------- ANDROID-ONLY --------
  android_icon?: number | null;
  android_pushReceiverModules?: string[] | null;
  android_pushChannelName?: string | null;
  android_pushChannelDescription?: string | null;
};
```

Parameter descriptions:

apiUrl `string` Required: Yes
Description: Altcraft API endpoint URL

---

rToken `string | null` Default: `null`
Required: No
Description: Altcraft role token (identifies a resource / database / account). Used when the only matching

---

appInfo `{ appID: string; appIID: string; appVer: string } | null` Default: `null` Required: No
Description: Basic application metadata.

---

enableLogging `boolean | null` Default: `null`
Required: No
Description: Enables or disables logging.

---

providerPriorityList `string[] | null` Default: `null`
Required: No
Description: Provider priority list (by index: 0 is the highest priority).

Usage examples:

Android

```
providerPriorityList: ['android-firebase', 'android-huawei', 'android-rustore']
```

iOS

```
providerPriorityList: ['ios-apns', 'ios-firebase', 'ios-huawei']
```

---

android_icon `number | null` *(Android only)* Default: `null`
Required: No

Description: `drawable` **resource ID used as the notification icon.**

---

android_pushReceiverModules `string[] | null` ***(Android only)*** **Default:** `null`
Required: No
Description: List of package names containing native `AltcraftPushReceiver` **implementations.**

---

android_pushChannelName `string | null` ***(Android only)*** **Default:** `null`
Required: No
Description: Base name of the push notification channel (visible in Android settings).

---

android_pushChannelDescription `string | null` ***(Android only)*** **Default:** `null`
Required: No
Description: Push channel description (visible in Android settings).

Performing initialization

Use the `initialize(config)` function to initialize the SDK:

```
await AltcraftSDK.initialize({
  apiUrl: 'https://pxl-example.altcraft.com',

  // optional
  rToken: null,
  enableLogging: true,
  providerPriorityList: ['android-firebase', 'android-huawei', 'android-rustore'],

  // android-only (optional)
  android_icon: null,
  android_pushReceiverModules: null,
  android_pushChannelName: null,
  android_pushChannelDescription: null,
});
```

SDK Functionality

Working with subscription statuses

Changing subscription status

```
AltcraftSDK (React Native)
└ pushSubscriptionFunctions
   // Subscribe (status = SUBSCRIBED)
   ├ function pushSubscribe(
   │     sync: boolean | null = true,
   │     profileFields: Record<string, unknown> | null = null,
   │     customFields: Record<string, unknown> | null = null,
   │     cats: CategoryData[] | null = null,
   │     replace: boolean | null = null,
   │     skipTriggers: boolean | null = null
   │   ): void
   // Suspend (status = SUSPENDED)
   ├ function pushSuspend(
   │     sync: boolean | null = true,
   │     profileFields: Record<string, unknown> | null = null,
   │     customFields: Record<string, unknown> | null = null,
   │     cats: CategoryData[] | null = null,
   │     replace: boolean | null = null,
   │     skipTriggers: boolean | null = null
   │   ): void
   // Unsubscribe (status = UNSUBSCRIBED)
   └ function pushUnSubscribe(
         sync: boolean | null = true,
         profileFields: Record<string, unknown> | null = null,
         customFields: Record<string, unknown> | null = null,
         cats: CategoryData[] | null = null,
         replace: boolean | null = null,
         skipTriggers: boolean | null = null
       ): void
```

- `function pushSubscribe()` — **subscribes to push notifications;**
- `function pushUnSubscribe()` — **unsubscribes from push notifications;**
- `function pushSuspend()` — **suspends the push subscription (no notifications are delivered, but no unsubsc**
- `function unSuspendPushSubscription()` — **used to implement** `LogIn` / `LogOut` **transitions.**

All functions in this group share the same signature and parameters:

---

sync: `Boolean`
Default: `true`
Required: No
Description: A flag that sets whether the request is executed synchronously.
Successful request execution:
On success, an event with code 230, 231, or 232 is created. Its `event.value` **depends on the sync flag:**

If `sync == true`

```
ResponseWithHttpCode
  ├ httpCode: 200
  └ response
     ├ error: 0
     ├ errorText: ""
     └ profile
        ├ id: "your id"
        ├ status: "subscribed"
        ├ isTest: false
        └ subscription
           ├ subscriptionId: "your subscriptionId"
           ├ hashId: "7f31a9c4"
           ├ provider: "android-firebase"
           ├ status: "subscribed"
           ├ fields
           │  ├ _os_ver: "{\"raw\":\"14\",\"ver\":\"[\\\"14.0\\\", \\\"0.0\\\", \\\"0.0\\\"]\"}"
           │  ├ _device_type: "mob"
           │  ├ _ad_track: "true"
           │  ├ _device_name: "Pixel"
           │  ├ _os_language: "en"
           │  ├ _os_tz: "+0100"
           │  ├ _os: "Android"
           │  ├ _device_model: "Pixel 7"
           │  └ _app_ver: "1.0.0"
           └ cats
              └ [ { name: "developer_news", title: "dev_news", steady: false, active: false } ]
```

For synchronous requests, the event value `event.value["response_with_http_code"]` contains:

- **httpCode — transport-level status code;**
- **response — an export type containing:**
  - `error: number | null` **— internal server error code *(0 if no errors)*,**
  - `errorText: string | null` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData | null` **— profile data if the request is successful:**
    - **profile info (ProfileData)**
    - **subscription (Subscription)**
    - **if the request fails, only `profile = null` is returned**

▶ **export type**

If sync == false

```
ResponseWithHttpCode
  ├ httpCode: number | null
  └ response: Response | null
     ├ error?: number | null
     ├ errorText?: string | null
     └ profile?: ProfileData | null // always null
```

For asynchronous requests, the event value `event.value["response_with_http_code"]` contains:

- **httpCode — transport-level status code;**
- **response — an export type containing:**
  - `error: number | null` **— internal server error code *(0 if no errors)*,**
  - `errorText: string | null` **— error text *(empty string if no errors)*,**
  - `profile: ProfileData | null` **— always `null` for async requests.**

Failed request execution:

If a request in this group fails, an event with one of the following codes is created:

Operations without automatic retry on the SDK side:

- **430 – notification subscription;**
- **431 — subscription suspension;**
- **432 — unsubscription.**

Operations with automatic retry on the SDK side:

- **530 – notification subscription;**
- **531 — subscription suspension;**
- **532 — unsubscription.**

Event contents:

- **only `httpCode` if the Altcraft server was unavailable;**
- **`error` and `errorText` if the server returned an error.**

Retrieving event values

```
const RESPONSE_CODES = new Set<number>([
  230, 231, 232,
  430, 431, 432,
  531, 532, 533,
]);

subscribeToEvents((event: SdkEvent) => {
  if (event.code == null) return;
  if (!RESPONSE_CODES.has(event.code)) return;

  const raw = event.value?.['response_with_http_code'] as unknown;
  const responseWithHttp = raw as ResponseWithHttpCode | null;
  if (!responseWithHttp) return;

  const httpCode = responseWithHttp.httpCode;

  const response = responseWithHttp.response;
  const error = response?.error ?? null;
  const errorText = response?.errorText ?? null;

  const profile = response?.profile ?? null;
  const profileId = profile?.id ?? null;
  const profileStatus = profile?.status ?? null;
  const profileIsTest = profile?.isTest ?? null;

  const subscription = profile?.subscription ?? null;
  const subscriptionId = subscription?.subscriptionId ?? null;
  const hashId = subscription?.hashId ?? null;
  const provider = subscription?.provider ?? null;
  const subscriptionStatus = subscription?.status ?? null;

  const fields = subscription?.fields ?? null;

  const cats = subscription?.cats ?? null;
  const firstCat = cats?.[0] ?? null;
  const catName = firstCat?.name ?? null;
  const catTitle = firstCat?.title ?? null;
  const catSteady = firstCat?.steady ?? null;
  const catActive = firstCat?.active ?? null;
});
```

---

profileFields: `Record<string, unknown> | null`
Default: `null`
Required: No
Description: An object containing profile fields.
The parameter can include system fields (for example, `_fname` — first name, `_lname` — last name) and optio

- **Scalar values:**
  - **String**
  - **Boolean**
  - **Number**
  - `null`
- **Objects:** `Record<string, unknown>`
- **Lists:** `unknown[]`

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: with field "field_name": Incorrect field
```

---

customFields: `Record<string, unknown> | null`
Default: `null`
Required: No
Description: An object containing subscription fields.
The parameter can include system fields (for example, `_device_model` — device model, `_os` — OS) and optio

- **String**
- **Boolean**
- **Number**
- `null`

If an invalid optional field is passed, the request fails:

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "field_name" is not valid: failed convert custom field
```

> **TIP: Note**
>
> **Most system subscription fields are automatically collected by the SDK and added to push requests. The**

---

cats: `CategoryData[] | null`
Default: `null`
Required: No
Description: Subscription categories.
Category structure:

```
  name?: string | null;
  title?: string | null;
  steady?: boolean | null;
  active?: boolean | null;
};
```

When sending a push request with categories, use only `name` **(category name) and** `active` **(category active**
Request example:

```
const cats: CategoryData[] = [
  { name: 'football', active: true },
  { name: 'hockey', active: true },
];
```

Categories used in the request must be created beforehand and added to the resource in the Altcraft platf

```
SDK error: 430
http code: 400
error: 400
errorText: Platform profile processing error: field "subscriptions.cats" is not valid: category not found in resource
```

---

replace: `boolean | null`
Default: `null`
Required: No
Description: When enabled, all subscriptions of other profiles with the same push token in the current dat

---

skipTriggers: `boolean | null`
Default: `null`
Required: No
Description: When enabled, the profile containing this subscription will be ignored by mailing and scenario

---

Request examples

Example: subscribing to push notifications

Minimal working call:

```
AltcraftSDK.pushSubscribe();
```

Passing all available parameters:

```
AltcraftSDK.pushSubscribe(
  true,
  { _fname: 'Andrey', _lname: 'Pogodin' },
  { developer: true },
  [{ name: 'developer_news', active: true }],
  false,
  false
);
```

> **TIP: Note**
>
> **For** `pushSubscribe`**,** `pushSuspend`**, and** `pushUnSubscribe`**, the SDK automatically retries the request if the HTTP s**

Function `unSuspendPushSubscription()`
The function `async function unSuspendPushSubscription(): Promise<ResponseWithHttpCode | null>` **is designed for** `LogIn`

- **searches for subscriptions with the same push token as the current one that do not belong to the prof**
- **changes the status of the found subscriptions from** `subscribed` **to** `suspended`**;**
- **changes the status of the subscriptions in the profile referenced by the current JWT from** `suspended` **to**
- **returns** `ResponseWithHttpCode | null`**, where** `response.profile` **is the current profile referenced by the JWT (i**

Recommended `LogIn` / `LogOut` flow

`LogIn` **flow**

- **An anonymous user opens the app. The user has `JWT_1`, pointing to database #1Anonymous;**
- **The user subscribes to push; a profile is created in #1Anonymous;**
- **The user registers and receives `JWT_2`, pointing to database #2Registered;**
- **Call `unSuspendPushSubscription()` — the anonymous subscription in #1Anonymous is suspended;**
- **The SDK searches for a profile in #2Registered to restore the subscription;**
- **Since a subscription with this push token does not exist in #2Registered, `unSuspendPushSubscription()` ret**
- **After receiving `null`, call `pushSubscribe()` to create a new profile in #2Registered.**

`LogOut` flow

- **The user logs out in the app (`LogOut`);**
- **The user receives `JWT_1`, pointing to database #1Anonymous;**
- **Call `unSuspendPushSubscription()`, which suspends the subscription in #2Registered and sets the status in**
- **The request returns profile #1Anonymous != null — the subscription already exists, no new one is requ**

Example implementation:

```
async function unSuspend(logIn: boolean): Promise<void> {
  setAuth(logIn);

  const result: ResponseWithHttpCode | null =
    await AltcraftSDK.unSuspendPushSubscription();

  if (!result) {
    AltcraftSDK.pushSubscribe();
    return;
  }

  const httpCode = result.httpCode ?? null;
  const subscription = result.response?.profile?.subscription ?? null;

  if (httpCode === 200 && subscription == null) {
    AltcraftSDK.pushSubscribe();
  }
}

  void unSuspend(true);
}

  void unSuspend(false);
}
```

Requesting subscription status

```
AltcraftSDK
└── function getStatusOfLatestSubscription(): Promise<ResponseWithHttpCode | null>
└── function getStatusForCurrentSubscription(): Promise<ResponseWithHttpCode | null>
└── function getStatusOfLatestSubscriptionForProvider(
       provider: string | null
    ): Promise<ResponseWithHttpCode | null>
```

Subscription status functions:

- **`function getStatusOfLatestSubscription()` — status of the profile's latest subscription;**
- **`function getStatusForCurrentSubscription()` — subscription status for the current token/provider;**
- **`function getStatusOfLatestSubscriptionForProvider()` — status of the latest subscription for a provider. If `null`**

`function getStatusOfLatestSubscription(): Promise<ResponseWithHttpCode | null>`

Retrieves the status of the profile's latest subscription. Returns `ResponseWithHttpCode | null`, containing `respo`

Example:

```
const result = await AltcraftSDK.getStatusOfLatestSubscription();
```

`function getStatusForCurrentSubscription(): Promise<ResponseWithHttpCode | null>`

Retrieves the subscription status for the current token/provider. Returns `ResponseWithHttpCode | null`, contain

Example:

```
const result = await AltcraftSDK.getStatusForCurrentSubscription();
```

`function getStatusOfLatestSubscriptionForProvider(provider: string | null): Promise<ResponseWithHttpCode | null>`

Retrieves the latest subscription status by provider. Returns `ResponseWithHttpCode | null`, containing `response`. `null` is returned.

Example:

```
const result = await AltcraftSDK.getStatusOfLatestSubscriptionForProvider(null);
```

Below is an example of extracting profile, subscription, and category data from the status response. This

Data from status-retrieval functions

```
async function readStatus(): Promise<void> {
  const result = await AltcraftSDK.getStatusForCurrentSubscription();
  if (!result) return;

  const httpCode = result.httpCode ?? null;

  const response = result.response ?? null;
  const error = response?.error ?? null;
  const errorText = response?.errorText ?? null;

  const profile = response?.profile ?? null;
  const subscription = profile?.subscription ?? null;

  const cats = subscription?.cats ?? null;
}
```

Provider push token management

```
AltcraftSDK
|
└   // Get current device token data
  ├──   function getPushToken(): Promise<TokenData | null>
  |
  |  // Save/update token manually for a specific provider
  └──   function setPushToken(provider: string, token: string | null): Promise<void>
```

SDK functions for working with a provider token:

- `function setPushToken()` — sets the device push token and provider;
- `function getPushToken()` — retrieves the current push token.

`setPushToken(provider, token)`

Sets the device push token and provider.

- In React Native, `context` is not passed — token storage and persistence are handled on the native side
- `token: null` clears the token for the specified provider.

Usage example:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

await AltcraftSDK.setPushToken('FCM', token);
```

Example of passing a token when a new token is received:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

async function onNewToken(provider: string, token: string) {
  await AltcraftSDK.setPushToken(provider, token);
}
```

`getPushToken()`

Returns the current device push token data and provider as `TokenData`.

Return format:

- `TokenData` — an object with:
  - `provider: string`
  - `token: string`

If the token is unavailable, `null` is returned.

Usage example:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

const tokenData = await AltcraftSDK.getPushToken();
```

Example of requesting push token data:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

async function readPushToken() {
  const data = await AltcraftSDK.getPushToken();

  const provider = data?.provider ?? null;
  const token = data?.token ?? null;

  return { provider, token };
}
```

Provider push token management

```
AltcraftSDK
|
└   // Get current device token data
  ├──   function getPushToken(): Promise<TokenData | null>
  |
  |  // Save/update token manually for a specific provider
  └──   function setPushToken(provider: string, token: string | null): Promise<void>
```

SDK functions for working with a provider token:

- `function setPushToken()` — **sets the device push token and provider;**
- `function getPushToken()` — **retrieves the current push token.**

`setPushToken(provider, token)`

Sets the device push token and provider.

- **In React Native,** `context` **is not passed — token storage and persistence are handled on the native side**
- `token: null` **clears the token for the specified provider.**

Usage example:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

await AltcraftSDK.setPushToken('FCM', token);
```

Example of passing a token when a new token is received:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

async function onNewToken(provider: string, token: string) {
  await AltcraftSDK.setPushToken(provider, token);
}
```

`getPushToken()`

Returns the current device push token data and provider as `TokenData`.

Return format:

- `TokenData` — **an object with:**
  - `provider: string`
  - `token: string`

If the token is unavailable, `null` is returned.

Usage example:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

const tokenData = await AltcraftSDK.getPushToken();
```

Example of requesting push token data:

```
// or
// import AltcraftSDK from '@altcraft-react-native-sdk/...';

async function readPushToken() {
  const data = await AltcraftSDK.getPushToken();

  const provider = data?.provider ?? null;
  const token = data?.token ?? null;

  return { provider, token };
}
```

Mobile events

```
// AltcraftSDK object function (React Native)

AltcraftSDK
└── function mobileEvent(
        sid: string,
        eventName: string,
        sendMessageId?: string | null,
        payload?: Record<string, unknown> | null,
        matching?: Record<string, unknown> | null,
        matchingType?: string | null,
        profileFields?: Record<string, unknown> | null,
        subscription?: Subscription | null,
        utm?: UTM | null
    ): void
```

Use the `mobileEvent()` function from `AltcraftSDK` to register a mobile event. The function accepts the followi

---

sid: `string`

Required: Yes Description: String identifier of the pixel to which mobile events are bound.

---

eventName: `string`

Required: Yes Description: Name of the mobile event.

---

sendMessageId: `string | null`

Required: No Description: SMID identifier of the sent message (if the event is related to a specific mailing)

---

payload: `Record<string, unknown> | null`

Required: No Description: Event data — a map with string keys. In the RN bridge, values are serialized to

- `string` **— passed as is**
- `number` / `boolean` / `bigint` — `String(value)`
- `object` / `array` — `JSON.stringify(value)`
- `null` / `undefined` **— the key is not passed (the field is removed from payload)**

> TIP: Note
>
> **Non-serializable objects (for example,** `Date` **without conversion, custom classes, functions) are passed as**

---

matching: `Record<string, unknown> | null`

Required: No Description: Map for passing values with matching types and identifiers (for example, `{"email`

---

matchingType: `string | null`

Required: No Description: Matching type.

---

profileFields: `Record<string, unknown> | null`

Required: No Description: Profile fields — a map with string keys. In the RN bridge, values are serialized to

> TIP: Note
>
> **The** `profileFields` **parameter is used only when JWT authorization is enabled.**

---

subscription: `Subscription | null`

Required: No Description: Parameter for adding a subscription for the selected channel.

The parameter value is one of the implementations of the `Subscription` **union type:**

- `EmailSubscription` **— email subscription**
- `SmsSubscription` **— SMS subscription**
- `PushSubscription` **— push subscription**
- `CcDataSubscription` **— subscription for Telegram, WhatsApp, Viber, Notify**

> TIP: Note
>
> **The** `subscription` **parameter is used only when JWT authorization is enabled.**

Common subscription fields (for all implementations)

Field	Type
`type`	`string`
`resource_id`	`number`
`status`	`string | null`
`priority`	`number | null`

Field	Type
`custom_fields`	`{ [key: string]: string } | null`
`cats`	`string[] | null`

Subscription variants

EmailSubscription (`type = "email"`)

Field	Type
`resource_id`	`number`
`email`	`string`

SmsSubscription (`type = "sms"`)

Field	Type	
`resource_id`	`number`	**Yes**
`phone`	`string`	**Yes**

PushSubscription (`type = "push"`)

Field	Type	
`resource_id`	`number`	**Yes**
`provider`	`string`	**Yes**
`subscription_id`	`string`	**Yes**

CcDataSubscription (`type = "cc_data"`)

Field	Type	
`resource_id`	`number`	**Yes**
`channel`	`string`	**Yes**
`cc_data`	`{ [key: string]: string }`	**Yes**

utm: `UTM | null`

Required: No Description: UTM tags. Passed as a `UTM` structure, where each UTM type is a separate field:

```
  campaign?: string | null;
  content?: string | null;
  keyword?: string | null;
  medium?: string | null;
  source?: string | null;
  temp?: string | null;
};
```

Usage examples

```
AltcraftSDK.mobileEvent('your sid', 'app_open');
```

Passing data via payload

```
AltcraftSDK.mobileEvent(
  'your sid',
  'purchase',
  null,
  {
    orderId: 'A-1001',
    amount: 3990,
    currency: 'RUB',
    items: [{ sku: 'sku-1', qty: 2 }],
  }
);
```

Passing UTM tags

```
AltcraftSDK.mobileEvent(
  'your sid',
  'app_install',
  null,
  null,
  null,
  null,
  null,
  null,
  {
    campaign: 'your campaign utm',
    content: 'your content utm',
    keyword: 'your keyword utm',
    medium: 'your medium utm',
    source: 'your source utm',
    temp: 'your temp utm',
  }
);
```

Passing `matching` / `matchingType`

```
AltcraftSDK.mobileEvent(
  'your sid',
  'profile_update',
  null,
  { source: 'settings' },
  { email: 'user@example.com' },
  'email'
);
```

***Passing `subscription` (JWT)**

```
const sub: PushSubscription = {
  type: 'push',
  resource_id: 10,
  provider: 'android-firebase',
  subscription_id: 'provider-sub-id',
  status: 'active',
  priority: 1,
  custom_fields: { utm: 'your utm tag' },
  cats: ['promo', 'news'],
};

AltcraftSDK.mobileEvent(
  'your sid',
  'jwt_bind_subscription',
  null,
  null,
  null,
  null,
  null,
  sub,
  null
);
```

> **TIP: Note**
>
> **When passing `subscription` via the RN bridge, some fields are serialized to strings (for example, `custom_fie`**

Receiving SDK events in the app

In React Native, SDK events are passed from the native layer via `NativeEventEmitter`. Only one active subscr
Events are emitted under the name `AltcraftSdkEvent`. Each event received in JS has the following format:

```
  function: string;                // SDK function name
  code: number | null;             // event code
  message: string;                 // event message
  type: 'event' | 'error' | 'retryError';
  value?: Record<string, unknown> | null; // extra payload
};
```

Event types:

- `event` **— informational events and successful operations;**
- `error` **— execution errors;**
- `retryError` **— errors of operations that the SDK retries automatically.**

Subscribing to events

To subscribe, use `subscribeToEvents`.
When you subscribe again, the previous subscriber is automatically removed.

```
subscribeToEvents((event) => {
  switch (event.type) {
    case 'event':
      // handle success event
      break;

    case 'error':
      // handle error
      break;

    case 'retryError':
      // handle retryable error
      break;
  }
});
```

Unsubscribing from events

To stop receiving events, use `unsubscribeFromEvent`.

```
unsubscribeFromEvent();
```

After calling it, the subscriber is removed and SDK events are no longer delivered to JavaScript.

List of all SDK events

Event list

Code	
200	**SDK configuration is installed**
201	**push provider set**
202	**received a notification unrelated to the Altcraft Platform**
203	**received Altcraft push notification**
204	**push is posted**
205	**SDK data has been cleared**
206	**waiting for SDK initialization to complete**
207	**waiting for push token update to complete**
220	**receiver override event**
230	**subscribe request succeeded**
231	**push suspend request succeeded**
232	**push unsubscribed request succeeded**
233	**push update request succeeded**
234	**push unsuspend request succeeded**
235	**profile status request succeeded**
236	**push event delivered successfully**
237	**mobile event delivered successfully**
401	**the configuration is not set**
402	**userTag is null. It is impossible to identify the user**
404	**SDK initialization timeout has expired**
405	**mobile event parts is null**
422	**unsuspend request data is null**
423	**profile request data is null**
430	**subscribe request failed**
431	**suspend request failed**
432	**unsubscribed request failed**
433	**update request failed**
434	**unsuspend request failed**
435	**status request failed**
436	**push event delivery failed**
437	**mobile event request failed**
450	**push data is null**
451	**uid in the push data is null or empty, it is impossible to send a push event to the server**
452	**error uploading the notification image**
453	**couldn’t create notification**
454	**foreground info is null**
455	**the notification channel has not been created**
471	**invalid provider. Available - android-firebase, android-huawei, android-rustore**
472	**invalid customFields: not all values are primitives**
480	**subscribe retry limit reached (request removed from DB)**
484	**push event retry limit reached (request removed from DB)**
485	**mobile event retry limit reached (request removed from DB)**
501	**config data is null**
502	**current push token is null**
503	**userTag is null. It is impossible to identify the user**
504	**no permission to send notifications**
505	**no internet connection, retry when connection is restored**
506	**failed to update the push token. The subscription request was rejected**
520	**push subscribe request data is null**
521	**token update request data is null**
524	**push event request data is null**
525	**mobile event request data is null**
529	**common data is null**
530	**subscribe request failed (retryable)**
531	**token update request failed (retryable)**
534	**push event delivery failed (retryable)**
535	**mobile event delivery failed (retryable)**

Code	
540	JWT token is null
541	matching mode is null
542	JWT payload exceeds allowed size (16 KB limit): input rejected to prevent DoS
543	JWT does not contain a payload
544	auth data is null
545	matching claim does not contain a matching ID
560	response data is null

Additional SDK functions

Clearing SDK data

```
AltcraftSDK.clear();
```

The function clears SDK data and cancels all background tasks that are waiting to be executed.

Manual push event registration

> **TIP: Note**
>
> **Use these functions only if you implement your own notification handling logic and do not pass notificati**

Android

In React Native, manual push event registration functions are available on the JS side:

- `deliveryEvent(message?: Record<string, string> | null, messageUID?: string | null)` **— registers notification deliv**
- `openEvent(message?: Record<string, string> | null, messageUID?: string | null)` **— registers notification open.**

Parameters:

- `message: Record<string, string> | null` **— notification data payload as a dictionary of strings.**
- `messageUID: string | null` **— message identifier.**

Passing rules:

- **you can pass only** `message` **(the SDK will take the message ID from the payload if it exists there);**
- **you can pass only** `messageUID` **(if the payload is unavailable but the ID is known);**
- **if both parameters are** `null`**, the event will not be registered correctly (no data to identify the messag**

iOS

Manual push event registration on the React Native side is not implemented. If manual event registration
See the iOS guide (manual push event registration).

Public SDK API

default export AltcraftSDK

```
AltcraftSDK
// SDK initialization and configuration setup
├─ function initialize(config: AltcraftConfig): Promise<void>
// Full SDK data cleanup
├─ function clear(): Promise<void>
// Request notification permissions (platform-dependent)
├─ function requestNotificationPermission(): void

// Push subscription (fire-and-forget)
├─ function pushSubscribe(
│    sync?: boolean | null,
│    profileFields?: Record<string, unknown> | null,
│    customFields?: Record<string, unknown> | null,
│    cats?: CategoryData[] | null,
│    replace?: boolean | null,
│    skipTriggers?: boolean | null
│  ): void
├─ function pushSuspend(...): void
└─ function pushUnSubscribe(...): void

// Subscription status (async)
├─ function unSuspendPushSubscription(): Promise<ResponseWithHttpCode | null>
├─ function getStatusOfLatestSubscription(): Promise<ResponseWithHttpCode | null>
├─ function getStatusOfLatestSubscriptionForProvider(provider?: string | null): Promise<ResponseWithHttpCode | null>
└─ function getStatusForCurrentSubscription(): Promise<ResponseWithHttpCode | null>

// SDK events (single JS-side subscriber)
├─ function subscribeToEvents(handler: (event: SdkEvent) => void): void
└─ function unsubscribeFromEvent(): void

// Mobile event sending
└─ function mobileEvent(
sid: string,
eventName: string,
sendMessageId?: string | null,
payload?: Record<string, unknown> | null,
matching?: Record<string, unknown> | null,
matchingType?: string | null,
profileFields?: Record<string, unknown> | null,
utm?: UTM | null
): void

// Push provider token
├─ function setPushToken(provider: string, token: string | null): Promise<void>
└─ function getPushToken(): Promise<TokenData | null>

// Android: pass incoming push payload to SDK
└─ function takePush(message: Record<string, string>): void

// Manual push event registration (delivery / open) on RN side
├─ function deliveryEvent(message?: Record<string, string> | null, messageUID?: string | null): void
└─ function openEvent(message?: Record<string, string> | null, messageUID?: string | null): void

// iOS: write value to UserDefaults / App Group
└─ function setUserDefaultsValue(suiteName: string | null, key: string, value: unknown): void
```

Event Classes

```
SdkEvent
// Universal SDK event delivered to JS via NativeEventEmitter.
function: string
code: number | null
message: string
type: 'event' | 'error' | 'retryError'
value?: Record<string, unknown> | null
```

Exported Types

```
// Subscription category
  name?: string | null;
  title?: string | null;
  steady?: boolean | null;
  active?: boolean | null;
};

// Device push provider token
provider: string | null;
token: string | null;
};

// Application metadata
appID: string;
appIID: string;
appVer: string;
};

// API response with HTTP status code
httpCode: number | null;
response: Response | null;
};

error?: number | null;
errorText?: string | null;
profile?: ProfileData | null;
};

id?: string | null;
status?: string | null;
isTest?: boolean | null;
subscription?: SubscriptionData | null;
};

subscriptionId?: string | null;
hashId?: string | null;
provider?: string | null;
status?: string | null;
fields?: { [key: string]: string } | null;
cats?: CategoryData[] | null;
};

// UTM parameter container
campaign?: string | null;
content?: string | null;
keyword?: string | null;
medium?: string | null;
source?: string | null;
temp?: string | null;
};
```

AltcraftConfig

```
  apiUrl: string;

  rToken?: string | null;
  appInfo?: AppInfo | null;
  providerPriorityList?: string[] | null;
  enableLogging?: boolean | null;

  // -------- ANDROID-ONLY --------
  android_icon?: number | null;
  android_pushReceiverModules?: string[] | null;
  android_pushChannelName?: string | null;
  android_pushChannelDescription?: string | null;
};
```

Provider setup

Provider setup follows the instructions for the native SDKs of each platform. More details are available in

Android:

- **Firebase Cloud Messaging**
- **Huawei Mobile Services**
- **RuStore**

iOS:

- **Apple Push Notification Service**
- **Firebase Cloud Messaging**
- **Huawei Mobile Services**

Administrator documentation: Contents

Contents of this section.

Getting Started

Functional characteristics

System requirements

Processing HTTP/HTTPS traffic

Documentation Archive

Architecture Overview

Component Description

Basic scheme

Fail-safe scheme

Typical Placement in Infrastructure

Nodes

Tariffs

External data configuration

LDAP

Tasks

Schedule Jobs

Global Stop Lists

Webversion Store Policies

Accounts

Account Users

Account Virtual Senders

Account Database Indexes

Databases

Notifiers

Admins

API tokens

MTA

Default rules

Retry rules

Lock rules

Bounce patterns

Strategies

Keys

ISPs

Pools

Email

SMS

Event generator

Integration with Sendsay

Audit Journal

Data Usage

ARF decoder

URL decoder
SMID decoder
License
Manual installation
Running the platform in a Docker container
Configuration file
Domain settings
LDAP access configuration
Sending Email via SMTP relay
Pixel and push domain configuration
Cluster and Replication Setup
System notifications configuration
Processes UNIX sockets configuration
HTTPS Configuration
Migrating from MongoDB Community Edition to Percona Server for MongoDB
Adding sender IP addresses
Deduplication request settings
PostgreSQL database for account data
Proxy server settings
Keycloak Integration with Altcraft
Getting HTTP service statuses
Configuration MongoDB logs rotation
Configuration of system constants and directories
Filtering bot actions
Directory of gender markers
Personnel requirements
Platform maintenance processes
Platform updating
Platform service monitoring
RabbitMQ migration
Backup and recovery
Transferring the platform to a new server
Creating, deleting, and populating tables for statistics in ClickHouse
Usage of the aktool utility
Users and directories engaged by the platform
Overview
Creating and configuring a channel
Entity field configuration
Template language
Entities
Pipe: Basic Concepts
Result
Errors
Log

Pack
Unpack
HTTP Request
Store Set
Store Get
Selector
SQL
Eventer
Scheduler
Pipeline: Basic Concepts
Message
Schedule
Listener
Moderate
Stop
System page customization
Send Message ID
Instructions for migrating history to ClickHouse
Utility for importing push subscriptions to Firebase project
Utility for importing push subscriptions to Firebase project
Administrator API
Get accounts list
Add a new account
Delete the account
Update an Existing Account
Add a new account
Delete a user
Get a list of users
Sending a Welcome Email
Synchronize node MTA configuration
Get nodes list
Get node MTA status
Activate node MTA
Deactivate node MTA
Create or update AKMTA sender
Get AKMTA sender information
Assign account to sender
Get senders list
Delete sender
Restore sender
Get sender queue information
Hold sender queue
Release sender queue
Clear sender queue

Get virtual senders list

Get virtual sender information

Create virtual sender

Update virtual sender

Clone virtual sender

Delete virtual sender

Getting Started

This article will help you start using the Altcraft Marketing Platform. It provides a standard workflow for i

Technical Requirements and Staffing Requirements

> **TIP**
>
> **You can read more about the technical requirements in this article of the documentation.**

Verify that your systems configuration meets the requirements necessary for running the platform. Keep i

Hardware Requirements

Server / Resource
Customer database size
Daily delivery volume
Management server — CPU
Management server — RAM
Management server — Disk
Sending server — CPU
Sending server — RAM
Sending server — Disk

*** Assumes core allocation without considering hyper-threading level of Intel Xeon Gold 6240R or Intel Xec**
**** Additional memory allocation may be required when building suboptimal segments and automation mec**
***** Assumes write speed availability of 80 MB/s for HDD, 450 MB/s for SSD SATA, 1500 MB/s for NVME disk limited.**
****** The disk size is calculated for an average Email message size of 70 KB and a message lifetime of up**

Software Requirements

Management and sending servers are guaranteed to work in the following environments:

- **Ubuntu Server 18.04 LTS**
- **Ubuntu Server 20.04 LTS**
- **Ubuntu Server 22.04 LTS**

When using Docker containers (see Running the platform in a Docker container):

- **CentOS 7+**
- **Red Hat Enterprise Linux 7+**
- **Debian 9+**
- **Astra Linux Common Edition 1.7+**
- **Astra Linux Special Edition 1.8+ (**

  certified)
- **SUSE Linux Enterprise Server 12+**
- **RED OS 7.3+ (**

  certified)

During installation, the following open-source software is used:

Software	
MongoDB	**8.0**
PostgreSQL *	**16.1**
RabbitMQ	**3.12.9 (4.0+ also supported)**
Kvrocks	**Latest**
ClickHouse	**24.8+ (LTS)**
SSDB	**1.8.0**
xvfb	**Any current version**
cutycapt	**Any current version**

*** PostgreSQL is optional and required when using the “Market” section.**
**** The ability to install this software from official repositories must be ensured during system installation**

Staff Requirements

Specialists responsible for monitoring platform performance, administration, and infrastructure support sl

- **technical education;**
- **knowledge of TCP/IP protocols;**

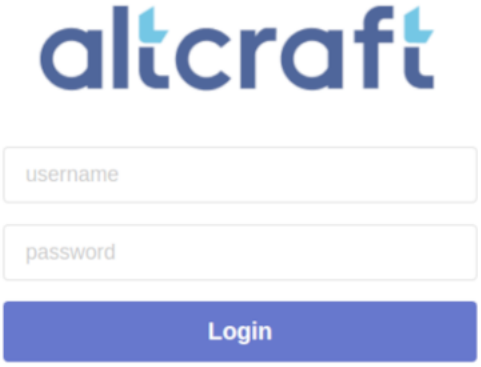
- **understanding of basic DBMS principles;**
- **at least 1 year of Linux administration experience;**
- **knowledge of the platform's functionality;**
- **experience working with MongoDB;**
- **experience working with ClickHouse;**
- **knowledge of database recovery tools and server performance monitoring tools.**

Platform Installation

To work with Altcraft, you need to install it on your server. You can read more about all installation metho
The platform is distributed as a compressed archive with the `.tar.gz` extension.

Accessing the Admin Panel

The Admin Panel address is set in the `main.json` configuration file located at `/opt/akd/config/`:
`{ADMIN_WEBCONTROL_LOCAL_IP}:{ADMIN_WEBCONTROL_LOCAL_PORT}`
or
`{WEBCONTROL_HOSTNAME}:{WEBCONTROL_PUBLIC_PORT}/admin`



After installation, an administrator account is automatically created:
Username: `admin`
Password: `admin`

> CAUTION
> **After the first login, you must change the password for the `admin` account.**

Configuring the Platform in the Admin Panel

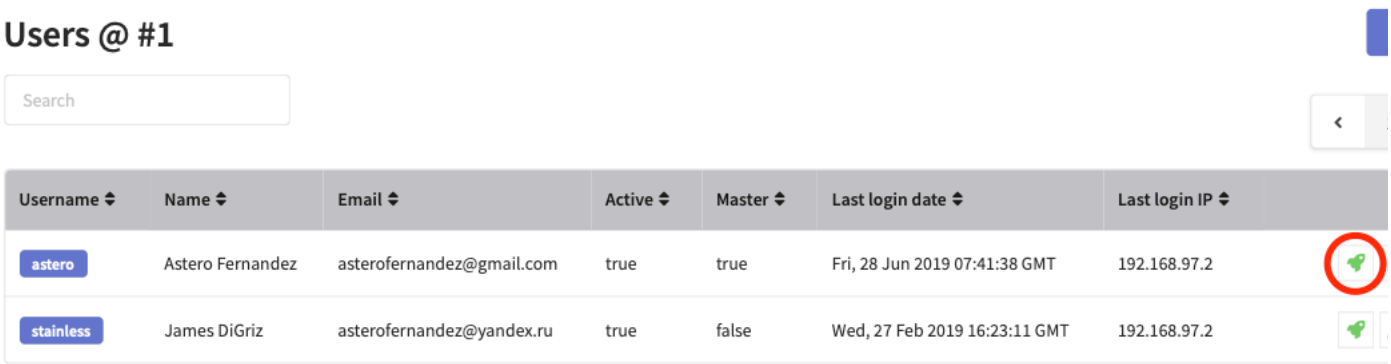
To ensure correct platform operation, you must configure several required entities in the Admin Panel. Th
In the Admin Panel, you need to create:

- **Sender — configuration of the messaging service. Senders are configured for email and SMS channels**
- **Node — the network interface of the sending server used by AKMTA senders. A node can be on the sa**
- **Tariff — a set of restrictions for a platform account. The Altcraft Marketing Platform license agreeme**
- **Account — the workspace for marketers and related specialists working with the Altcraft User Interfa**

Logging in to the User Interface

Access from the Admin Panel

You can log into an account directly from the Altcraft Admin Panel. To do this, pop-ups must be allowed in



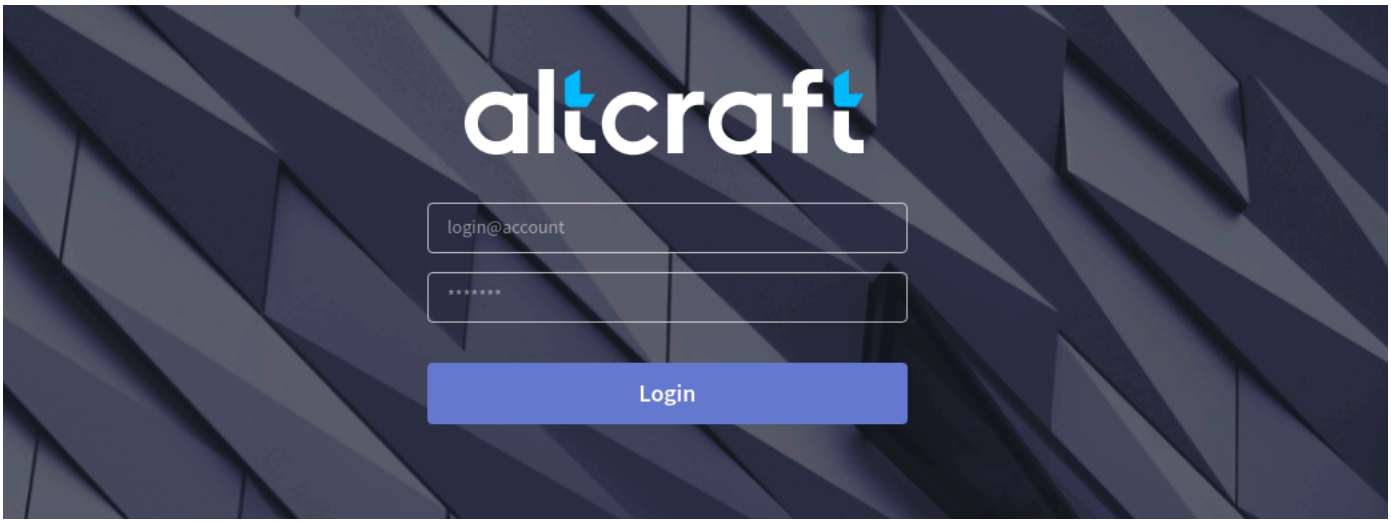
Access via Link

The User Interface of an account is available at the address set in the main.json configuration file:

`{WEBCONTROL_HOSTNAME}:{WEBCONTROL_PUBLIC_PORT}/`

or

`{WEBCONTROL_PUBLIC_IP}:{WEBCONTROL_PUBLIC_PORT}/`



To log in, enter your credentials in the appropriate fields.

Functional characteristics

Brief description

The software is intended to automate the collection, processing, storage and management of customer d

Operational purpose

The software is intended for the following end users:

- **CRM managers**
- **Marketers**
- **Analysts**

Functional characteristics

Customer data

- **Collecting data and building a unified customer profile**
- **Creating the structure of the database of client profiles**
- **Client's card**
  - **Editing profile fields**
  - **Client multi-subscription management**
  - **View customer activity history**
  - **Interaction with the profile:**
    1. **add a relation**
    2. **add to segment**
    3. **add to script**
    4. **attach a promo code**
    5. **send transactional message**
- **Client database management**
- **Import clients in the following ways:**
  - **Manually**
  - **From the clipboard**
  - **From file (csv, xls, xlsx, txt)**
  - **Via API**
  - **On schedule from external SQL databases**
- **Copying customer profiles from one database to another**
- **Database export to file**
- **Collecting customer activity in all channels and on the site**

Audiences

- **Creating and editing clients' audiences**
- **Forming dynamic or fixed audience**
- **Targeting the audience according the following data:**
  - **Personal and demographic data**
  - **Behavioral data, activity history**
  - **Transactional message**
  - **Data on interests and registered goals**
  - **Data on client devices**
  - **Multiple subscriptions and identifiers from different channels in the unified customer profile**
  - **Data on participating in loyalty programs**
  - **Data on relationship between different customers**
- **Client audience export to file**
- **Synchronization of the client audience with external SQL databases**

Content

- **Universal editor for multichannel messages**
- **Personalization in messages:**
  - **Substitution of data from the built-in database of client profiles**
  - **Substitution of data from external SQL databases**
  - **Substitution of data from external HTTP services**
- **Support for boolean conditions and many other operators in message templates**
- **Email message drag-and-drop builder**

- Rich Push Message Builder
- SMS editor
- Professional HTML editor for Email messages
- Generating message fragments for reuse in templates

Omnichannel communications

- Bulk sending and delivery of messages to channels:
  - Email
  - SMS
  - Telegram
  - Web Push
  - Mobile Push
  - Viber
  - Notify
  - WhatsApp*
  - Client's personal account
  - Custom channel
- Different types of messaging:
  - Disposable
  - Regular
  - Transactional / Triggered
- A/B/n and MVT testing
- Customizable newsletter calendar
- Ability to add a database of observers to the mailing list
- The ability to set limits on mailing and automatically increase them

Scenarios

- Builder of automated scenarios for customer interaction
- Automatically add a client to the scenario by event
- The following automation nodes are available:
  - Pause
  - Run a trigger mailing
  - Run a placement mailing
  - Check for condition
  - Import to database
  - Update customer profile
  - Manage subscriptions in a customer profile
  - Send HTTP(s) request to any API
  - Registering a goal in a customer profile
  - Best send time
  - Allocation node
  - Filter by data passed via API
  - Attach a promo code
  - Add / Remove from suppression list
  - Add / Remove from static segment

Retargeting

- Facebook* audience synchronization on schedule
- Google Ads audience synchronization on schedule
- Yandex.Audience audience synchronization on schedule
- VK Ads audience synchronization on schedule

Loyalty programs

- Promo codes
  - Generation of promo codes for a given template
  - Automatic attaching of promo codes in personal messages
  - Fetching promo codes from an external service
  - Import from file
  - Export to file

Forms and surveys

- Form&Survey Constructor

- Managing colors and styles of the form
- Publishing the form on the website or sharing it via URL
- Automatic collection of applications through the form posted on the website
- Viewing and downloading survey results
- NPS calculation

Pop-ups

- Easy-to-use editor for creating pop-ups
- 4 types of pop-ups and 18 ready-made templates
- Customization of pop-ups in the code editor
- Placement and management of pop-ups on the website through the Tag Manager
- Manual management of pop-ups through scripts
- Automatic collection of applications through the pop-up posted on the website
- Valuable insights with visual analytics

Message sending

- High-performance module for sending Email, Push and other messages
- Managing the rules for sending messages to different Email services
- Managing the rules for resending messages to different Email services
- Distributed sending across multiple servers
- View and manage sending queues

Tracking

- Tracking website visitors
- Collecting the goal events on the website
- Tracking customer actions in the message
- Tag manager for placing third-party scripts on the website

Analytics

- Channel report
- Traffic report
- Audience growth report
- Cohorts report
- LTV report
- Funnel
- Goals report
- Bounces report
- Undeliveries report
- Click map report (Email messages)

Security

- Group and role-based access rights policies
- Configurable password policy
- User actions logging
- Connecting an LDAP server for authorization
- Configurable sending policies for mailings
- Two-factor authentication when logging into the platform

Integrations

- Fetching events from Yandex Appmetrica
- Sending Push Messages via Google Firebase, Apple Notification Service, RuStore, Huawei Mobile Serv
- Integration with Amazon SNS
- Receiving applications from LPGenerator, Tilda
- Sending SMS via SMSCenter, BulkSMS, Svyaznoy Loading, Devino Telecom, SMSTraffic, Infobip, MTS, S
- Sending any events to external APIs through the WebHooks mechanism

mSDK

Altcraft mSDK is a mobile programming interface (SDK) for integrating push notifications into mobile apps

- Automatically subscribe users to push notifications

- **Support sending notifications with text, images and action buttons**
- **Update and refresh device tokens on the server for seamless notification delivery**
- **Track notification delivery and user interaction with notifications**
- **Automatically retry failed requests and work in the background**

Market

- **Importing data about products and orders into the market platform (via file or through API)**
- **Customer segmentation based on purchase history**
- **Adding dynamic content to messages (order items, cost)**
- **Sending trigger messages when the order status changes**

Interface

To work with the platform, the user needs a device (personal computer or phone) with a browser. The plat found in "User documentation".

Interface language

The platform is designed to support multiple user languages. Now two languages are available on the plat

- **English**
- **Russian**

If necessary, it's possible to add support for any language as soon as possible.

** The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as*

System requirements

User PC requirements

Operating systems

- **Windows Vista/7/8/10**
- **Mac OS X 10.5 - 11.x**
- **Linux, any family OS**

Browsers

- **Google Chrome, version 87 and up**
- **Firefox, version 83 and up**
- **Microsoft Edge, version 79 and up**
- **Opera, version 58 and up**

Hardware requirements

Altcraft platform can be installed on a physical server as well as on a virtual machine.
It is necessary to choose hardware based on the specific needs of your tasks. To clarify the requirements

Server / Resource
Customer database size
Daily message sending volume
Control server - CPU
Control server - RAM
Control server - Disk
Sending server - CPU
Sending server - RAM
Sending server - Disk

*** Assumes core allocation without considering hyper-threading level of Intel Xeon Gold 6240R or Intel Xec**
**** Additional memory allocation may be required when building suboptimal segments and automation mec**
***** Assumes write speed availability of 80 MB/s for HDD, 450 MB/s for SSD SATA, 1500 MB/s for NVME disk limited.**
****** The disk size is calculated for an average Email message size of 70 KB and a message lifetime of up**

For cluster deployment

For cluster deployment, at least a threefold increase in resources will be required, taking into account the

Infrastructure requirements

A frequent question concerns public IPv4 addresses, so the requirement is explained as a separate parag
In its standard configuration, the AKD server uses three public (white) IP addresses. One is intended for tl web, as well as for implementing web layer functions, displaying forms, and popups. An SSL certificate wi
The AKMTA email sender can use the same IP addresses as the control host for operation if it is hosted di determination, consultation with a marketer specializing in email mailings is required. For a volume of up
Here and in the following, we are talking about IPv4 addresses, as IPv6 is currently not fully supported by

License access requirements

To operate the application, continuous access to Altcraft license servers must be organized:

- **lic1.altkraft.com:443 TCP**
- **lic2.altkraft.com:443 TCP**
- **lic3.altkraft.com:443 TCP**

Access can be organized through a proxy server and must be provided with each platform update and lice

Software requirements

The control and sending servers are designed to work reliably in the following environments:

- **Ubuntu Server 22.04 LTS**
- **Ubuntu Server 24.04 LTS**

The platform is compatible with the following operating systems:

- **CentOS 7+**
- **Red Hat Enterprise Linux 7+**
- **Debian 9+**

- **Astra Linux Common Edition 1.7+**
- **Astra Linux Special Edition 1.8+ (**

  certified)
- **SUSE Linux Enterprise Server 12+**
- **RED OS 7.3+ (**

  certified)

INFO

You can install it either directly on the system or using containers (see Running the platform in a Docker

The following open-source software is used during the system installation:

Software	
MongoDB	**8.0**
PostgreSQL *	**16.1**
RabbitMQ	**3.12.9 (4.0+ versions are also supported)**
ClickHouse	**24.8+ (LTS)**
Kvrocks	**Latest**
SSDB	**1.8.0**
xvfb	**Latest**
cutycapt	**Latest**

*** PostgreSQL is not mandatory and is required to work with the “Market” section.**

**** The ability to install this software from official repositories must be ensured during system installation**

TCP/IP port access requirements

AKD Ports (Control module)

Description
RabbitMQ Message Broker - Message Exchange
RabbitMQ Message Broker - control
Tracking user actions in messages (dedicated IP)
Control panel (dedicated IP)
Tracking actions in web and web layer (dedicated IP)
MongoDB for storing all system configurations
PostgreSQL for using the “Market” section
SSDB for storing email addresses that cannot be delivered
SSDB for storing notifications
ClickHouse for storing profile history
Notification of reports and security system
Kvrocks for checking duplicates (contacts) in mailings

AKMTA ports (Sending module)

Description
For send messages
For receive and process complaints and FBL
For auto remote interact with the sender

*** All ports can be reassigned.**

**** The choice of HTTPS or SSH senders management method is optional.**

Processing HTTP/HTTPS traffic

NOTE

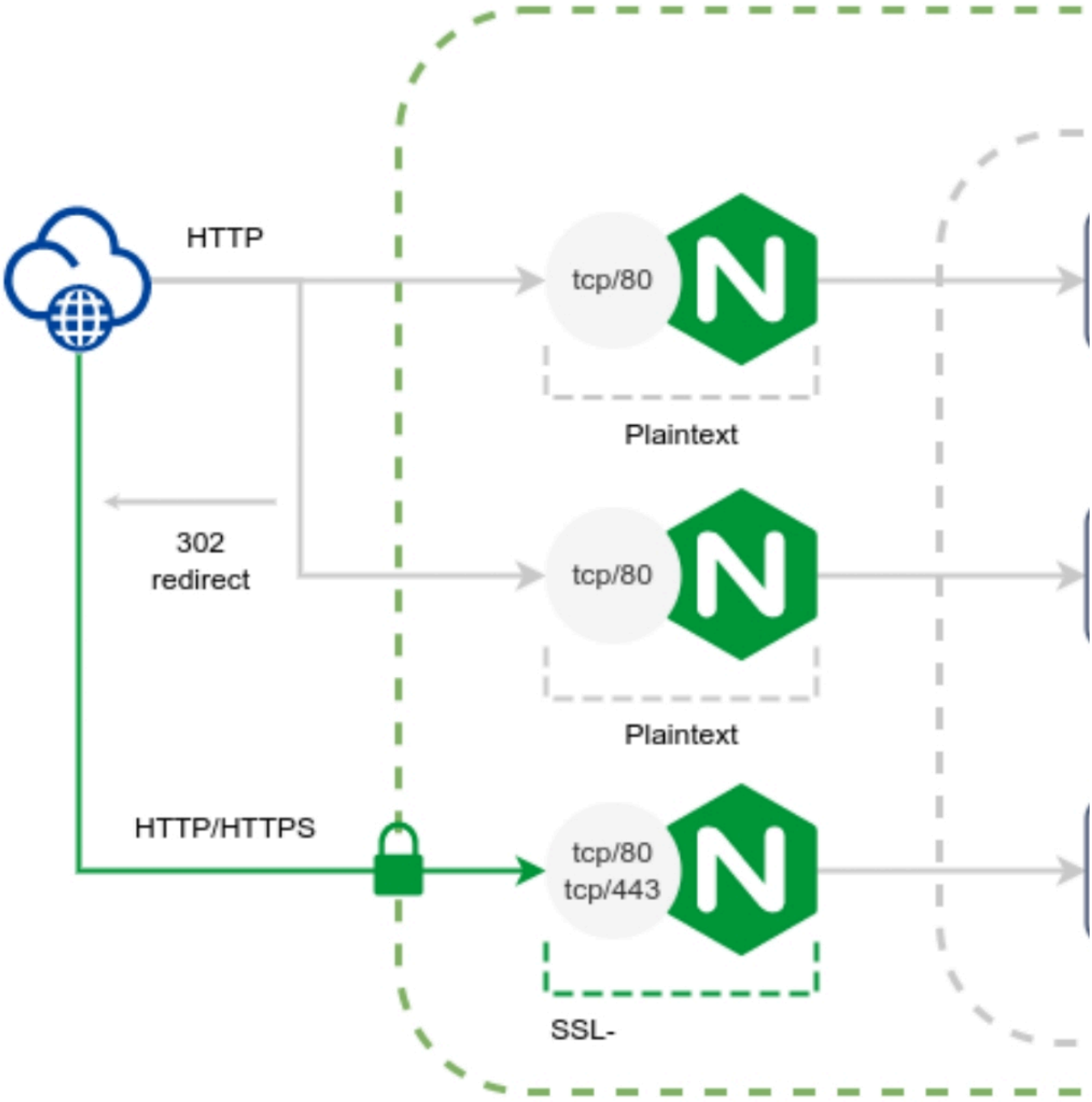
For a complete list of all configurable parameters, see the section - Configuration file. To apply changes

TIP

To get free Let’s Encrypt certificates, you can use the certbot utility.

Default traffic processing scheme on the Altcraft side

By default, HTTPS is required for pixels to work and custom cookies to be processed. Setting the COOKIE

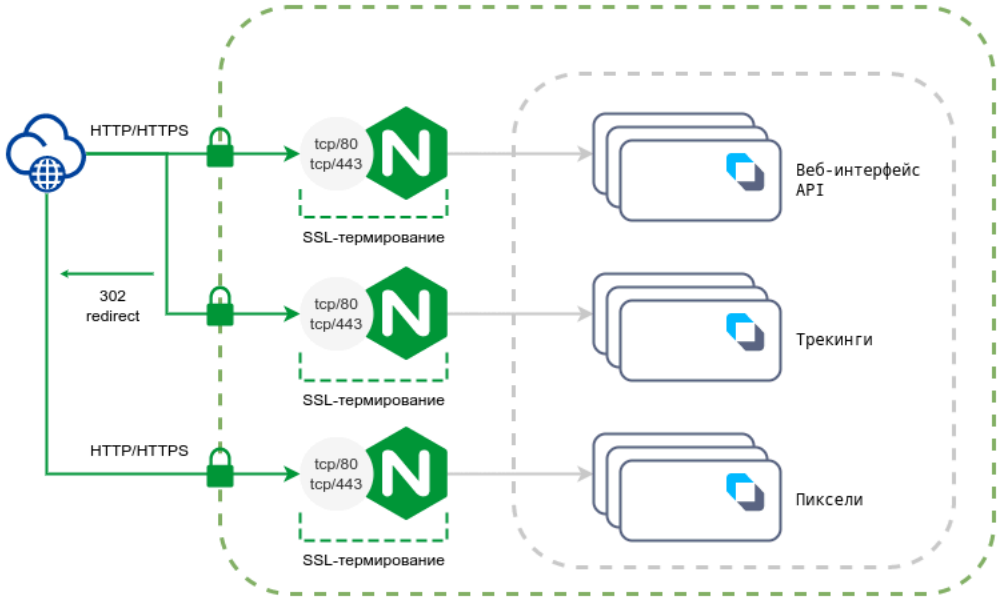


Required parameters

Parameter	Type	
SSL_ON	**bool**	`false`
TRACKING_SSL_ON	**bool**	`false`
COOKIESAVER_SSL_ON	**bool**	`true`

Parameter	Type	
COOKIESAVER_SSL_CRT_PATH	**string**	`/etc/letsencrypt/live/pxl.examp`
COOKIESAVER_SSL_KEY_PATH	**string**	`/etc/letsencrypt/live/pxl.examp`

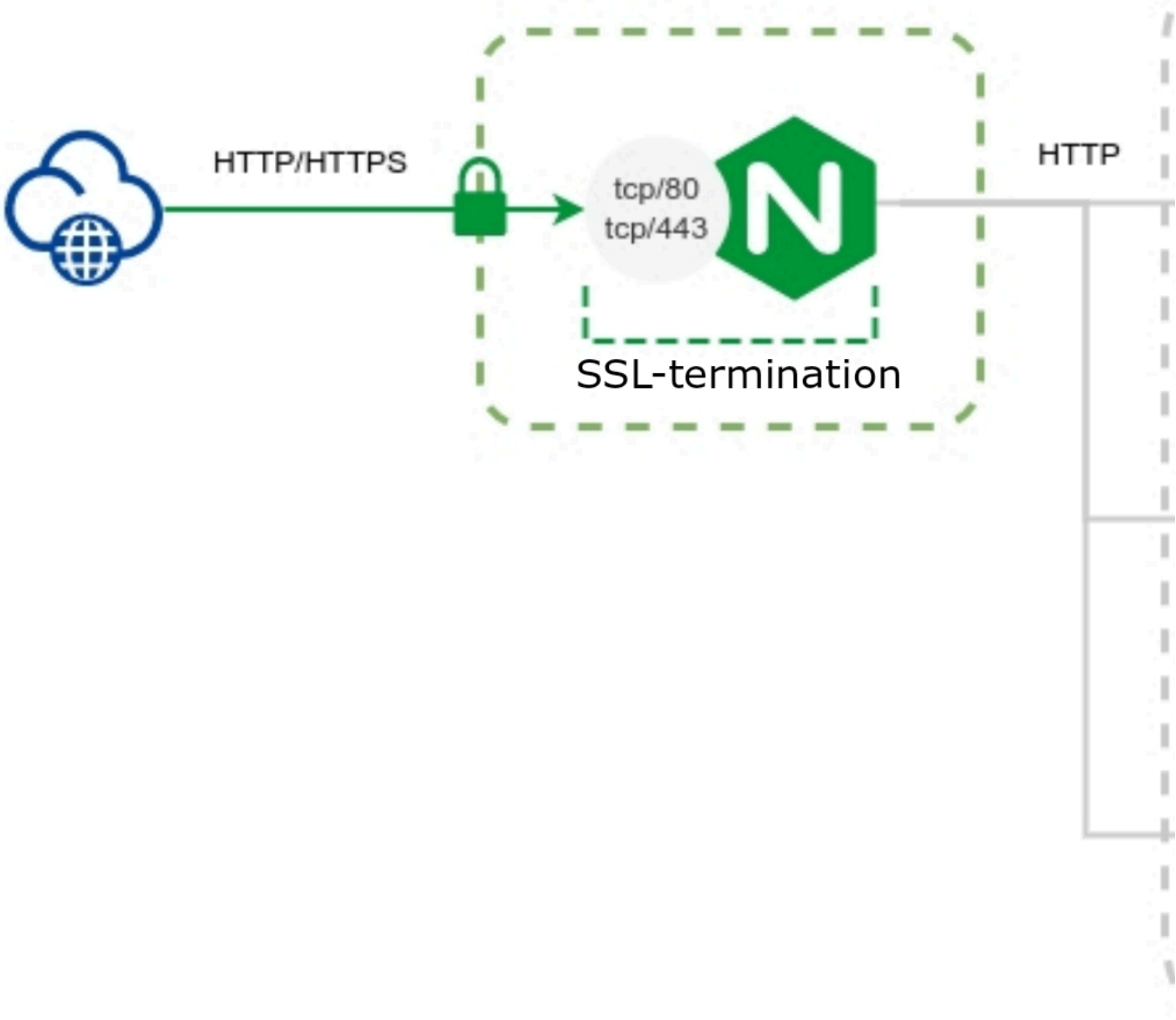
Scheme for processing traffic with full SSL termination on the Altcraft side



Required parameters

Parameter	Type	
SSL_ON	**bool**	`true`
SSL_CRT_PATH	**string**	`/etc/letsencrypt/live/altcraft.example.com/fullch`
SSL_CRT_KEY	**string**	`/etc/letsencrypt/live/altcraft.example.com/privke`
TRACKING_SSL_ON	**bool**	`true`
TRACKING_SSL_CRT_PATH	**string**	`/etc/letsencrypt/live/click.example.com/fullchain`
TRACKING_SSL_CRT_KEY	**string**	`/etc/letsencrypt/live/click.example.com/privkey.p`
COOKIESAVER_SSL_ON	**bool**	`true`
COOKIESAVER_SSL_CRT_PATH	**string**	`/etc/letsencrypt/live/pxl.example.com/fullchain.p`
COOKIESAVER_SSL_KEY_PATH	**string**	`/etc/letsencrypt/live/pxl.example.com/privkey.pem`

Traffic processing scheme with full SSL termination on the external side



Required parameters

Parameter	Type	
SSL_ON	**bool**	**false**
TRACKING_SSL_ON	**bool**	**false**
TRACKING_SSL_EXTERNAL_ON	**bool**	**true**
COOKIESAVER_SSL_ON	**bool**	**false**
NGINX_TRUSTED_PROXIES	**array**	**[“Внешний IP-адрес про**
NGINX_REALIP_HEADER	**string**	**“X-Forwarded-For”**

External NGINX Configuration Example

```
server {
    listen 80;
    listen 443 ssl;
    server_name altcraft.example.com;

    if ($scheme != "https") { return 307 https://$host$request_uri; }
    ssl_certificate /etc/letsencrypt/live/altcraft.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/altcraft.example.com/privkey.pem;

    ssl_session_cache shared:le_nginx_SSL:10m;
    ssl_session_timeout 1440m;
    ssl_session_tickets off;

    ssl_protocols TLSv1.2 TLSv1.3;

    location / {
        proxy_pass http://172.16.0.10:80;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_http_version 1.1;
        proxy_read_timeout 900;
    }
}

server {
    listen 80;
    listen 443 ssl;
    server_name click.example.com;

    if ($scheme != "https") { return 307 https://$host$request_uri; }
    ssl_certificate /etc/letsencrypt/live/click.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/click.example.com/privkey.pem;

    ssl_session_cache shared:le_nginx_SSL:10m;
    ssl_session_timeout 1440m;
    ssl_session_tickets off;

    ssl_protocols TLSv1.2 TLSv1.3;

    location / {
        proxy_pass http://172.16.0.11:80;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_http_version 1.1;
        proxy_read_timeout 900;
    }
}

server {
    listen 80;
    listen 443 ssl;
    server_name pxl.example.com;

    if ($scheme != "https") { return 307 https://$host$request_uri; }
    ssl_certificate /etc/letsencrypt/live/pxl.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/pxl.example.com/privkey.pem;

    ssl_session_cache shared:le_nginx_SSL:10m;
    ssl_session_timeout 1440m;
    ssl_session_tickets off;

    ssl_protocols TLSv1.2 TLSv1.3;

    location / {
        proxy_pass http://172.16.0.12:443;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_http_version 1.1;
        proxy_read_timeout 900;
    }
```

```
}
```

Documentation Archive

With the release of new versions of Altcraft Platform, the documentation presented on the website is upd documentation, user documentation, and the API reference combined into a single PDF file.

- **Version 75 — PDF**
- **Version 74 — PDF**
- **Version 73 — PDF**
- **Version 72 — PDF**
- **Version 71 — PDF**

INFO: Important

The documentation versions listed above may contain factual inaccuracies. No changes are made to arc

Architecture Overview

Description

**Altcraft is a scalable marketing automation platform built on a microservice architecture. The microservi
Some parts of the system can be distributed across network zones (e.g., Internet, DMZ, Data Storage) for
The system requires the following components:**

- **MongoDB — the primary database for storing accounts, users, and settings;**
- **ClickHouse — stores events, activity history, and analytics;**
- **RabbitMQ — message broker for communication between services;**
- **SSDB and Kvrocks — high-performance key-value stores for auxiliary data;**
- **PostgreSQL (optional) — used in the "Market" module to store products and orders;**
- **other standard components available in OS repositories.**

**User interaction with the system is carried out via a web interface, while traffic routing and SSL terminat
A more detailed description of all components is available in a separate article (see System Components)**

Main Tasks of the Platform

User and API Call Handling

Handled by the `webcontrol`, `webadmin`, and `api` services. Cookies and JWT are used for identification and se

- **`webadmin` — for administrators (account creation, user and settings management);**
- **`webcontrol` — for users (business logic management).**

Marketing Mailing Logic Processes

Include tasks such as message preparation, customer data processing, and performing requests to exterr

Message Preparation

Implemented in the `proctrigger` and `mailing` services:

- **`proctrigger` generates trigger (usually transactional) messages;**
- **`campaign` handles mass mailings and runs on schedule.**

Message Delivery

**Messages are pulled from the broker, prepared for sending in a local queue, and delivered according to co
to speed up deployment.**

Customer Action Tracking

Implemented using pixels, HTTP redirects, and cookies. Access control and scalability are managed via tl

Event Processing and Analytics

Data collection, aggregation, and transmission to reports and external services. Nearly all Altcraft servic

Periodic Tasks

Customer data processing, import, and export are performed on schedule by dedicated Altcraft services.

Utility Applications

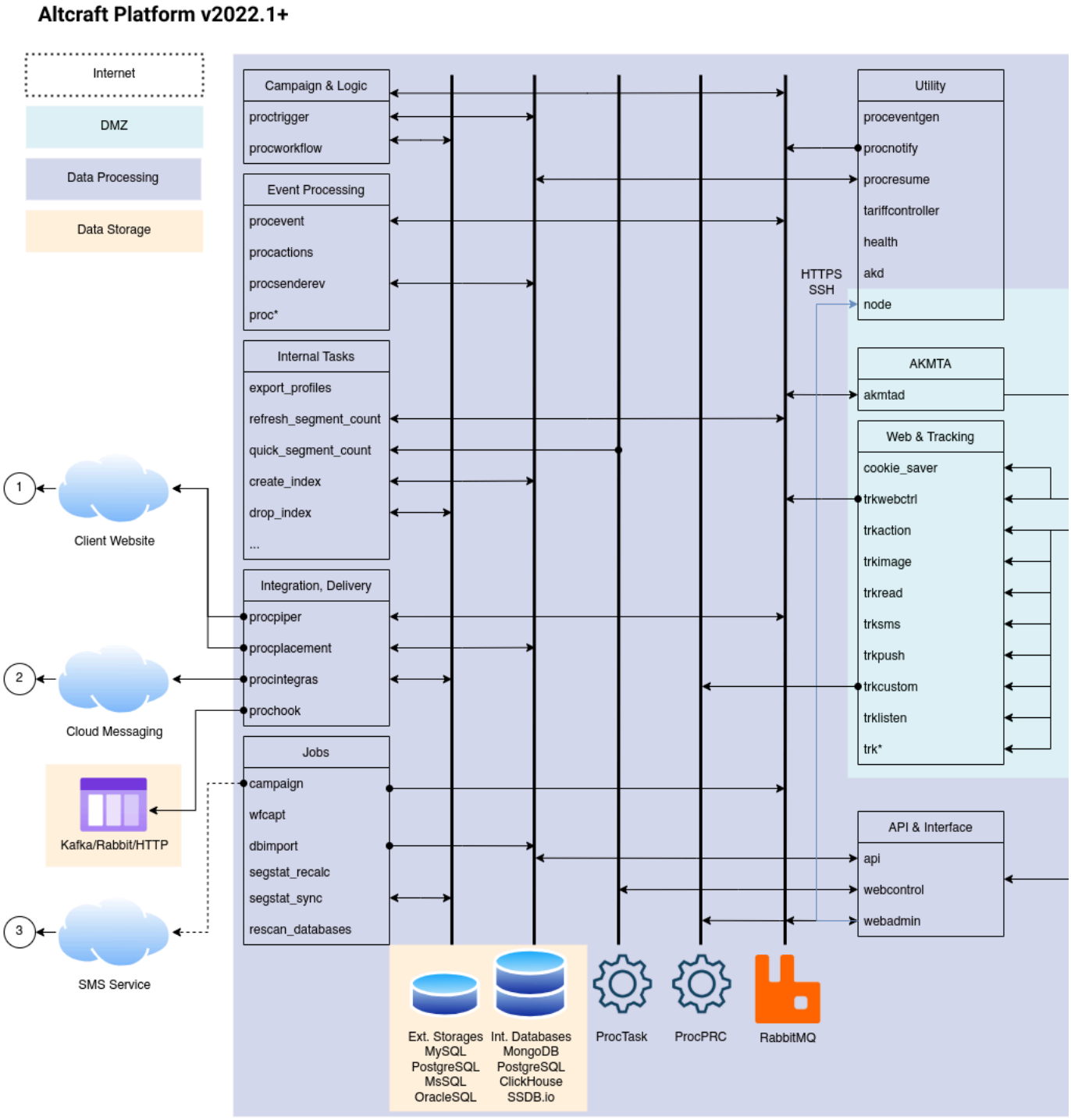
Examples:

- **`akd` — starts and stops services;**
- **`altool` — performs migrations and manages certain entities created in Altcraft;**
- **`health` — evaluates service health.**

Architecture Diagram

The diagram highlights the main zones where parts of the platform are located:

- **Internet — public segment with limited protection, usually hosting services that accept incoming mes**
- **DMZ — demilitarized zone for receiving HTTP and SMTP traffic. Services located in the DMZ are isolat**
- **Data Processing — data processing zone where the core business logic of the platform resides. This i**
- **Data Storage — secure segment for storing system and customer data (including external sources).**



In the diagram, platform components are grouped by functional purpose. Some services run continuously

- **Mailing & Logic — services for processing marketing mailings:**
  - `proctrigger` **— trigger mailings;**
  - `procworkflow` **— workflow task processing.**
- **Event Processing — services for handling events: collection, aggregation, and routing (e.g., tracking c**
- **Integration & Delivery — services for message delivery and integration with external systems:**
  - `procpiper`, `procintegras` **— deliver all message types and integrate with cloud services.**
- **Web & Tracking — services for processing HTTP traffic and serving content:**
  - `trkaction`, `trkread`**, and others — process tracking requests and serve static content.**
- **API & Interface — user and admin interface services:**
  - `webcontrol` **— user web interface;**
  - `webadmin` **— admin interface;**
  - `api` **— REST/HTTP API for external applications.**
- **AKMTA — email delivery; managed via node and the admin web interface.**
- **ProcTask — handler for periodic and one-off tasks (`proctask`), accepts gRPC requests.**

- **ProcRPC — gRPC service for tracking, providing access to database data.**
- **Utility Services — auxiliary system processes (`akd`, `procnotify`, `node`).**
- **Internal Tasks — periodic and one-off tasks executed inside services (`proctask`).**
- **Jobs — tasks requiring a separate process launch, such as mailings.**
- **Proxy — proxy servers for traffic routing and SSL termination. They can also run behind the customer'**

All components support horizontal scaling. However, some services (e.g., `procworkflow` and `proctask`) opera

Component Description

Services

API and Web Interfaces

Services providing access to the Altcraft platform for users and administrators, as well as interaction wit

- `api` — **the main API service (REST/HTTP) for working with Altcraft data and functions**
- `webadmin` — **the admin panel for managing accounts, users, and system settings**
- `webcontrol` — **the user panel for working with business logic: managing campaigns, mailings, and segm**
- `webloyalty` — **the loyalty program module interface**

Tracking and Analytics

Services that collect, route, and process tracking events (clicks, opens, subscriptions, forms, etc.), as we

External request processing

- `cookie_saver` — **performs the following:**
  - **handles user cookies**
  - **serves scripts and resources for Web SDK, Push SDK, and unsubscribe logic in emails**
  - **processes requests from Web SDK and Push SDK**
  - **processes requests from custom unsubscribe logic in emails**
  - **processes website requests (web-layer pixels)**
- `trk_amazon_sns` — **processes events from Amazon SES**
- `trkaction` — **performs the following:**
  - **tracking redirects for Email, SMS, and Push channels**
  - **processes user and system links (unsubscribe, subscription confirmation, suppress, open/read pix**
  - **processes trigger links (start trigger campaign)**
  - **serves scripts and resources for the subscription manager**
  - **serves scripts and resources for web versions of emails**
- `trkcustom` — **performs the following:**
  - **tracking redirects for custom channels**
  - **processes user and system links (unsubscribe, subscription confirmation, suppress, open/read pix**
  - **supports custom trigger links**
- `trkimage` — **image request handler, serves static images (e.g., inline in emails)**
- `trklisten` — **request handler for third-party services (integrations)**
- `trkpush` — **tracking redirect service for Push channel (deprecated)**
- `trkread` — **message read event processing service (deprecated)**
- `trkwebctrl` — **performs the following:**
  - **processes events in web forms and popups**
  - **serves scripts and resources for the Altcraft tag manager, popups, and web forms**
  - **processes user actions in web forms and popups**

Event aggregation and routing

- `procactions` — **processes tracking events (opens, clicks, subscriptions), updates profiles, generates ev**
- `procpixel` — **processes web-layer pixel events, imports and updates profiles, generates events for** `proc`
- `procpush` — **processes Push channel events, imports and updates profiles, creates push events**
- `procsenderev` — **processes events from senders: deliveries, bounces, returns, responses (deprecated)**
- `procsmsev` — **processes outgoing SMS gateway events, polls gateways, generates events for** `proctrigger`
- `procsmslisten` — **processes incoming SMS gateway events, receives provider hooks, generates events f**
- `proctrkweb` — **processes web form and popup events, writes data to the database, generates events fo**
- `procwebver` — **saves web versions of emails to storage**
- `procrpc` — **RPC service for receiving and processing requests from tracking services**

Tasks

Persistent services for scheduling and executing tasks:

- `procworkflow` — **processes and executes workflow tasks**
- `proctask` — **schedules and runs background tasks (segment calculations, campaign launches, etc.)**

Worker services (started on demand for specific resource-intensive work and stopped afterward):

- `campaign` — **manages execution of mass campaigns: preparing content and sending it to the message**
- `segstat_recalc` — **recalculates static segments**
- `rescan_databases` — **recalculates statistics and metrics of profile databases**

- `dbimport` — **scheduled profile import from external SQL tables**
- `wfcapt` — **captures profiles into a workflow on schedule**
- `segstat_sync` — **synchronizes static segment data with external advertising platforms and audiences (e**

Data Processing

Services for internal data processing, integrations, and subscription management.

- `procresume` — **manages subscriptions: automatically unfreezes (resumes) subscriptions for profiles afte**
- `proctrigger` — **processes and manages trigger (event-based) campaigns**
- `prochook` — **manages outgoing webhooks: generates and sends notifications to external systems base**
- `procpiper` — **processes pipelines and data in custom (user-defined) channels**
- `procleadsaver` — **performs the following:**
  - **background operations with profiles and data**
  - **deferred profile updates and imports**
  - **asynchronous counter updates in databases and segments**
  - **add/remove profiles in static segments (in development)**
  - **mark profiles in supercampaign audiences (in development)**
- `procevent` — **performs the following:**
  - **writes events to the analytics storage**
  - **generates and routes events for** `proctrigger` **and** `prochook`
- `proceventgen` — **generates test events for debugging and testing Altcraft**
- `procintegras` — **performs the following:**
  - **manages integrations and message delivery**
  - **integrations with external analytics services (AppMetrica, Facebook Audiences)**
  - **synchronizes segments with external SQL databases**
  - **sends messages through all SMS gateways**
  - **sends messages through third-party Email gateways**
  - **sends Push notifications**
- `procplmt` — **performs the following:**
  - **manages indirect campaigns in custom channels**
  - **processes moderation events in custom channels**
  - **automatically unpublishes after TTL (Time to Live) expiration**

Proxy

Separate Nginx instances for proxying network traffic to Altcraft services.

- `nginx-trk` — **proxies and routes requests to tracking services**
- `nginx-sites` — **proxies requests to API and web interfaces**
- `nginx-cookisaver` — **proxies requests to the** `cookie_saver` **service**

Databases

Main databases that require independent installation and maintenance.

MongoDB

Primary DB for storing all account entities: customer data, users, campaigns, templates, settings, and oth
https://www.mongodb.com/

ClickHouse

ClickHouse — an analytical DBMS used to store subscriber activity history and Altcraft events (and in som
https://clickhouse.com/

Kvrocks

Key-value store used for deduplication (contact data) in campaigns.
https://github.com/apache/kvrocks

SSDB

Key-value store used for storing short-term hourly account statistics and global stop lists (based on MD5 l
https://github.com/ideawu/ssdb

PostgreSQL

Installed and connected additionally for the marketplace module, used for storing product and order data.
https://www.postgresql.org/

Bundled Databases

Delivered as static files with Altcraft, do not require separate installation or management.

SexDB

Used for automatic gender detection by profile name during data import.

GeoDB

Used to determine geolocation by IP address.

BadPass

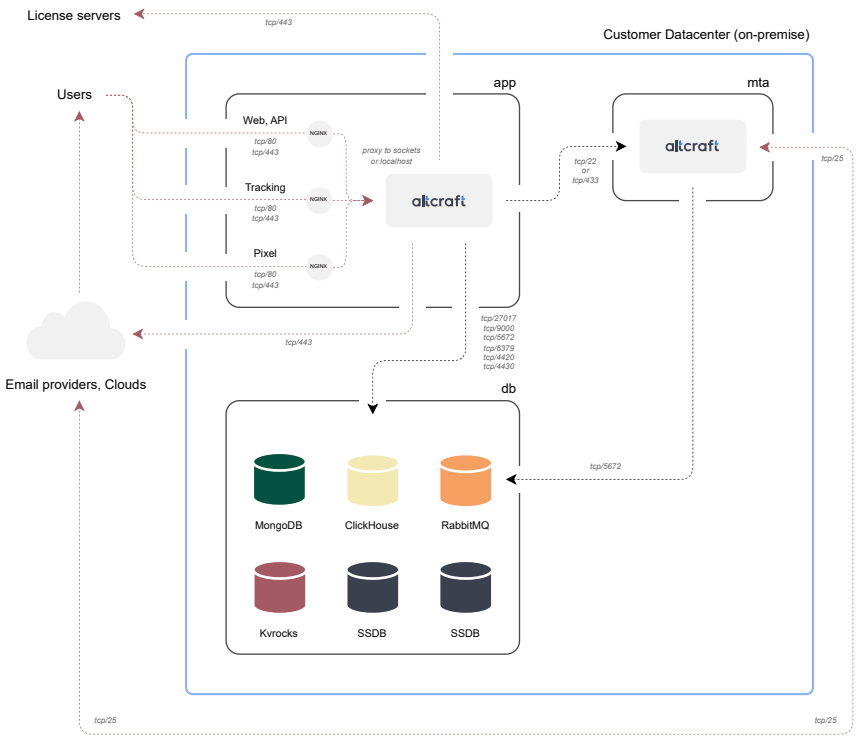
Used to detect weak and compromised user passwords.

Message Brokers

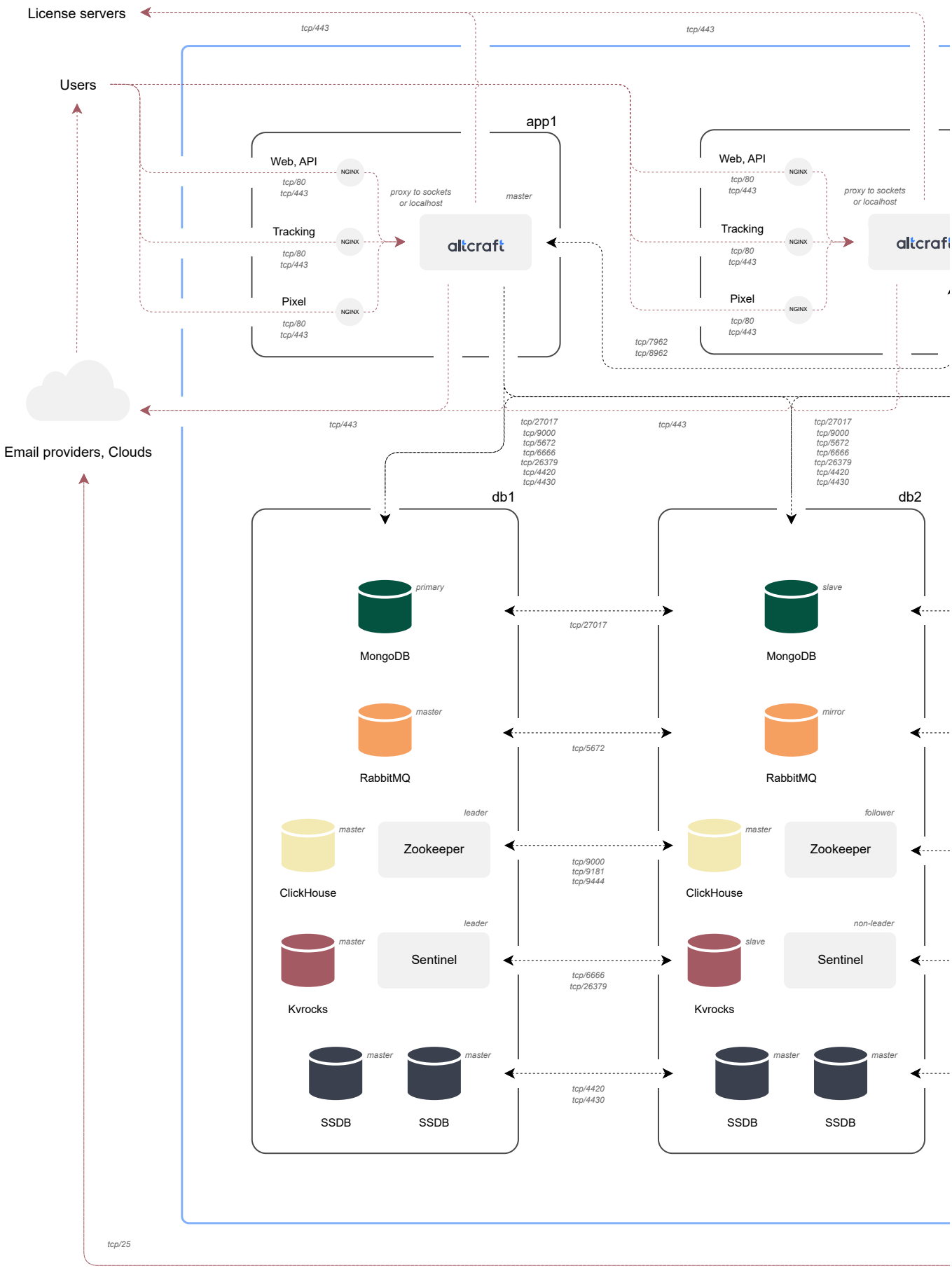
RabbitMQ

The main message broker, used for various interactions between Altcraft services.
https://www.rabbitmq.com/

Basic scheme



Fail-safe scheme



Typical Placement in Infrastructure

Description

Depending on the tasks and operational requirements, Altcraft can be deployed in different ways. This pa system design, on the basis of which a specific architecture is then formed according to project requirem

- **Control server — server(s) hosting Altcraft services and databases.**
- **Sending server — server(s) hosting the `akmtad` and `node` services, which are part of Altcraft.**
- **MTA (Mail/Message Transfer Agent) — the `akmtad` mail agent included in Altcraft, responsible for email**
- **Web resources — the customer's or partners' web resources where visitor tracking is performed.**
- **Users — system users: administrators, marketers, designers, etc.**
- **Applications — external or internal customer services that interact with Altcraft.**
- **Cloud messaging — cloud services for sending PUSH notifications (Google Firebase, Apple APNs, etc.)**
- **Subscribers — end users: buyers, subscribers, website visitors. In Altcraft they are tracked as *profile***
- **Email services (ISP) — email providers (e.g., Gmail, Yandex, etc.) that receive email messages.**

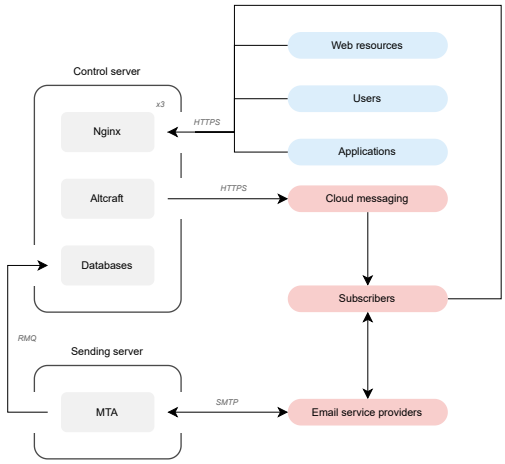
Regardless of the deployment method, the following questions must be addressed in advance:

- **Which domain will be used for the control panel?**
- **Which domain will be used for tracking, pixel?**
- **Which domains will be used for From and MAIL FROM in email mailings?**

Without Load Balancing

This scheme is suitable for the following cases:

- **test or pilot deployment;**
- **deployment in external networks;**
- **implementation in small organizations;**
- **working with relatively small volumes of email mailings.**



If necessary, access to the control panel and API can be restricted to the internal network. A distinctive f

> **INFO**
>
> **At the moment, Altcraft runs three separate Nginx instances, which requires three IP addresses on the s**

The MTA interacts with ISPs directly, which allows the use of bypass mechanisms (various retry rules for

> **INFO**
>
> **At least two IP addresses are required for sending. Depending on mailing volume and content type, more**

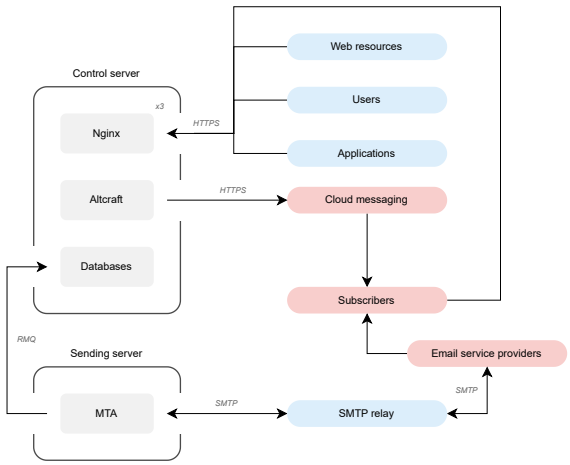
Using an SMTP Relay

This scheme is suitable for the following cases:

- **test or pilot deployment;**
- **deployment in external networks;**
- **implementation in small organizations;**
- **working with relatively small volumes of email mailings;**
- **need to use an SMTP relay.**

When using an SMTP relay, the Altcraft MTA must receive delivery reports (DSN, Delivery Status Notificati IP addresses configured in the Altcraft sender (or the domain must resolve to the relay only for external s

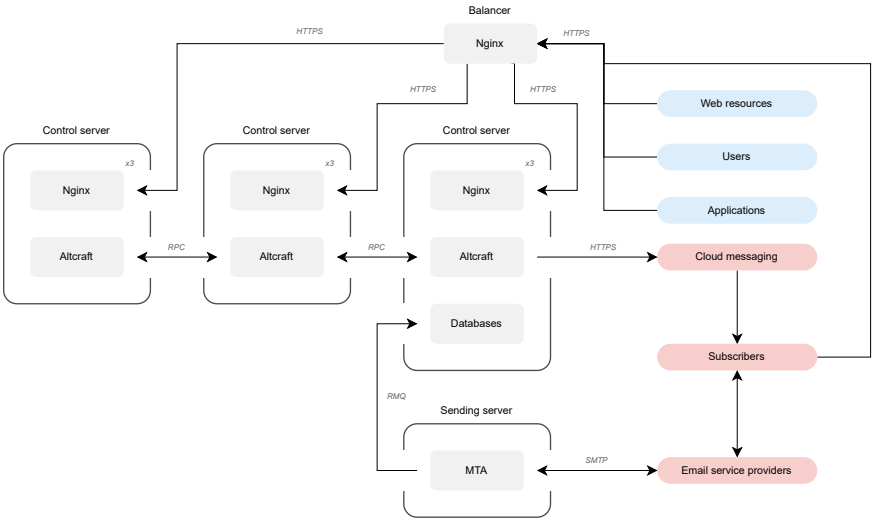
or non-delivery event.



With Load Balancing

This scheme is suitable for the following cases:

- **implementation in large organizations;**
- **deployment in external or private networks;**
- **planned clustering;**
- **working with relatively large volumes of email mailings.**



In this case, Altcraft can operate in a private network, while message delivery and HTTP interaction are h

INFO

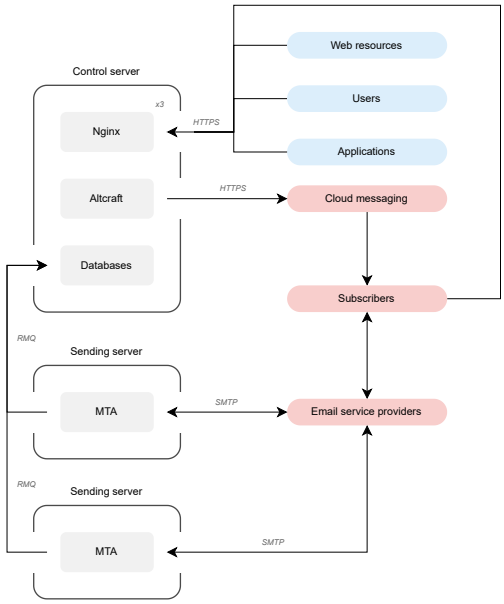
- **All databases can be replicated, thereby improving the overall fault tolerance of the solution.**
- **The load balancer must pass the client's real address in the `X-Real-IP` or `X-Forwarded-For` header, and i**
- **When deploying the service in multiple instances, configure sticky sessions on the load balancer. Th**

Other Options

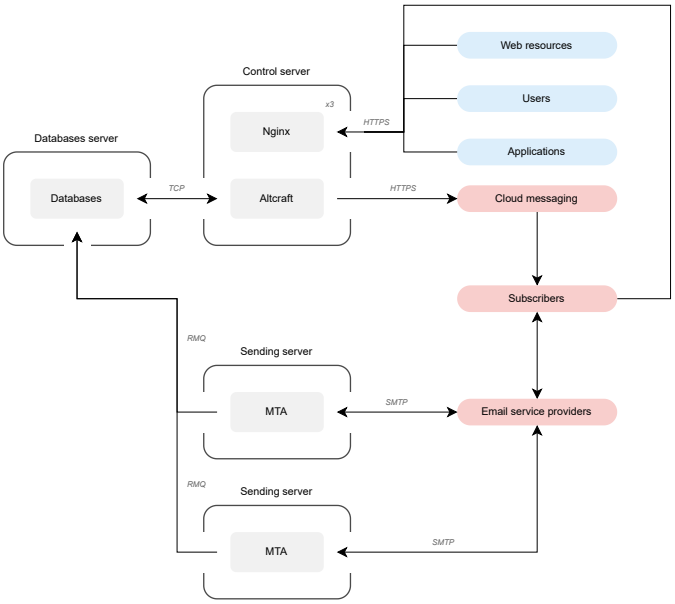
For example, to ensure high-performance sending (millions of emails per hour), multiple MTA servers can b

INFO

If a server has multiple IP addresses, one of them can be used for sending via an SMTP relay by creating



And to simplify maintenance, all databases can be moved to a dedicated server and resources redistribute



The specific scheme depends on the tasks at hand — for example, a prioritized delivery channel. Details o

Nodes

Description

A Node is a network interface of the sending server used by AKMTA senders. A node can be located on th

NOTE

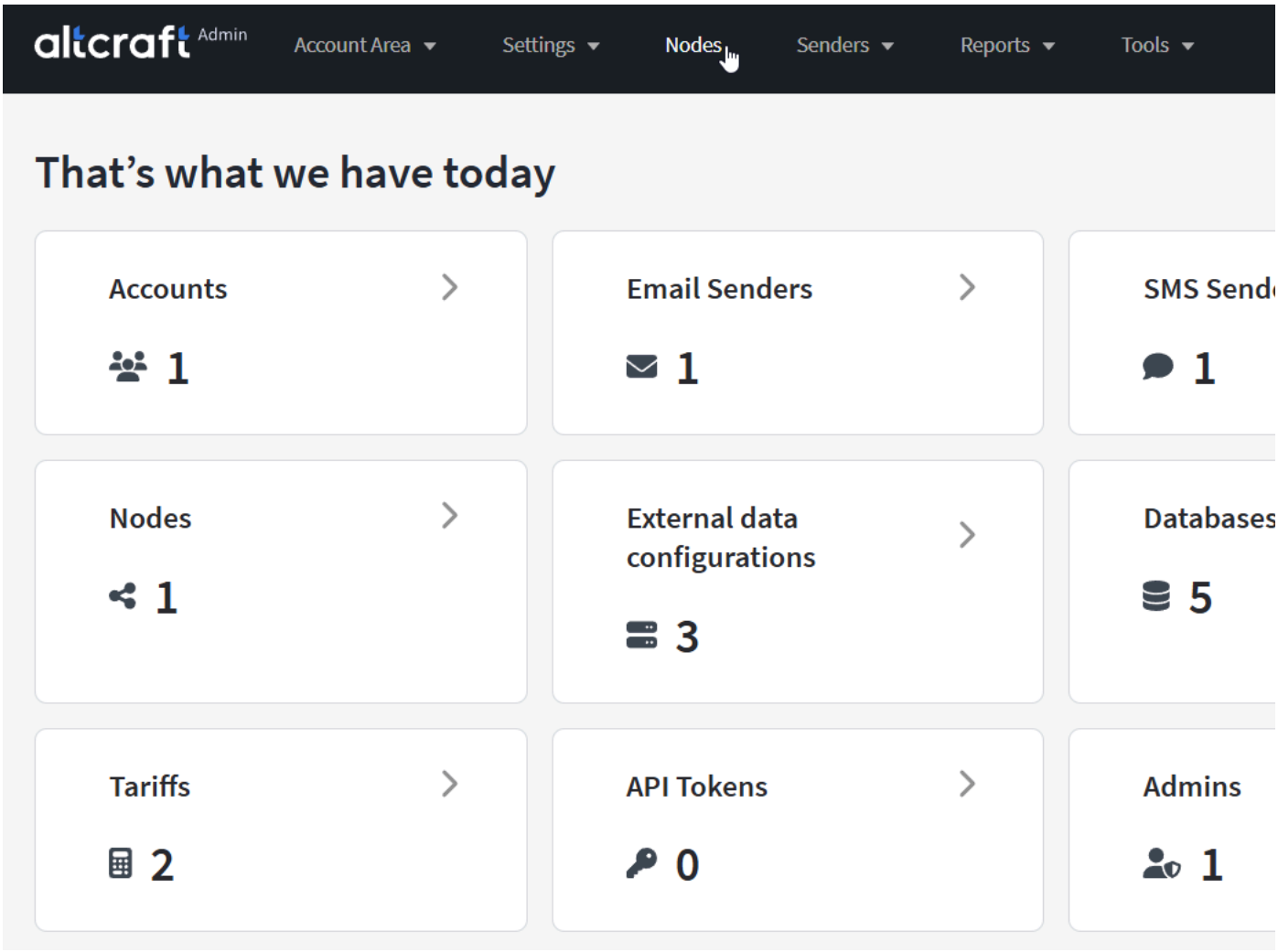
Different Altcraft deployment options are described in this article.

TIP

Instructions for adding IP addresses to a node are available here.

Creation, Editing, and Deletion

To manage nodes, in the main menu select Nodes.

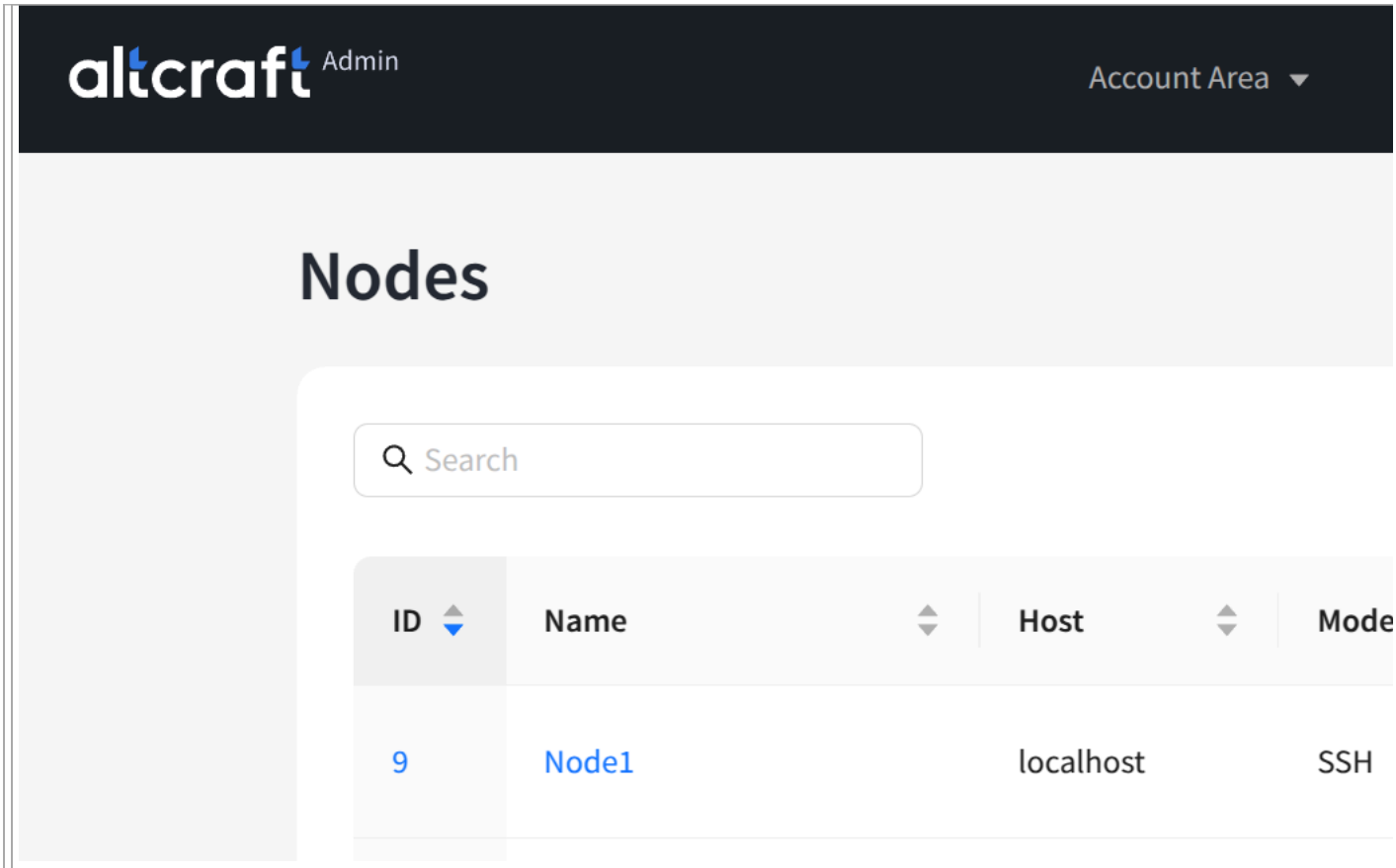


The list of existing nodes will appear. On the right side of the table, there are buttons for viewing error log

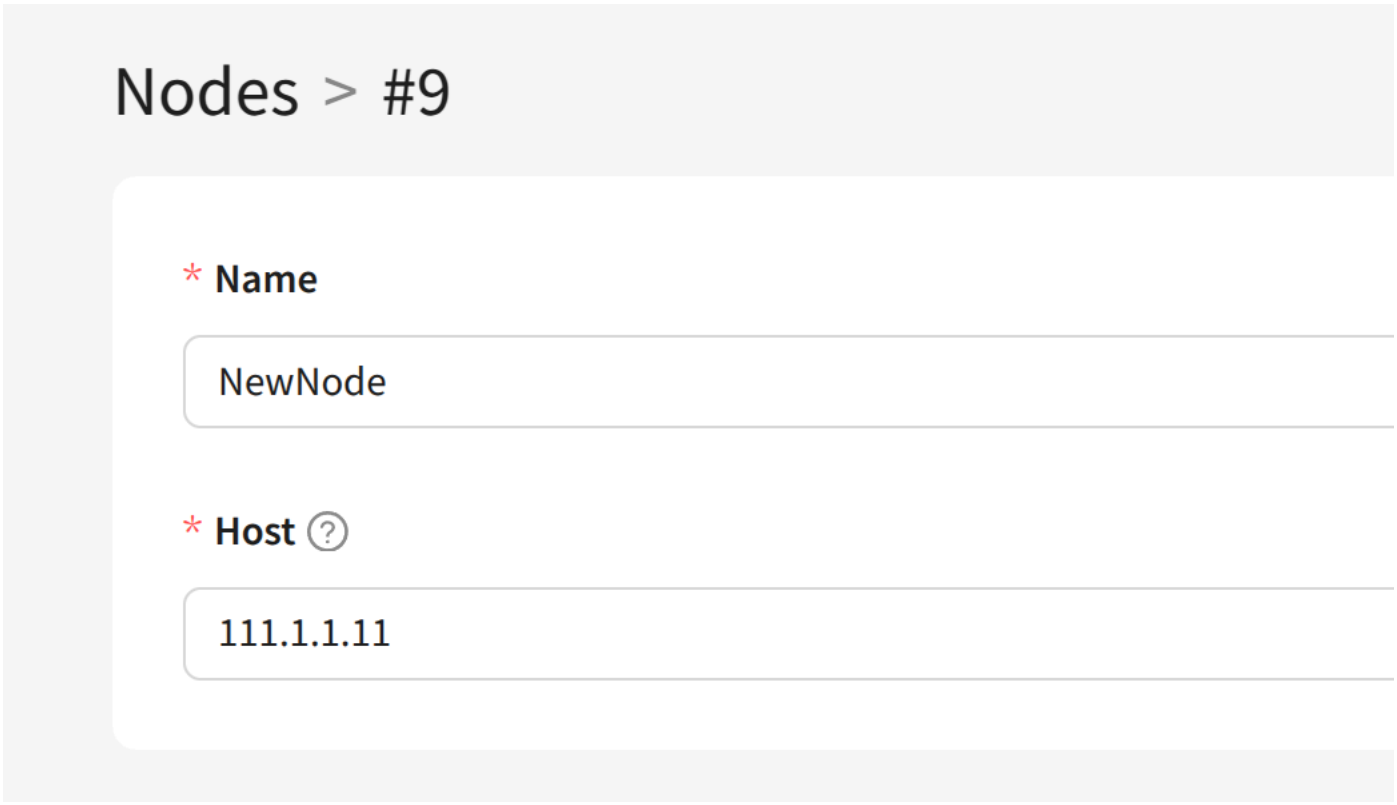
NOTE

In the node list, you can start or stop the MTA (mailer) on a specific node or on all available ones. After n

To create a new node, click + Create:

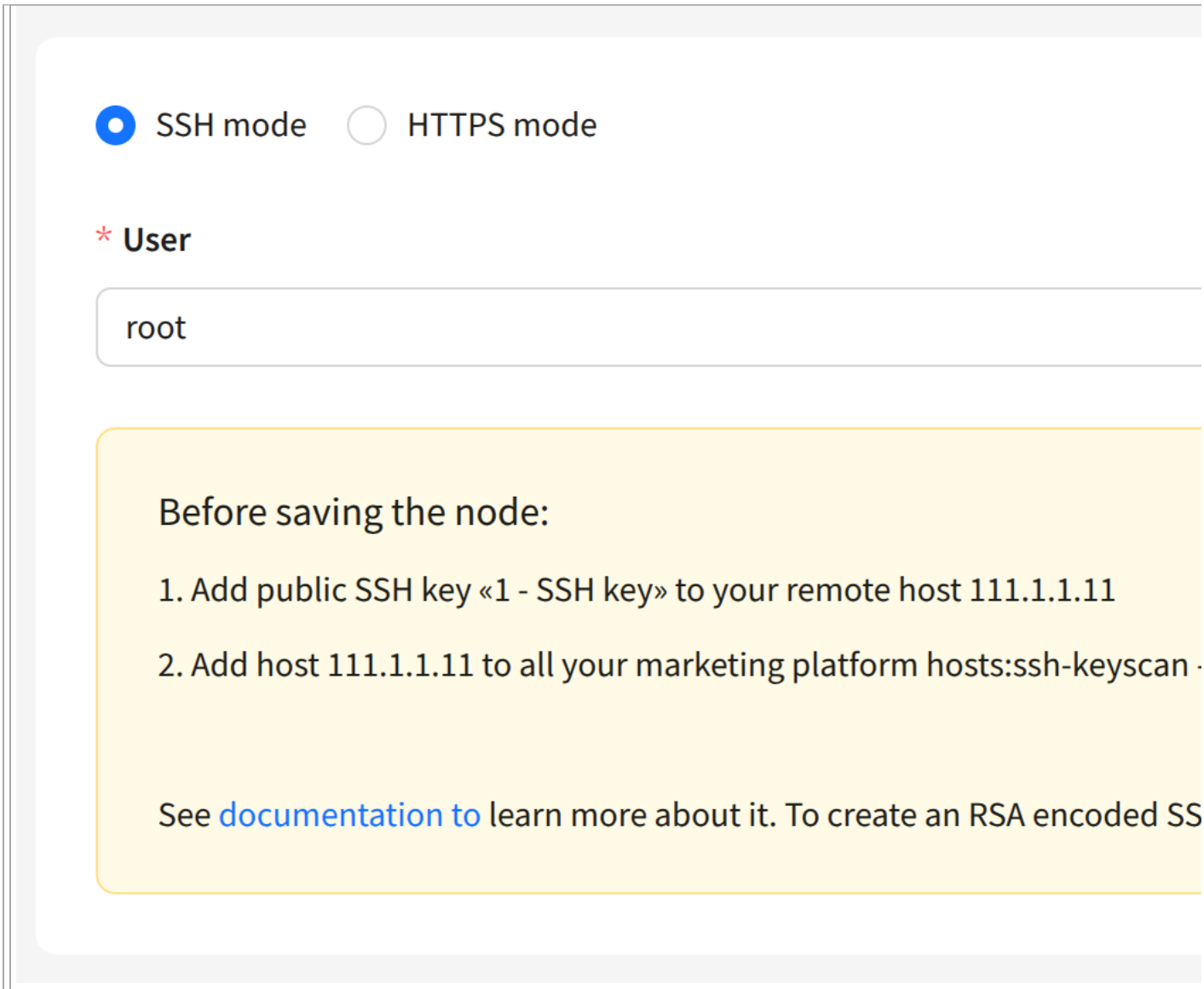


For the new node, specify a unique Name, the network Host address, and the Node installation directory c



Node with SSH Access

By default, SSH protocol is used to access the server with the node. For this type of connection, you must

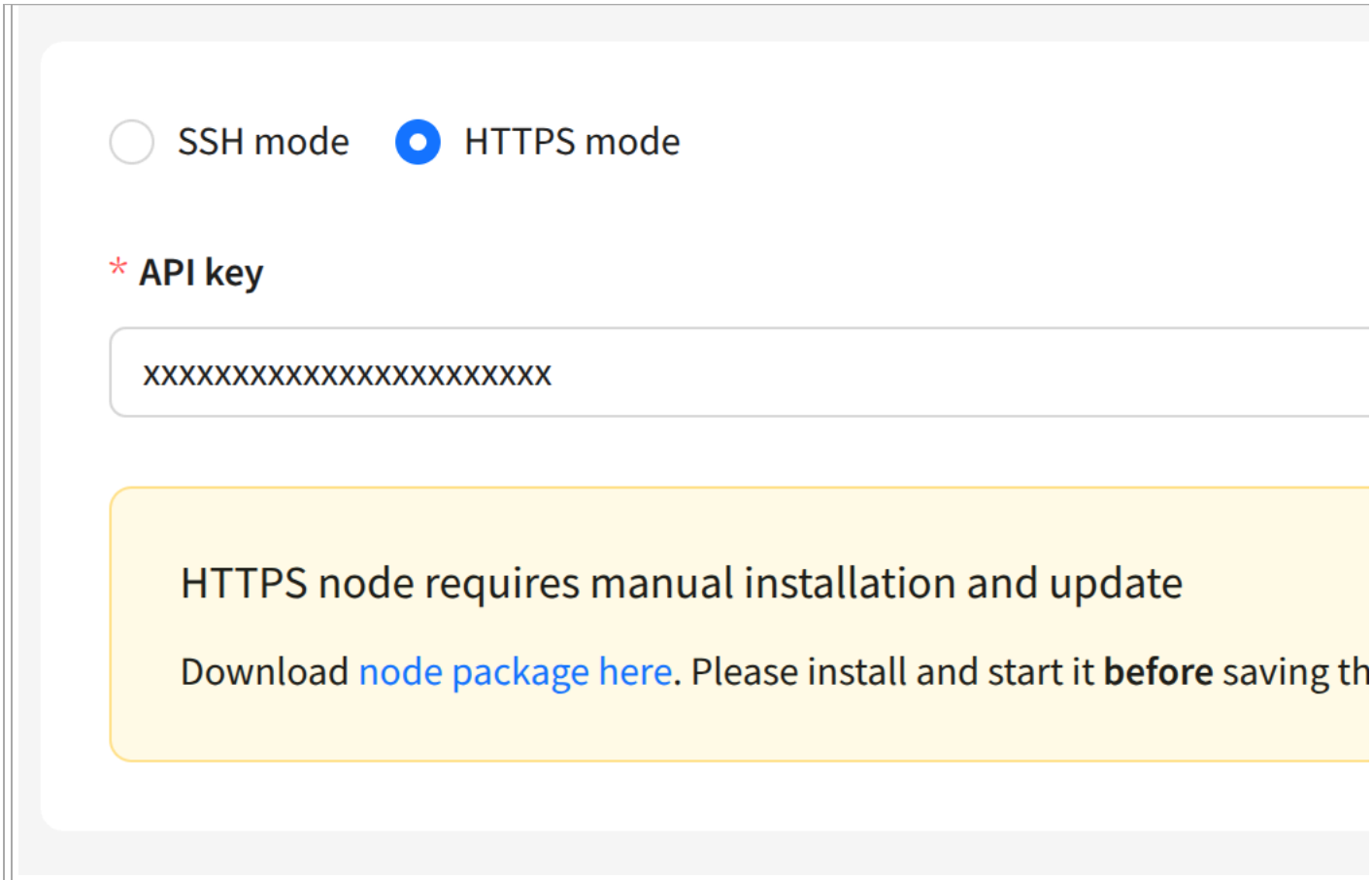


TIP

The SSH key must be created in advance in the platform. Learn more about how to do this here.

Node with HTTPS Access

A node using HTTPS connection must be manually installed on the sending server. The installation archive



Unpack the node archive on the sending server and configure the node in the node.json file:

```
{
  "LISTEN": "127.0.0.1:4433",
  "CERT_PATH": "/opt/akmta_node/conf/server.crt",
  "KEYFILE_PATH": "/opt/akmta_node/conf/server.key",
  "APIKEY": "O46FtVEU1UdrMDwdCAORyDJelirt4r0W",
  "PID_FILE": "/opt/akmta_node/node.pid",
  "LOG_FILE": "/opt/akmta_node/node.log",
  "BASEDIR": "/opt/akmta_node"
}
```

Example of starting the node:

```
./bin/node -d --https --config=/opt/akmta_node/conf/
```

Send Module Settings

For the send module using the node, you can configure data channel throughput, data retention time, and

MTA Settings

* Priority channel size

1000

* Other ch

1000

* Number of spool writers

4

* Retry cha

1000

* Spool cleaner timeout (days)

3

* DNS look

10

* Number of RMQ consumers per sender

1

* Number

4

* ESMTP connection timeout

15

* Fetcher p

1

* Override FBL header name

X-AKMTA-FBL

POP3 listen only interfaces

127.0.0.1:110, default listening on all sending IP

Write SMTP responses to system log

Enable profiling

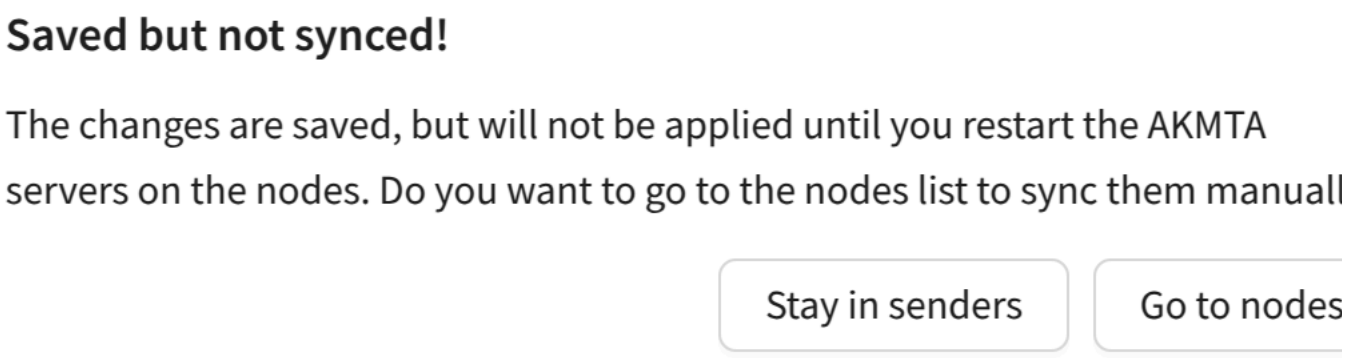
Data channel settings:

- **Priority channel size — size of the high-priority sending channel (incoming).**
- **Other channel size — size of the main sending channel (incoming).**
- **Spool channel size — size of the outgoing sending channel (from the spool).**
- **Number of spool writers — number of concurrent threads writing to the spool from RabbitMQ.**
- **Retry channel size — size of the retry sending channel.**
- **Event channel size — size of the event processing channel.**
- **Spool cleaner timeout (days) — number of days before cleaning the spool (queue) on the node.**
- **DNS lookup timeout (sec) — DNS request timeout.**
- **DNS lookup cache expire (sec) — DNS cache expiration time in seconds.**
- **Number of RMQ consumers per sender — number of parallel RabbitMQ connections.**
- **Number of spool reader — number of spool readers.**
- **Number of eventer reader — number of event channel readers.**
- **ESMTP connection timeout — ESMTP connection establishment timeout.**
- **Fetcher prior spool timeout — buffer channel size for processing priority messages.**
- **Rebuilder workers count — number of internal service workers responsible for rebuilding expired mes**
- **Override FBL header name — override the service header name for receiving complaint reports (FeedB**
- **SMTP administrator email — administrator email address.**
- **POP3 listen only interfaces — select IPs for incoming mail responses (all selected by default).**
- **SMTP listen only interfaces — select IPs for outgoing mail responses (all selected by default).**
- **Write SMTP responses to syslog — log SMTP responses to syslog.**
- **Enable MTA observer — enable automatic MTA restart on the node by the management server in case**
- **Enable profiling — enable profiling (pprof).**
- **Clear spool messages after delivery — with this option enabled, delivered messages will be automatic**

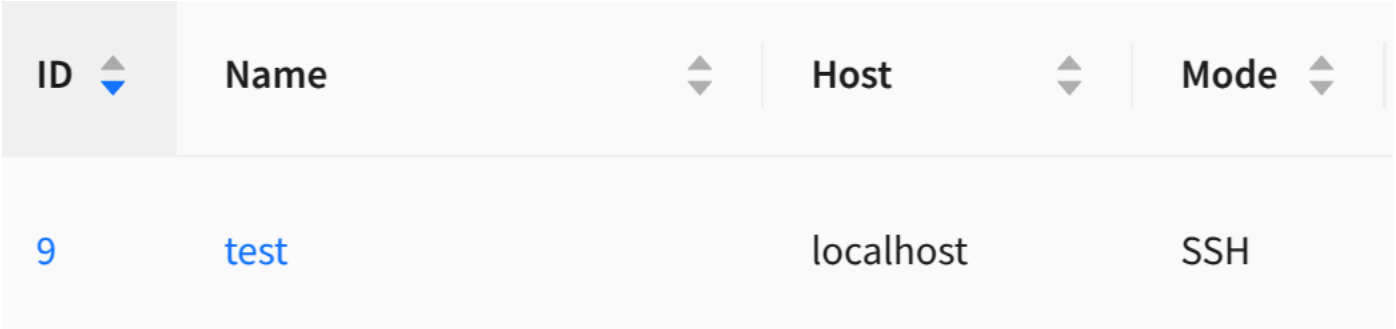
Debugging and Troubleshooting

Synchronizing Settings

After making changes to the sender configuration, you must synchronize the management and sending mo



The list of nodes will open. You must stop and start the sender on the desired node by clicking Stop and S



Tariffs

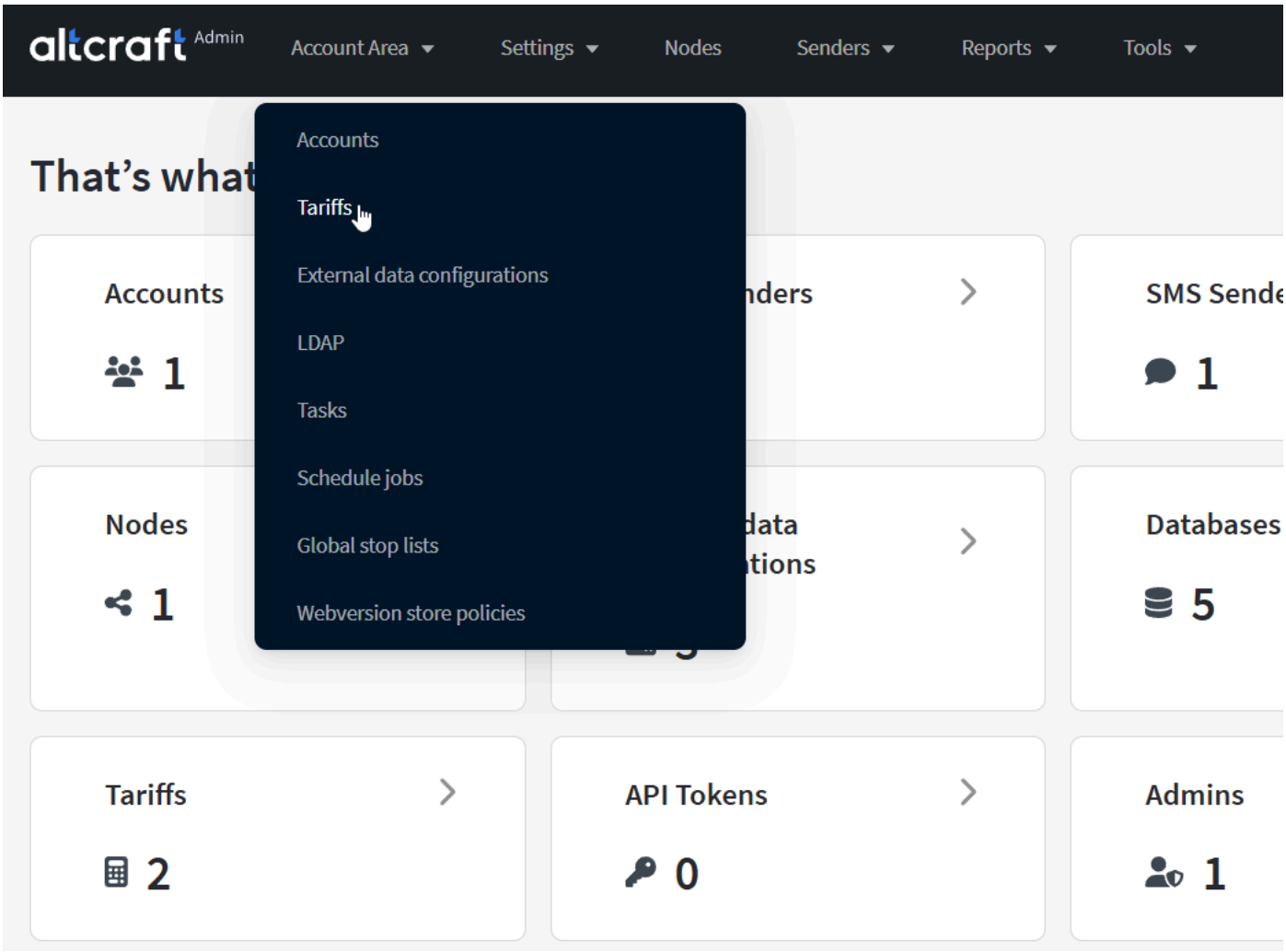
Description

A tariff is a set of limits for a platform account. The license agreement for the Altcraft marketing platform

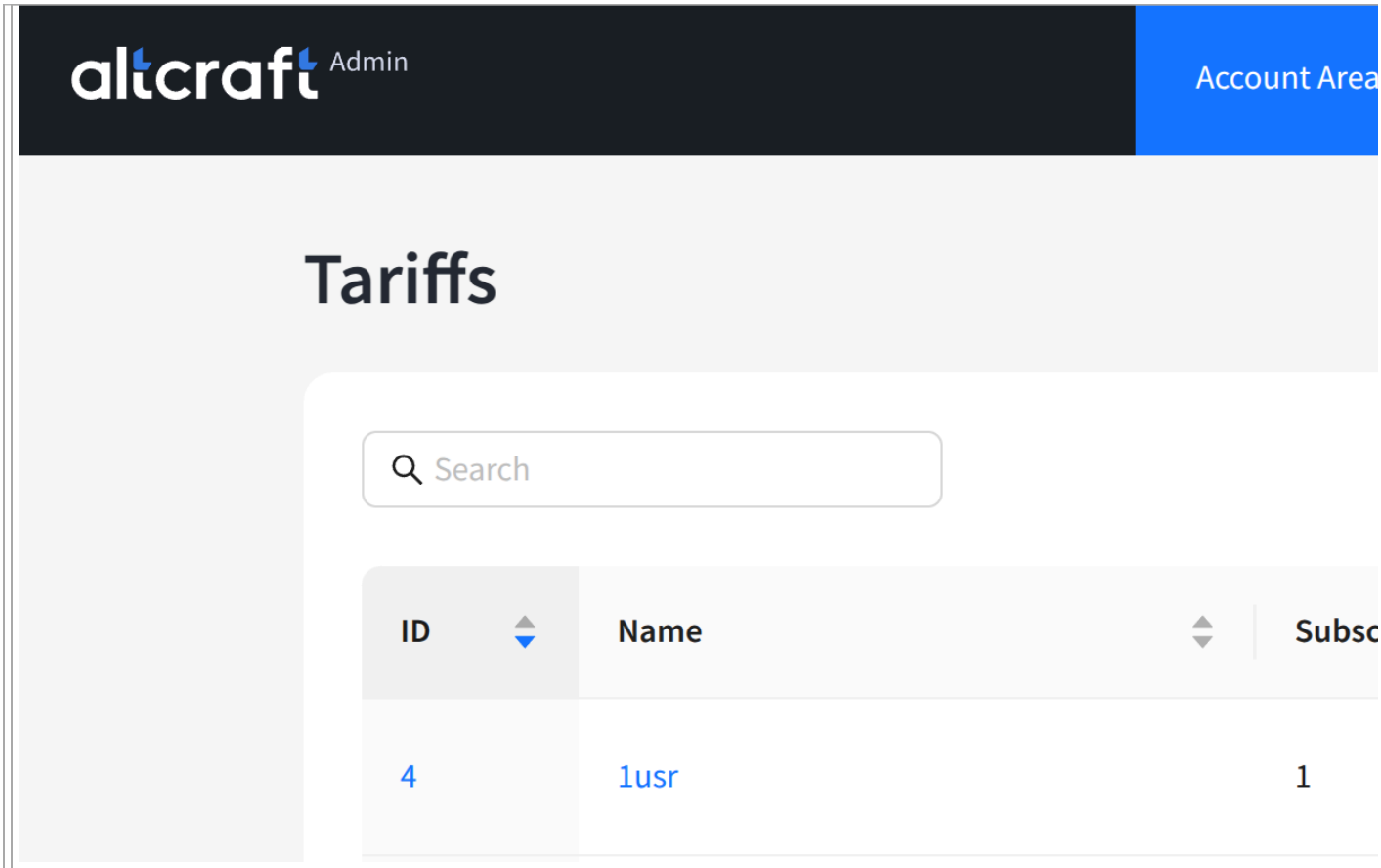
> **CAUTION**
> **A tariff is a mandatory attribute of an account. Create at least one basic tariff even if you do not plan to**

Creating, Editing, and Deleting

To manage tariffs, in the main menu select Account Area — Tariffs.

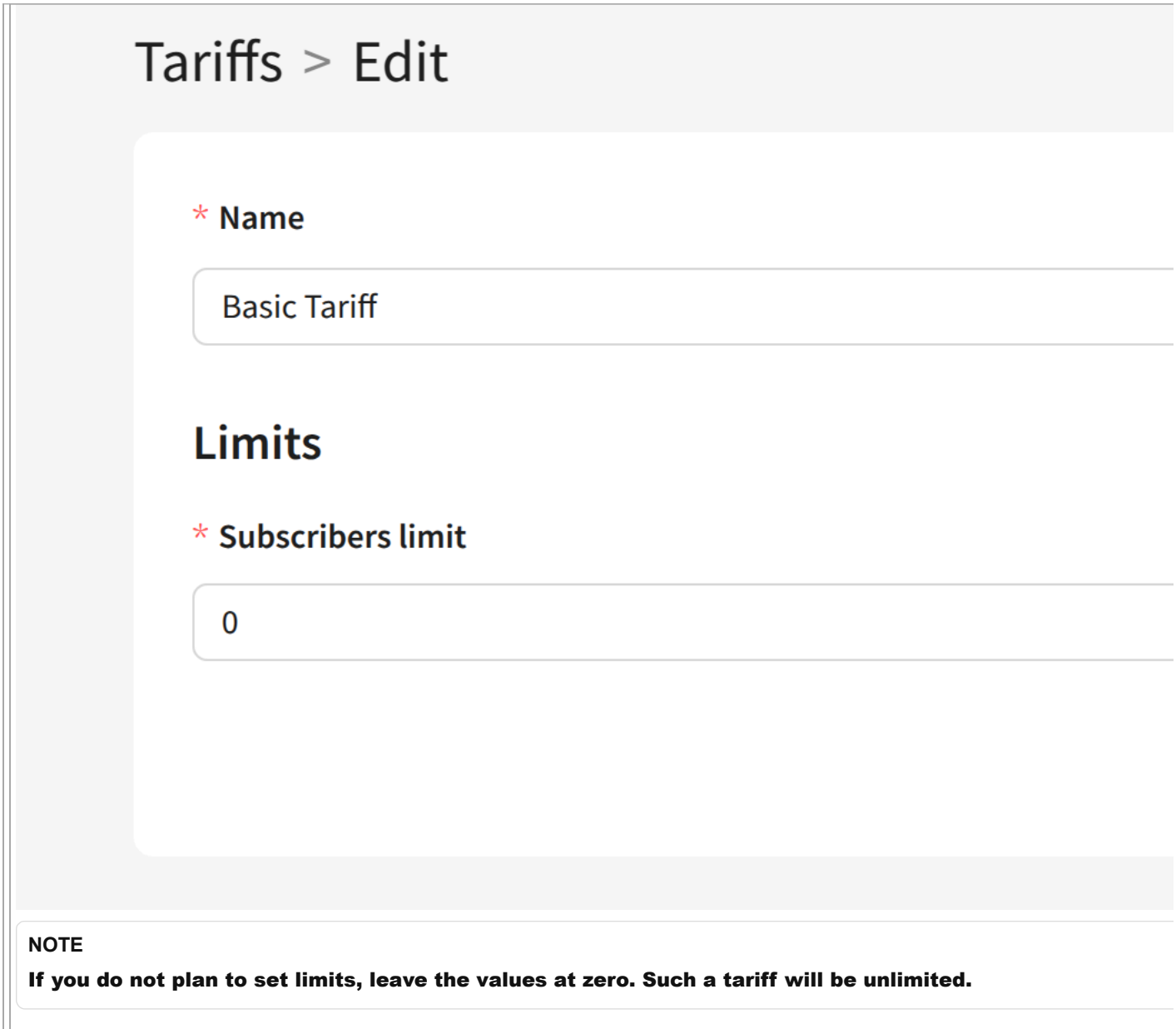


The list will display existing tariffs. On the right side of the table are the edit and delete buttons. You can



For each tariff, in addition to the name, configure two parameters:

- **Subscribers limit — limit on the total number of subscriber profiles in the account (all profile statuses**
- **Messages limit — limit on the number of messages sent (all communication channels are counted).**



NOTE

If you do not plan to set limits, leave the values at zero. Such a tariff will be unlimited.

External data configuration

Description

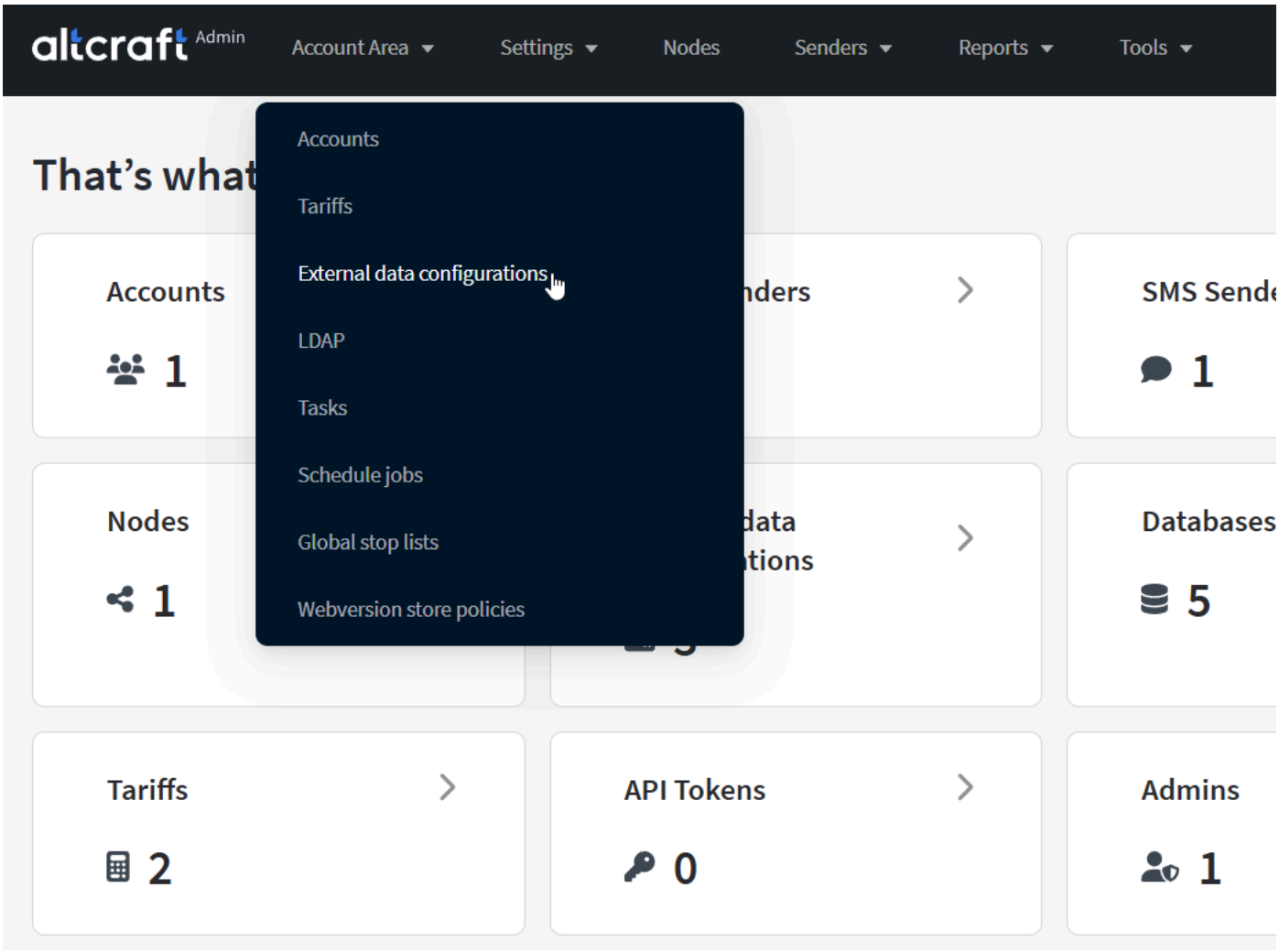
External Data Configuration is an Altcraft platform tool designed for integration with external SQL databa

Capabilities

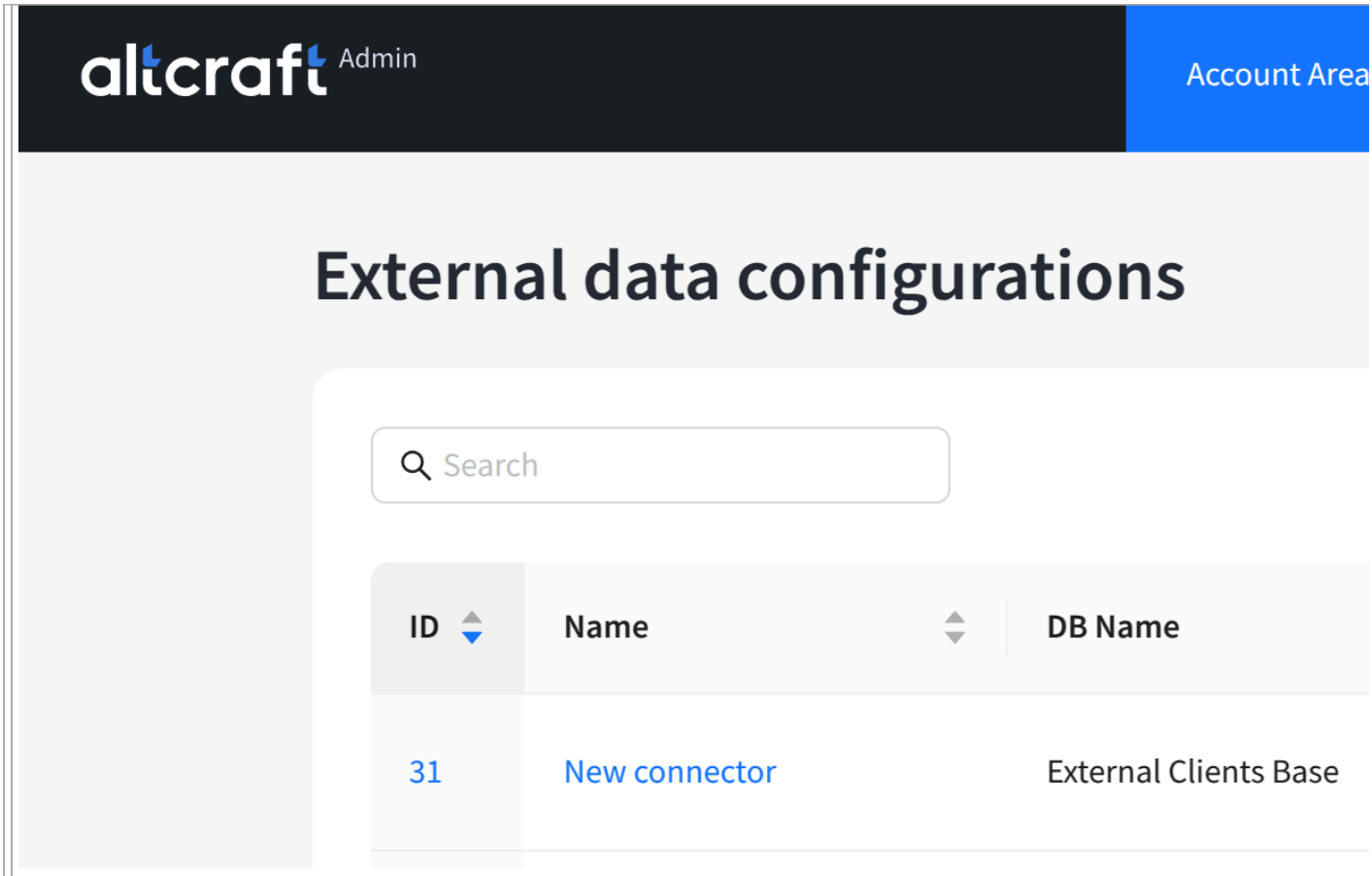
- **Scheduled import of customer data**
- **Segmentation of subscribers by external databases**
- **Dynamic content in message templates**

Creating, Editing, and Deleting

Integration with external SQL databases is configured in the Altcraft Admin Panel, while queries to databa
To manage external data source connections, in the main menu select Account Area — External data con



The list displays existing connectors to external databases. On the right side of the table are the edit and



For each connector, specify the following details:

- **Name — connector name.**
- **Type — the DBMS type of the external database. Supported types are **MySQL, PostgreSQL, MSSQL, (**
- **Database name — name of the external database.**
- **Connection timeout — connection query timeout to the database.**
- **Query timeout — SQL query execution timeout.**
- **Protocol — network protocol for connection. Supported protocols are TCP, UDP, and UnixDomain.**
- **Host — server IP address hosting the external database.**
- **Port — server port of the external database.**
- **Username, Password — login and password for database authorization.**
- **Additional dsn options — additional dsn parameters.**
- **With SSL — connect the database to the platform via SSL connection. Available for MySQL, PostgreSQ**

Hive DBMS supports different authentication types: PLAIN, GSSAPI (Kerberos), LDAP (also PLAIN), and cus

TIP

You can specify the Collation used for connecting to the database in the Additional dsn options field usin

External data configurations > Edit

* **Name**

New connector

Type

MySQL

* **Database name**

default

* **Connection timeout**

1

* **Query timeout**

1

* **Protocol**

TCP

* **Host**

localhost

* **Port**

1234

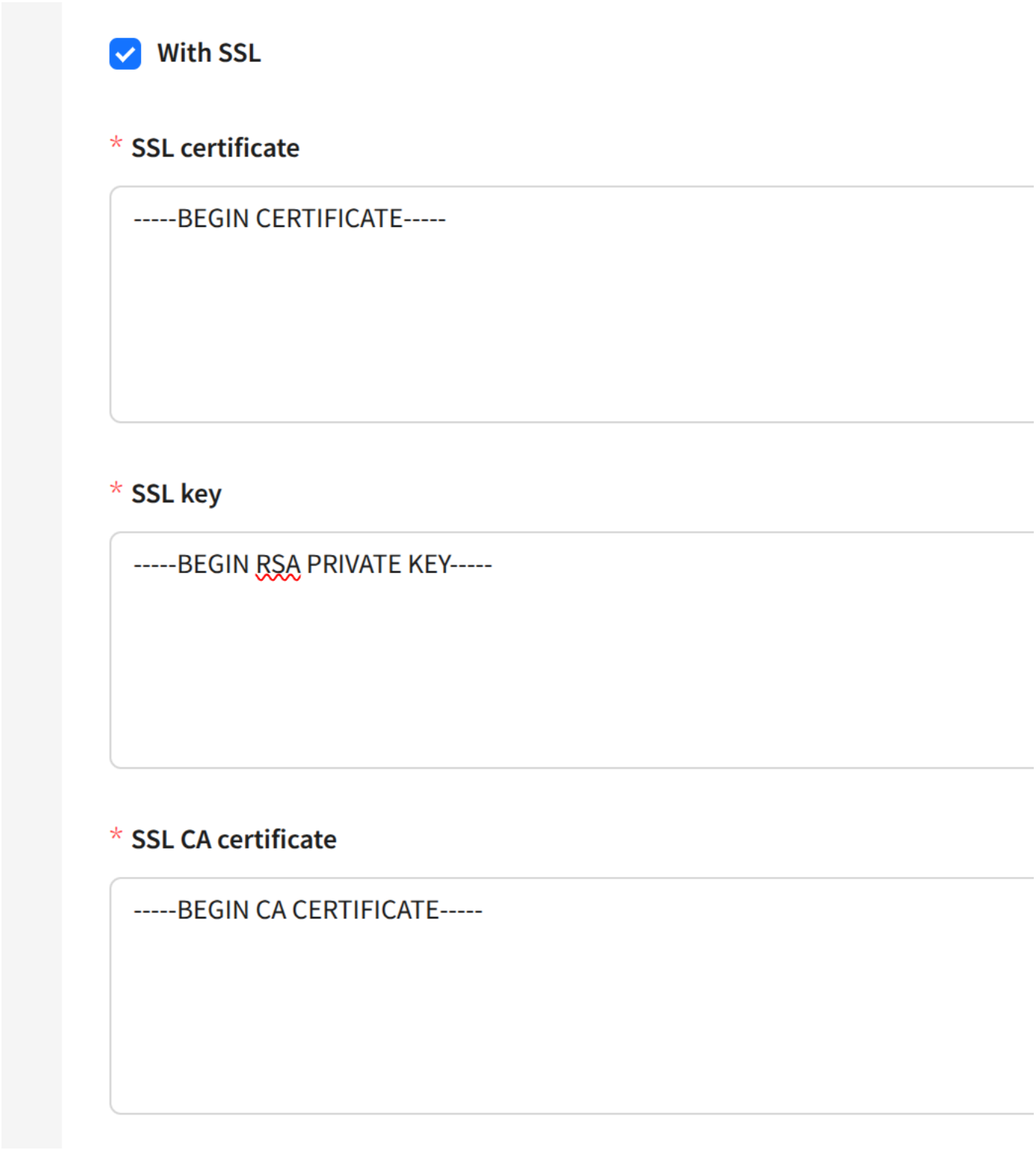
Username

Additional dsn options

key1=value1;key2=value2;

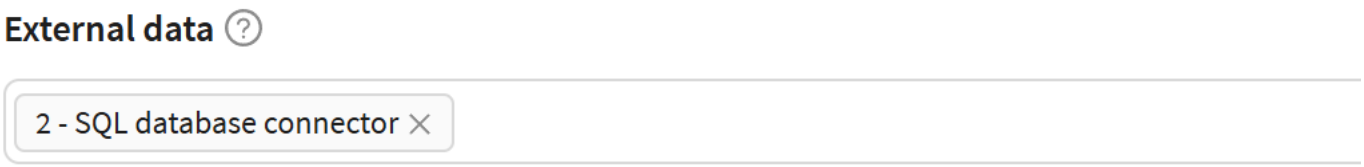
☐ **With SSL**

When using SSL connection, you must add an SSL certificate, SSL key, and SSL CA certificate in PEM forn



Connecting a connector to an account

After the connector is created, assign it to a platform account. In the admin panel, open Account Area — A



TIP

How to use external SQL tables for segmentation in the platform is described in detail in this article.

LDAP

TIP

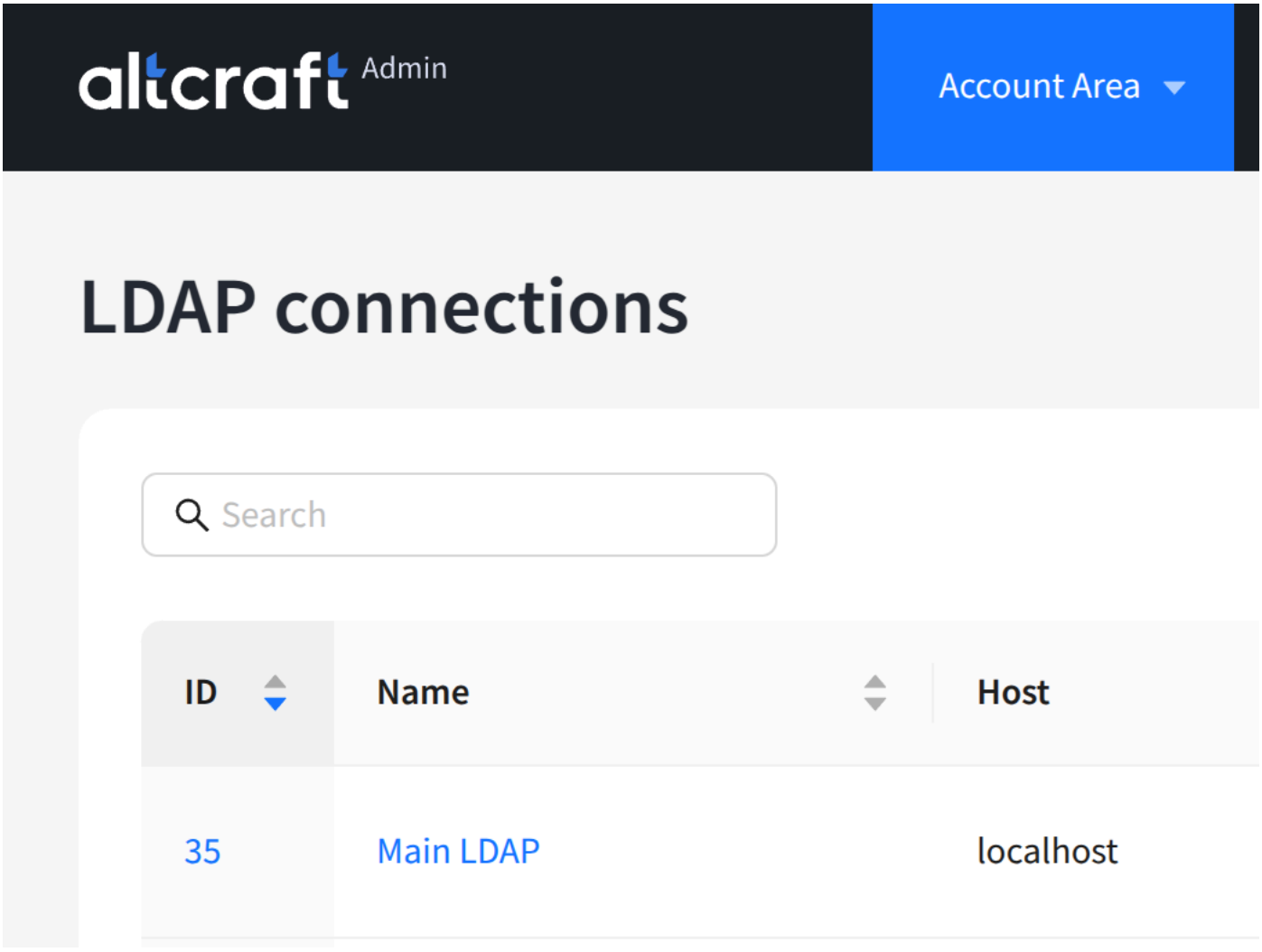
A step-by-step guide for configuring LDAP access to Altcraft is available here.

Access to Altcraft can be provided via LDAP (Lightweight Directory Access Protocol). This allows compan

NOTE

This article describes managing directory server connections for access to the Altcraft User Interface. L

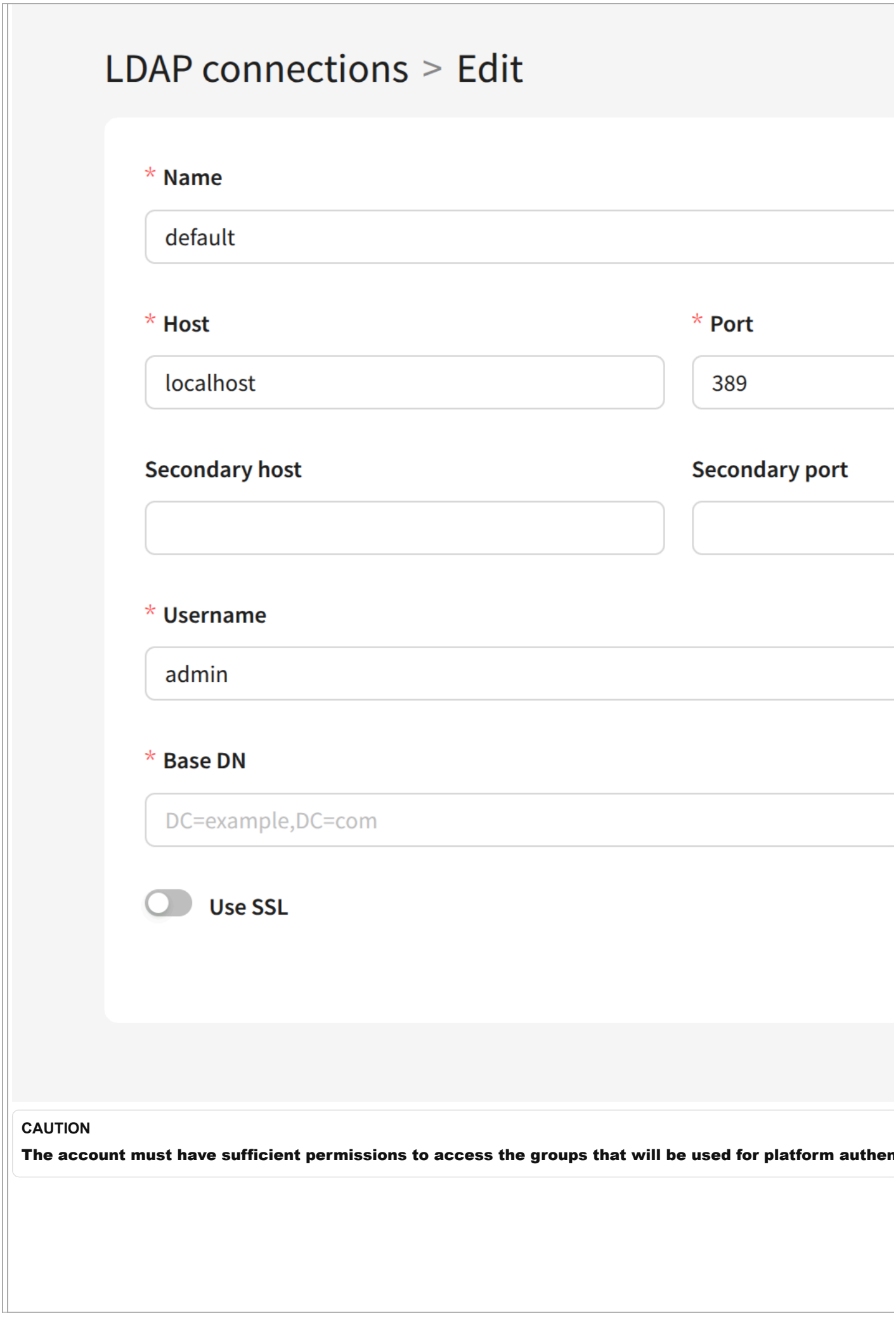
To manage LDAP connections, in the main menu select Account Area — LDAP.
The list displays existing LDAP connections. On the right side of the table are the edit and delete buttons.



To create a connection, configure the following parameters:

- **Name — connection name.**
- **Host — network address.**
- **Port — port for connection.**
- **Secondary host and Secondary port — address and port for reconnection if the primary connection fa**
- **Username and Password — directory server credentials.**
- **Base DN — base DN for user search.**

If you plan to use an encrypted connection, enable Use SSL and configure certificates on the LDAP server



CAUTION

The account must have sufficient permissions to access the groups that will be used for platform authen

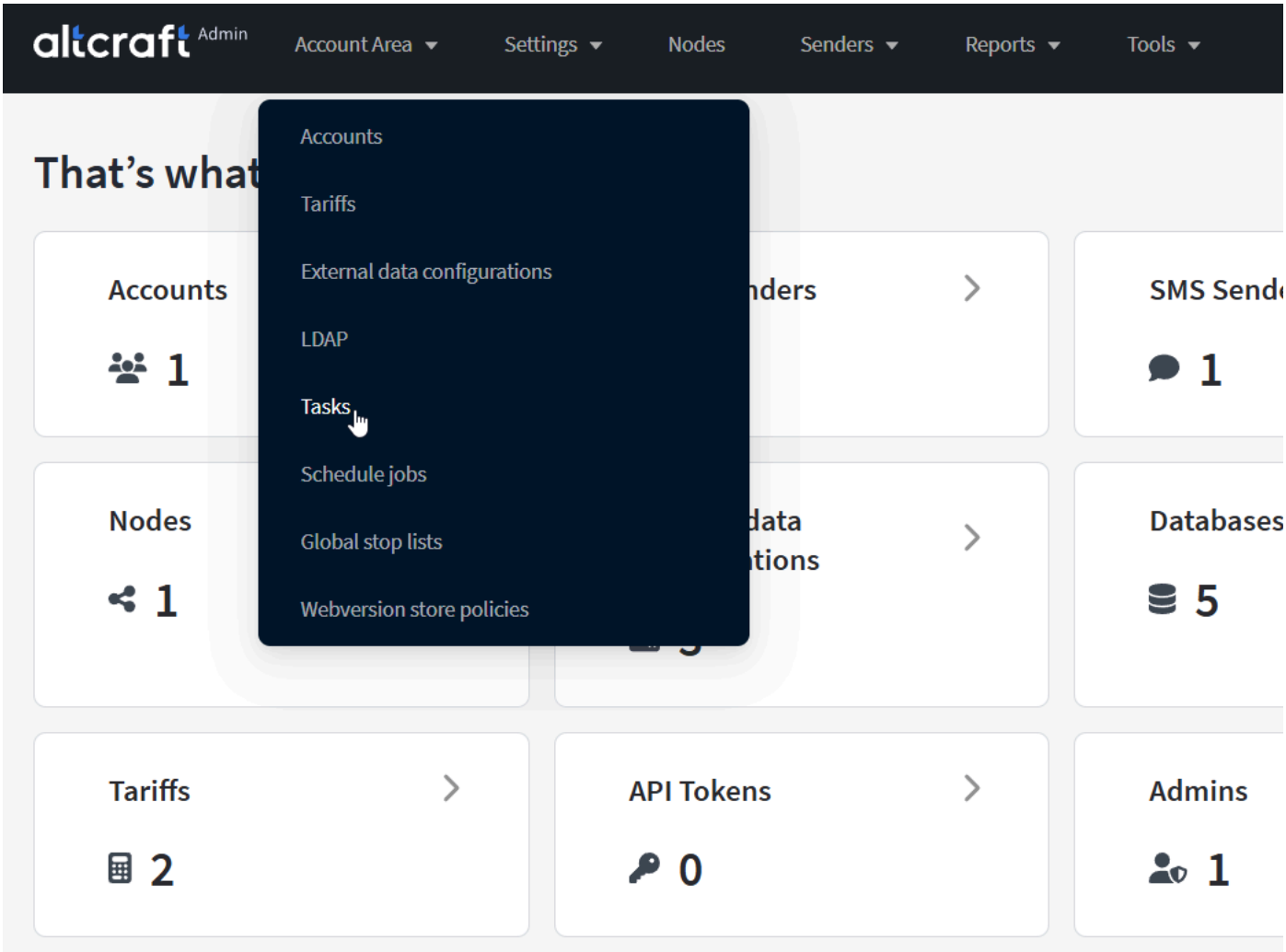
Tasks

Description

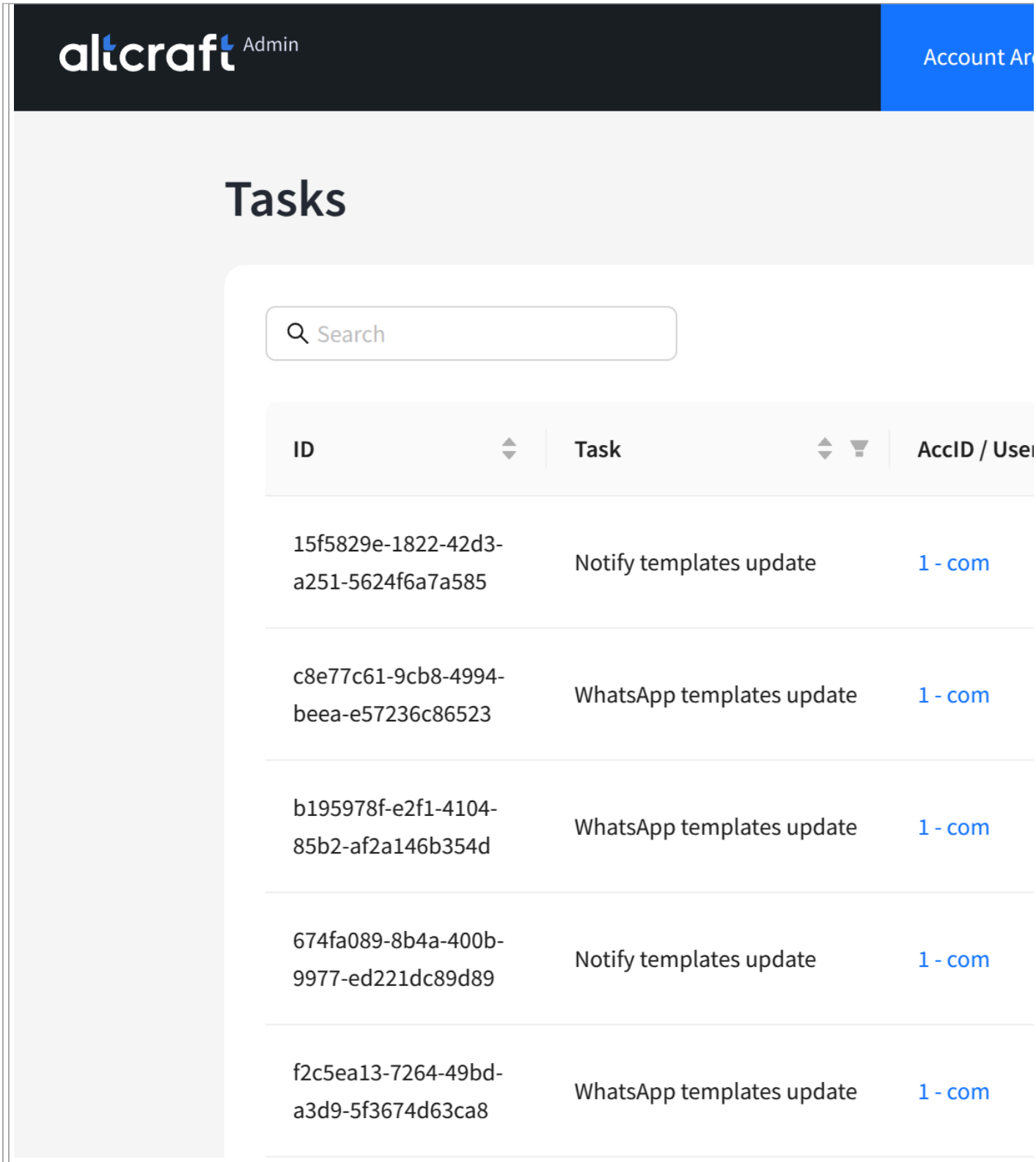
Tasks are processes that interact with Altcraft entities. They enable many of the platform's features.

Connection Settings

To view the tasks, in the main menu select Account Area — Tasks:



The list will display existing tasks.



Each task record contains the following data:

- **ID — task identifier.**
- **Task — task type.**
- **AccID / User — account and user who initiated the task.**
- **Machine ID — ID of the machine where the task was processed.**
- **State — task state.**
- **Created — task creation date.**
- **Updated by — user who updated the task.**
- **Actions — available actions: viewing process arguments and result parameters.**

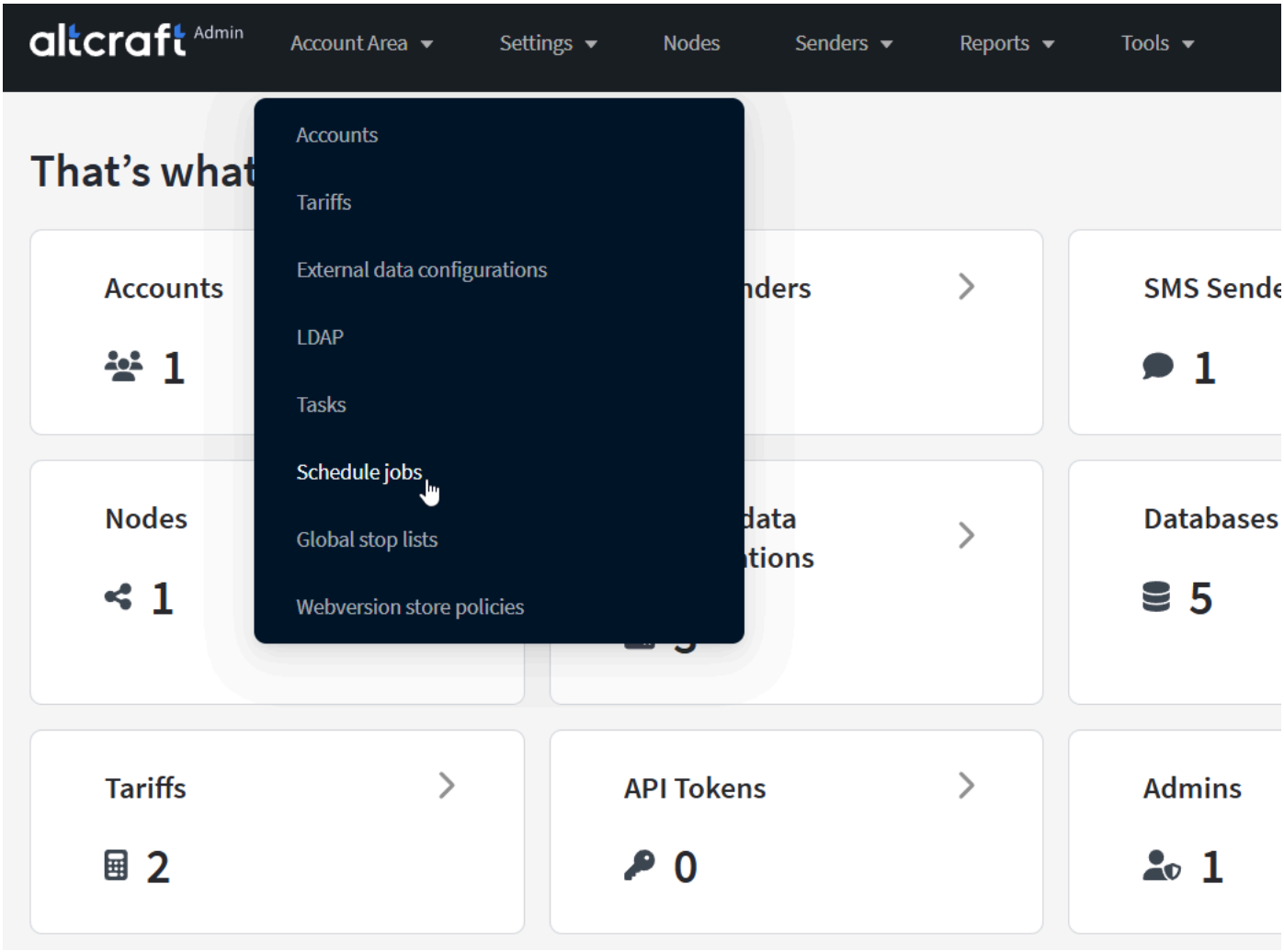
Schedule Jobs

Description

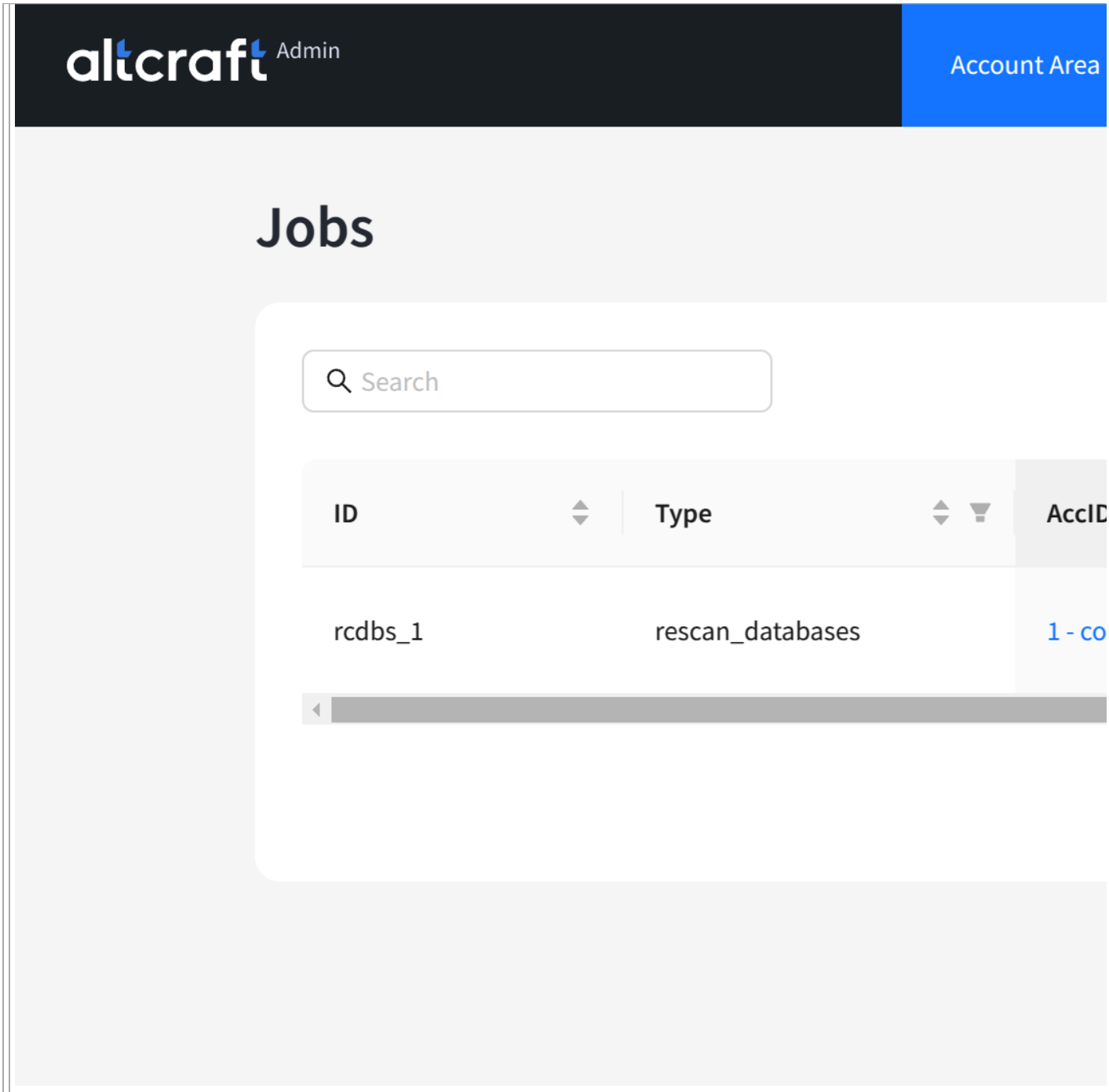
Scheduled jobs are processes launched periodically to support platform functionality. Schedule Jobs diffe

Connection Settings

To view the jobs, in the main menu select Account Area — Schedule Jobs:



The list will display existing scheduled jobs.



Each job record contains the following data:

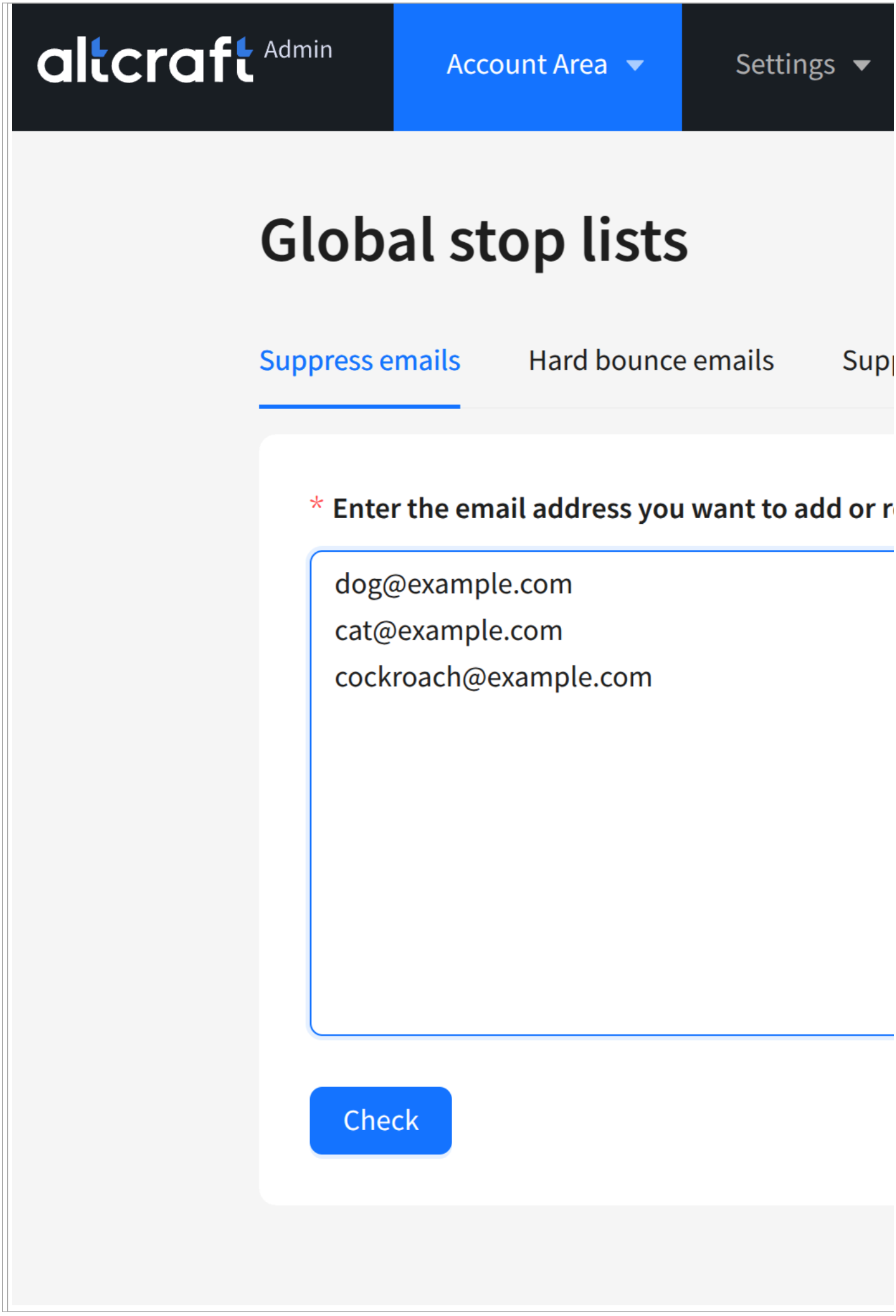
- **ID — job identifier.**
- **Task — job type.**
- **AccID / User — account and user who initiated the job.**
- **Machine ID — ID of the machine where the job was processed.**
- **Status — job status.**
- **Source ID — source identifier.**
- **Arguments — process arguments.**
- **Next launch — date of the next process run.**
- **Updated by — user who updated the job.**

Global Stop Lists

In the Altcraft Admin Panel, you can manage the platform's global stop list. The global stop list consists o

- **Suppress emails — the **blacklist** of email addresses. These email addresses may be **valid**, but rates.**
- **Hard bounce emails — a list of **invalid emails** that returned 5xx class errors from mail servers — i**
- **Suppress phones — the **blacklist** of phone numbers. These phone numbers may be valid, but for va**
- **Hard bounce phones — a list of invalid phone numbers that returned errors when sending SMS — in m**

To view the stop lists, in the main menu select Account Area — Global stop lists:


altcraft Admin
Account Area
Settings
Global stop lists
Suppress emails
Hard bounce emails
Sup
* Enter the email address you want to add or r
dog@example.com
cat@example.com
cockroach@example.com
Check

On the opened page you will see tabs corresponding to different stop lists.

If you want to add, remove, or check entries in the list, enter the phone numbers or email addresses you a

* **Enter the email address you want to add or remove from the suppr**

dog@example.com
cat@example.com
cockroach@example.com

Check

Result

Found:1 Not found:2

Emails
cat@example.com
cockroach@example.com
dog@example.com

You can remove or add entries to the list using the + and - buttons next to each item. The Add all not foun

Webversion Store Policies

Description

A web version storage policy is a set of rules that defines the directory and retention period for web versi

Pre-configuration

S3 storage is used to store web version objects. You can connect any S3-compatible storage such as Mini

```
"OBJECTS3_HOST": "",
"OBJECTS3_KEY_ID": "",
"OBJECTS3_KEY_SECRET": "",
"OBJECTS3_IS_SSL": false,
"OBJECTS3_INSECURE_REQUESTS": true,
"WEBVER_LEGACY_TTL_DAYS": 0
```

INFO

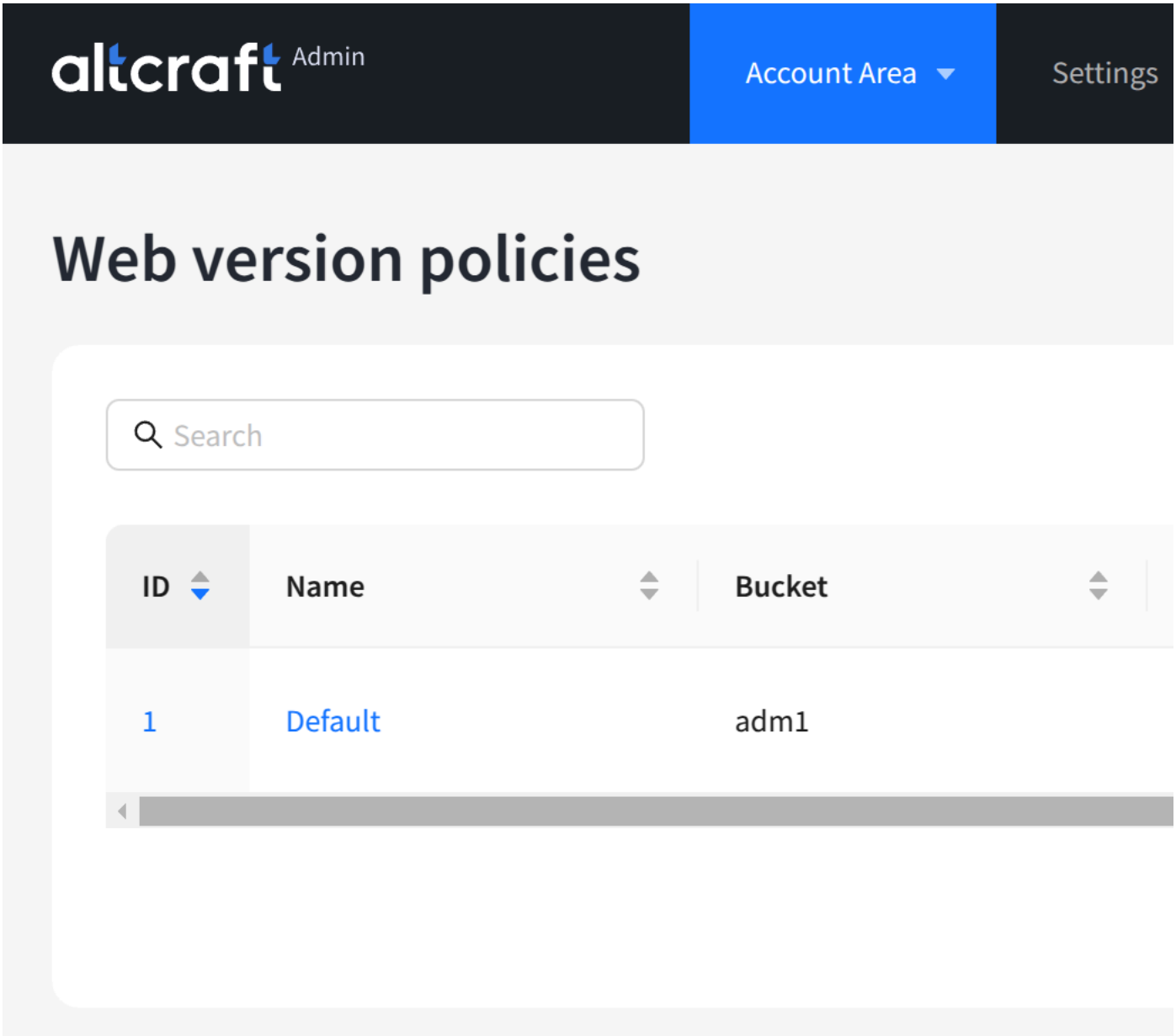
In version 66 and earlier, MongoDB is used to store web versions. In newer versions, this option is also a

```
"WEBVER_LEGACY_TTL_DAYS": 30 // Number of days to store web versions in Mongo if no policy is attached to the account
```

If at least one policy is created and attached to an account, web versions will be stored in S3 storage.

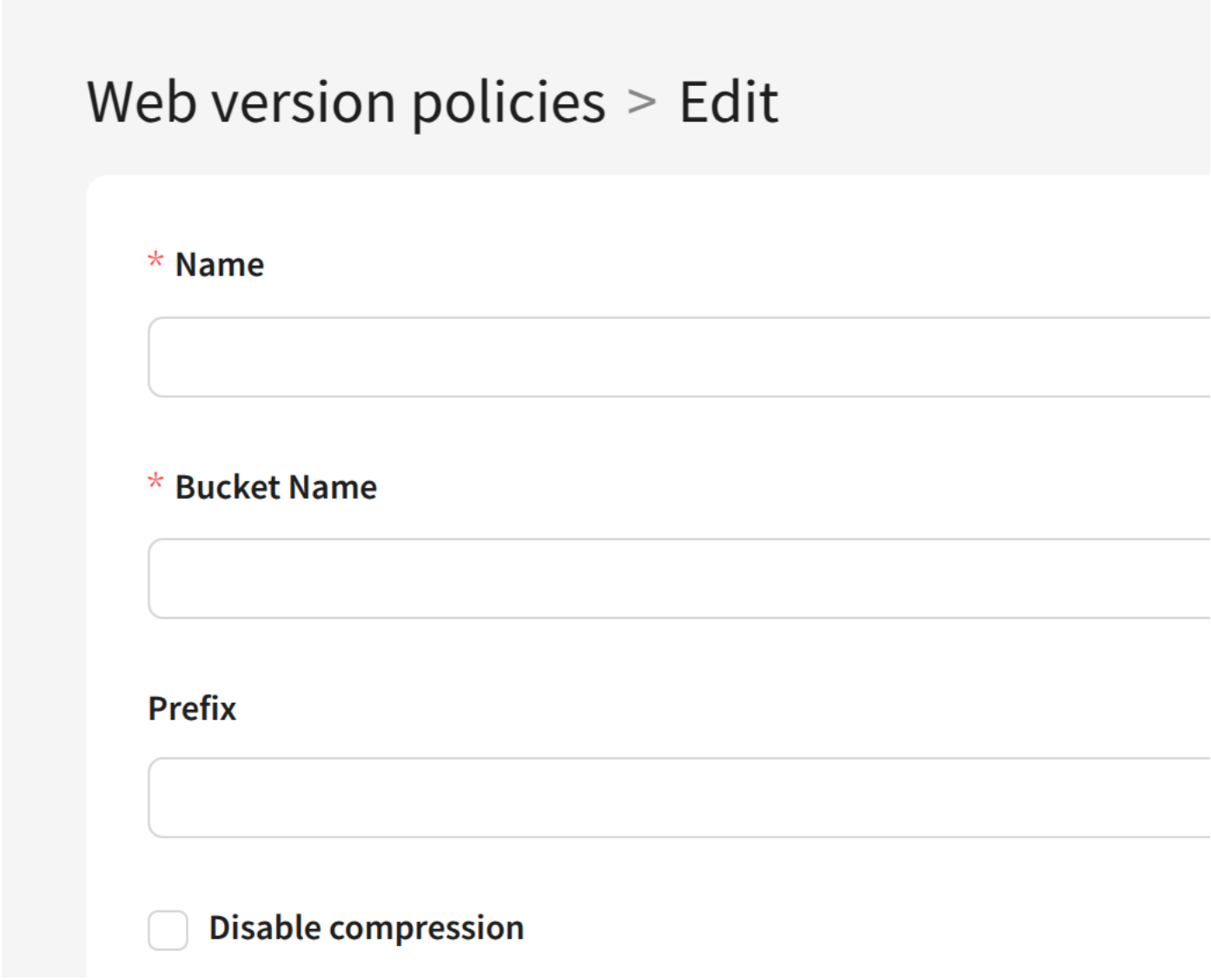
Creating a Policy

To manage policies, in the main menu select Account Area — Web version store policies.
The list displays existing policies. On the right side of the table are the edit and delete buttons. To edit, cl



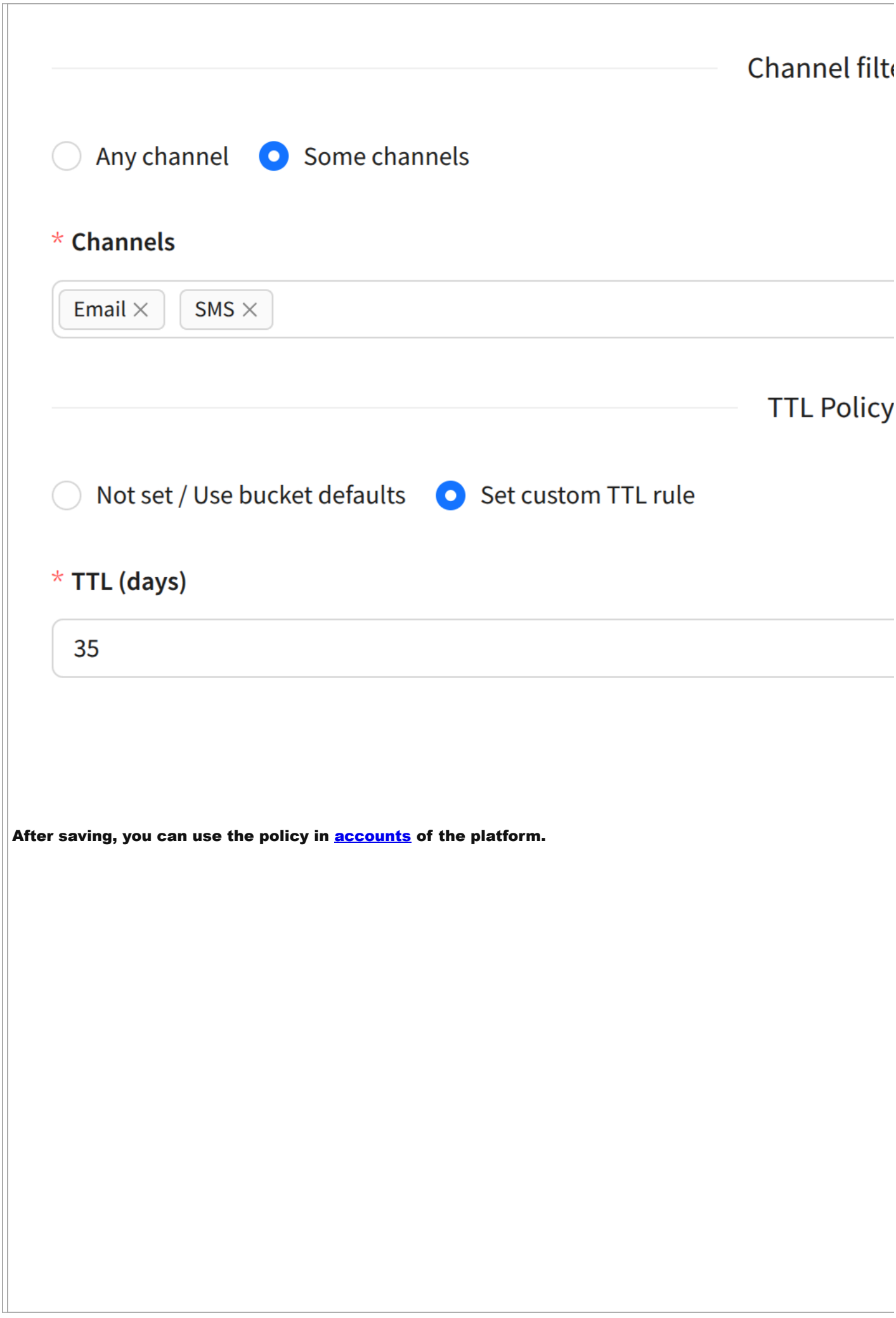
For each policy, set the following parameters:

- **Name — unique policy name.**
- **Bucket Name — the name of the bucket (container) where web version objects will be stored. The nar**
- **Prefix (optional) — a string at the beginning of the object key name, used to organize data inside the b**
- **The Disable compression option disables data compression within the policy.**



In the Channel filter block, select the channels where web versions of messages should be saved: all chan
In the TTL Policy block, set how long to store web versions:

- **Not set / Use bucket defaults — use the retention time set for the selected bucket.**
- **Set custom TTL rule — specify the retention time manually in the field below.**



After saving, you can use the policy in accounts of the platform.

Accounts

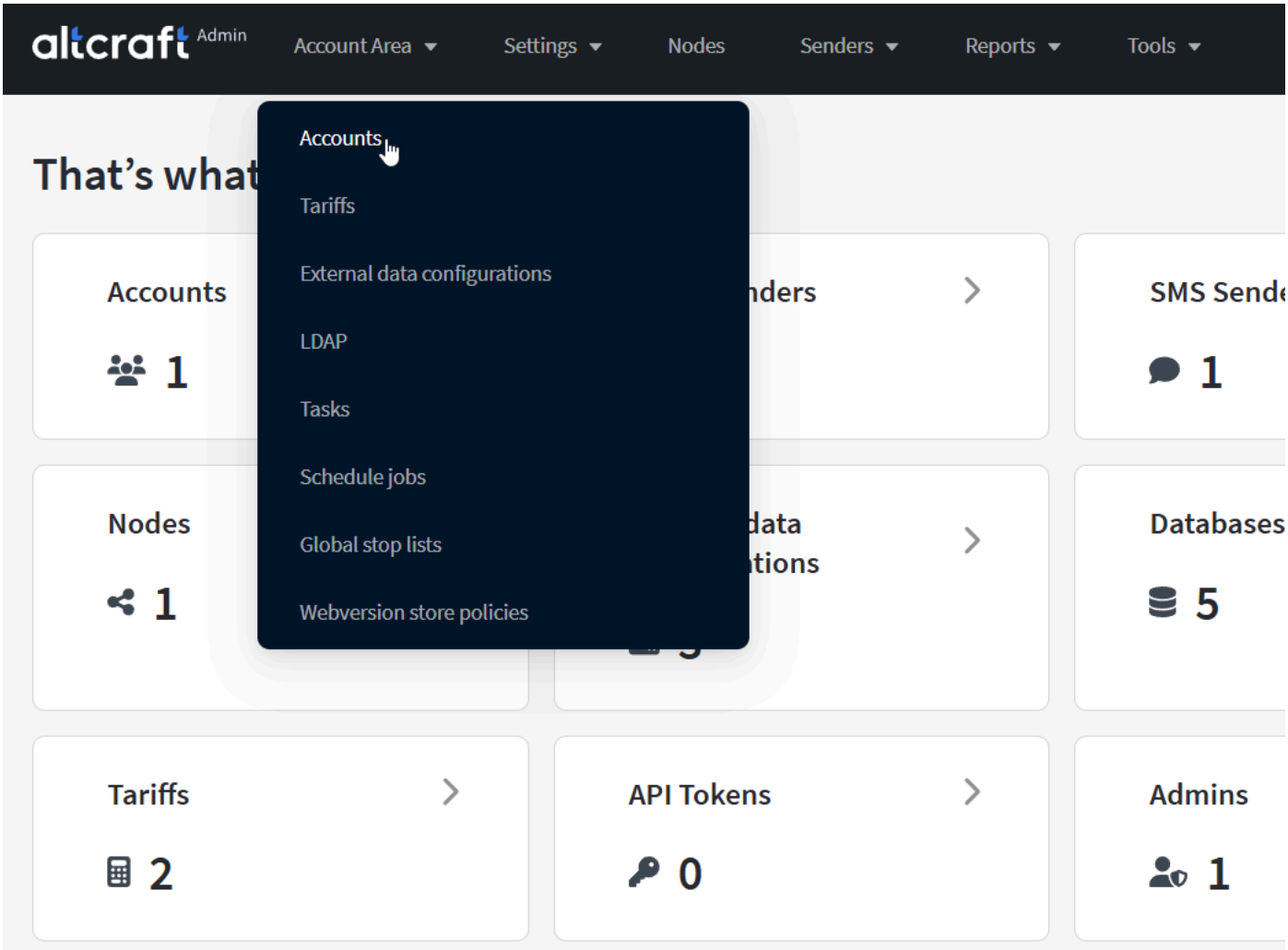
Description

Account is a unified environment for company marketers and related specialists to work with the User Int Master users of an account have access to all objects within the account's access groups, including cons An account uses shared internal and external platform resources:

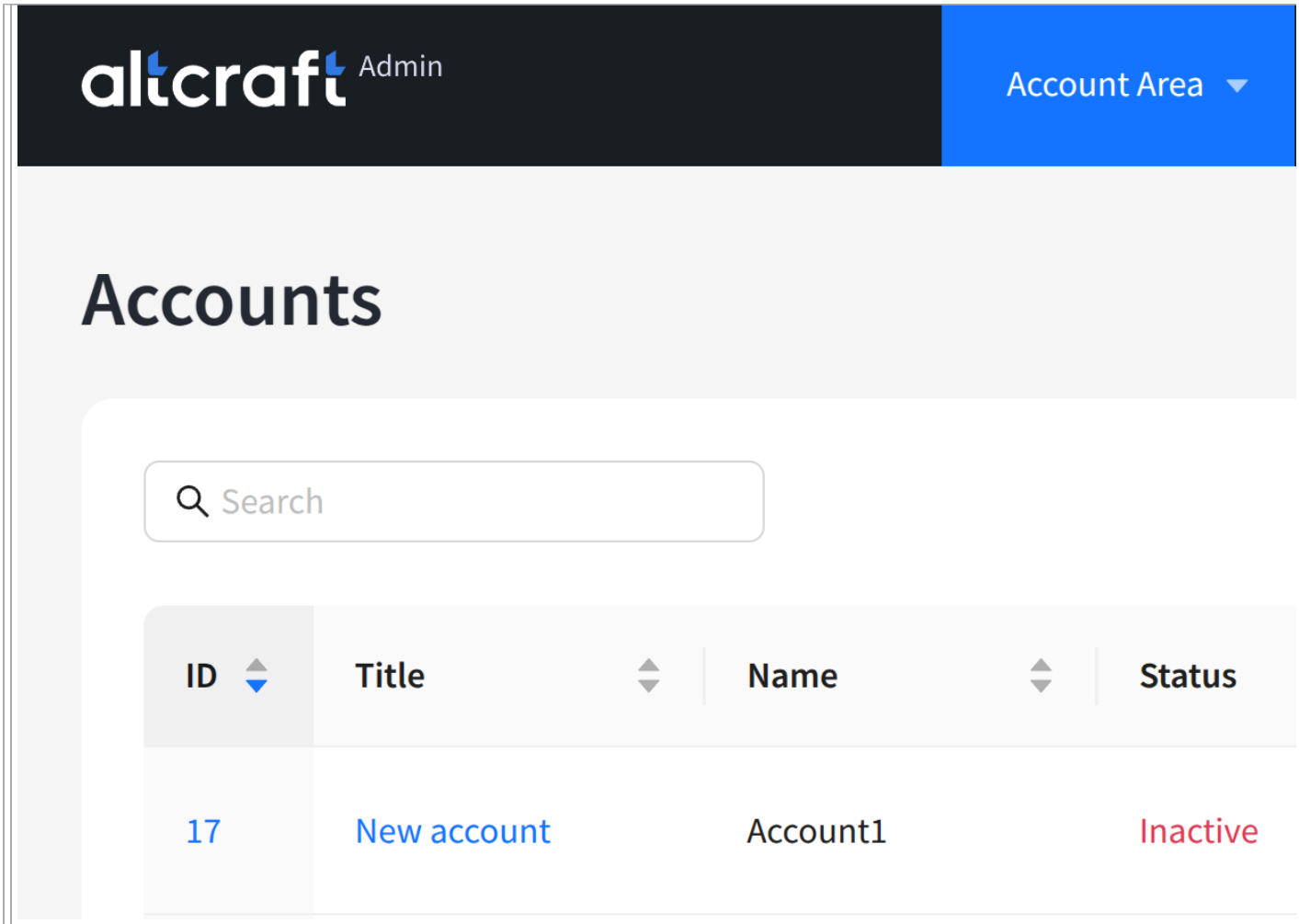
- **Tariffs**
- **Databases and Job Processing Server**
- **Senders**
- **Connectors to External Databases**
- **Connectors to External Directory Services (LDAP)**

Creating and Editing

To manage accounts, go to Account Area and select Accounts.



You will see a list of platform accounts:



To edit an account, click its identifier on the left. To create a new account, click + Create.

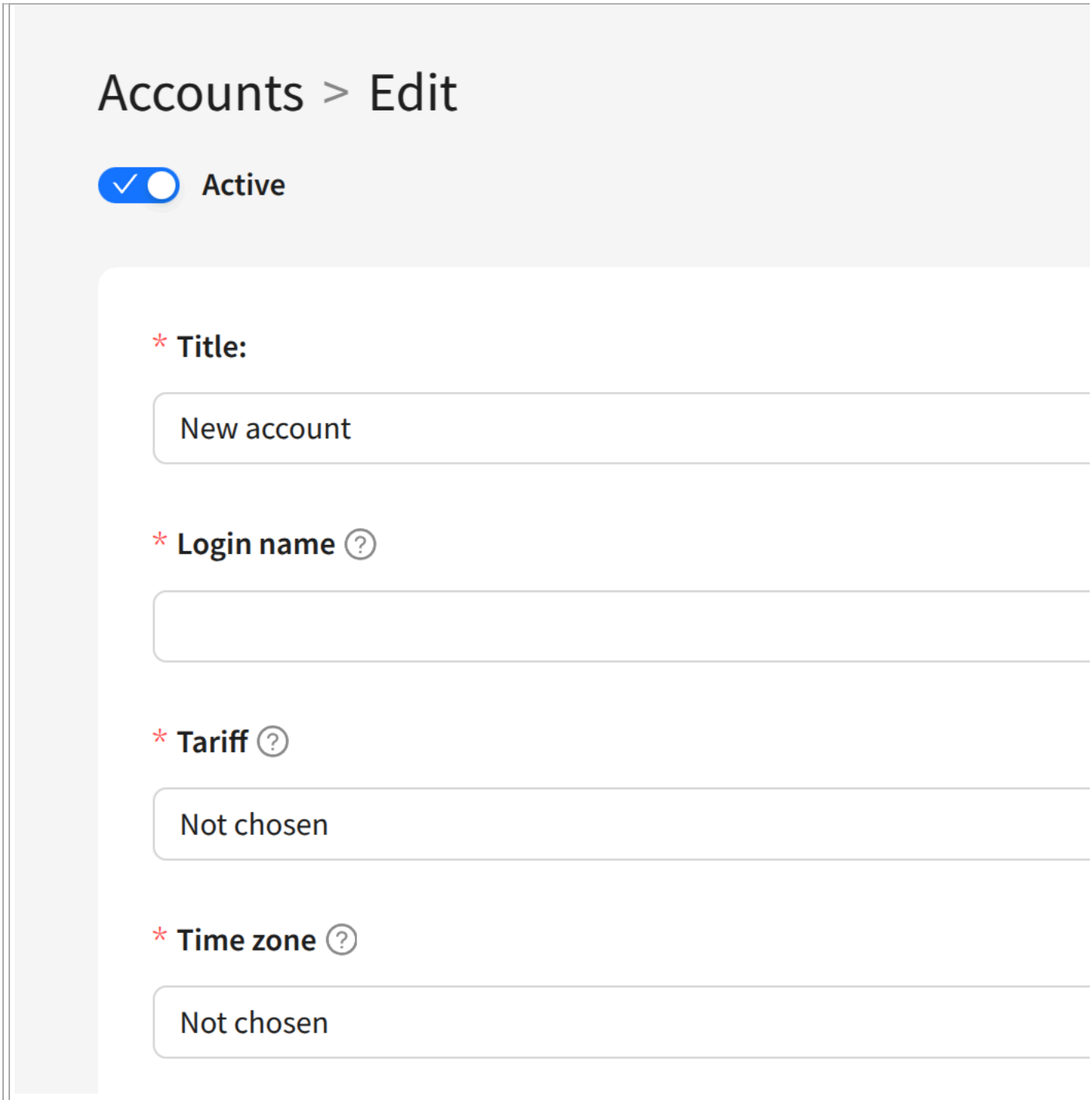
Main Settings

TIP

If needed, you can create an inactive account to activate it later when required. Use the Active/Inactive

For each new account, the following main settings are specified:

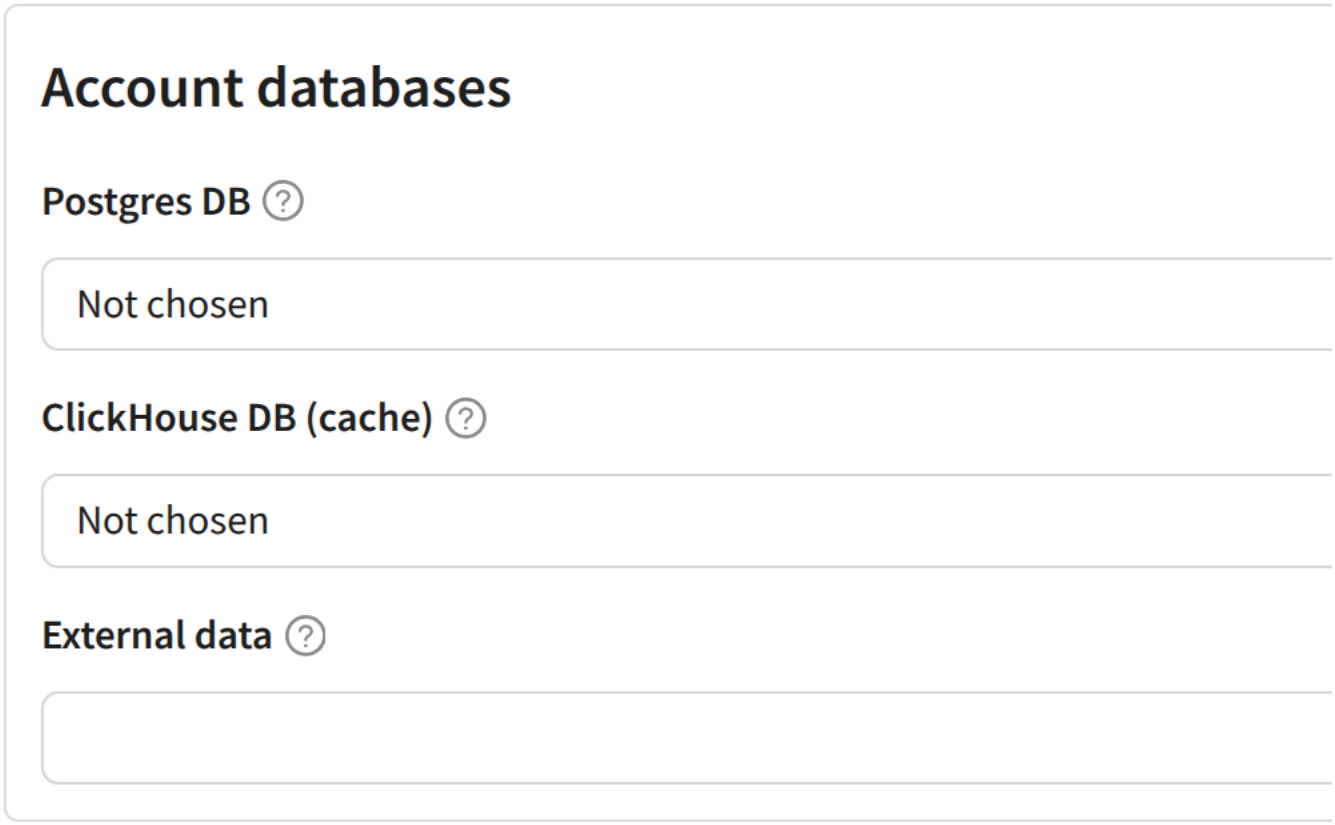
- **Title — the account name in the Admin Panel interface.**
- **Login name — the system name. Used in user logins of the account (username@login).**
- **Tariff — data and traffic limits for the account. Creating a tariff is described in this article.**
- **Time zone — the time zone used for timestamps in the account.**



Account Databases

In this section, specify previously created database connectors required for the platform:

- **Postgres DB — used by the Market module to store product and order data.**
- **ClickHouse DB — used to store history, statistics, and build custom reports. These data serve as the l**
- **External data — SQL connectors for integration with external infrastructure.**

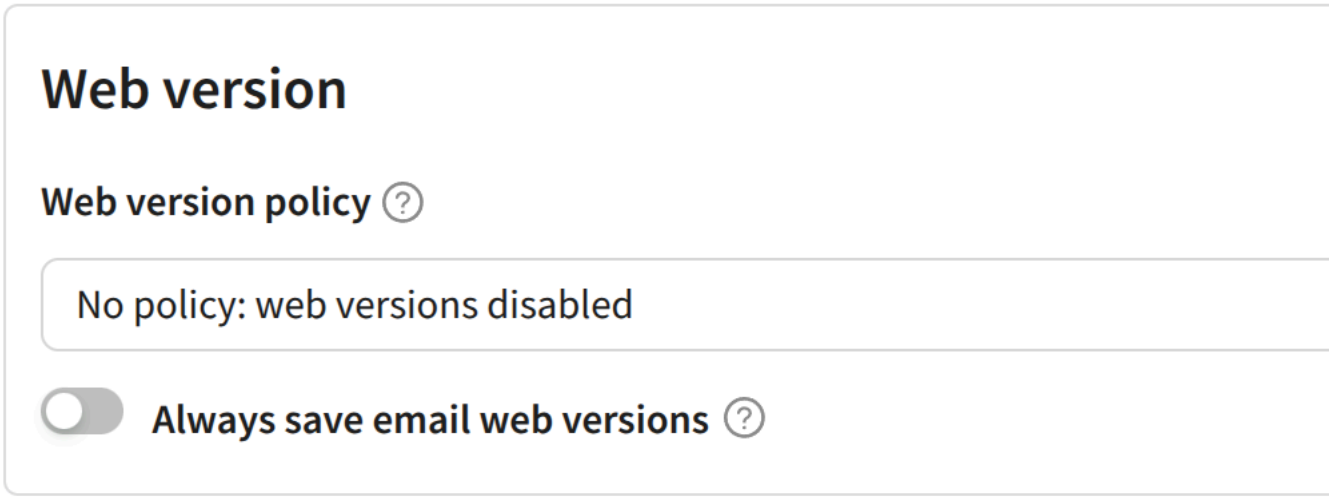


NOTE

When deploying platform components in a cluster to implement a fault-tolerant operation scheme, config

Web Version Settings

In this section, specify the web version storage policy for the account. Channels and retention periods are

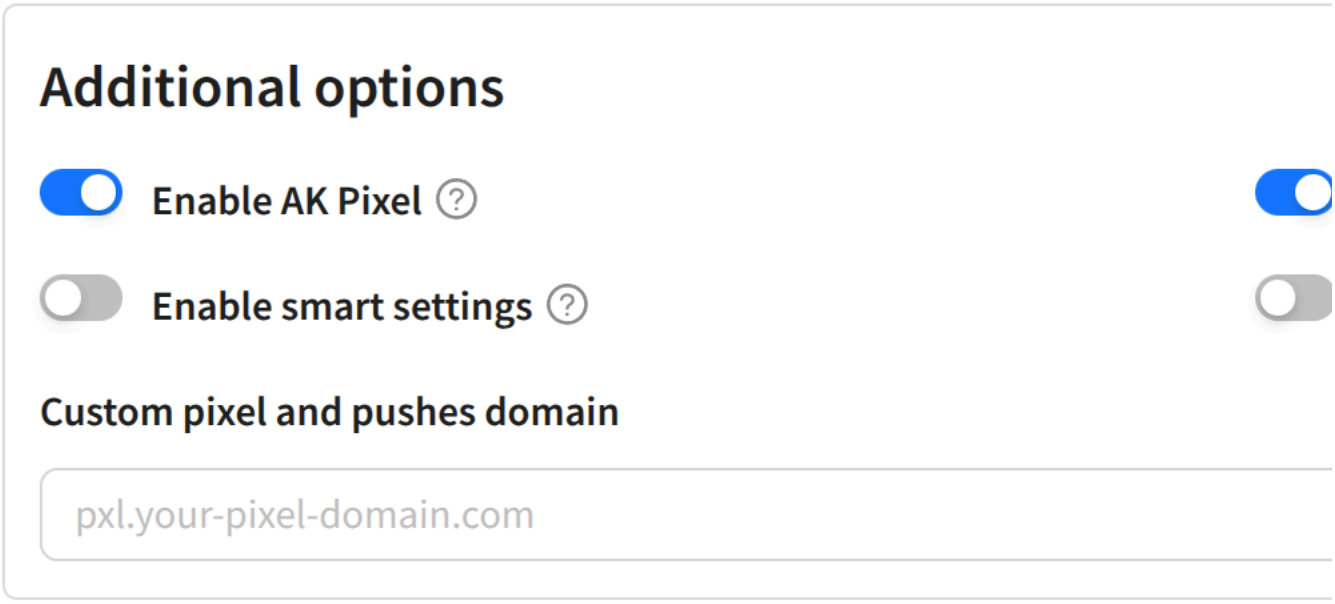


Additional Settings

The following additional account settings are available:

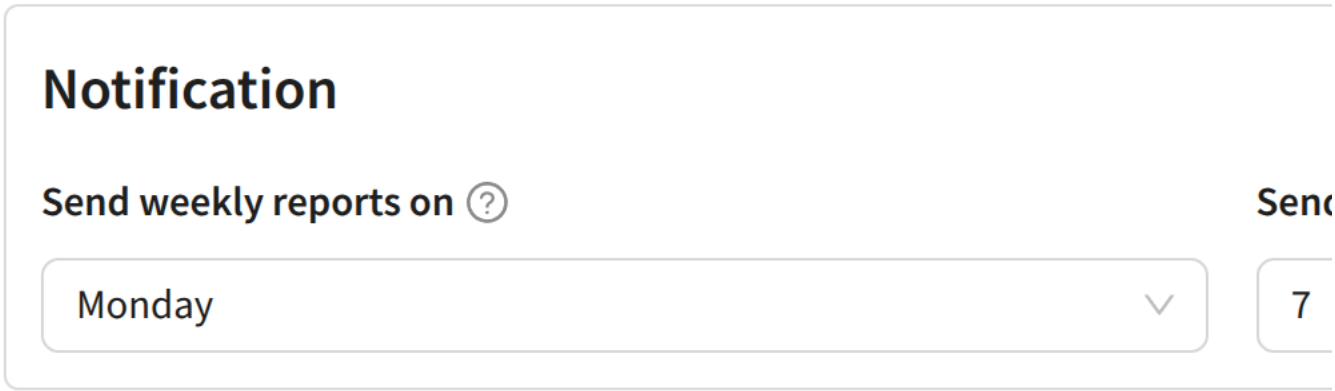
- **Enable AK Pixel — enable tracking pixels for the account.**
- **Enable smart settings — activate advanced mailing settings such as hidden copy and unconditional se**
- **Enable push channel — enable Web Push channel for the account.**
- **Capture email replies — enable collecting replies to email messages.**

For pixels and Push notifications to work, you must also specify a cookie processing domain.



Notifications

Additionally, you can set the day of the week for sending weekly email reports to account users (Send we



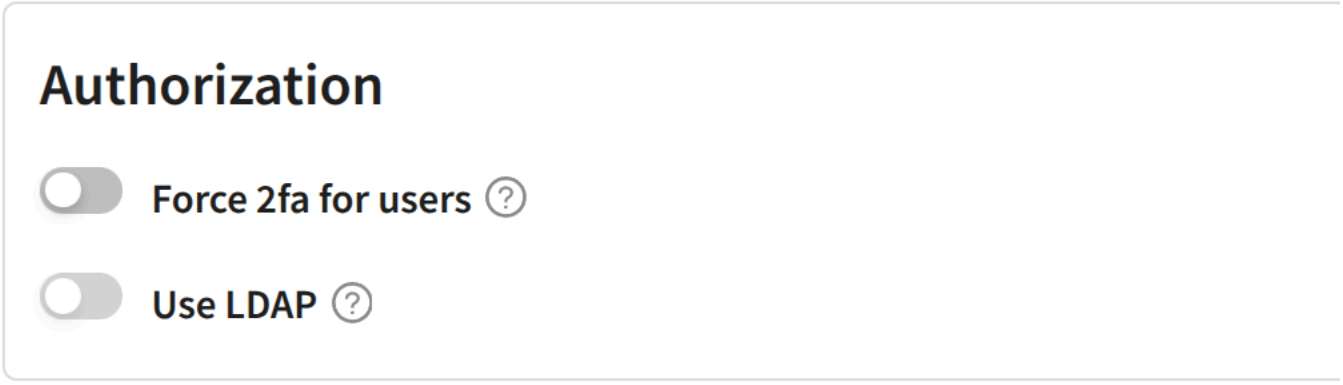
Authorization

In this section, you can configure the following parameters:

- **Force 2fa for users — enable mandatory two-factor authentication for account users.**
- **Use LDAP — allow employees to use corporate directory credentials, such as Active Directory, to log i**

TIP

A step-by-step guide for configuring LDAP access to Altcraft is available here.

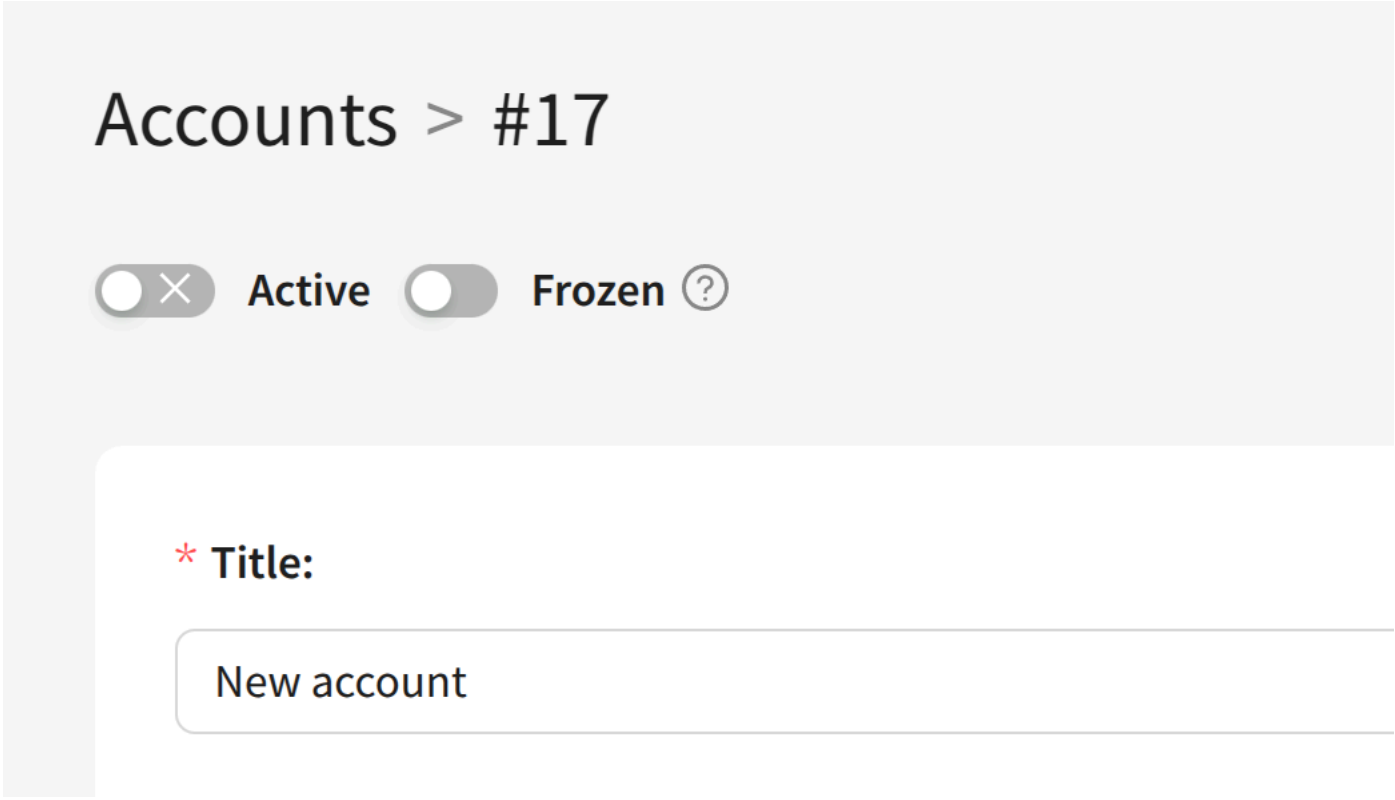


Managing an Existing Account

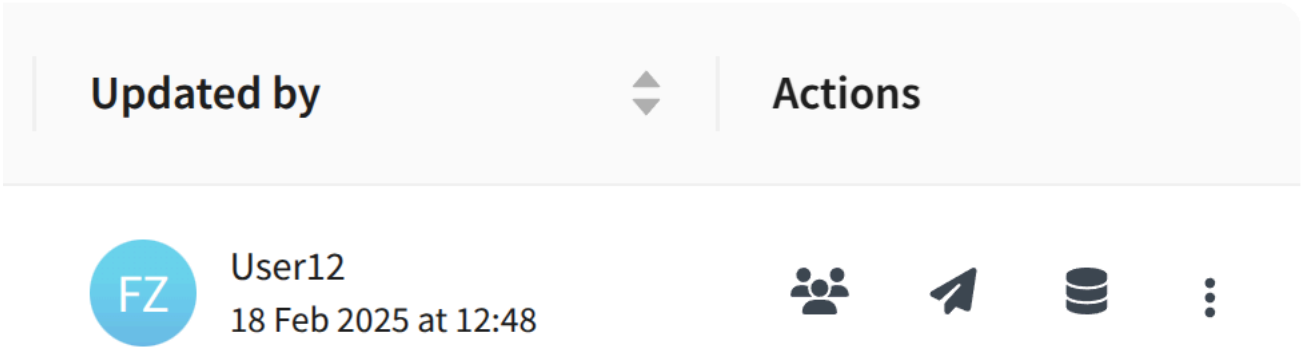
When the account is saved in the platform, the following features become available:

- **Managing account users (Users)**
- **Managing virtual senders (Virtual senders)**
- **Managing account databases (Account Databases)**
- **Activation and deactivation of the account. You can also configure notifications about deactivation to**
- **Freezing the account. When an account is frozen, creating and editing objects is not available.**

You can access these settings from the interface of an existing account:

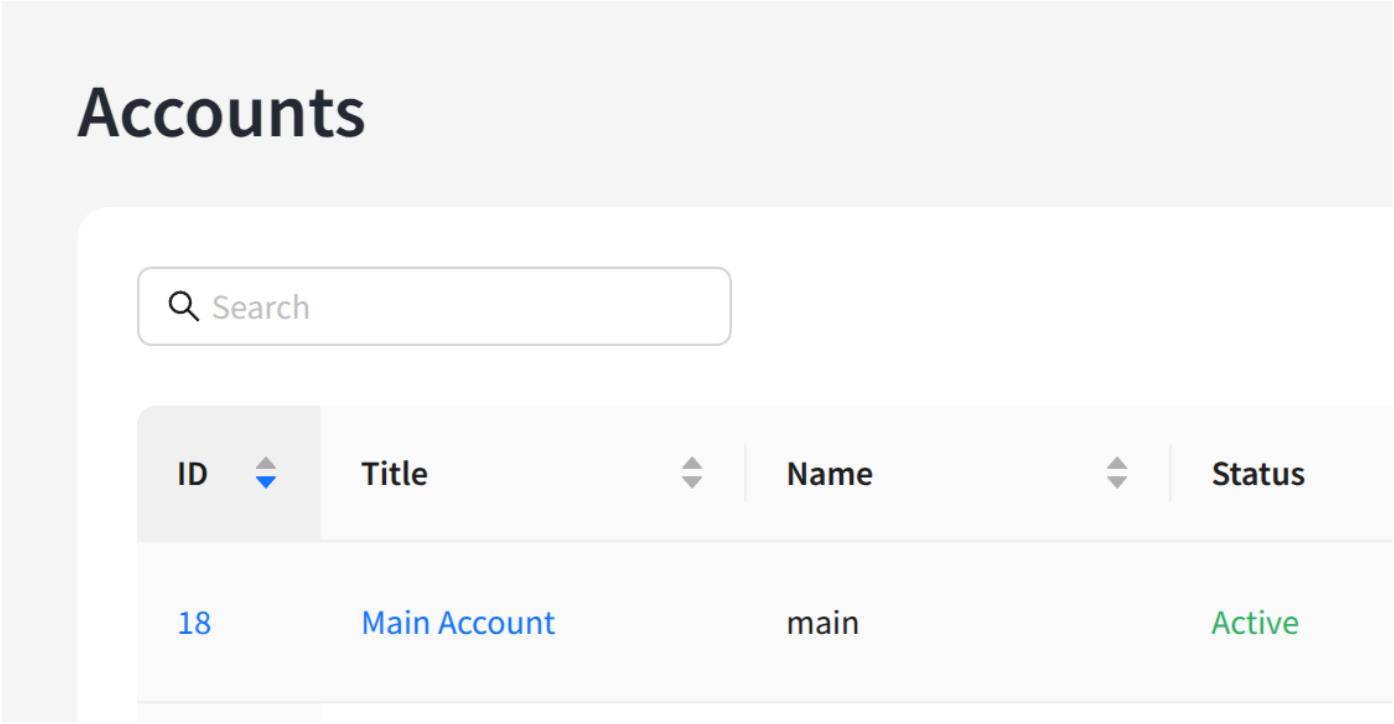


Or from the account list using the quick action buttons on the right side of the table:

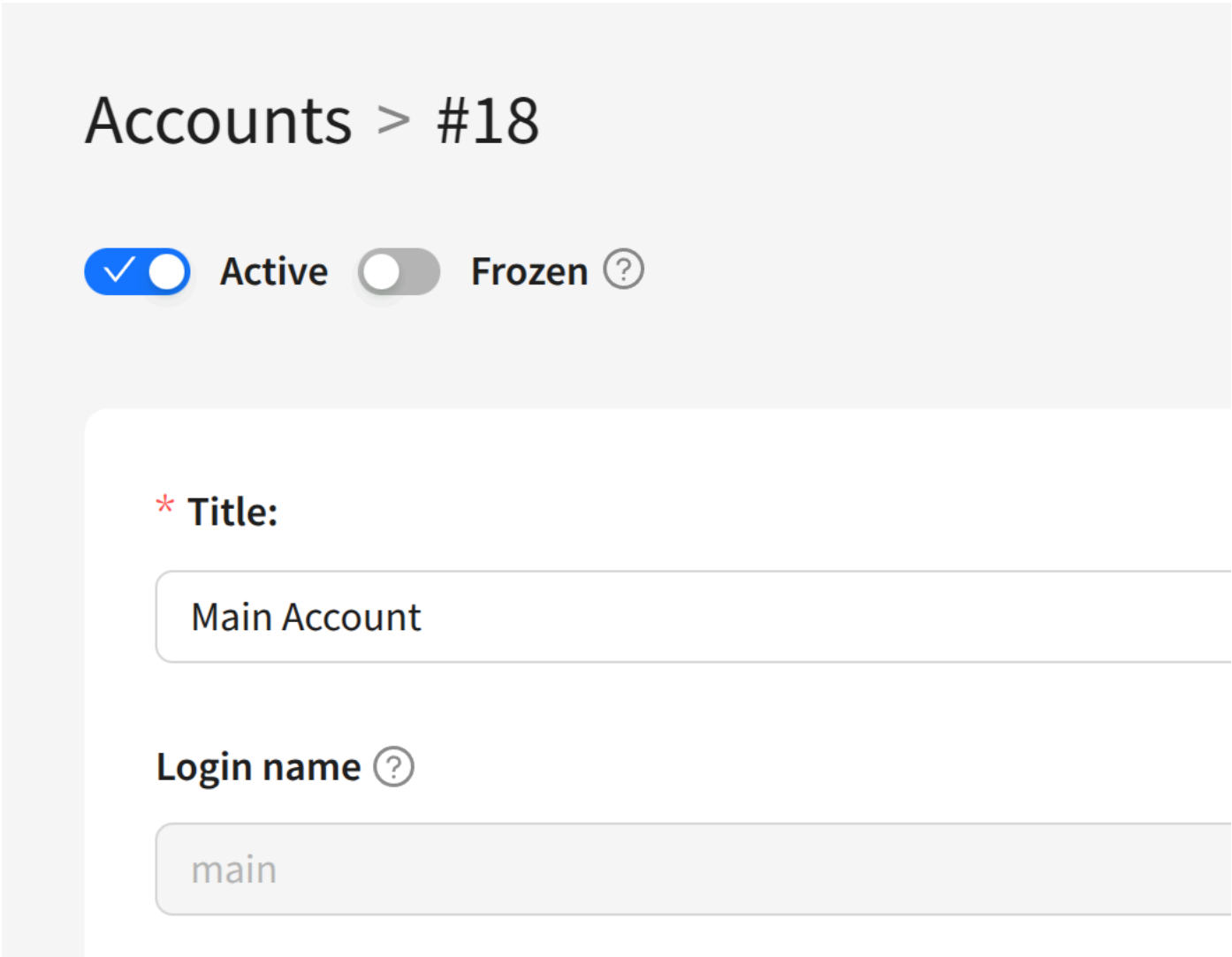


Account Users

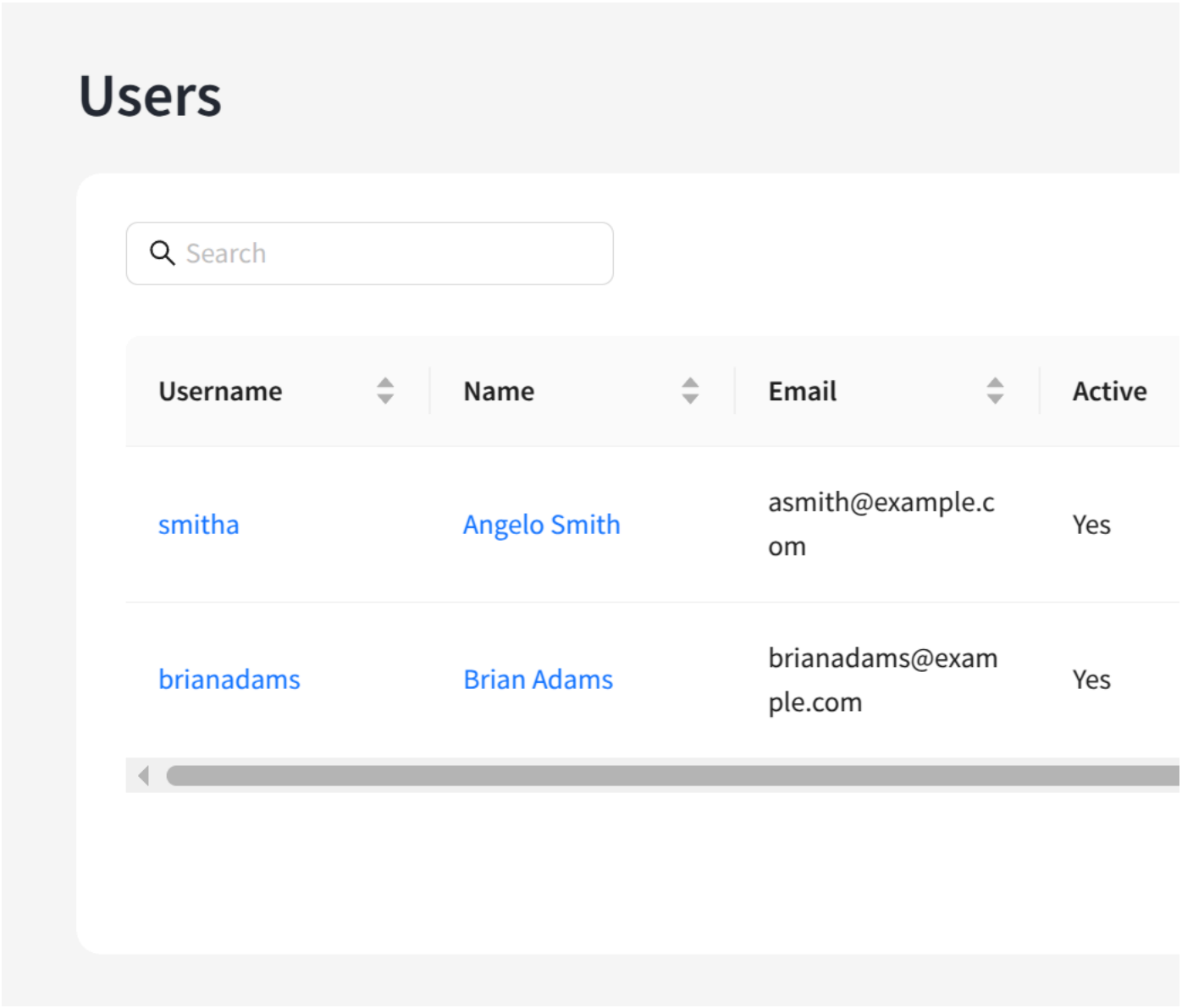
Managing account user accounts is available both in the Altcraft Admin Panel and in the User Interface. T
To manage users, in the list of accounts click the quick action button:



Or open the account editing menu and click the Users button:



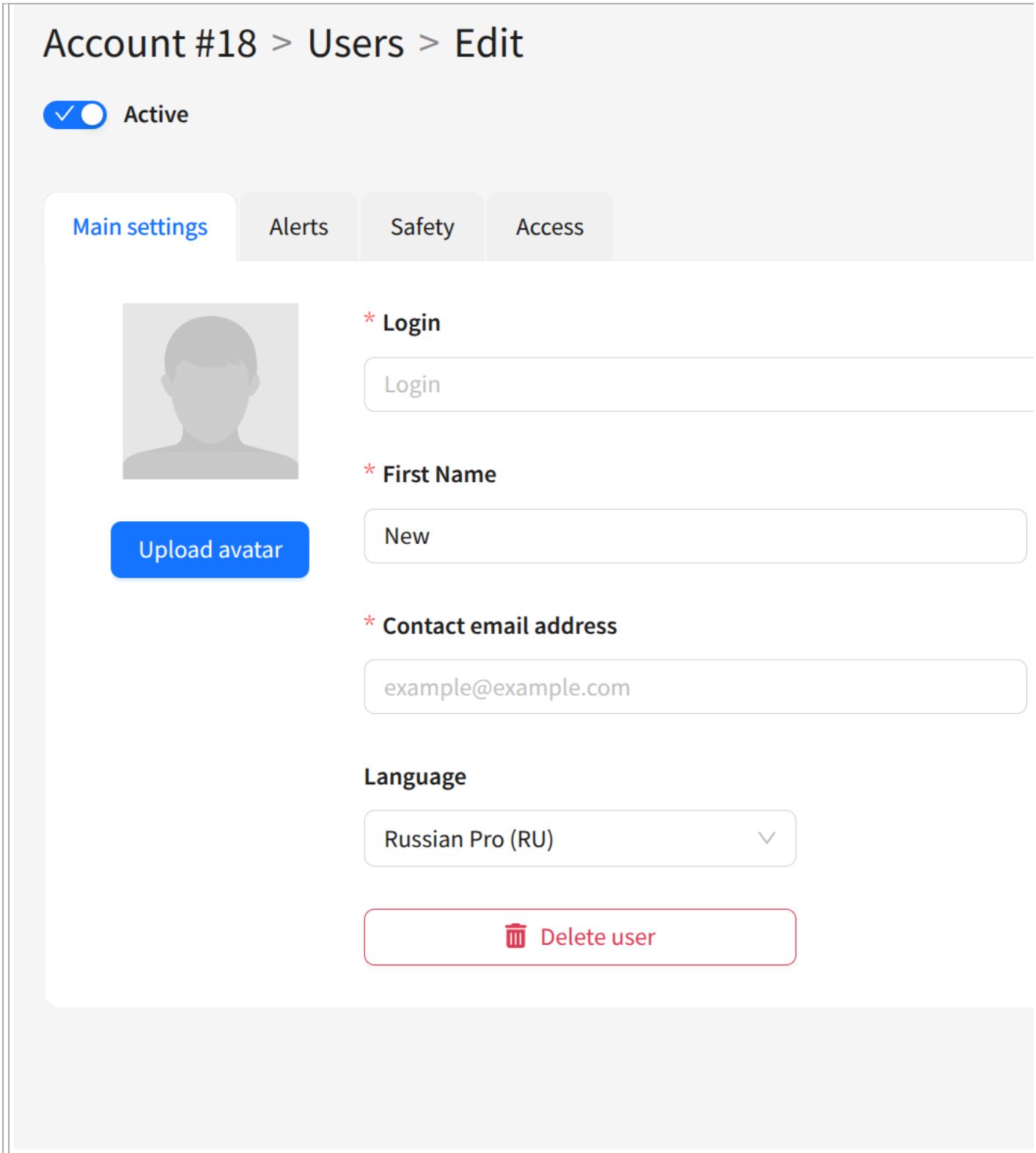
The list of users will open displaying existing accounts. On the right side of the table, there are buttons fo



General Settings

For each user, specify the following parameters:

- **Login — the system name. Used in the user login together with the system account name (login@acco**
- **First name — the user's first name.**
- **Last name — the user's last name.**
- **Contact email address — contact email address.**
- **Language — choose between Russian and English interface versions.**
- **Contact phone number — contact phone number (optional).**

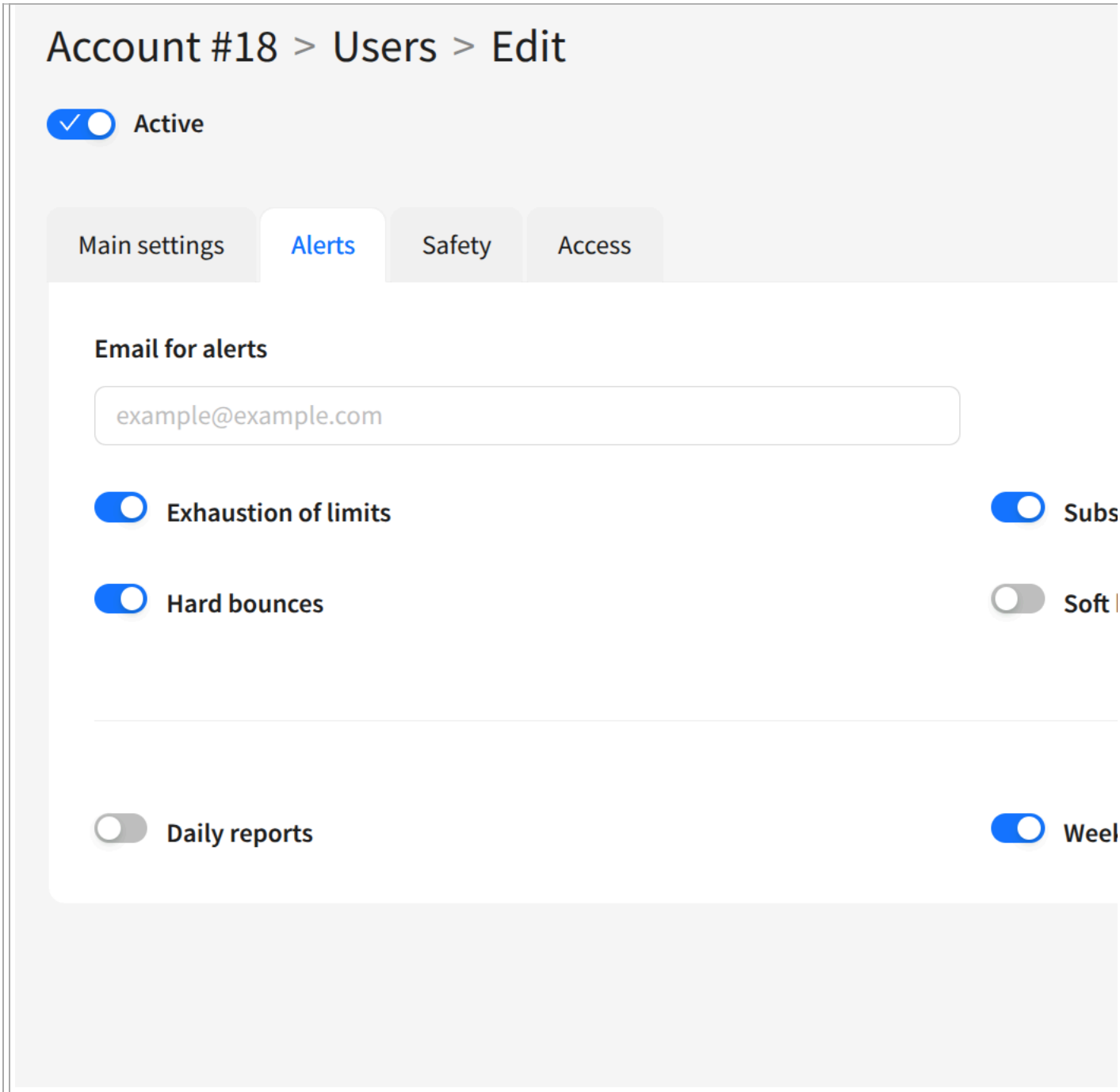


You can also set a user avatar here.

Notification Settings

In the Alerts block, you can specify the user's email for receiving Altcraft reports and notifications, and se

- **Exhaustion of limits — notifications about exceeding the limits of the tariff.**
- **Subscriber complaints — notifications about exceeding the safe threshold of subscriber complaints.**
- **Hard bounces — notifications about exceeding the safe threshold of sending to invalid emails.**
- **Soft bounces — notifications about exceeding the safe threshold of temporary delivery issues.**
- **Daily reports — enables sending daily reports on account mailings.**
- **Weekly reports — enables sending weekly reports on account mailings.**

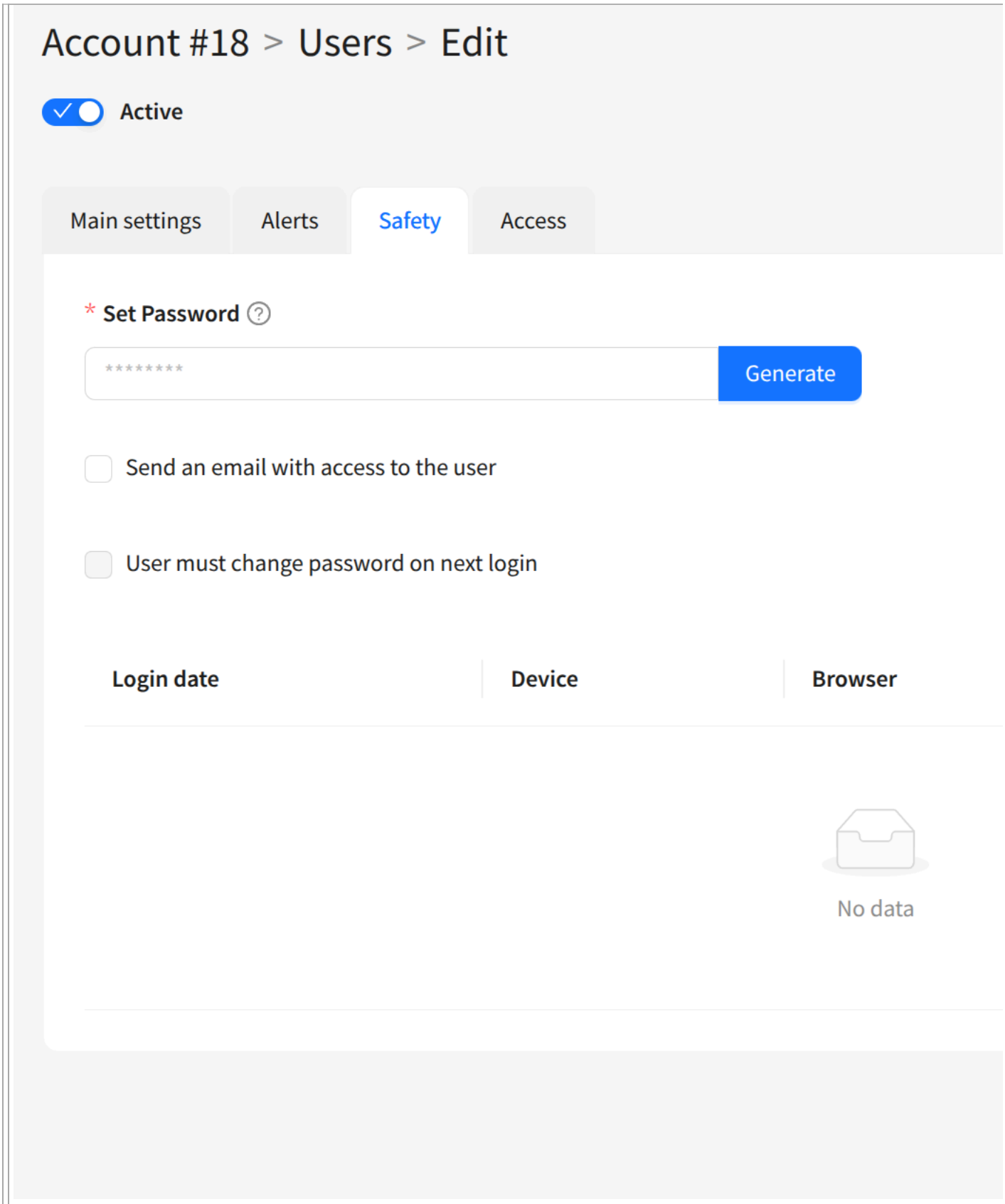


Safety Settings

The Safety section contains security settings for the user:

- **Set Password — set a password for the user.**
- **Send an email with access to the user — if enabled, an email with login details will be sent after savir**
- **User must change password on next login — when enabled, the user must change the password at the**

At the bottom of the screen, you can view the user's login history in the platform showing login date, brow



Access Settings

For each user, you can configure access to specific account objects and Altcraft features. More about acc

- **Assigned roles — assign access Roles to the user for account objects. Roles are created and edited ir**
- **Assigned groups — grant the user access to Groups of account objects. Master users can create obje**
- **Master user — grant the user access to the Main Group of the account, management of other users, a**
- **Is LDAP user — use the [LDAP protocol](/en/admin-guide/admin-panel/account-area/ldap) for user authe**

Account #18 > Users > Edit

Active

Main settings | Alerts | Safety | Access

* Assigned groups

All groups

Assigned roles

Master user

Is LDAP user

Account Virtual Senders

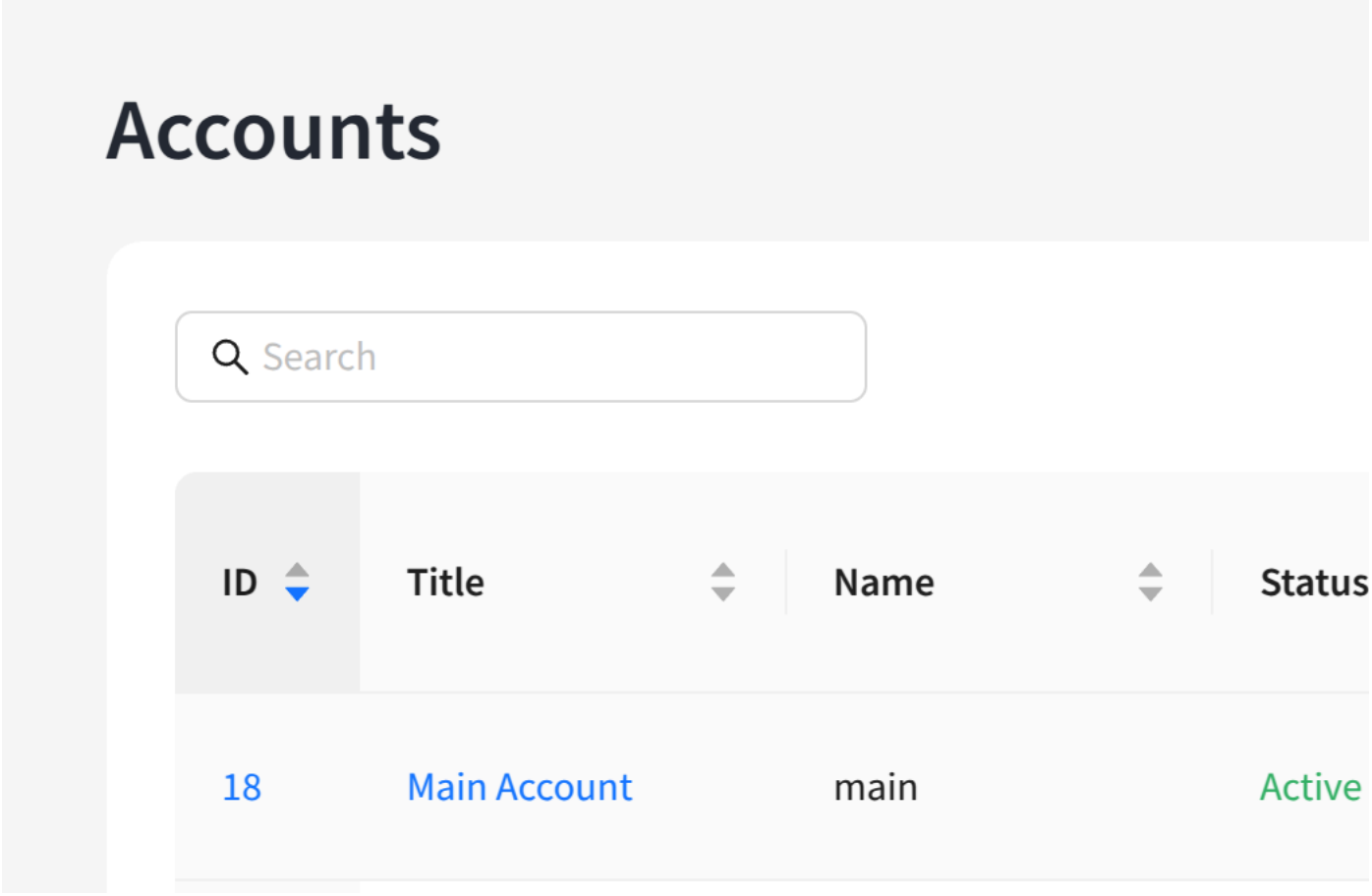
Description

Virtual senders group the platform's real senders and set rules and proportions for sending through them.

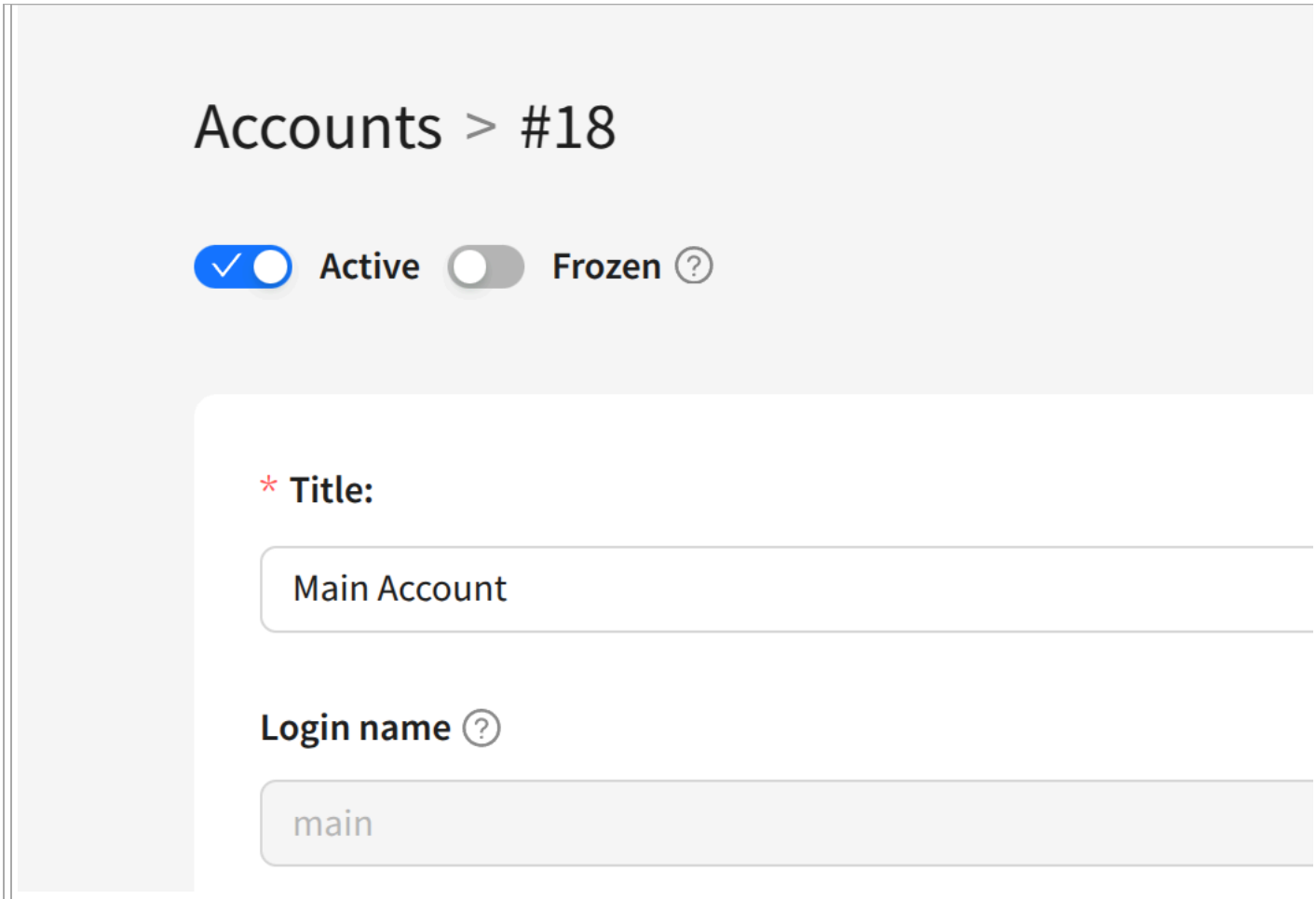
> **NOTE**
> **Users can select a virtual sender in mailing Smart settings. To enable this, the Smart settings option mu**

Creating, Editing, Deleting

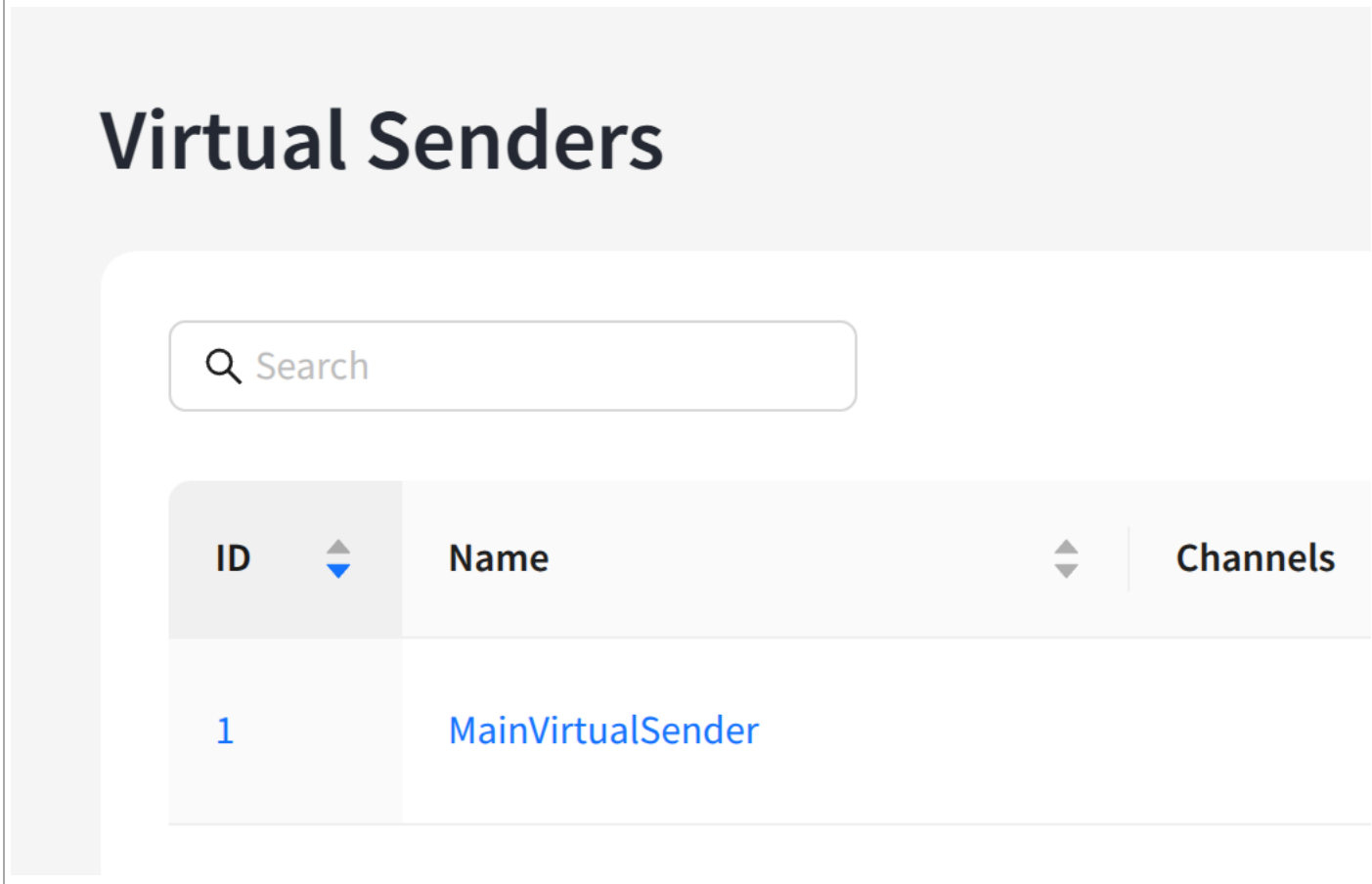
To manage virtual senders, in the list of accounts click the quick action button:



Or open the account editing screen and click the Virtual senders button:

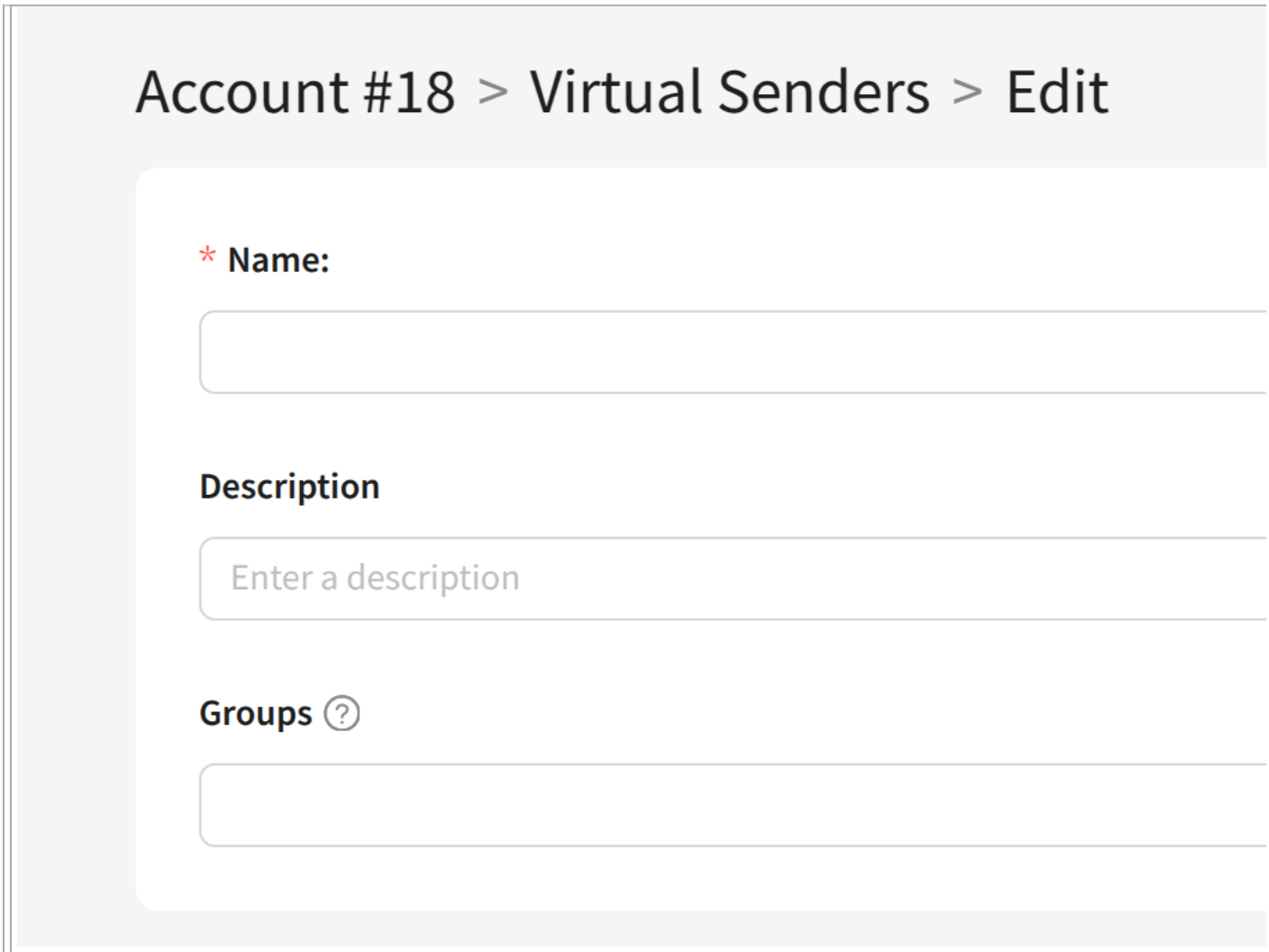


The list of existing virtual senders will open. On the right side of the table are the edit and delete buttons



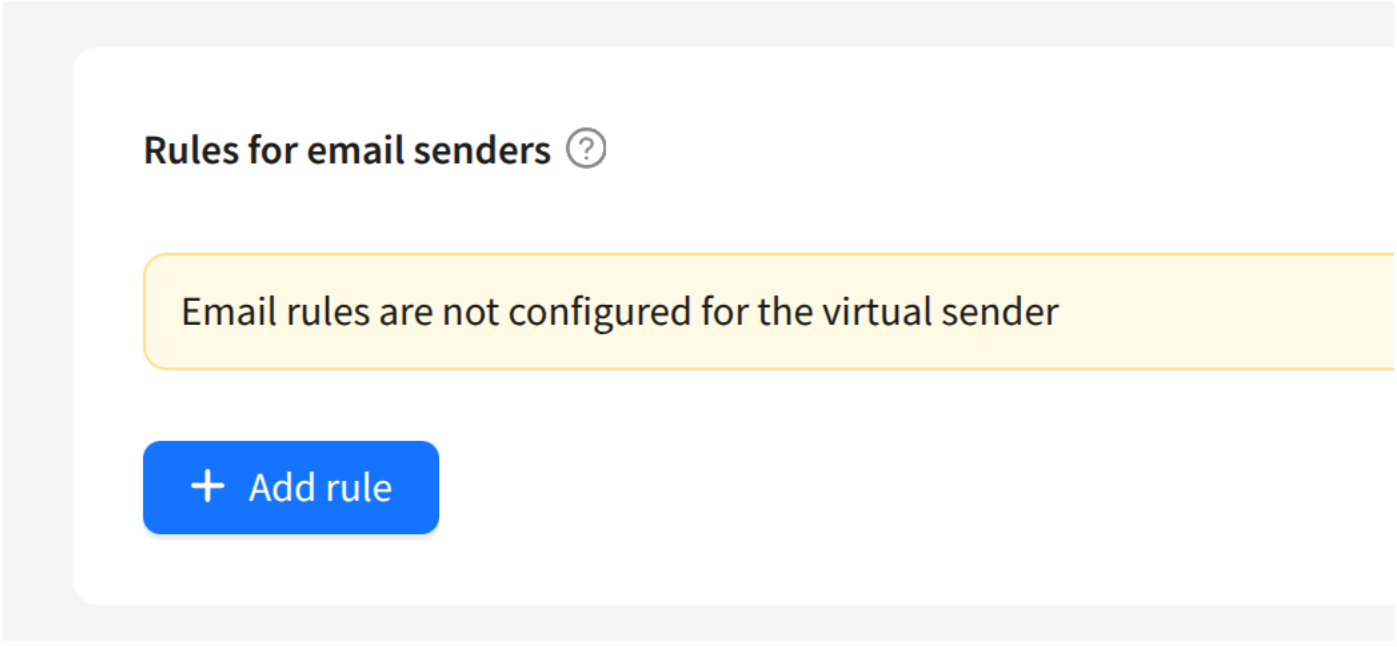
General Settings

In the general settings, specify a unique Name for each virtual sender and assign the account object Grou



Email Sender Rules

Email sender rules determine which real senders and in what proportions will be used under certain cond



Select the appropriate condition in the Rule dropdown menu:

- **Match by Group — applies to email mailings in a specific account object Group.**
- **Match by From domain — applies to sends from a specific domain.**
- **Match by To domain — applies to sends to specific email domains.**
- **Default (match all) — applies to all sends.**

Rules for email senders

Rule

Default (match all)

Selected sender

40 - Email senders

Percent

100

Override "From email"

Override "Reply-to email"

Remove sender

Add sender

Remove rule

Rule priority is determined by their position on the page: top to bottom. The Default (match all) rule must a

CAUTION

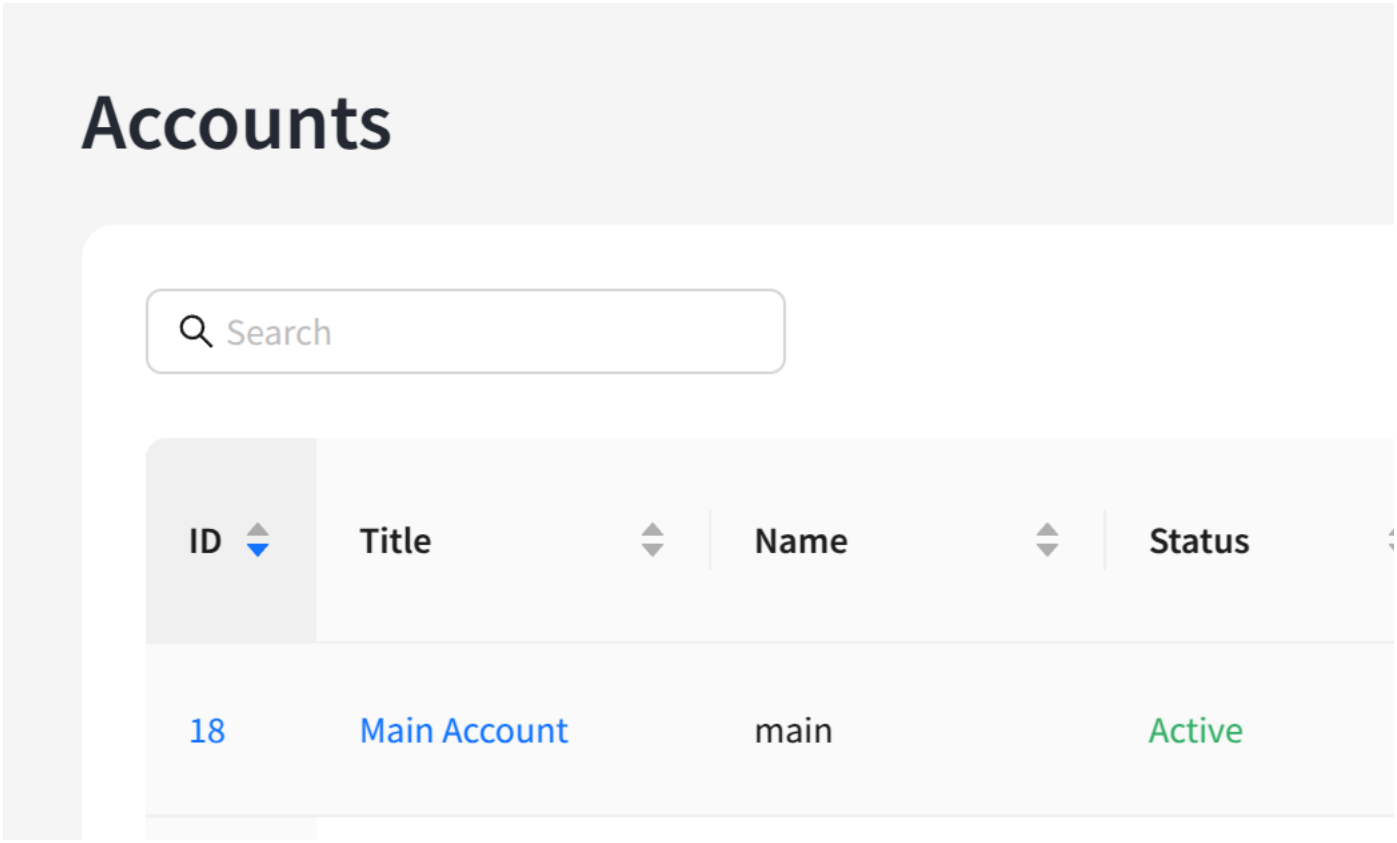
The default rule Default (match all) cannot be deleted or moved up because it handles all case

Account Database Indexes

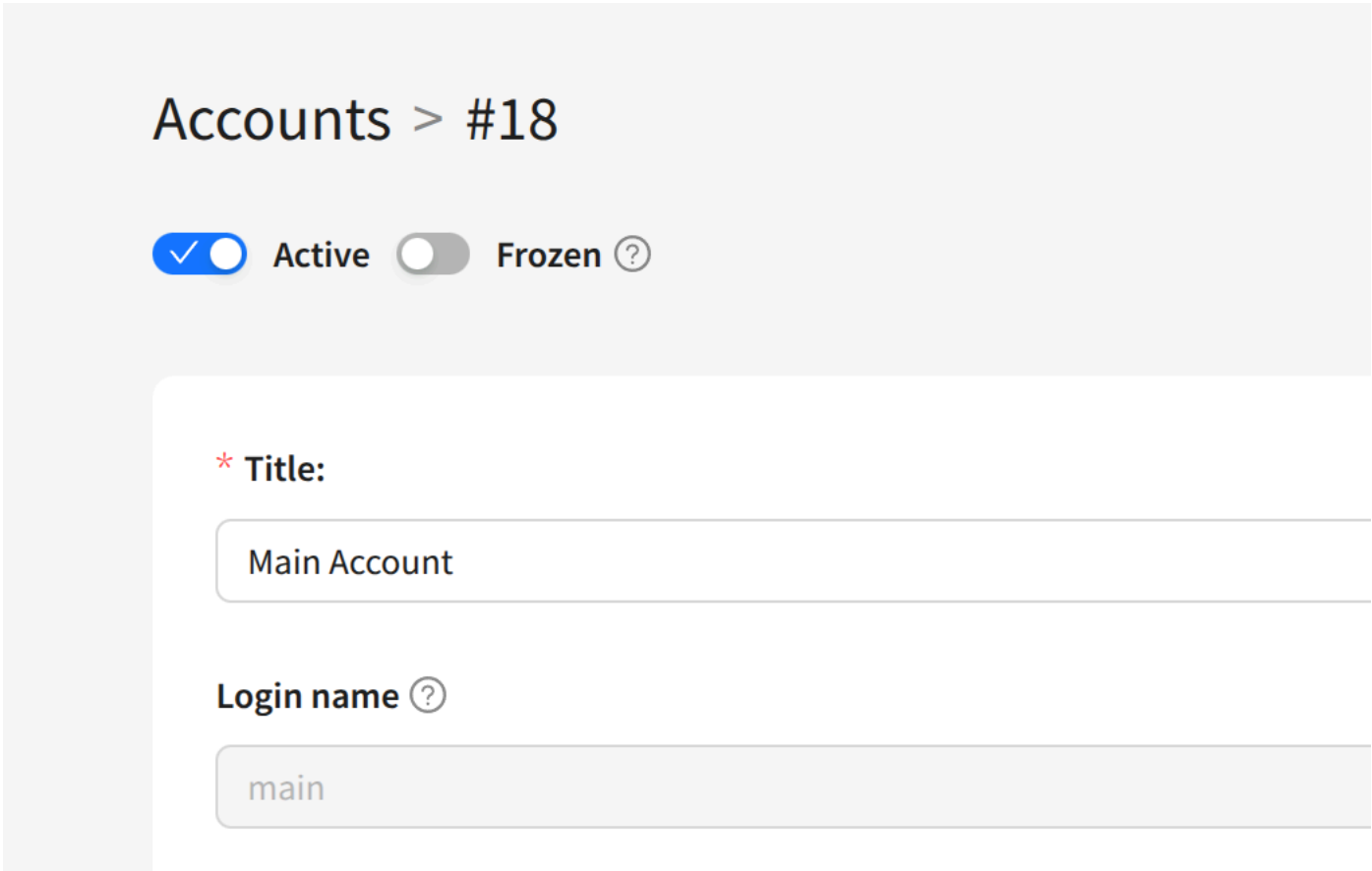
Database field indexes accelerate segmentation in Altcraft and profile search during bulk import by the fi

Creating, Editing, Deleting

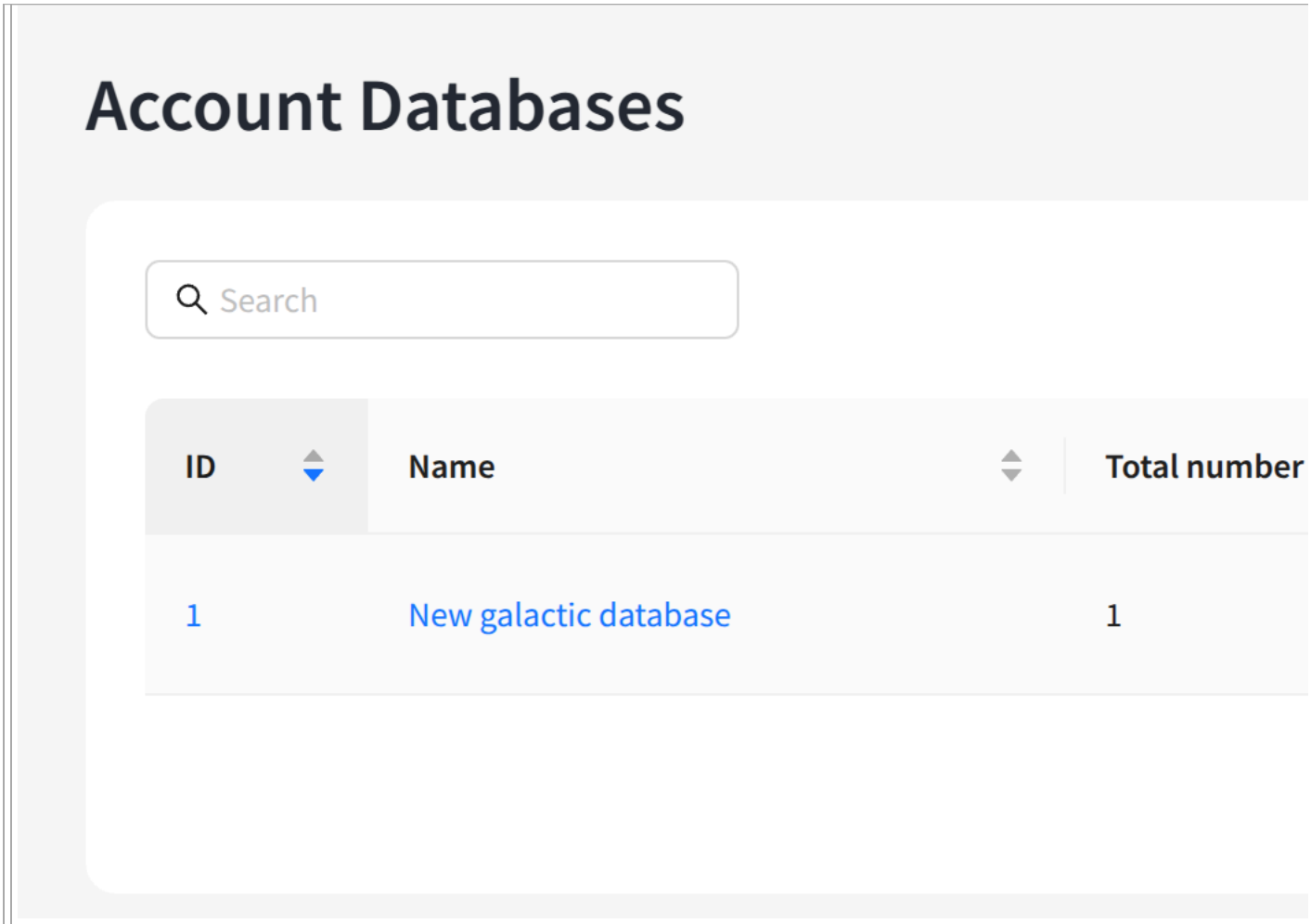
To manage indexes, in the list of accounts click the quick action button:



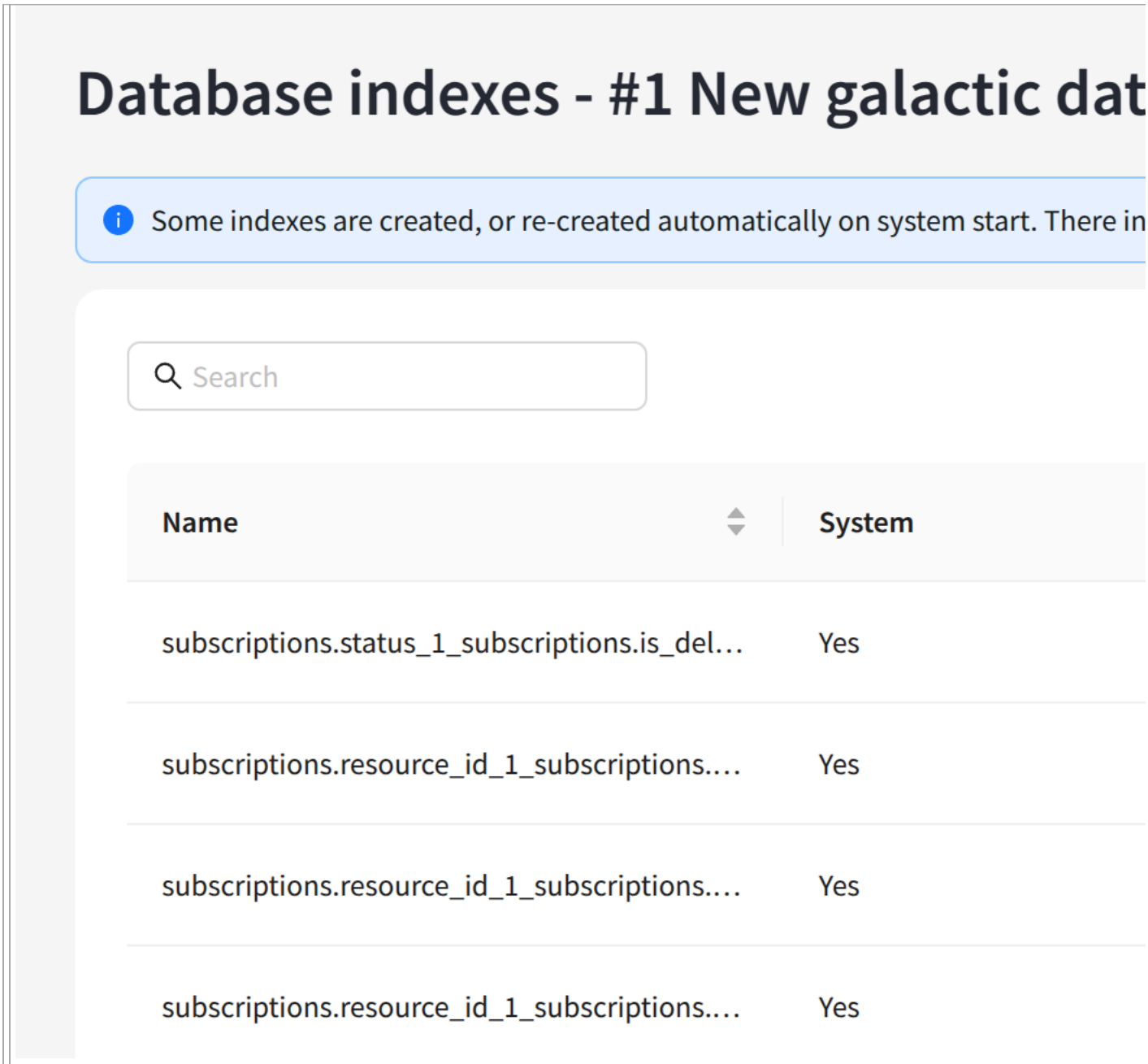
Or open the Account for editing and click the Databases button:



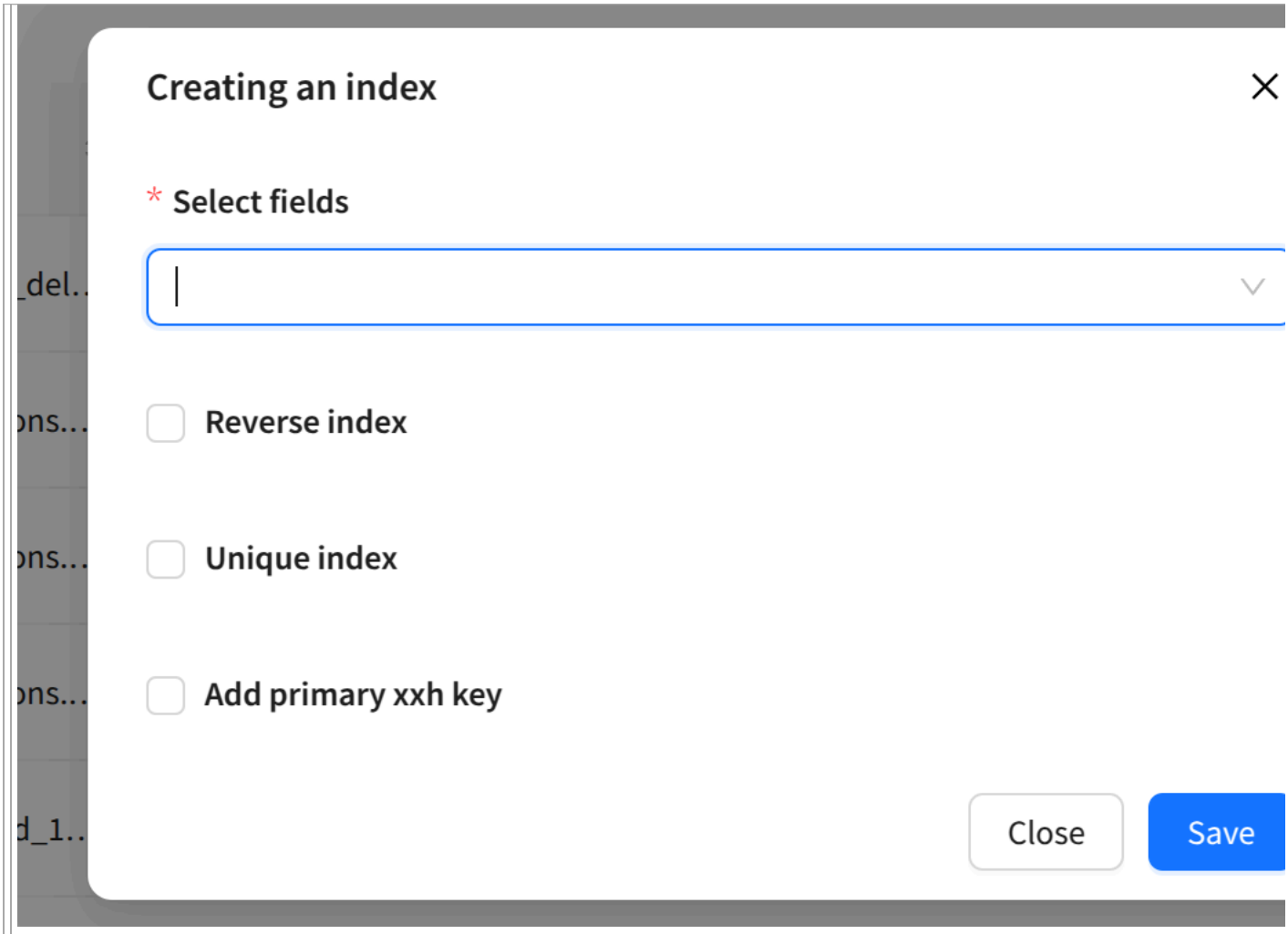
A list of all account databases containing customer profiles will open. For each database in the list, inforr



A list of all database field indexes will open. Most of them are system indexes and cannot be managed. Th



In the modal window that opens, select the required database fields and add indexes:



The following options are available for an index:

- **Reverse index — indexes for records are set in reverse order. Used when reverse search occurs more**
- **Unique index — guarantees that each record will have its own unique index. It will also be impossible**
- **Add primary xxh key — adds a history identifier to the index. Used when searches by this field are oft**

Databases

Description

Databases is a section in the Altcraft platform designed for managing connections. In this section, you co

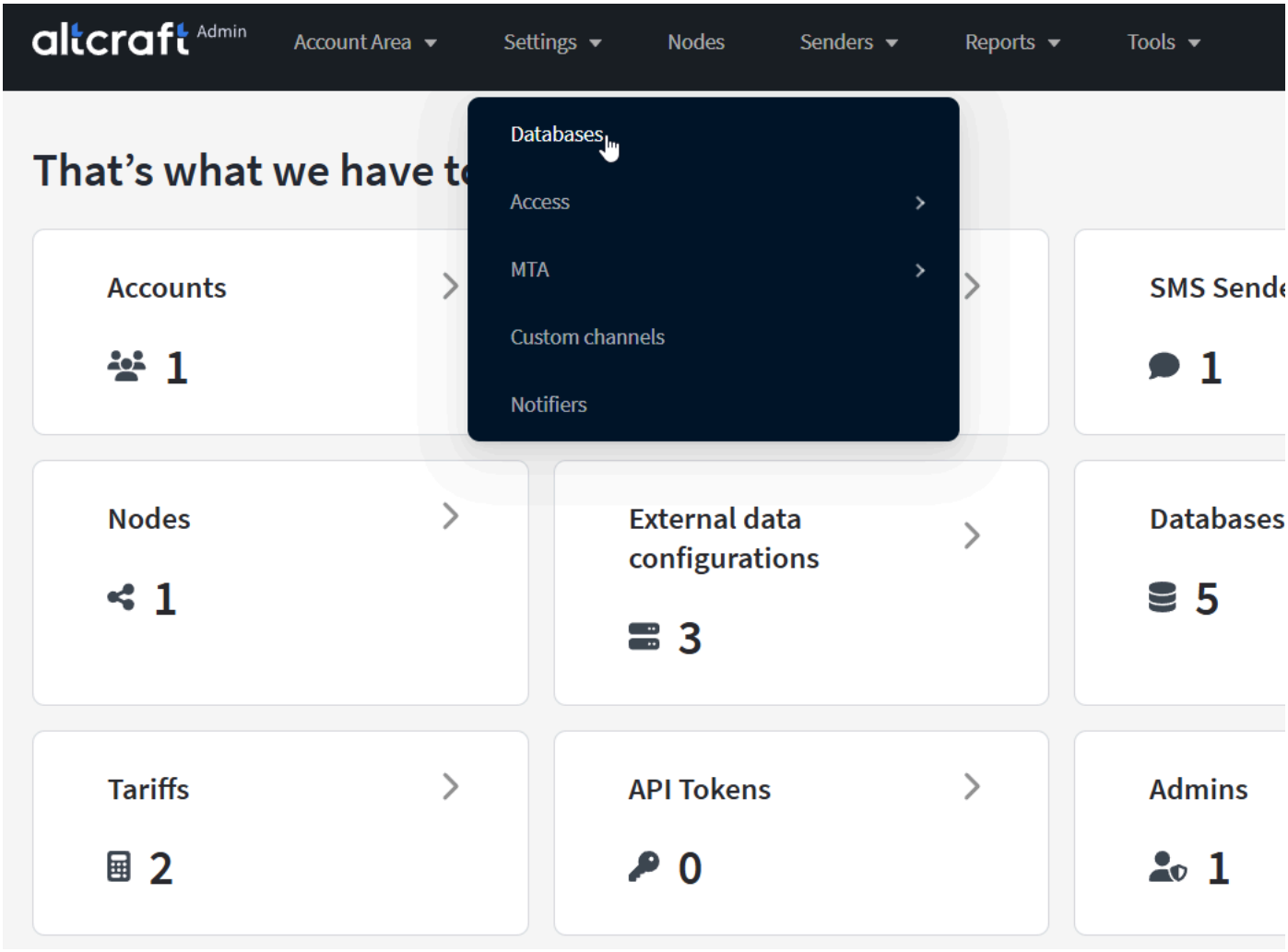
Database Types

For each account in the Altcraft marketing platform, you can configure connectors to the following databa

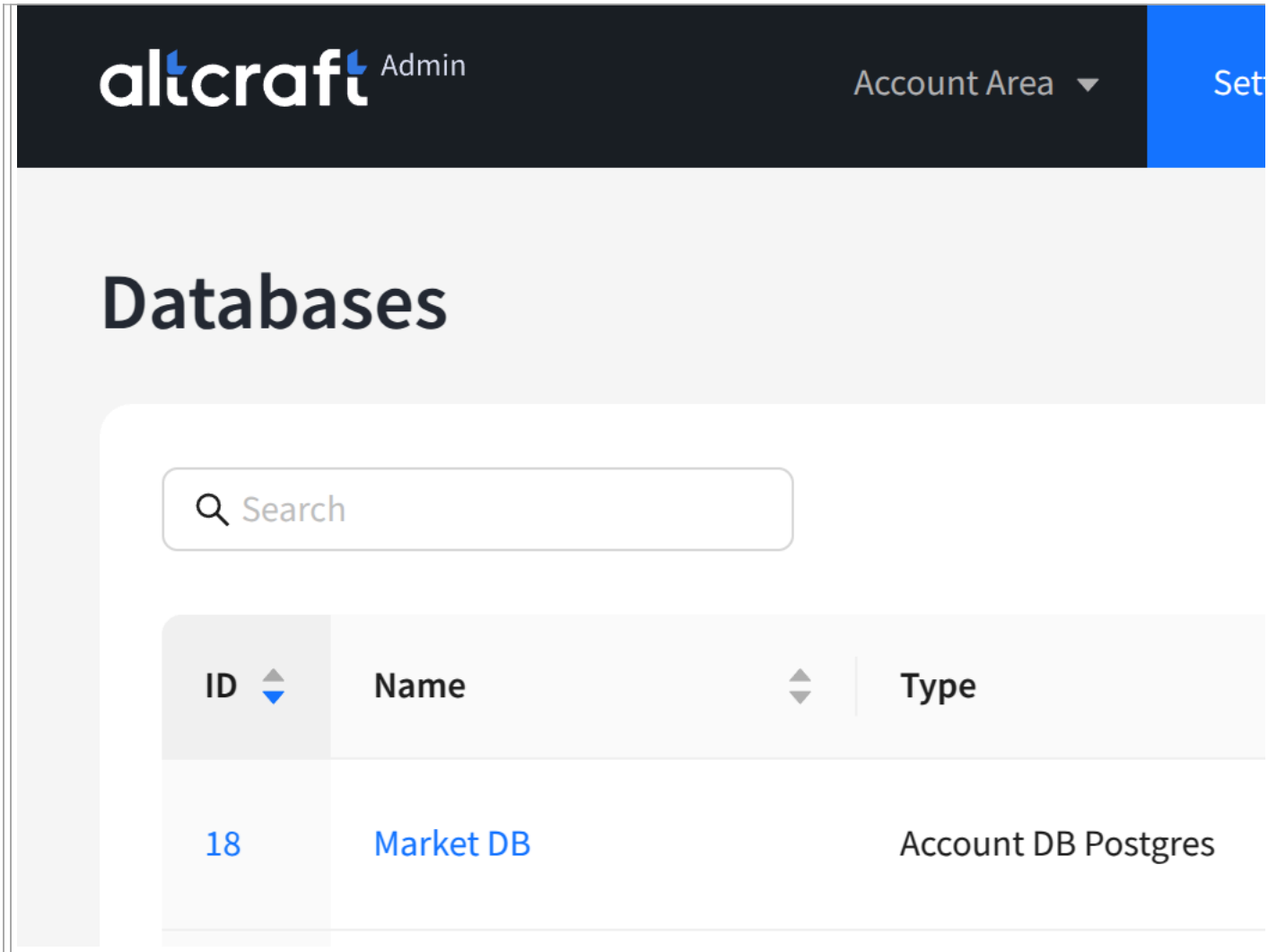
- **Account DB Postgres — used by the Market module for storing data about products and orders.**
- **Account DB ClickHouse — used to store data about profile relations and cache queries to external dat**

Creating, Editing, and Deleting

To manage database connectors, in the main menu select Settings — Databases.

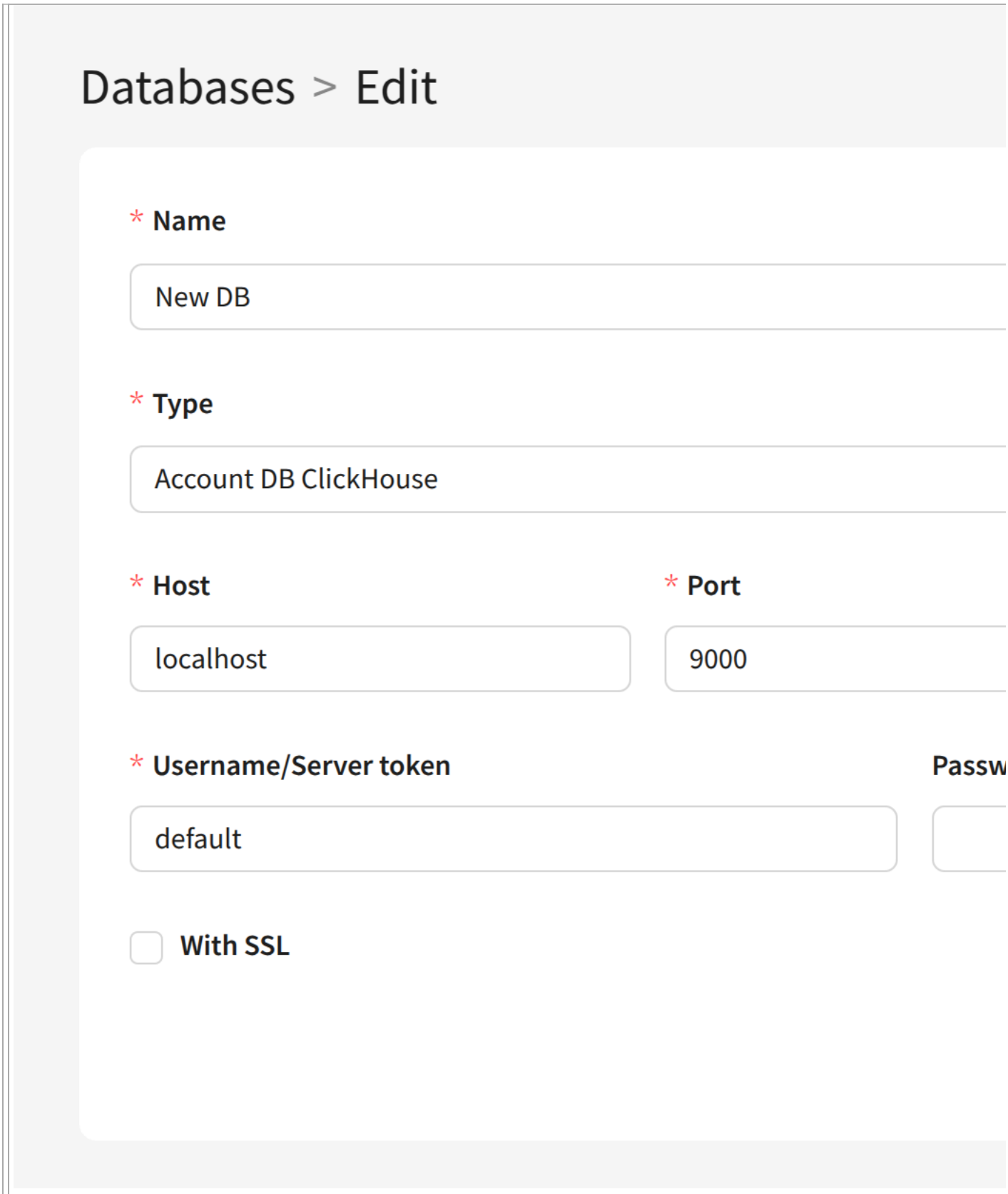


The opened list displays existing connectors. On the right side of the table are the edit and delete buttons



For each connector record, you need to enter a unique name and specify the following parameters:

- **Name — the connector name.**
- **Type — select the connector type depending on the target database.**
- **Host and Port — the network address and port for connecting to the system component.**
- **Username/Server token and Password/Secret — authentication data for connecting to the database. A**



For ClickHouse databases, you can enable SSL support. To do this, activate the Via SSL connection paran

With SSL

* SSL certificate

-----BEGIN CERTIFICATE-----

* SSL key

-----BEGIN RSA PRIVATE KEY-----

* SSL CA certificate

-----BEGIN CA CERTIFICATE-----

NOTE

When deploying platform components in a cluster to implement a high-availability deployment scheme, y

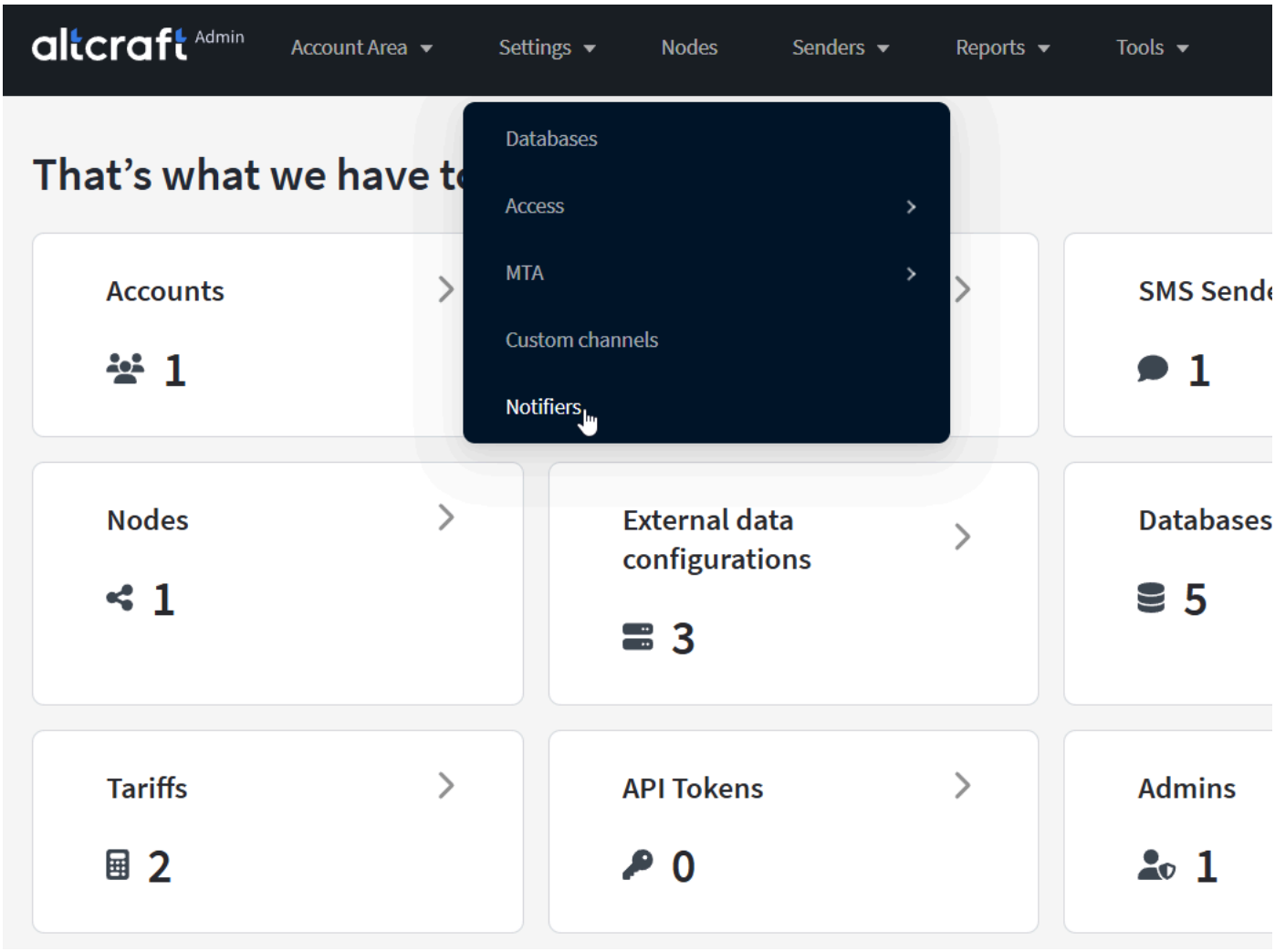
Notifiers

Description

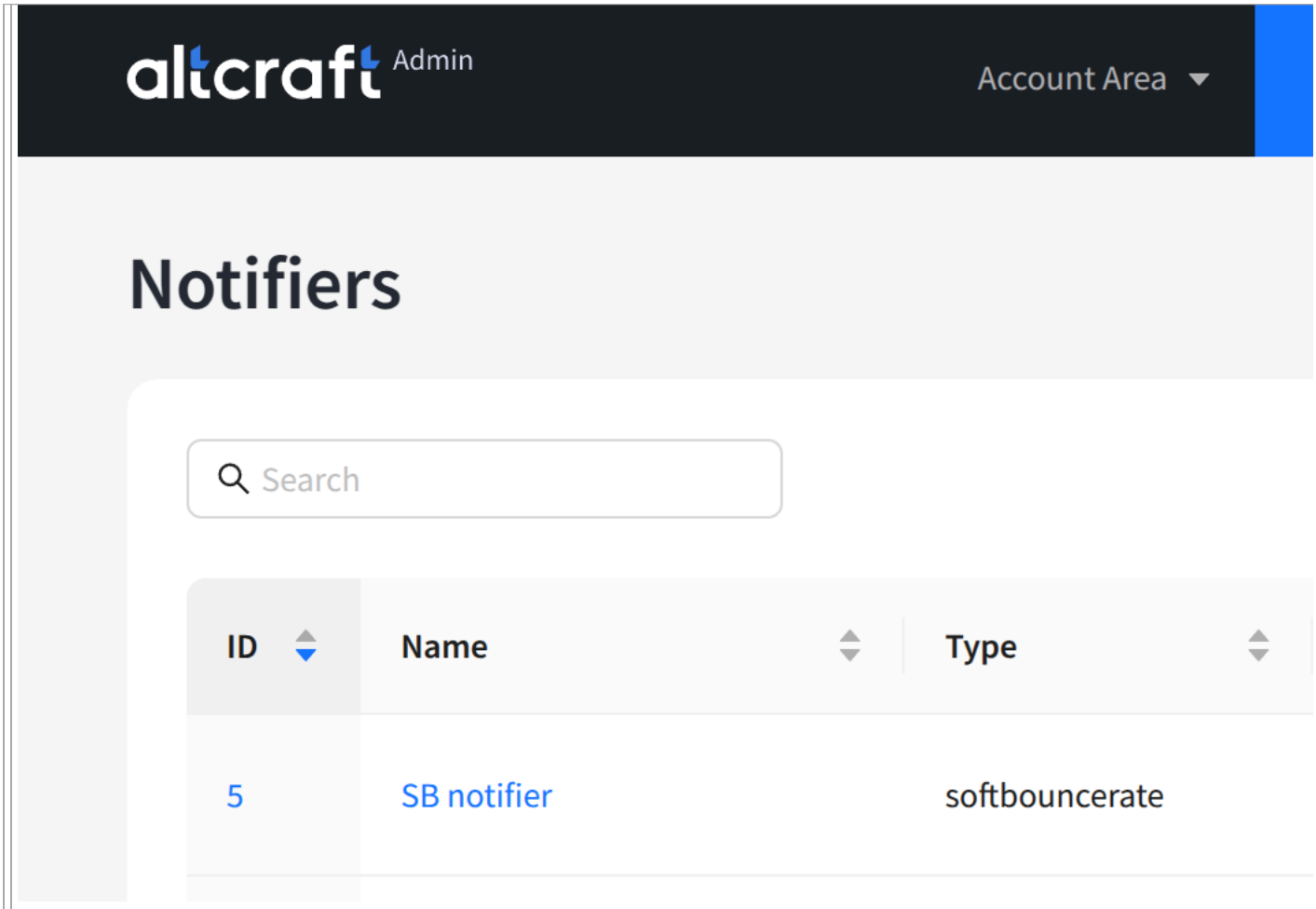
Notification Service is a mechanism in the Altcraft platform designed to automatically inform administrat
It allows you to promptly respond to failures and ensure the stable operation of the platform.

Creating, Editing, and Deleting

To manage notification settings, in the main menu select Settings — Notifiers.



The opened list displays existing notification settings. On the right side of the table are the edit and delet



To configure notifications, you need to specify the following parameters:

- **Name — the name of the notification settings**
- **Sender — the senders to be monitored. If you want to monitor all senders, select Capture all senders.**
- **Type — the error type for the notification. The following options are available:**
  - **When Hard bounce rate is higher than threshold — notification will be sent if the Hard bounce rat**
  - **When Soft bounce rate is higher than threshold — notification will be sent if the Soft bounce rate**
  - **When Complaints rate is higher than threshold — notification will be sent if the complaints rate ex**
  - **When Undelivery rate is higher than threshold — notification will be sent if the undelivery rate exc**
  - **When Performance rate is lower than threshold — notification will be sent if the performance rate**
- **Threshold — the threshold for the selected error metric**
- **Stats interval (minutes) — the statistics collection interval for metrics**
- **Minimum actions — the minimum number of registered events required to send a notification**
- **Send notification emails — the email addresses where the notification will be sent**
- **Freeze interval (minutes) — the time during which a repeat notification will not be sent**

Notifiers > Edit

Active

* **Name**

New notifier

Sender

Capture all senders

Type

When Undelivery rate is higher than threshold

* **Threshold**

0.01

* **Stats interval (minutes)**

120

Actions

* **Send notification emails**

* **Freeze inter**

120

Admins

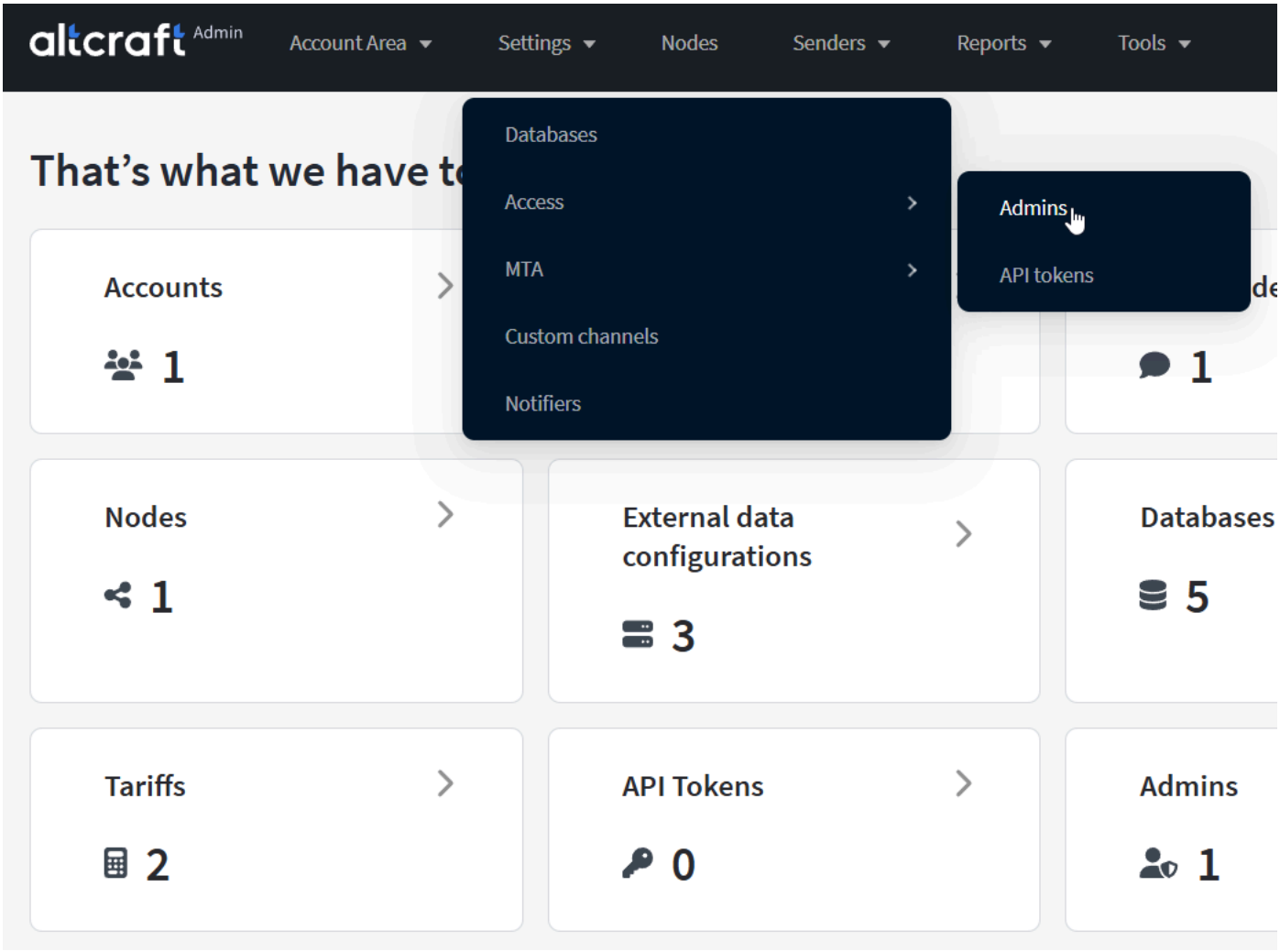
Description

Administrator in Altcraft is an account in the Admin Panel. Administrator records are created separately f
After installing the platform, an administrator account is created automatically:
Username: `admin`
Password: `admin`

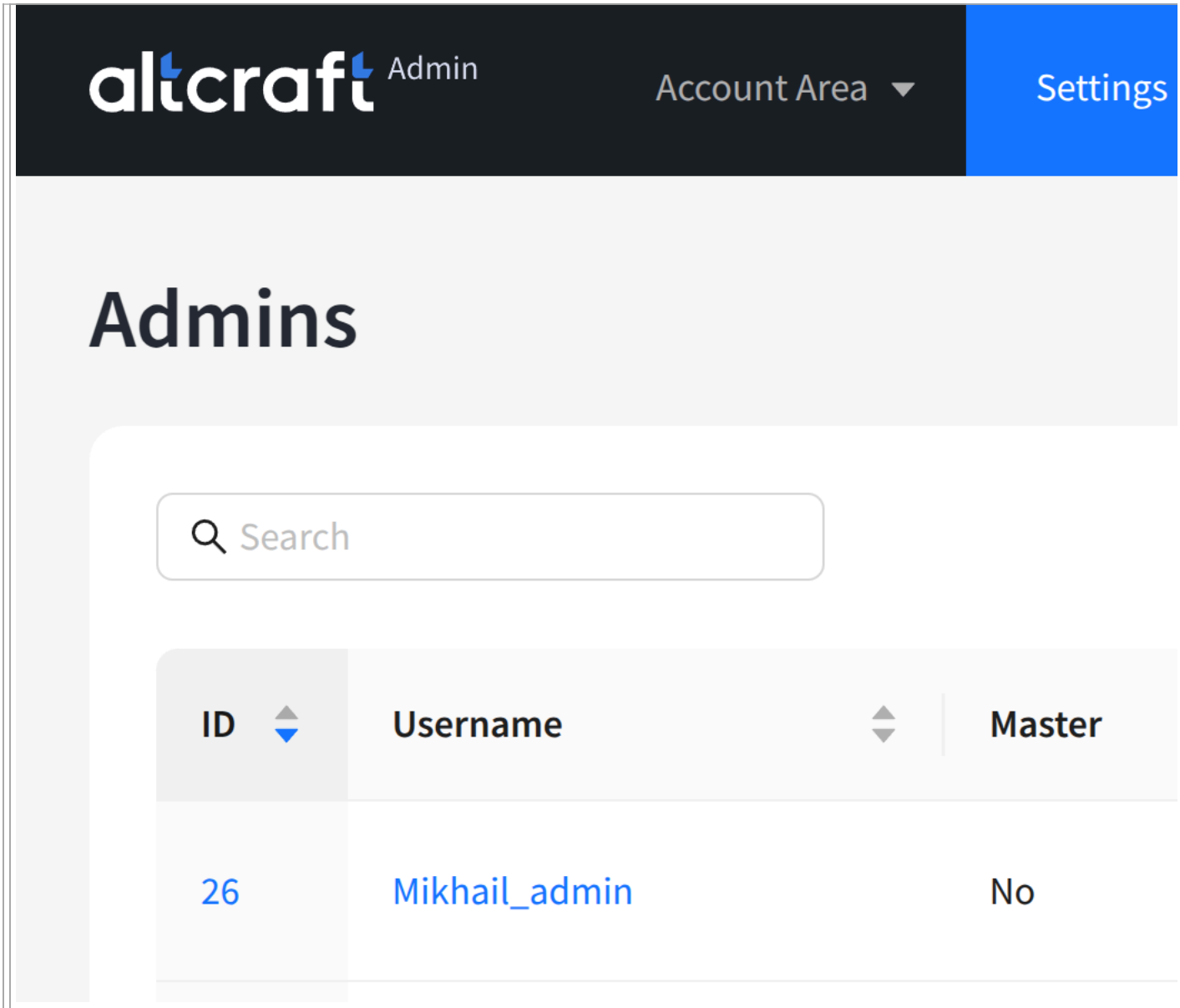
> DANGER
> **After the first login, you must change the password of the admin account.**

Creating, Editing, and Deleting

To manage administrators, in the main menu select Settings — Access — Admins.

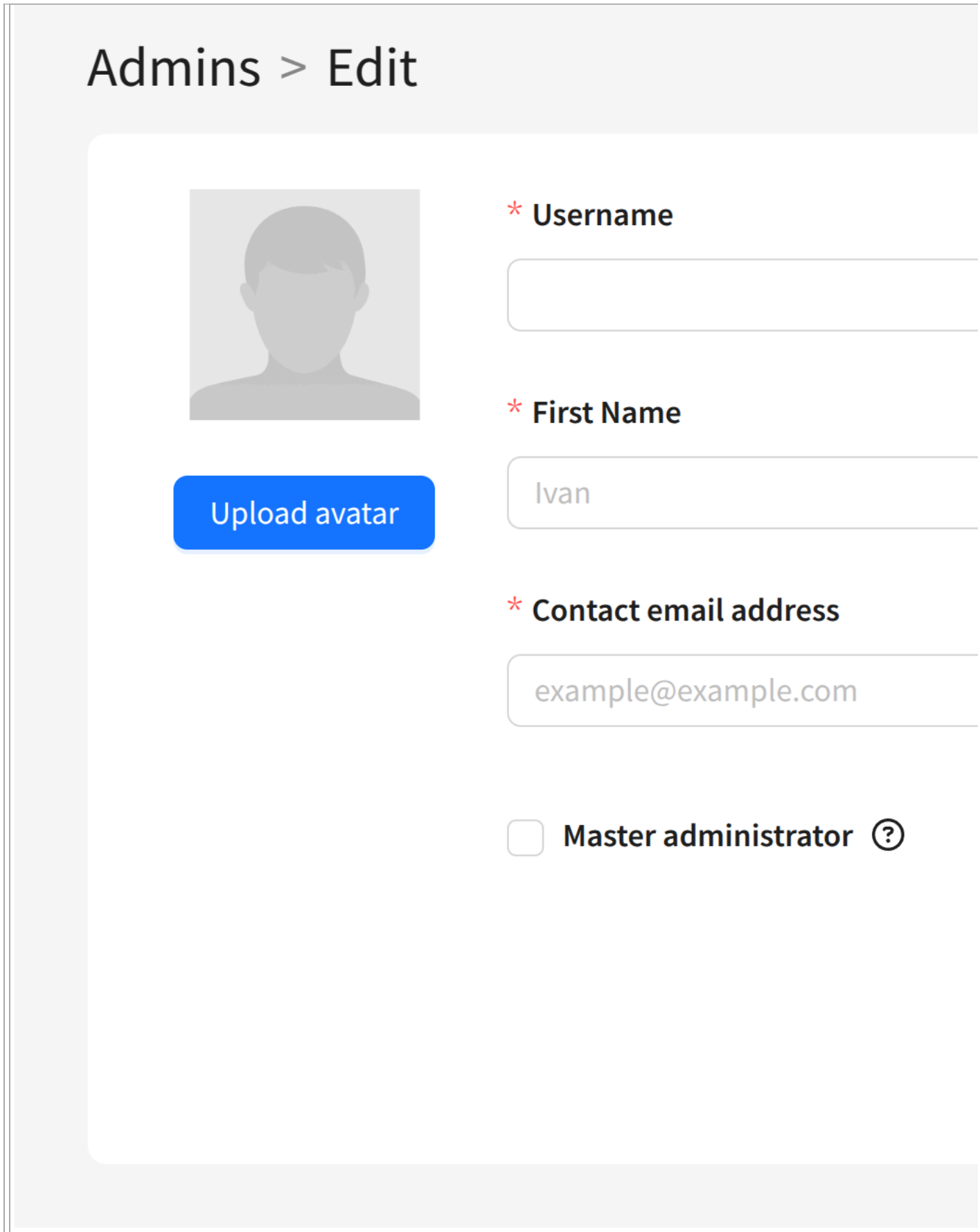


The opened list displays existing administrator accounts. On the right side of the table, there are edit and



The following parameters are configured for an administrator account:

- **Username and Password — username and password. These credentials are used to log in to the Admi**
- **First name and Last name — administrator's first and last name.**
- **Contact email address — administrator's contact email.**
- **Language — Admin Panel interface language.**



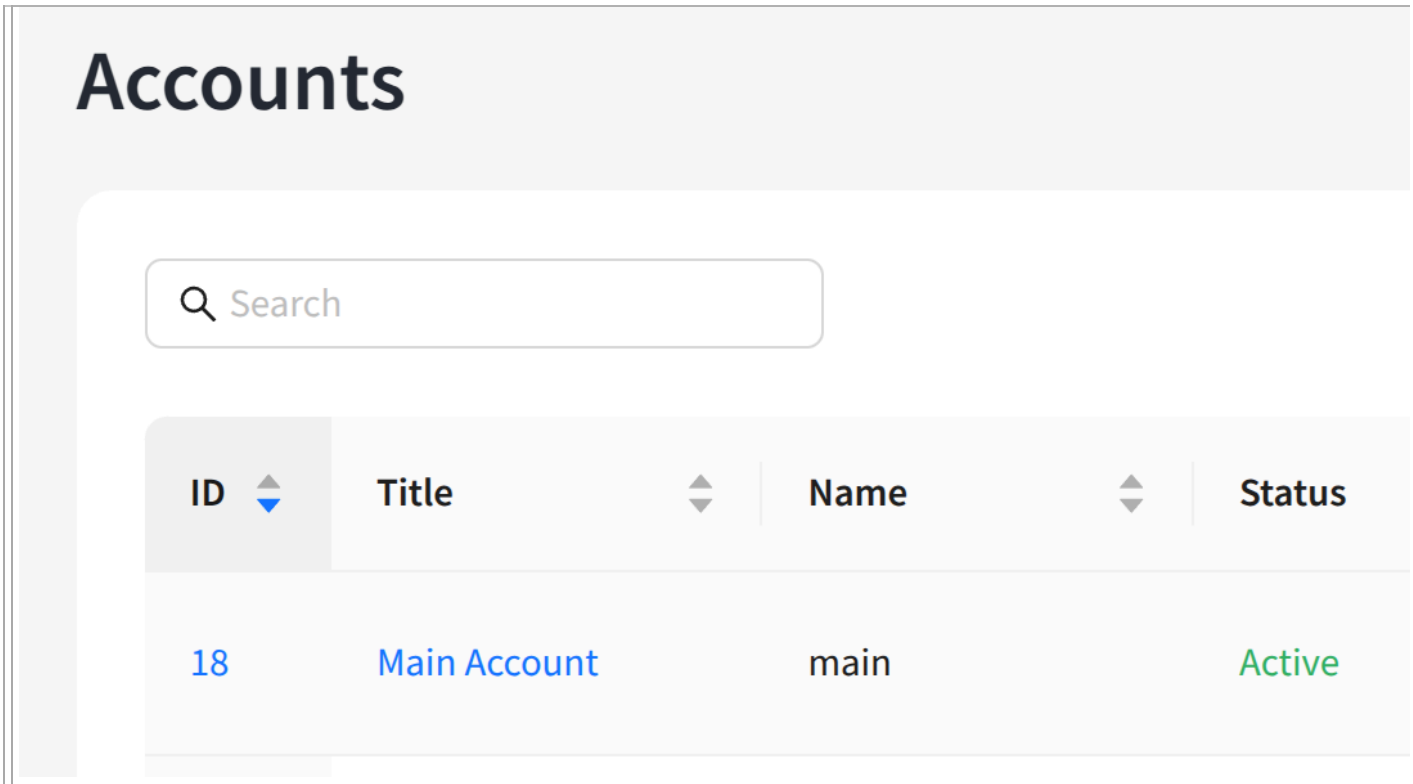
Select the Master administrator option if you want to allow the administrator to manage other accounts ir

NOTE

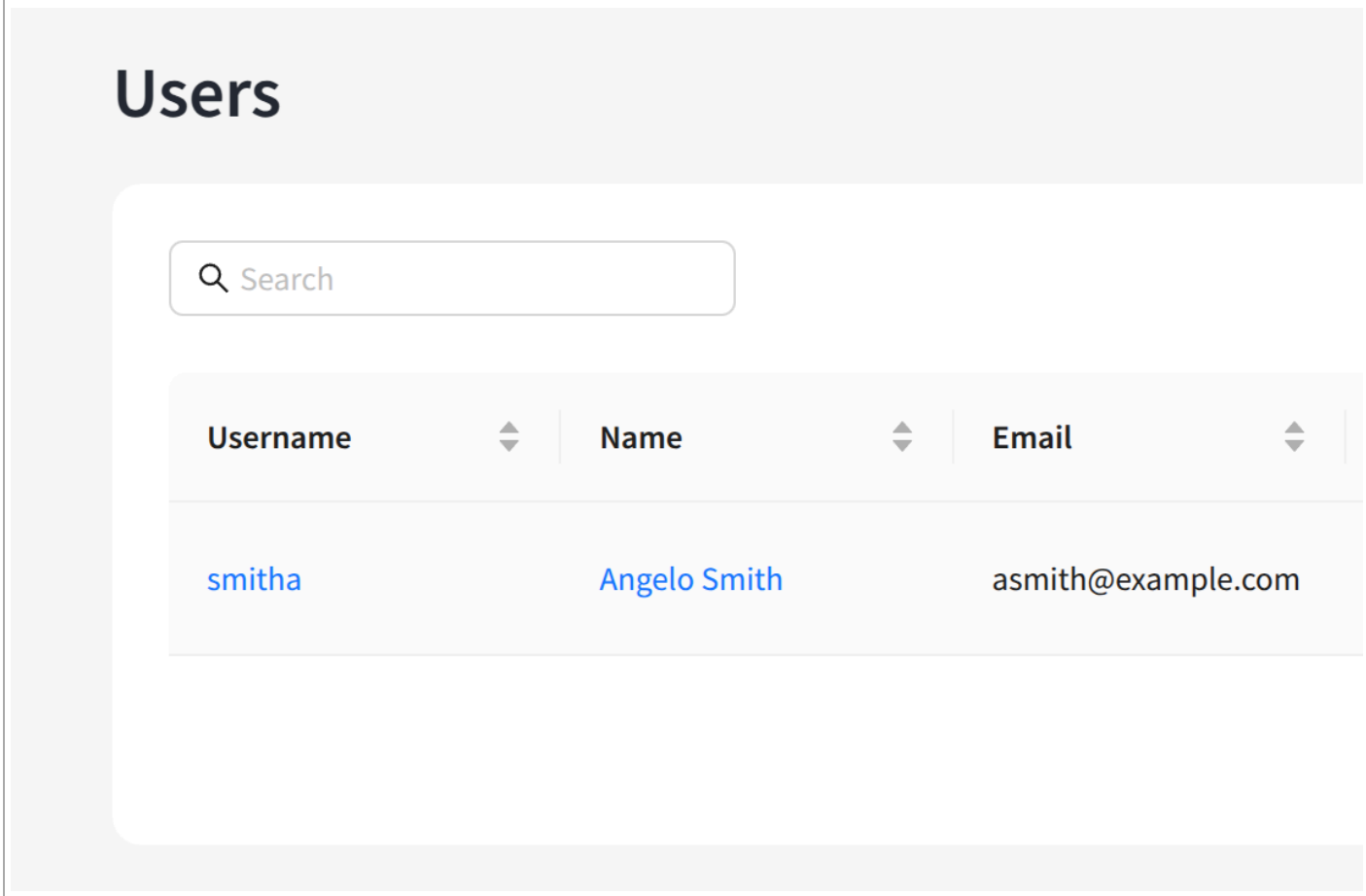
LDAP authentication is also available for administrators. More details in this article.

Gaining access to platform accounts from the Admin Panel

An Altcraft administrator can access any user account of any account created on the platform. To do this



Click the rocket button to access the selected Altcraft user account:



NOTE

Your web browser must allow pop-ups.

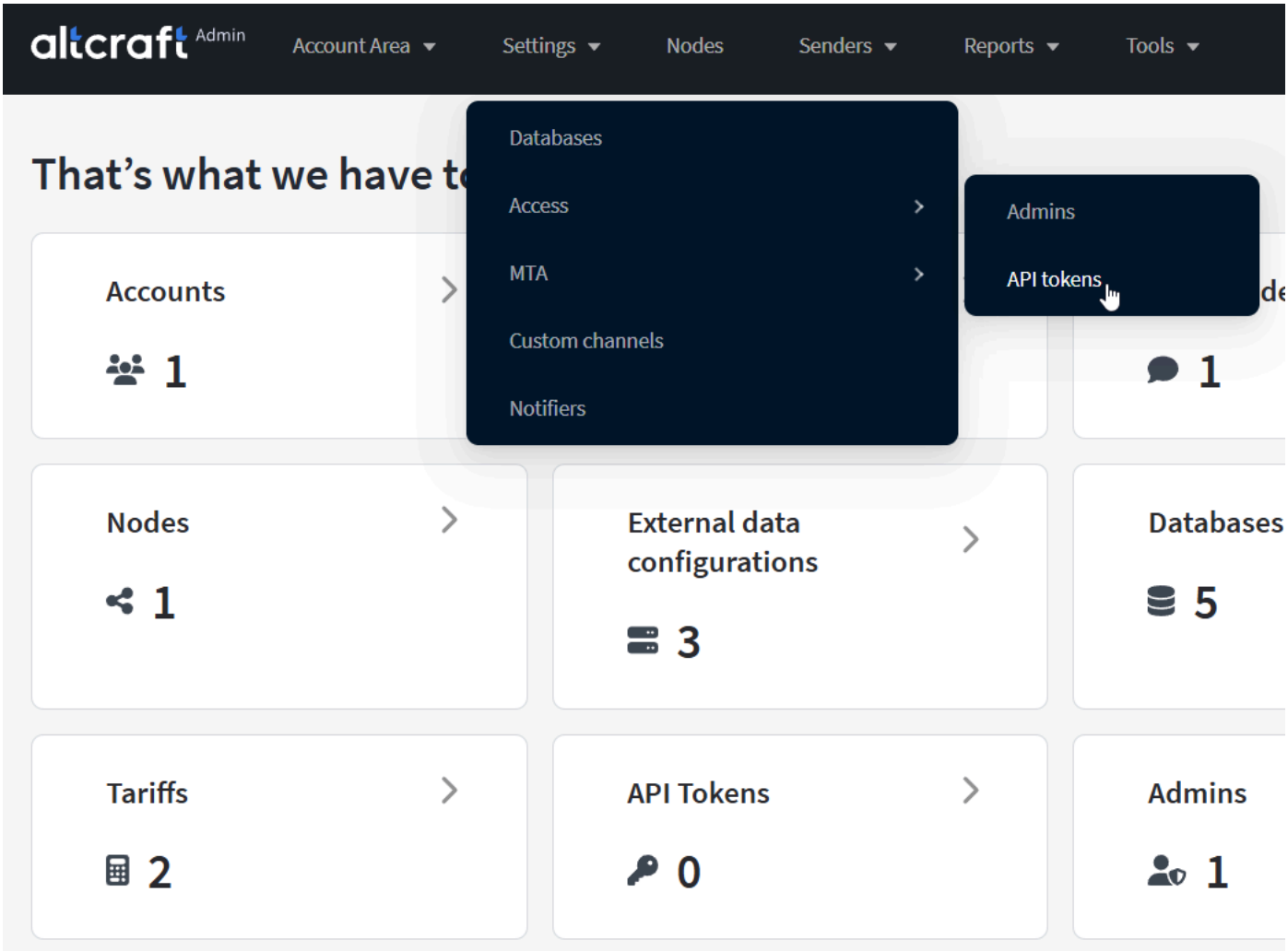
API tokens

Description

API tokens are tools used for API requests, mainly related to managing senders (message delivery agents

Creating, Editing, and Deleting

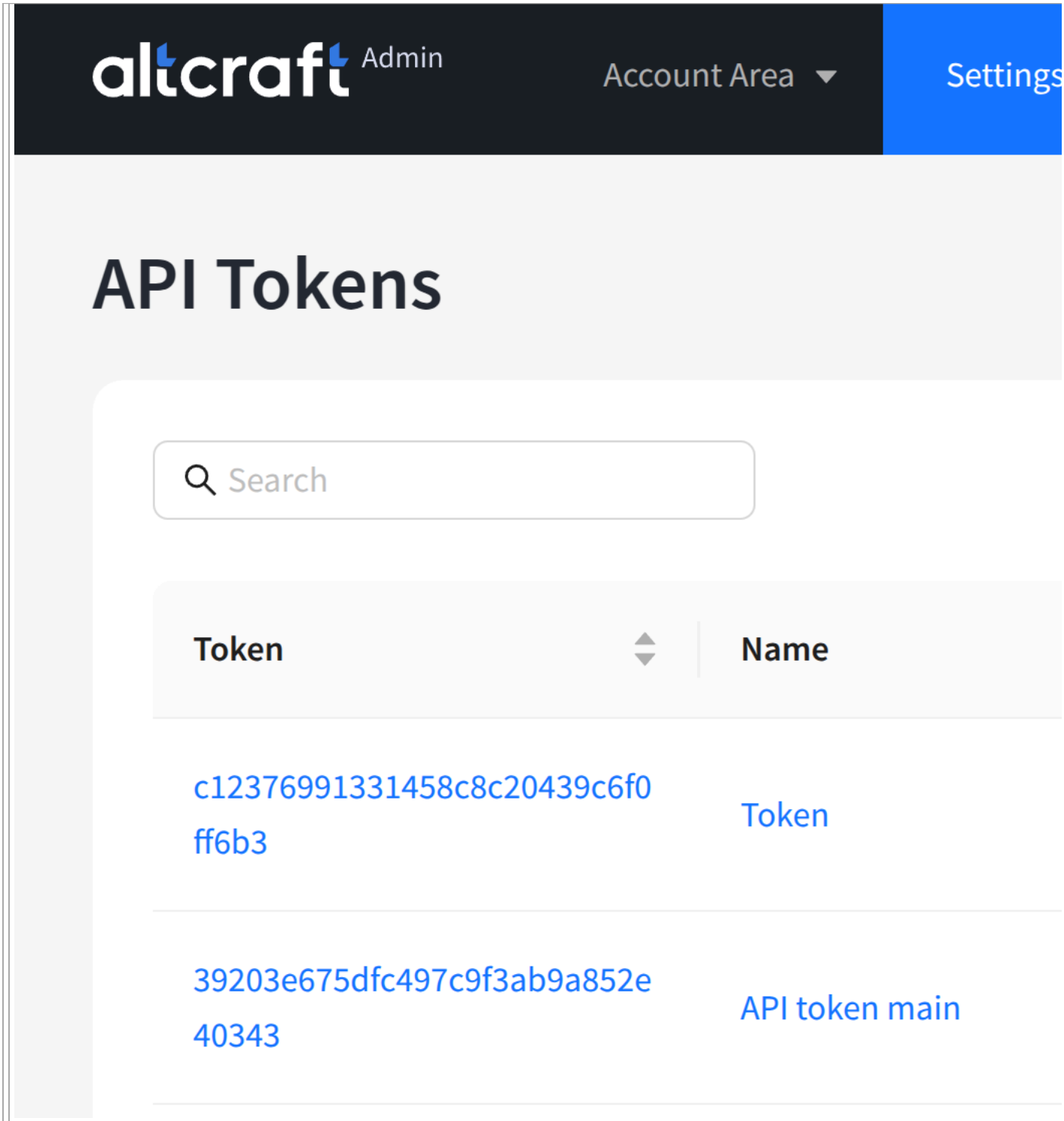
To manage API tokens, in the main menu select Settings — Access — API Tokens.



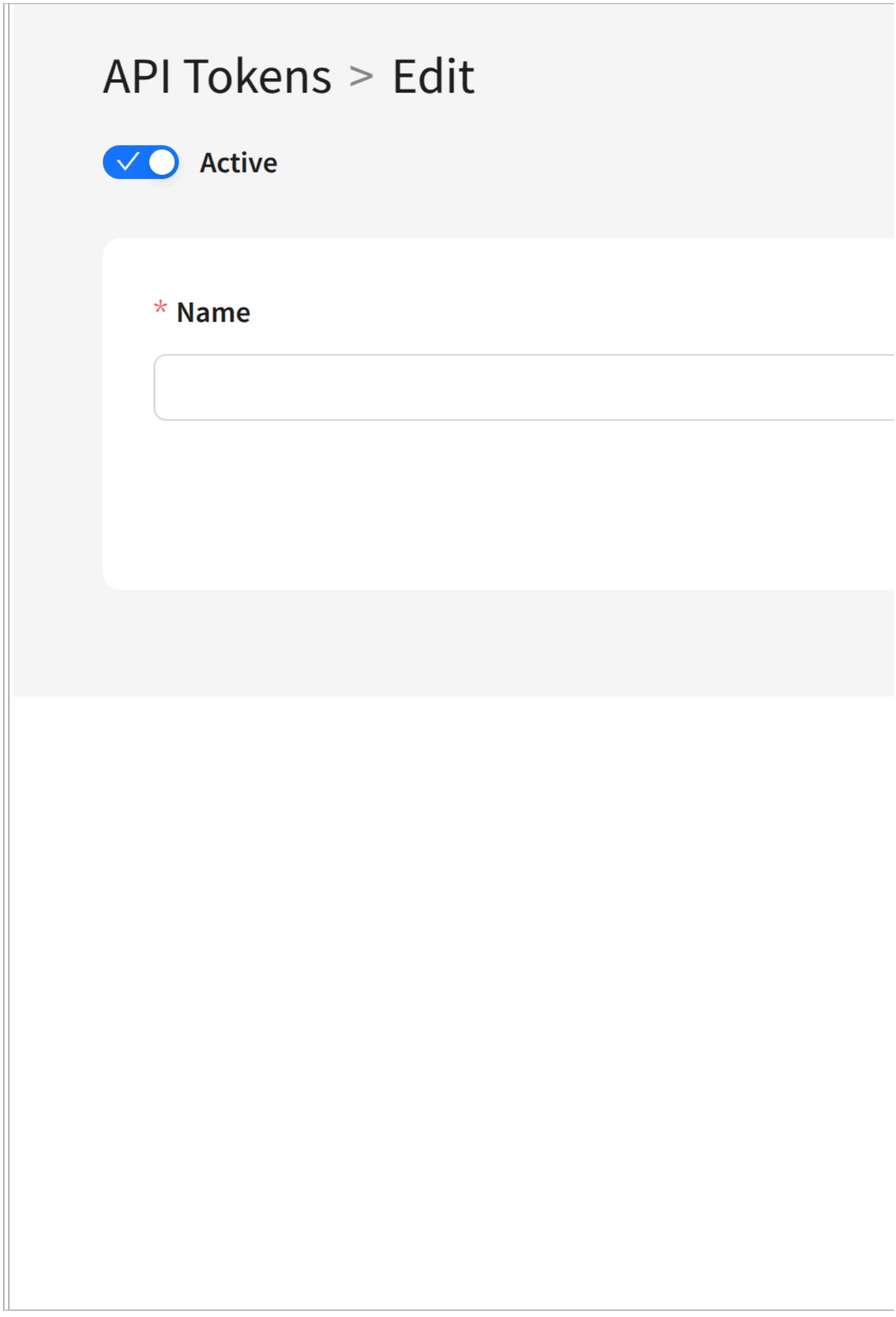
The opened list displays existing tokens. On the right side of the table, there are edit and delete buttons.

NOTE

The secret part of the token used in API requests is displayed in the "Token" column of the table.



You can set a unique name for the token and define whether the token is active or not. Using multiple adm


API Tokens > Edit
Active
* Name

MTA

Description

AKMTA (Altcraft Mail Transfer Agent) is an email sender developed by the Altcraft team and included in th maximize deliverability to subscribers.

Sending Rules Configuration

AKMTA supports the configuration of the following sending rules:

- **ISPs — delivery through different email providers**
- **Strategies — sending strategies (speed)**
- **Retry rules — rules for retrying message delivery**
- **Lock rules — rules for blocking message delivery**
- **Bounce patterns — templates of ISP server responses**
- **Default rules — delivery scenarios**

Additionally, to work with MTA you configure:

- **Private key — DKIM or SSH key required to access the sending server.**
- **Pool — a set of resources needed to send emails from a specific domain. The pool allows you to simul**

For more details on configuring each element, see the corresponding articles:

Caching DNS Server

Default Resolver Problem

Sending a large volume of messages can overload network resources, leading to delays and potential bott

When the system initiates a DNS query to convert a domain name into an IP address, it usually relies on tl often requires frequent DNS lookups to resolve recipient domain names. Default resolvers can inefficientl service disruptions.

By storing previously resolved domain-to-IP mappings, DNS caching servers reduce the need for redundan DNSSEC signatures and protecting against certain types of cyber threats, such as DNS spoofing.

Unbound Setup

Installation

Unbound must be installed on the sender servers:

```
sudo apt-get update
sudo apt-get install unbound
```

Configuration

The configuration file can have any name and must be located in the *`/etc/unbound/unbound.conf.d/`* directo

Example configuration:

```
server:
    # Send minimum amount of information to upstream servers to enhance
    # privacy. Only sends minimum required labels of the QNAME and sets
    # QTYPE to NS when possible.
    # See RFC 7816 "DNS Query Name Minimisation to Improve Privacy" for
    # details.
    qname-minimisation: yes

    # The following line will configure unbound to perform cryptographic
    # DNSSEC validation using the root trust anchor.
    auto-trust-anchor-file: "/var/lib/unbound/root.key"

# Remote control config section.
remote-control:
    # Enable remote control with unbound-control(8) here.
    # set up the keys and certificates with unbound-control-setup.
    control-enable: yes

forward-zone:
    name: "."
    forward-addr: 1.1.1.1
    forward-addr: 1.0.0.1
```

After creating the configuration, check it, restart Unbound, and enable it on boot:

```
unbound-checkconf
sudo systemctl start unbound.service
sudo systemctl enable unbound.service
```

Detailed information about the Unbound server can be found in the official documentation.

AKMTA Log Components

AKMTA logs are text records that capture all events related to email delivery. These records are maintain responses from email providers, and promptly adjust settings to improve deliverability.

Each entry in the AKMTA logs contains structured information about the event, which helps quickly under

AKMTA logs are written to syslog in the following format:

```
2024/10/28 12:16:31 '1|2|Gmail|18029cdf9235cc8d','10','1','Gmail','231.127.65.9','noreply@templating.net','templating.net','zxhcgvxhjx32423@gma
```

- **Event date and time:** `'2024/10/28 12:16:31'`
- **Local storage identifier:** `'1|2|Gmail|18029cdf9235cc8d'`
- **Template ID:** `'10'`
- **Sender (MTA) ID:** `'1'`
- **ISP name or domain:** `'Gmail'`
- **Sender IP address:** `'231.127.65.9'`
- **Sender email address:** `'noreply@templating.net'`
- **Sender domain:** `'templating.net'`
- **Recipient email address:** `'zxhcgvxhjx32423@gmail.com'`
- **MX connection IP:** `'116.69.16.162'`
- **MX server:** `'gmail-smtp-in.l.google.com.'`
- **Server response:** `'500 dial tcp 231.127.65.9:0->116.69.16.162:25: i/o timeout'`
- **Status:** `'undefined'`
- **Command sent before response:** `'DIAL'`
- **Account ID:** `'3'`
- **Mailing ID:** `'68'`
- **Database ID:** `'7'`
- **Resource ID:** `'1'`
- **Full message ID:** `'abcdefghIJklmnoP-123QrstuvwXYz'`
- **Email-safe message ID:** `'asikjbyrinnvjfikdig123-dfgretyhvvbvasdaa'`
- **Additional error message if needed:** `'<nil>'`

> **TIP**
>
> **For convenient and structured work with AKMTA logs, you can enable output in JSON format.**
> **To do this, add the `JSON_SYSLOG` parameter to the main platform configuration file and restart the platform.**

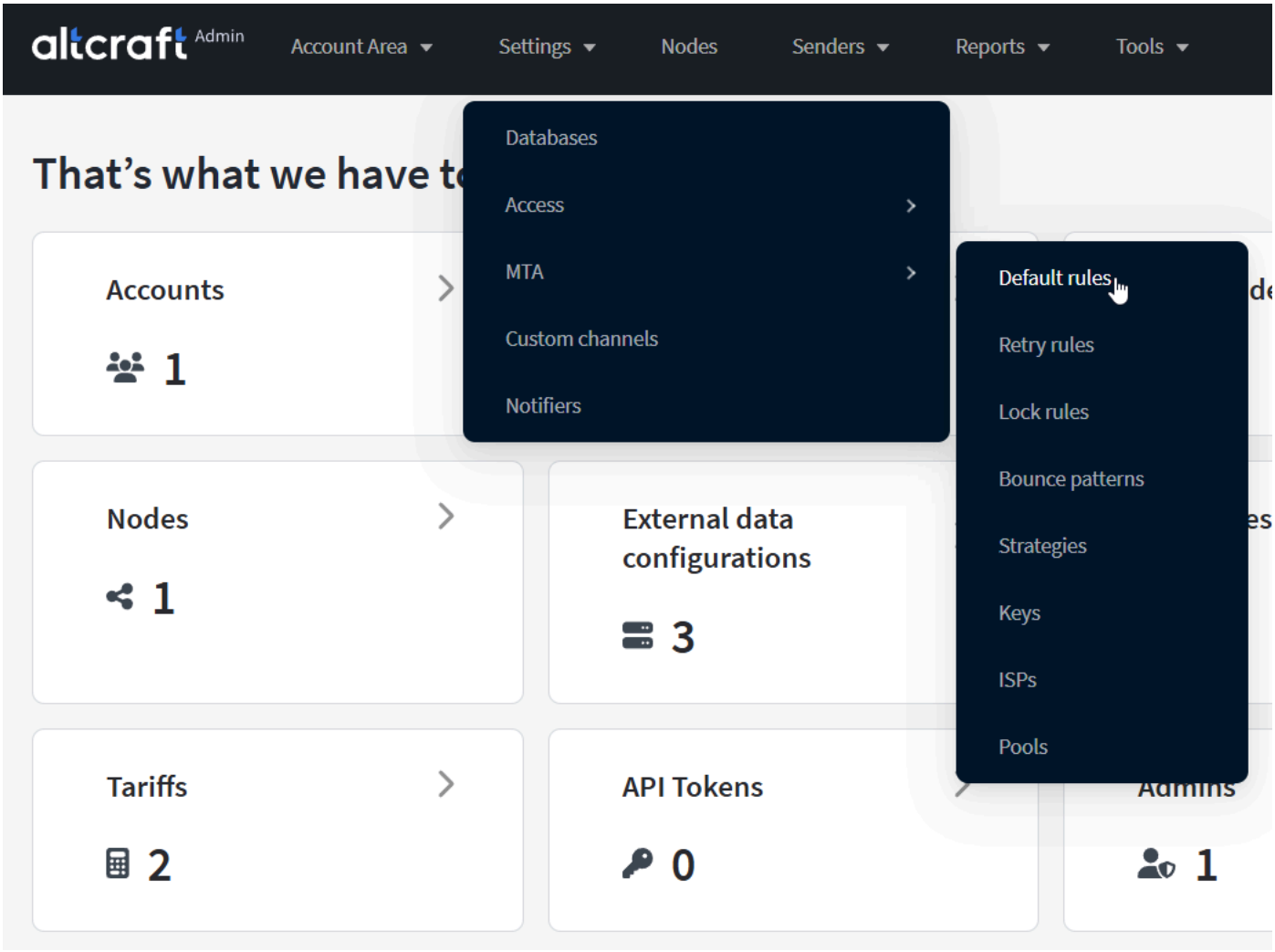
Default rules

Description

Default rules are configuration records that define AKMTA sending scenarios when receiving negative res

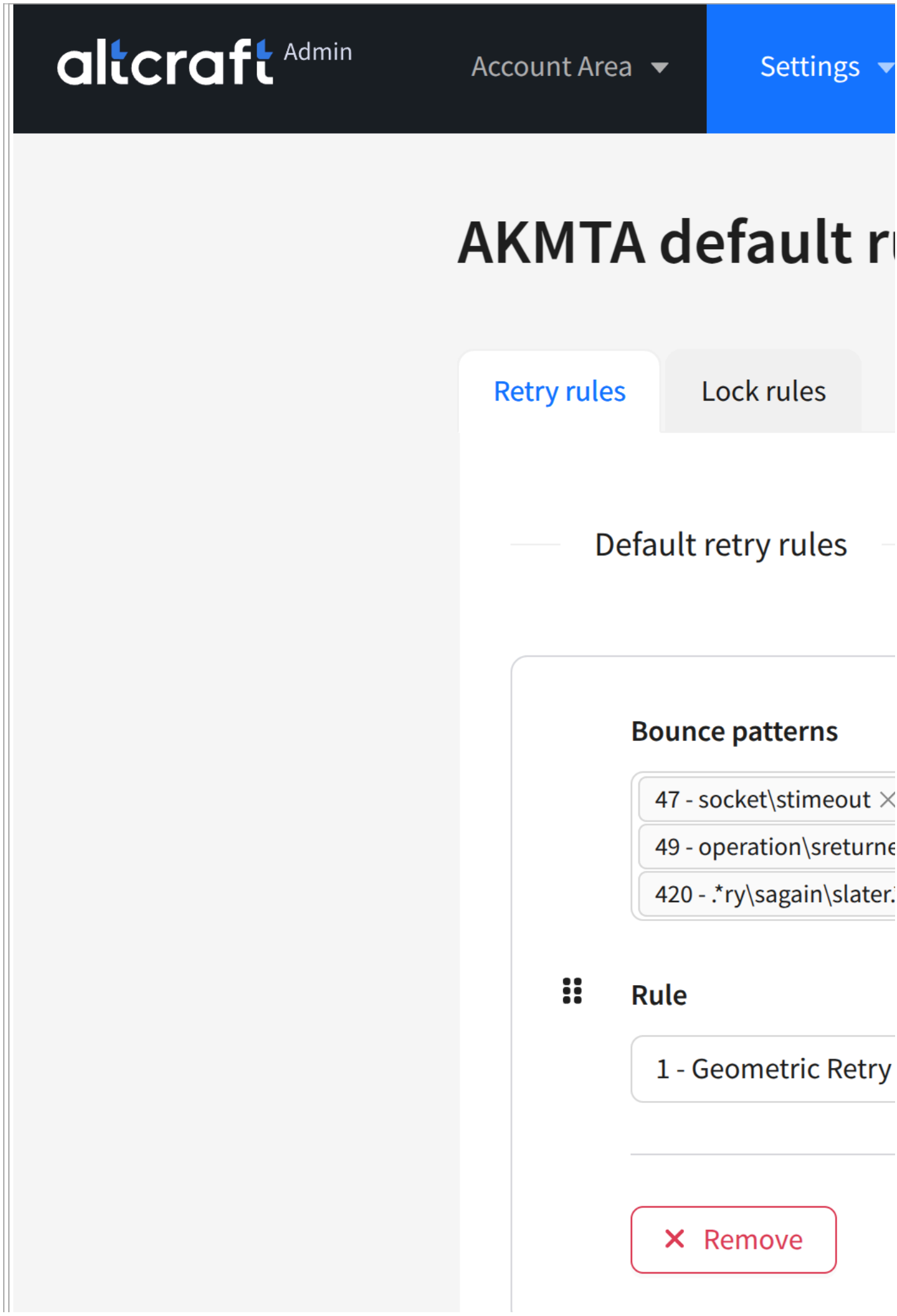
Connection settings

To manage sending scenarios, in the main menu select Settings — MTA — Default rules.

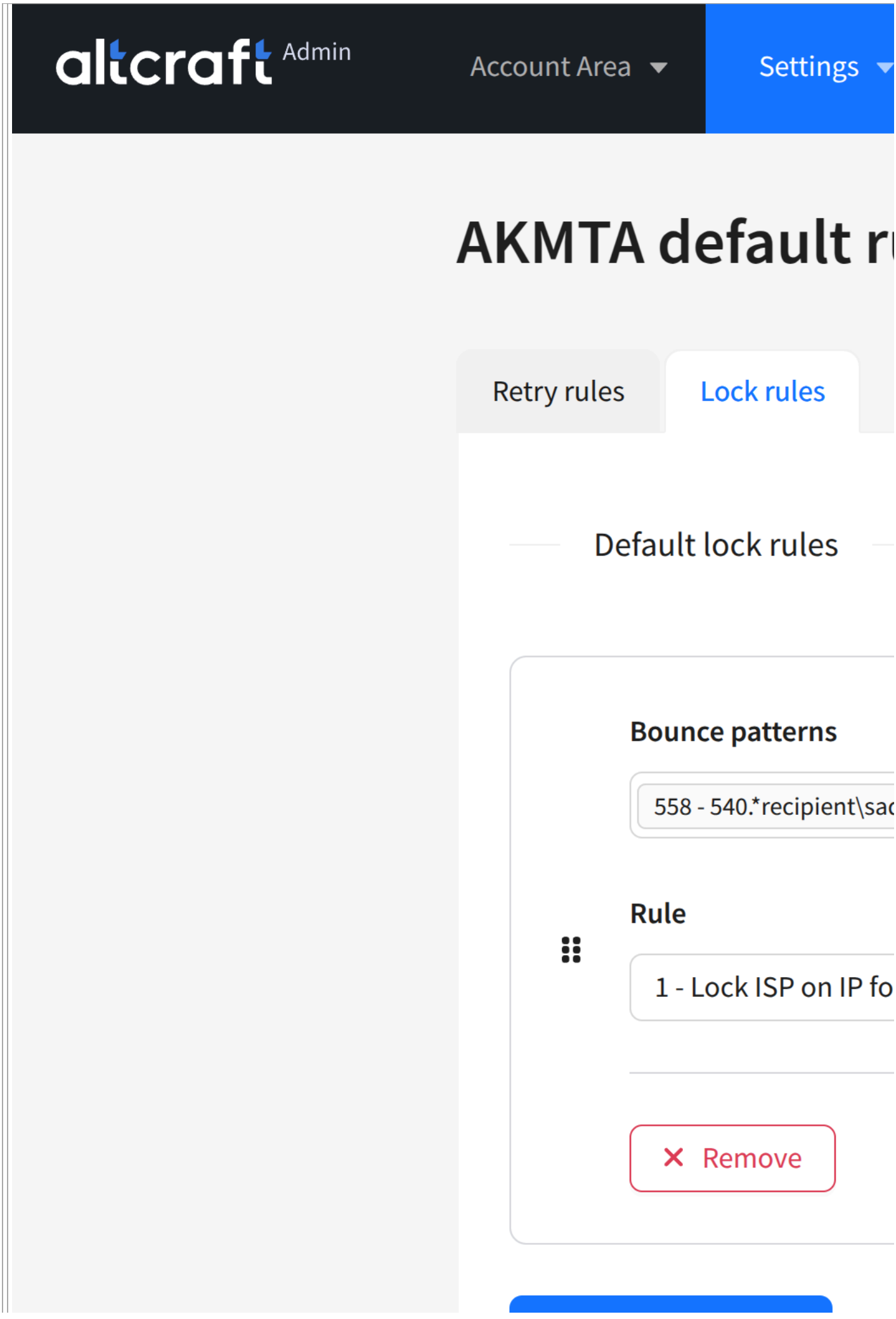


The Default rules page contains two tabs:

- **Retry rules — scenarios that use retry rules.**


altcraft Admin
Account Area
Settings
AKMTA default r
Retry rules
Lock rules
Default retry rules
Bounce patterns
47 - socket\stimeout
49 - operation\sreturne
420 - .*ry\sagain\slater.
Rule
1 - Geometric Retry
Remove

- **Lock rules — scenarios that use sending lock rules.**


altcraft Admin
Account Area
Settings
AKMTA default r
Retry rules
Lock rules
Default lock rules
Bounce patterns
558 - 540.*recipient\sac
Rule
1 - Lock ISP on IP fo
Remove

Add bounce patterns

Most MTA behavior scenarios for negative ISP server responses are already configured by Altcraft special

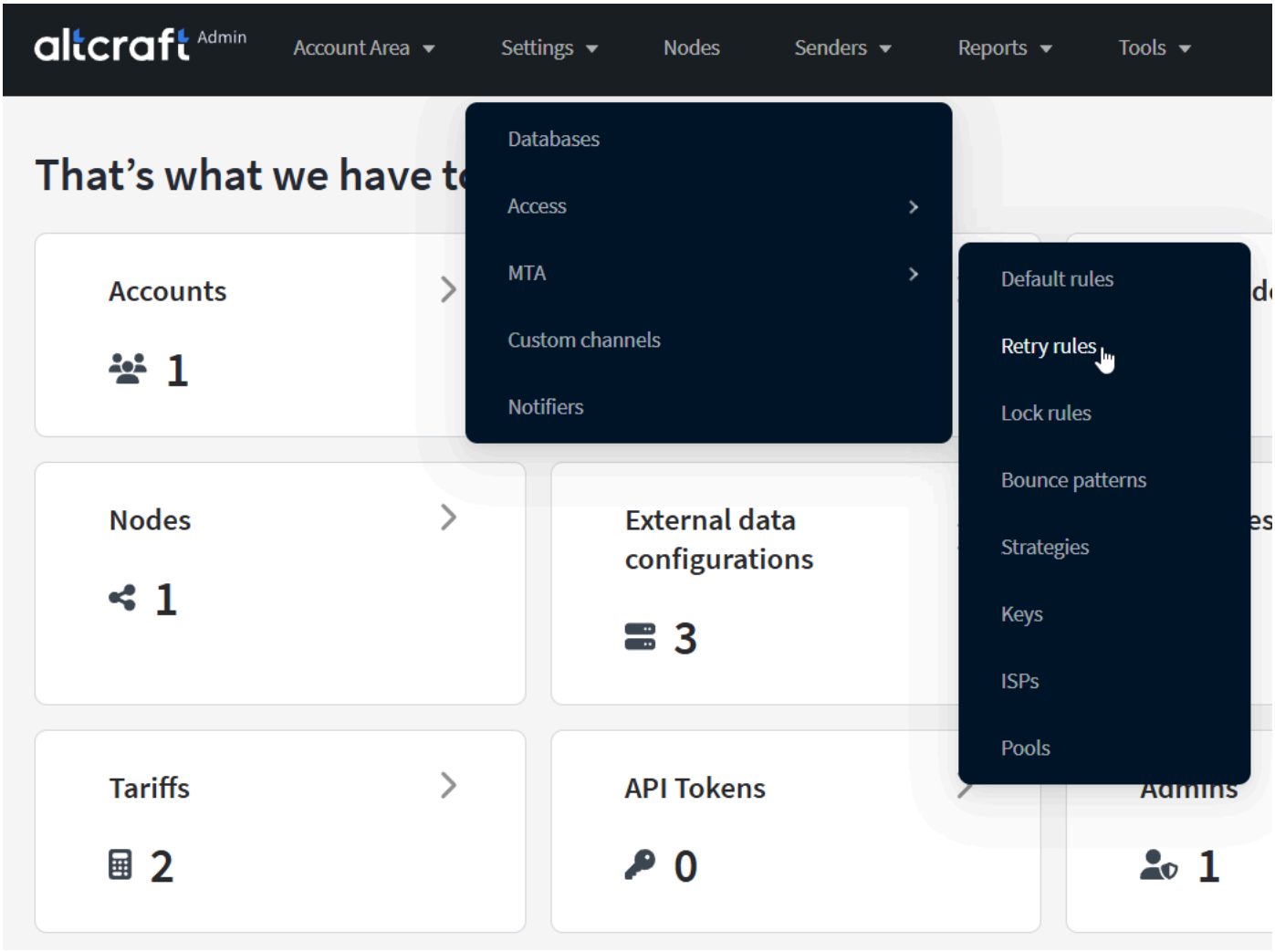
Retry rules

Description

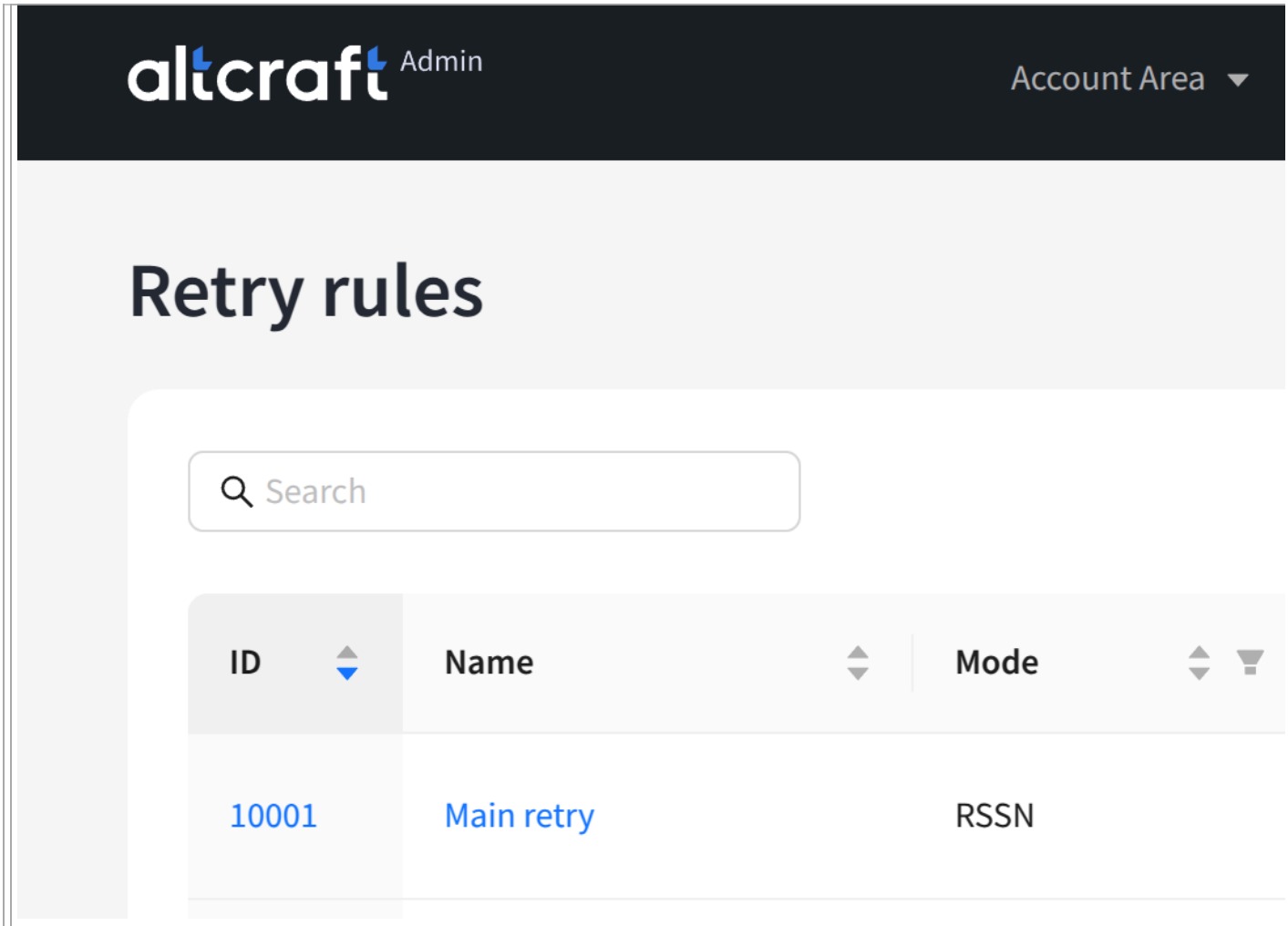
Retry rules are a tool used in cases when the ISP server responds that the recipient's mailbox is tempora

Creating, Editing, and Deleting

To manage retry rules, in the main menu select Settings — MTA — Retry rules.



The opened list displays existing AKMTA retry rules. On the right side of the table are the edit and delete



The following parameters are available for each rule:

- **Name — the rule name. The more informative it is, the more convenient it is to use in sending scenari**
- **Mode — IP selection for retry:**
  - **Retry from the same IP of the same node — the same IP address on the same node,**
  - **Retry from any other IP of the same node — any other IP on the same node,**
  - **Retry from any other IP of any node — any other IP on any node.**
- **Number of retries — the number of delivery attempts.**
- **Time interval (min) — the interval between attempts.**
- **Type — the type of interval increase between attempts:**
  - **Fixed (no increase) — fixed interval without increase,**
  - **Geometric increase delay — geometric interval increase exponentially,**
  - **Calculated random delay — interval increase by a random value with a modifier.**
- **Exponent — exponent for interval increase in geometric progression.**
- **Random modifier — modifier for calculating the random interval increase.**

NOTE

The minimum random value for interval increase is the interval itself. The maximum is the interval multi

Retry rules > Edit

* Name

Mode

Retry from the same IP of the same node

Type

Geom

* UserNumber of retries

6

* Time i

5

* Exponent

2

Random

12

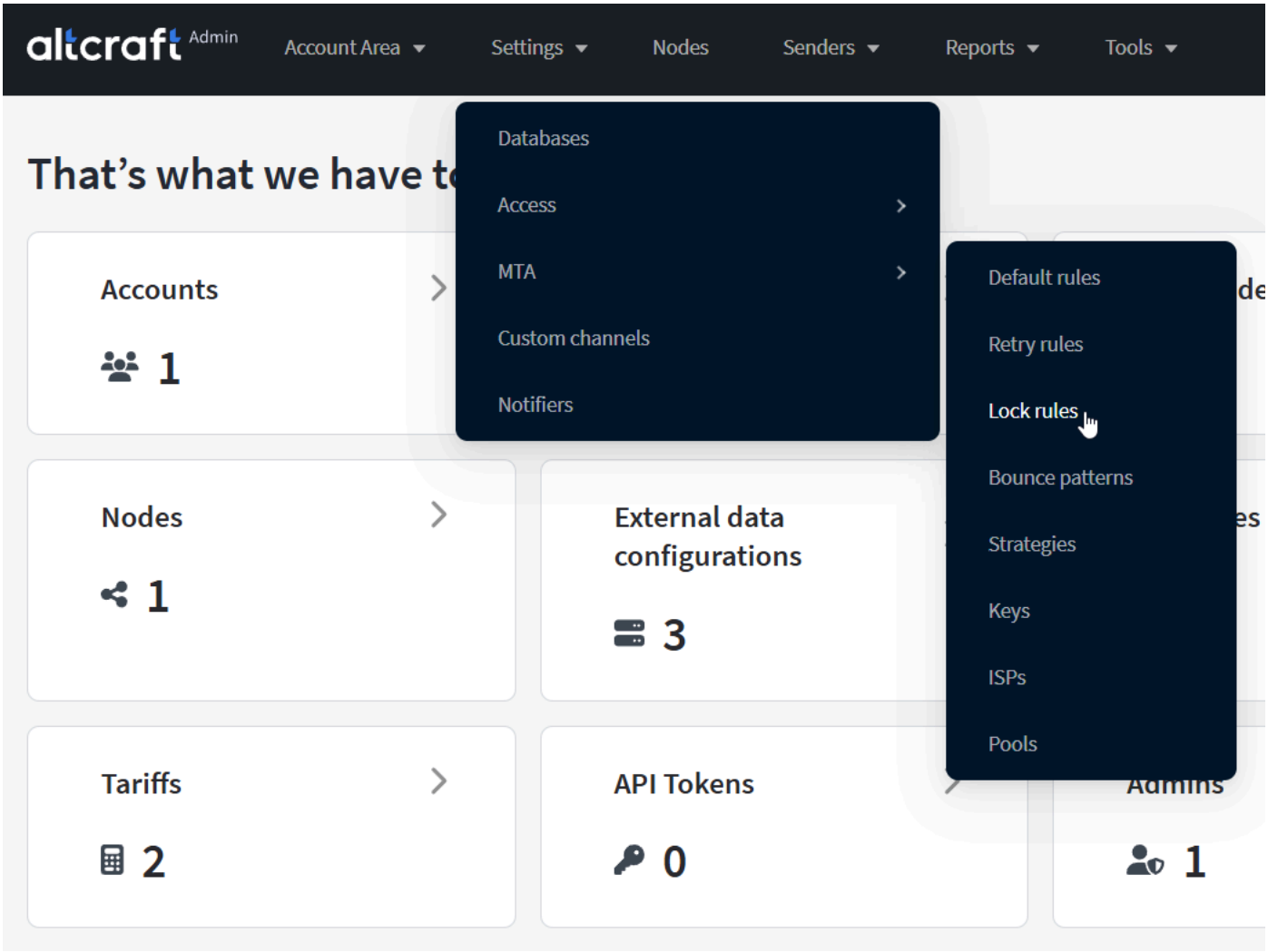
Lock rules

Description

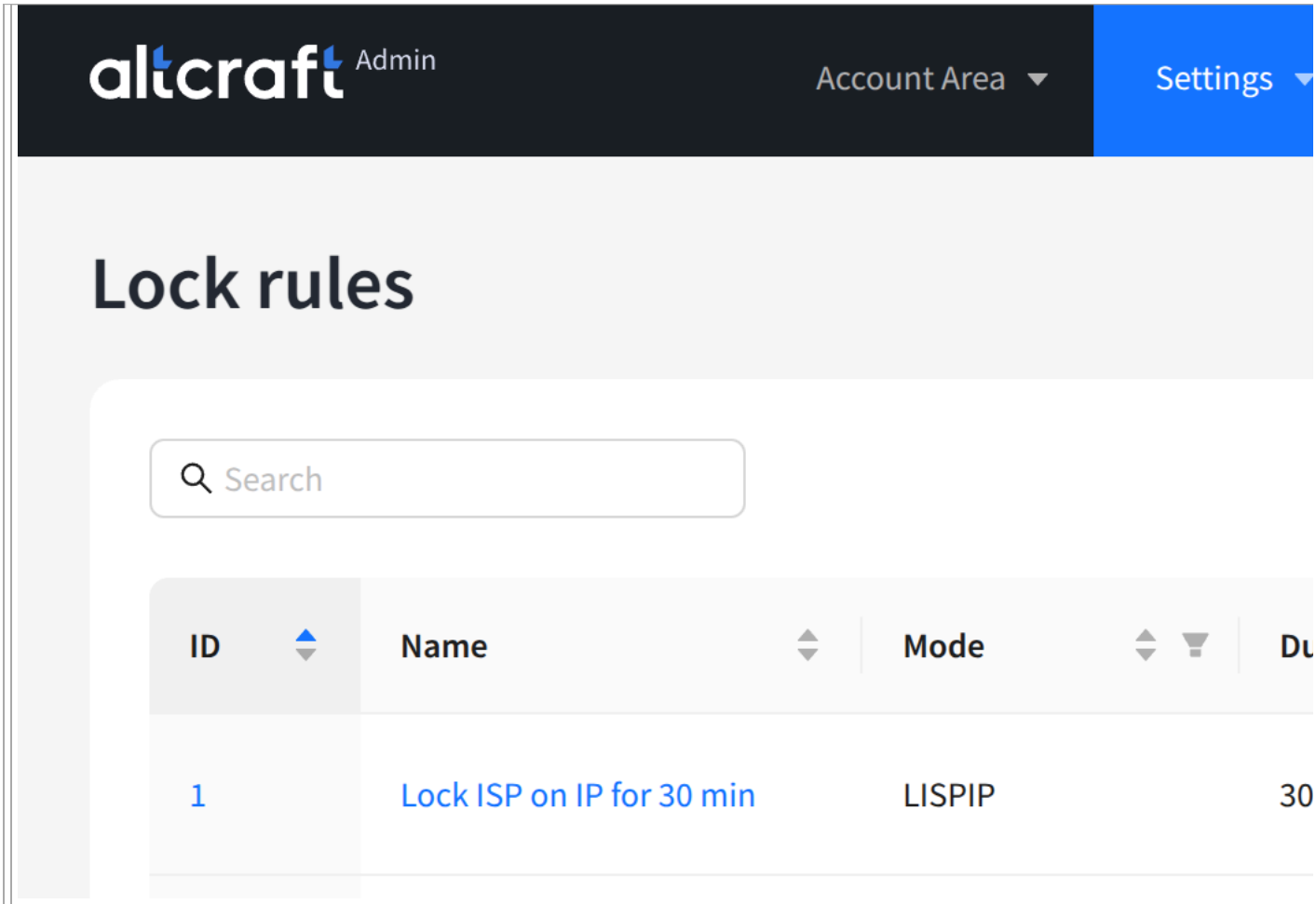
Lock rules are a tool that temporarily prohibits sending messages to a domain group of specific ISPs.

Creating, Editing, and Deleting

To manage lock rules, in the main menu select Settings — MTA — Lock rules.



The opened list displays existing AKMTA lock rules. On the right side of the table are the edit and delete b



The following fields are configured for a lock rule:

- **Name — the rule name. The more informative it is, the easier it is to use in sending scenarios.**
- **Mode — the type of lock:**
  - **Lock only the current IP — blocks only the current IP address from sending,**
  - **Lock ISP delivery from all IPs completely — blocks sending from any IP addresses to the ISP's dor**
- **Lock duration (minutes) — the lock time in minutes.**
- **Number of bounces — the number of negative responses from the ISP server.**
- **Time interval — the waiting period for negative responses from the ISP server, in minutes.**

Lock rules > Edit

* **Name**

Mode

Lock only the current IP

* **Lock duration (min)**

30

* **Number bounces**

3

Bounce patterns

Description

Bounce patterns are templates of responses from email providers (ISPs) about delivery problems. AKMTA

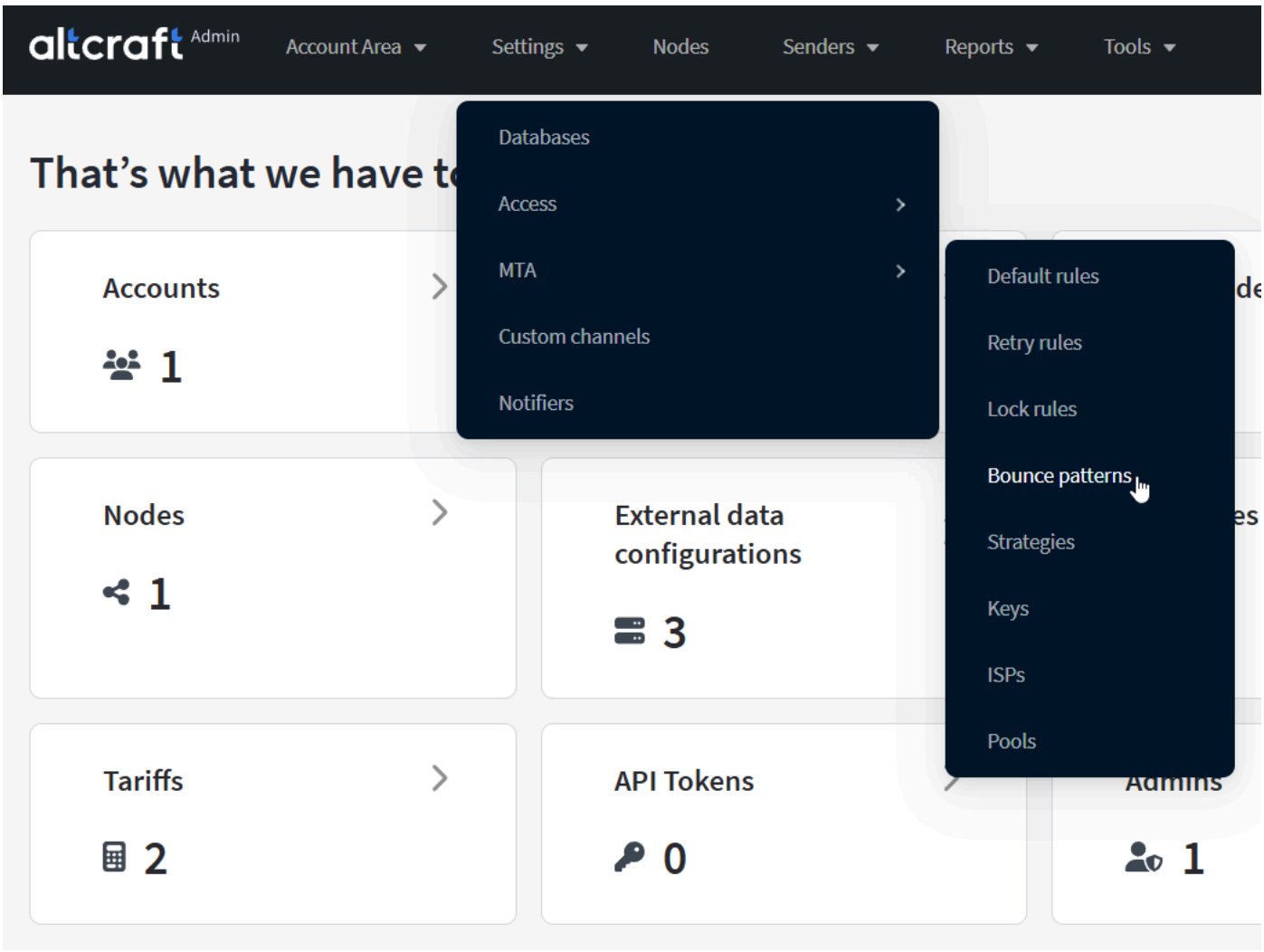
Types of negative ISP responses

Negative response templates are divided into three main categories:

- **Hard bounce (HB) includes server-class errors (5xx), which mean the recipient's mailbox does not exi**
- **Soft bounce (SB) contains client-class errors (4xx), indicating temporary unavailability of the mailbox.**
- **Bounce (B) — this type includes other delivery errors that are not Hard or Soft bounce. Typically, retry**

Creating, Editing, and Deleting

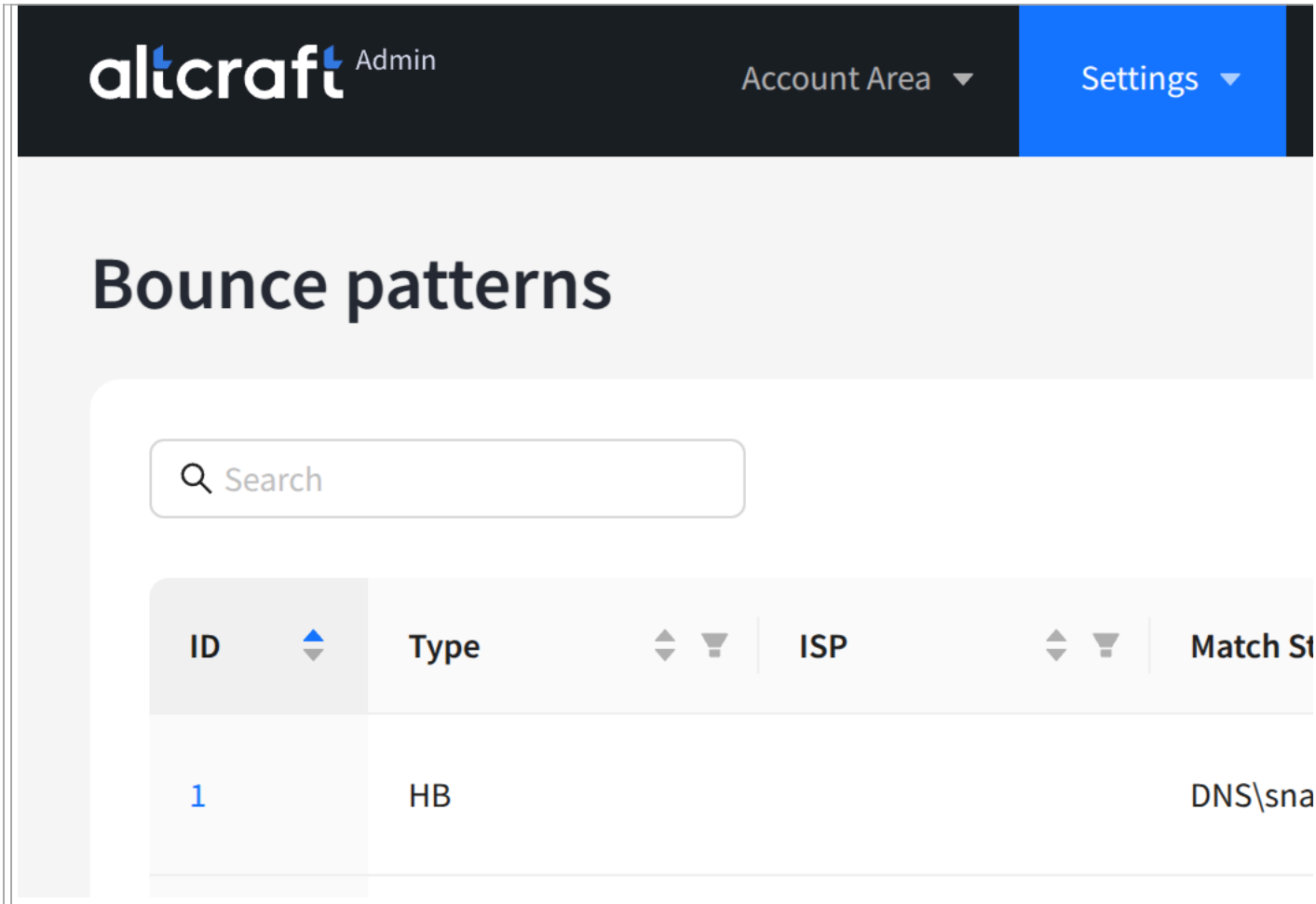
To manage ISP response templates, in the main menu select Settings — MTA — Bounce patterns.



The opened list displays existing templates of negative ISP server responses. For most responses, entries Create.

CAUTION

Only custom ISP response templates can be deleted.



Each response template is configured with:

- **Name — the template name.**
- **Type — the type of server response (Hard bounce, Soft bounce, Bounce).**
- **ISP — the email provider. You can select an ISP record or the All ISPs option to match responses from**
- **Pattern — the pattern used to match the response.**
- **Message — the delivery error message text.**

Bounce patterns > Edit

Name

Type

Soft bounce

* **ISP**

92 - iCloud

* **Pattern**

* **Message**

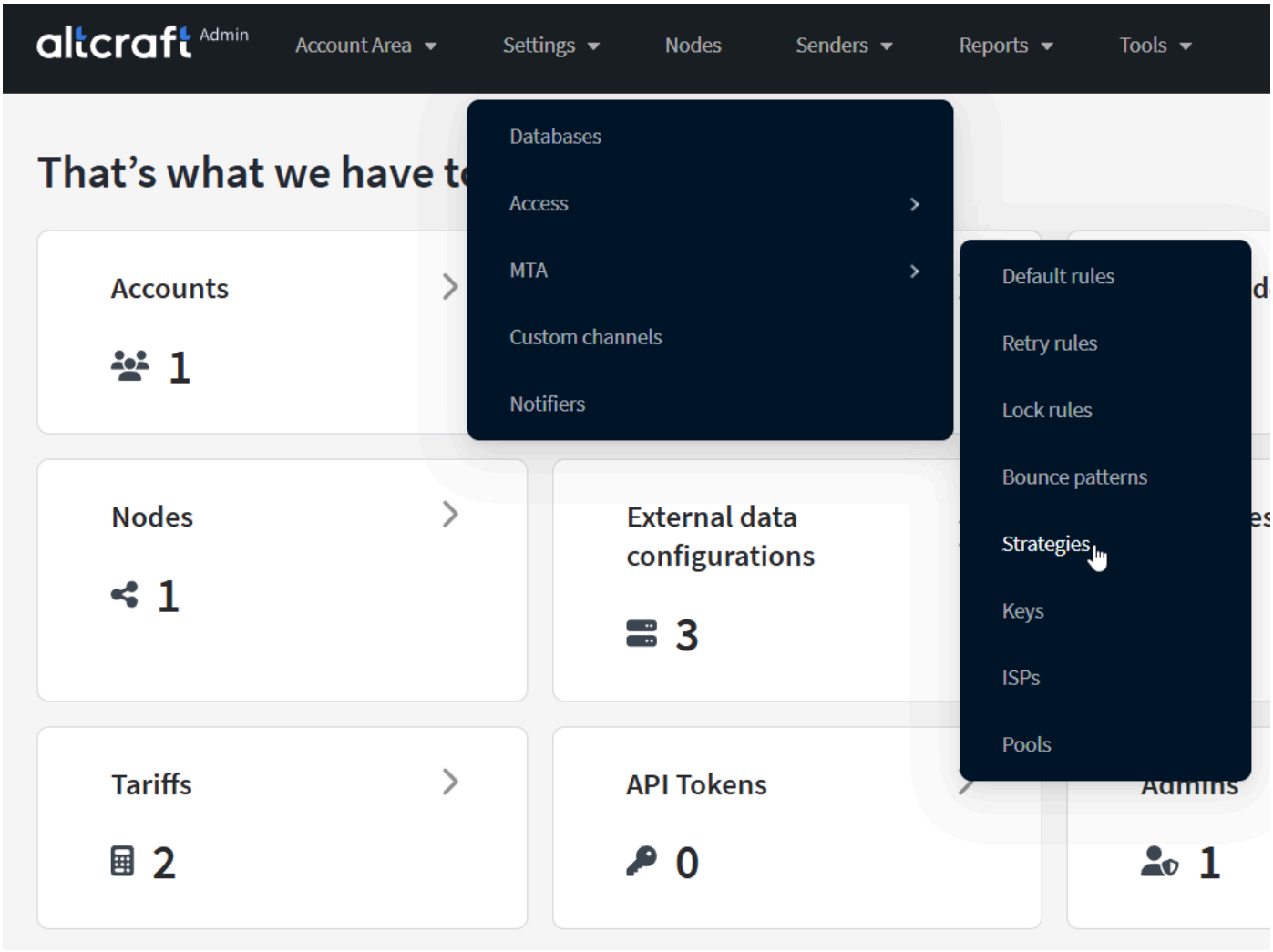
Strategies

Description

Strategies are a set of rules and settings that define message delivery parameters via AKMTA strategy. T

Creating, Editing, and Deleting

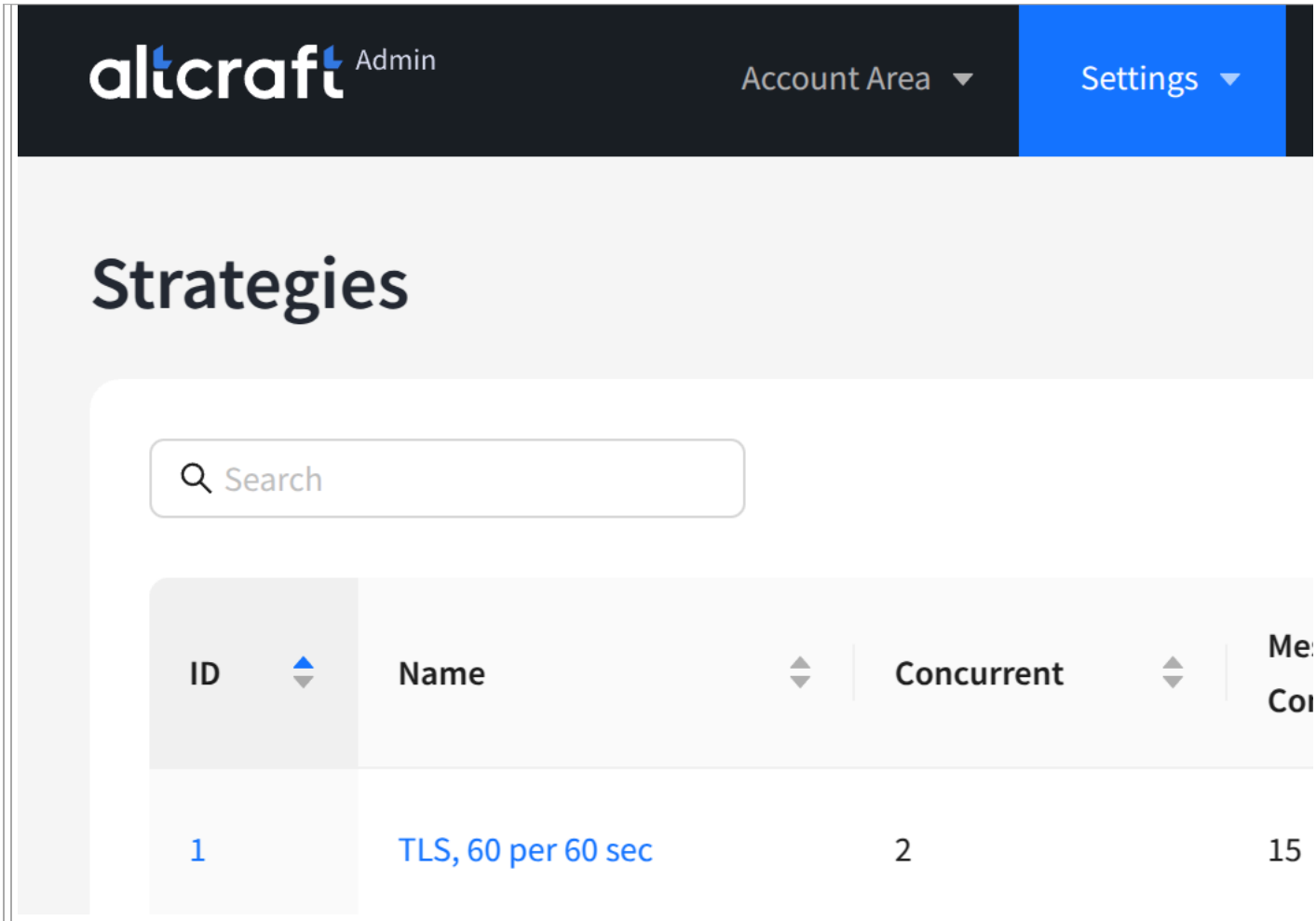
To manage sending strategies, in the main menu select Settings — MTA — Strategies.



The opened list displays existing AKMTA sending strategies. By default, the system has a 60:60 strategy - + Create.

CAUTION

The default strategy cannot be deleted from the system. However, if you only need one configuration, yo



Sending Strategy Settings

For each sending strategy, you must specify the following settings:

- **Name — the strategy name.**
- **Concurrent connections — the number of simultaneous connections.**
- **Messages per connection — the number of messages per connection.**
- **Number of connection attempts — the number of connection attempts within the time interval.**
- **Time interval (seconds) — the time interval in seconds.**

Strategies > Edit

* **Name**

* **Concurrent connections**

1

* **Messa**

1

* **Connection attempts**

1

* **Time i**

60

Average strategy speed per IP address

In 1 day	In 1 hour
1.4K / 1 day	60 / 1 hour

HELO type

Greet with IP PTR

- [x] **Try TLS first**
- [ ] **Dro**

The number of messages sent per interval equals the product of the first three parameters. The Average s

Within the strategy, you can also configure the HELO command. Three greeting types are available:

- **Greet with IP PTR — using the reverse PTR record of the IP address.**
- **Greet with From domain — using the sender domain (From email).**
- **Greet with IP — using the sender's IP address.**

The Try TLS first option will attempt to use an encrypted connection when connecting.

> **DANGER**
>
> **The Drop message option deletes all messages from the sender’s queue instead of delivering them. This**

Keys

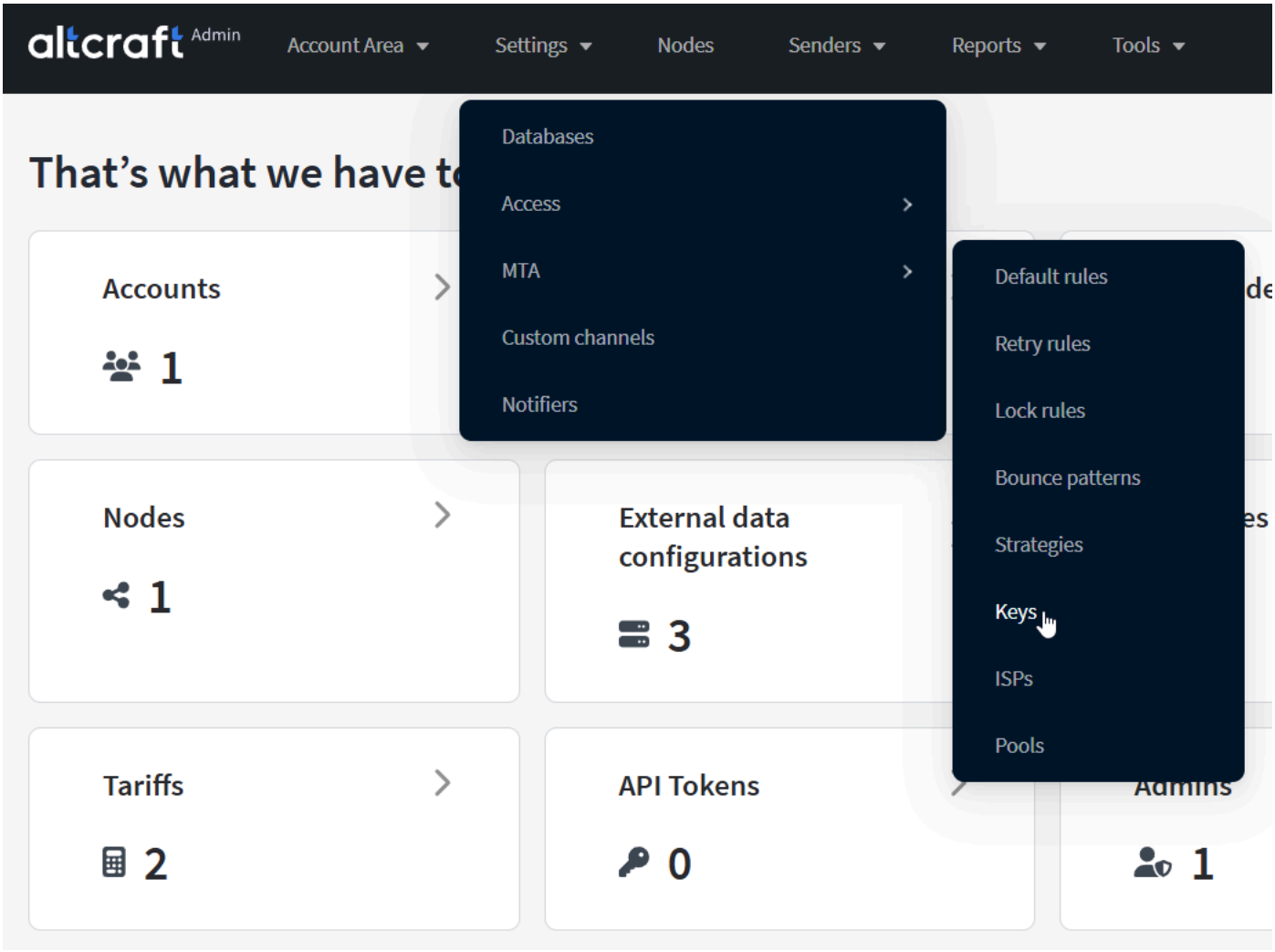
Description

Keys are cryptographic tools used to ensure the secure transfer of data.
The platform supports two types of keys:

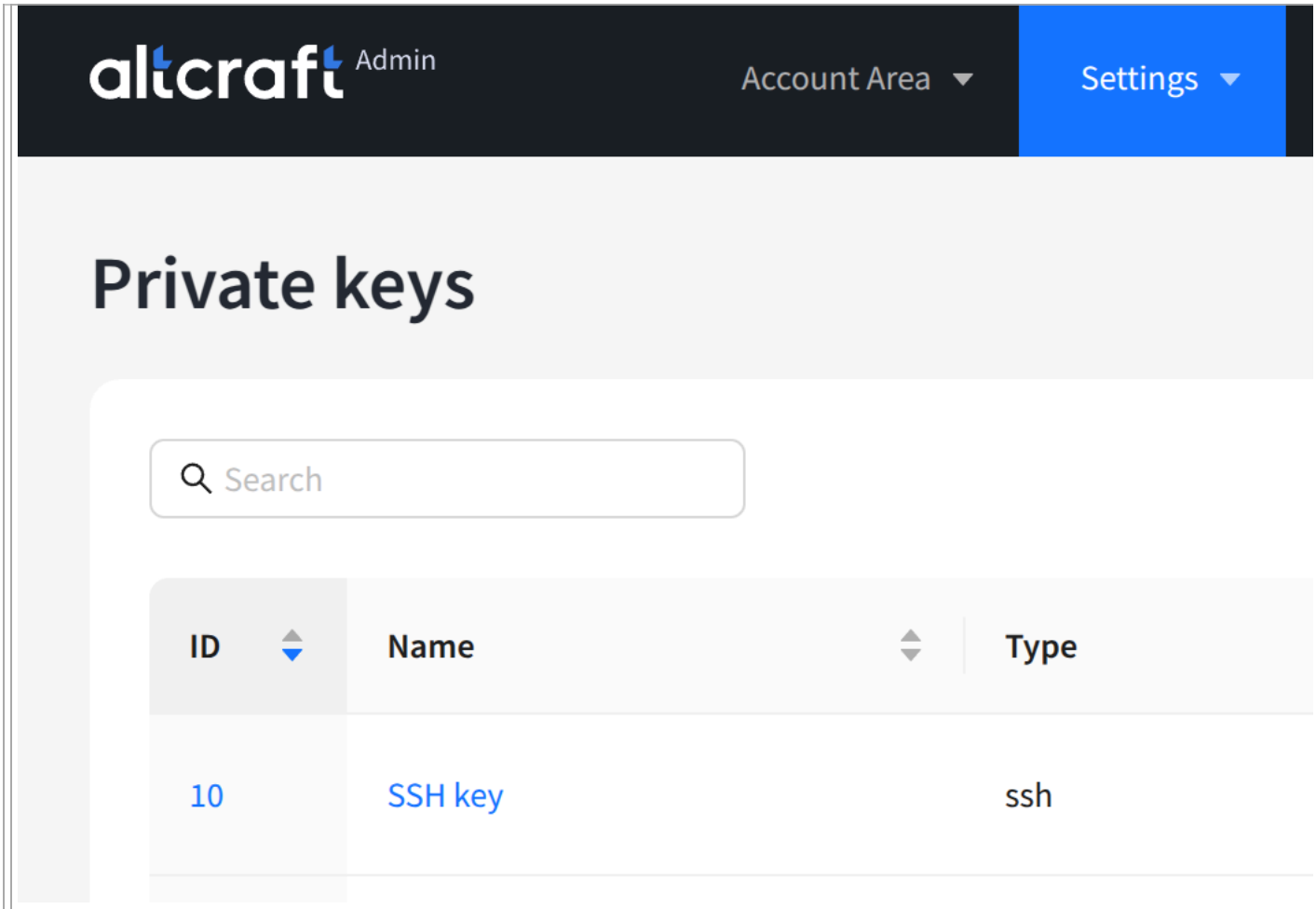
- **DKIM (DomainKeys Identified Mail) — used to authenticate the sender of email messages and protect**
- **SSH (Secure Shell) — used for securely connecting to the server for administration.**

Creating, Editing, and Deleting

To create keys, in the main menu select Settings — MTA — Keys.



The opened list displays existing SSH and DKIM key pairs. On the right side of the table are the edit and d



Select the key type (DKIM or SSH). After that, you can paste existing keys into the Public key and Private

Private keys > Edit

Name

new

Type

SSH key

Generate key pair

or paste keys

Public key

Paste public key

Private key

Paste private key

Save the key. Later, you can use it when configuring other entities in the Admin Panel.

ISPs

Description

ISPs are email providers through which email messages are sent. The platform allows you to configure ru

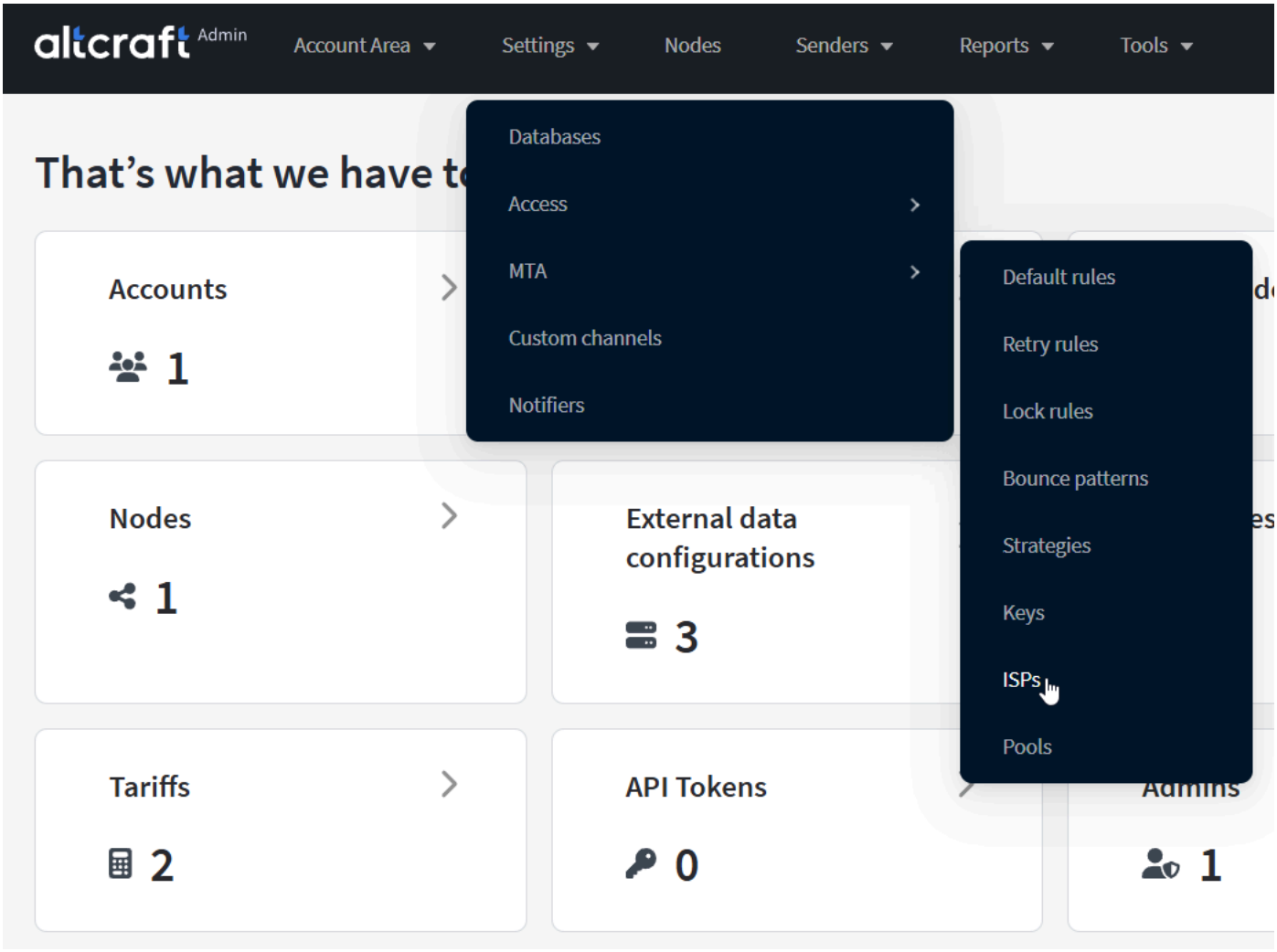
When is customization needed?

Configuring ISPs can be useful in the following cases:

- **Sending limits — some providers limit the number of incoming messages as part of bulk mailing polici**
- **Improving domain and IP reputation — setting individual rules for ISPs helps improve email deliverabil**
- **Specific response codes from providers — some ISPs use unique error codes and delivery statuses.**
- **Adding service headers — some providers may require including additional headers in emails.**

Creating, Editing, and Deleting

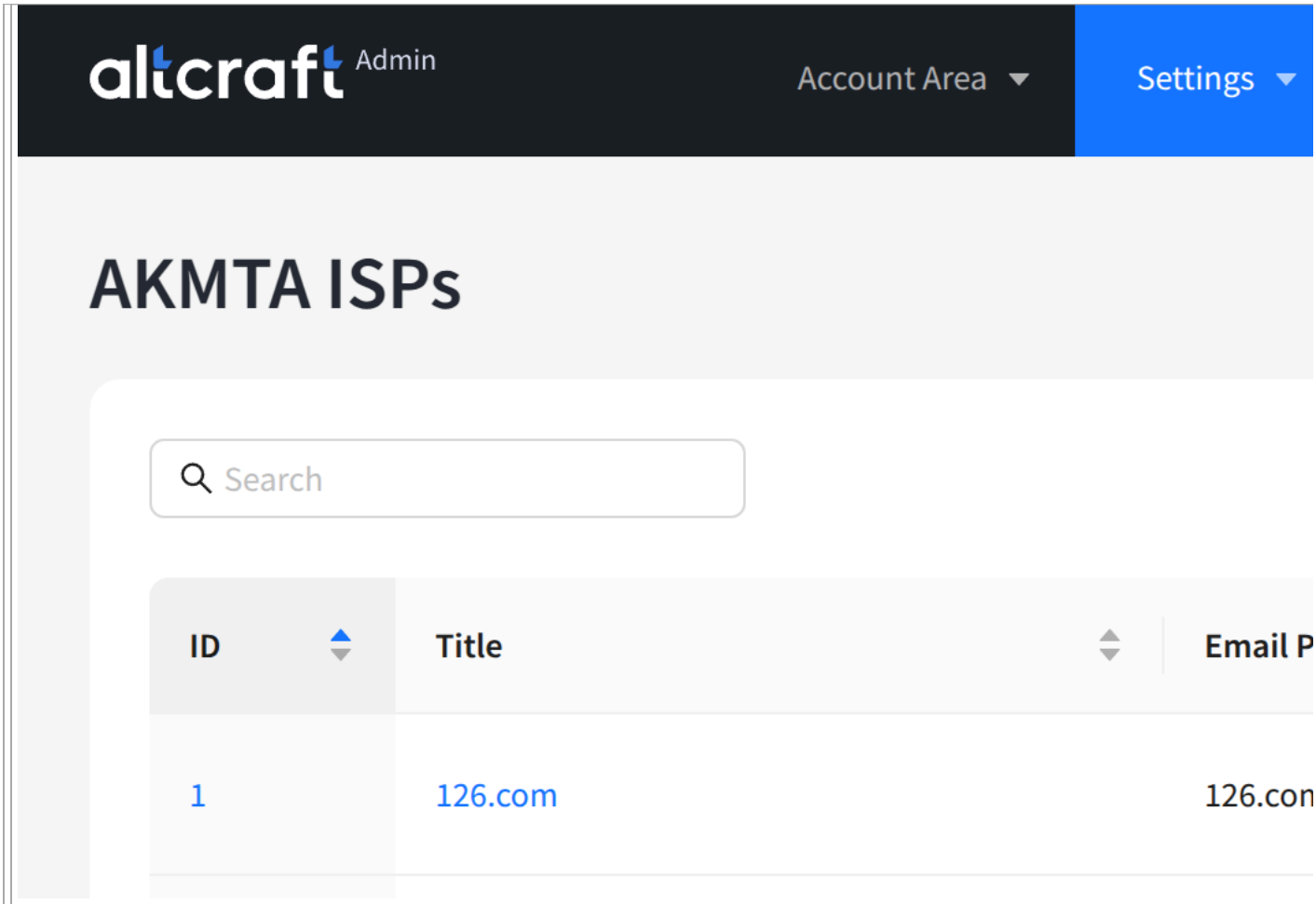
To manage email providers, in the main menu select Settings — MTA — ISPs.



The opened list displays existing ISP records in the AKMTA module. For most email providers, entries alre
Create:

CAUTION

Only custom email provider records can be deleted.



ISP Record Settings

Each ISP record is configured with:

- **Name — the email provider name.**
- **Encoding (Global, HTML, Text) — the message data encoding type. Available options: Quoted-Printable**
- **Domains — the domains of the email provider's mailboxes.**

AKMTA ISPs > Edit

* **Name**

Global encoding

Sender encoding

HTML encoding

Sender encoding

Te

* **Domains**

One domain per line

Headers overriding

Added headers

+ Add header

Deleted headers

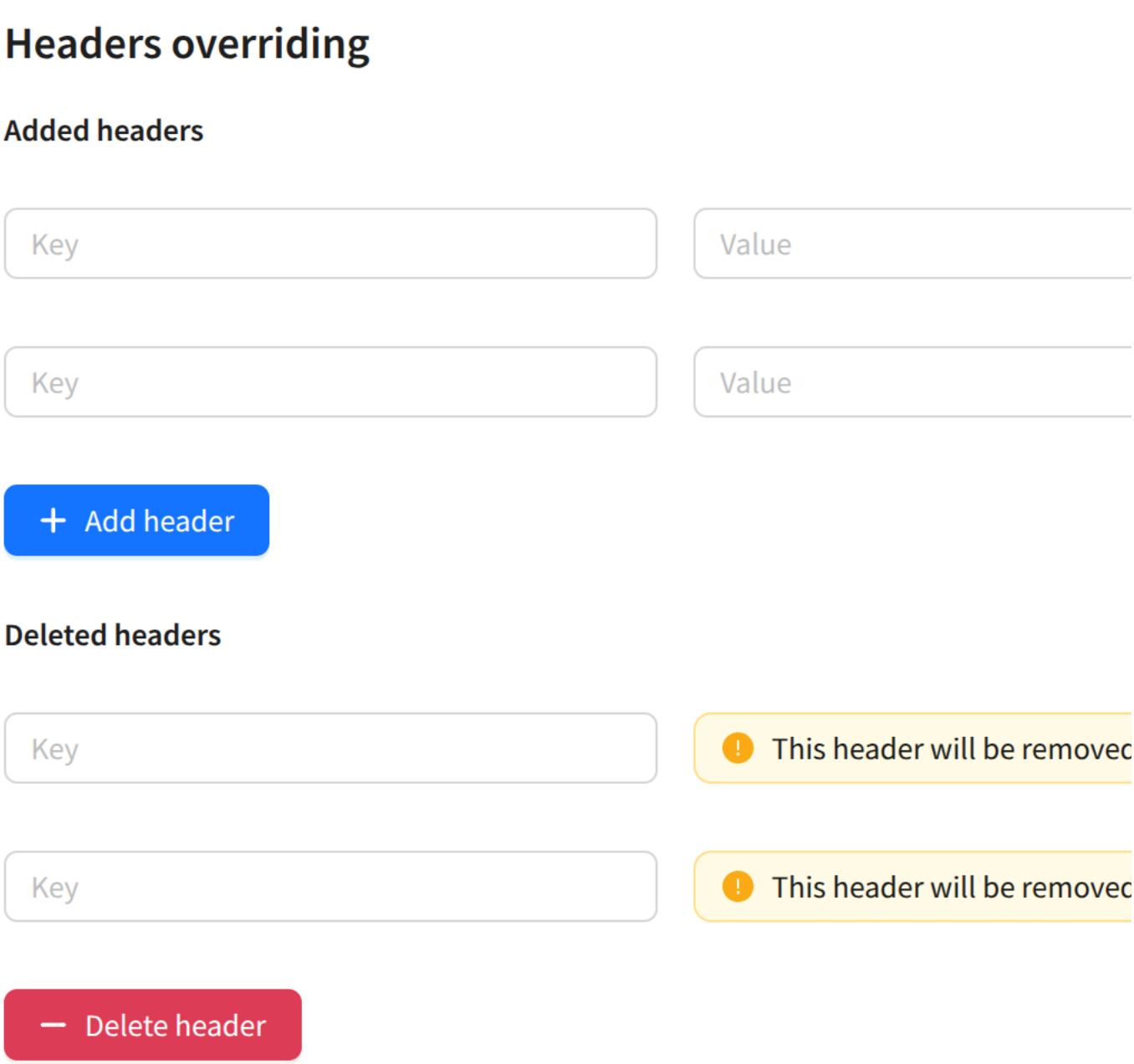
– Delete header

If you need to separately configure delivery to part of the email provider's domains, create a separate ISP

Overriding Headers

In AKMTA ISP records, you can modify email headers:

- **Add a header — by specifying the key and its value.**
- **Delete a header — by specifying the key you want to remove.**



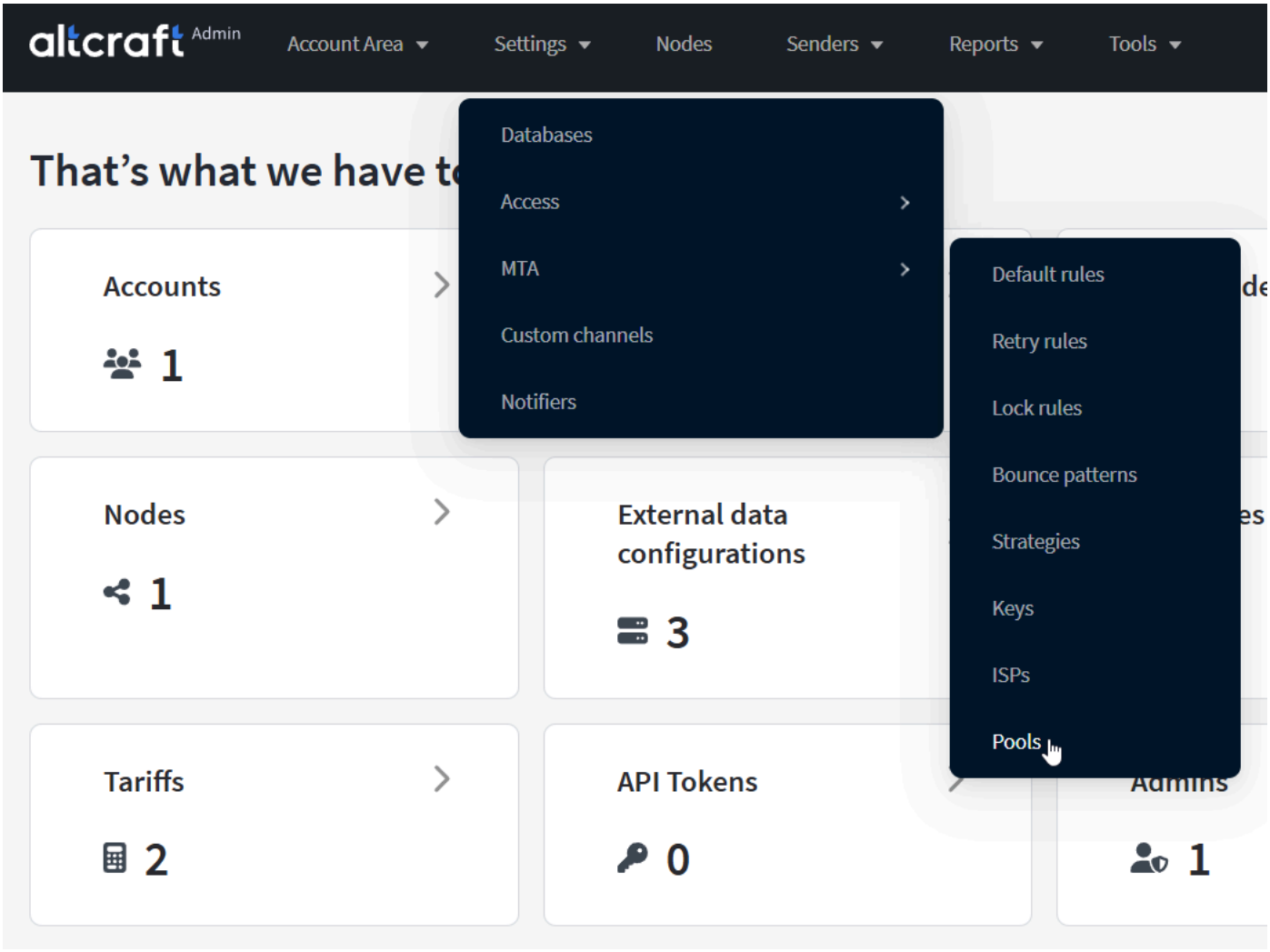
Pools

Description

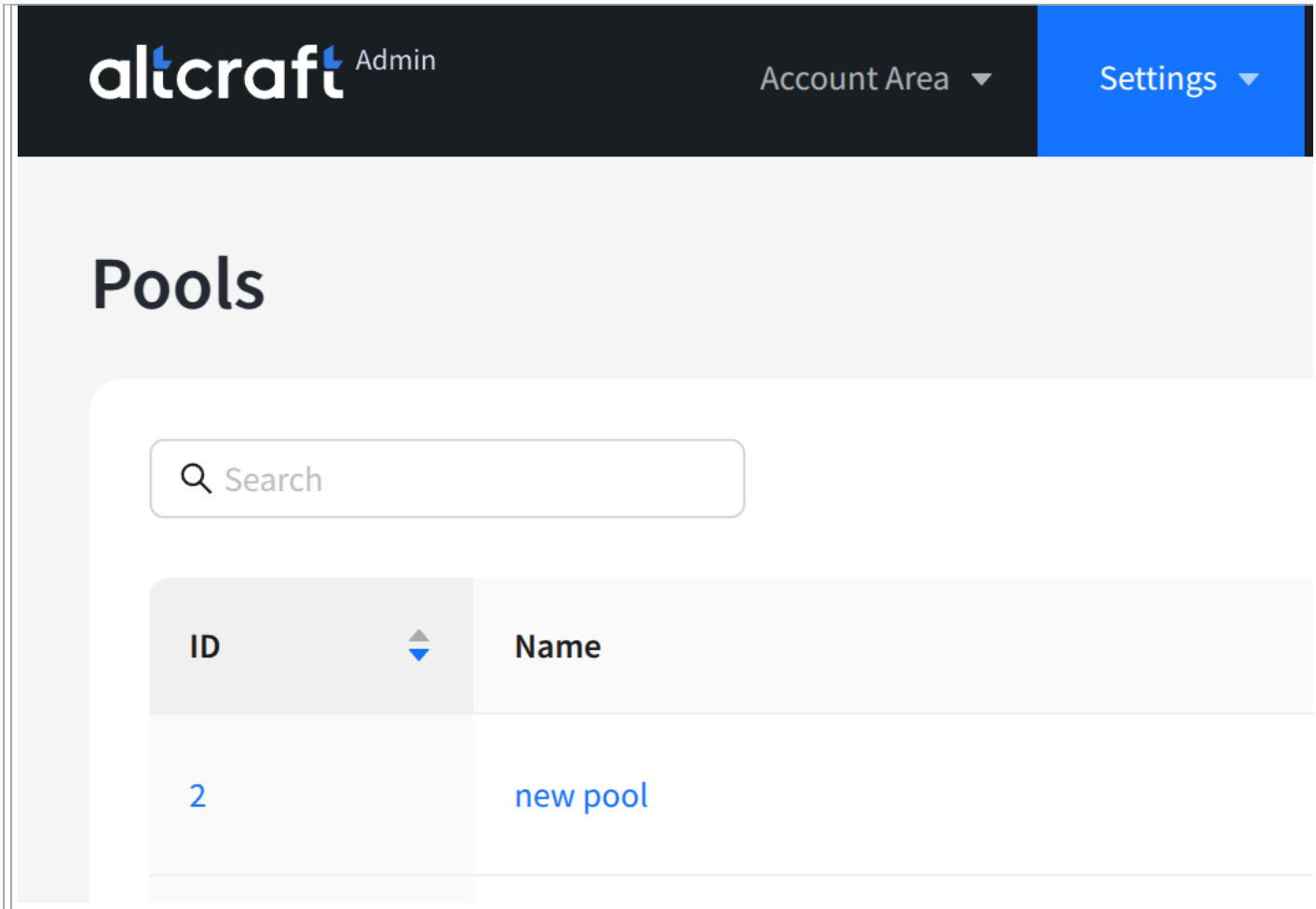
A Pool is a set of resources required to send emails from a specific domain. The pool allows you to simult

Creating, Editing, and Deleting

To manage pools, in the main menu select Settings — MTA — Pools.

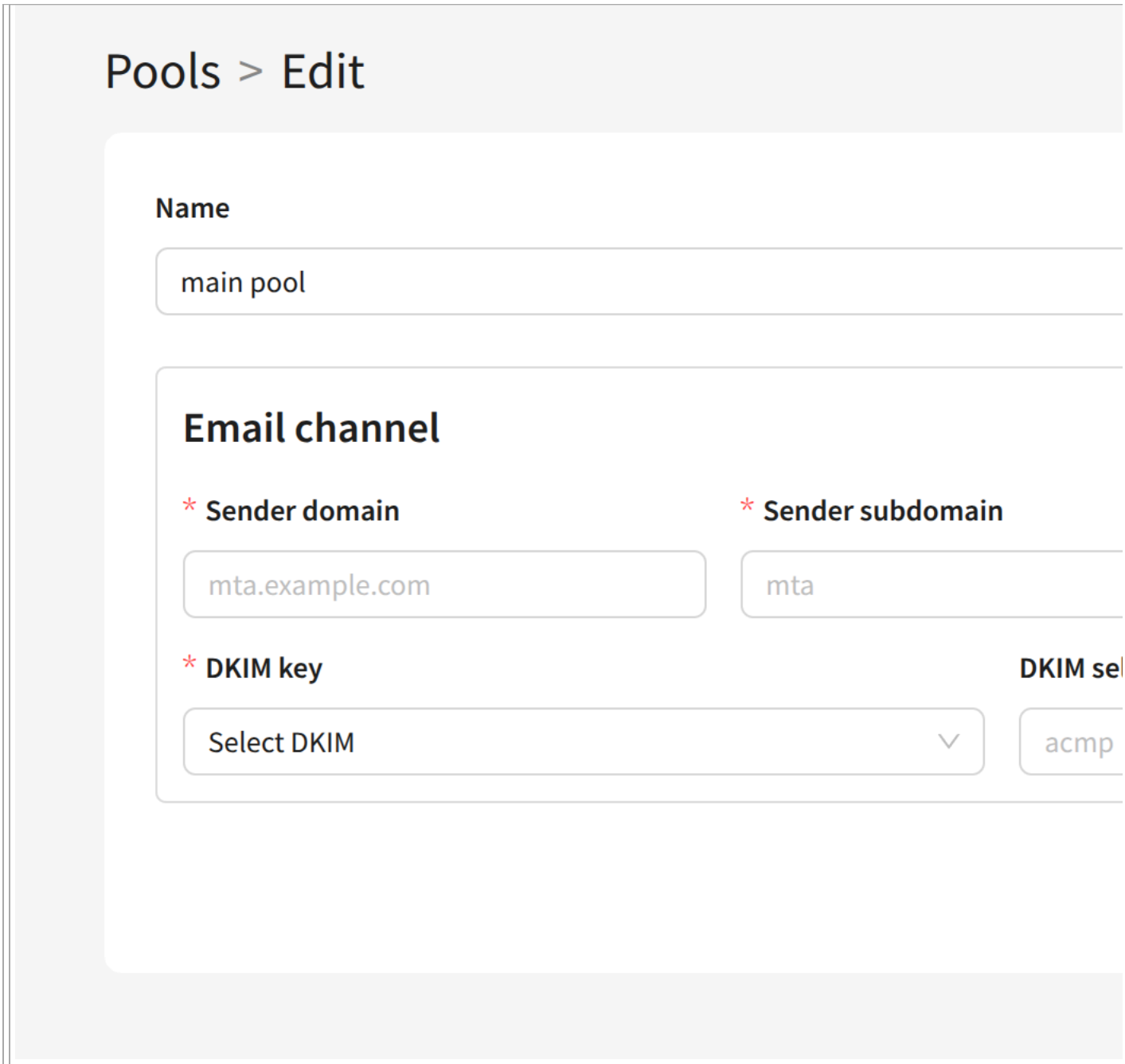


The opened list displays existing pools. On the right side of the table are the edit and delete buttons. You



To create a pool, you need to specify the following settings:

- **Name — the pool name**
- **Sender domain — the sending module's domain**
- **Sender subdomain — the sending module's subdomain**
- **SPF domain — the sender's SPF domain**
- **DKIM selector — the DKIM key selector of the sending domain**
- **DKIM key — the DKIM key for the sending domains**



After saving, the created pool will appear in the list. You can use it when configuring other entities in the

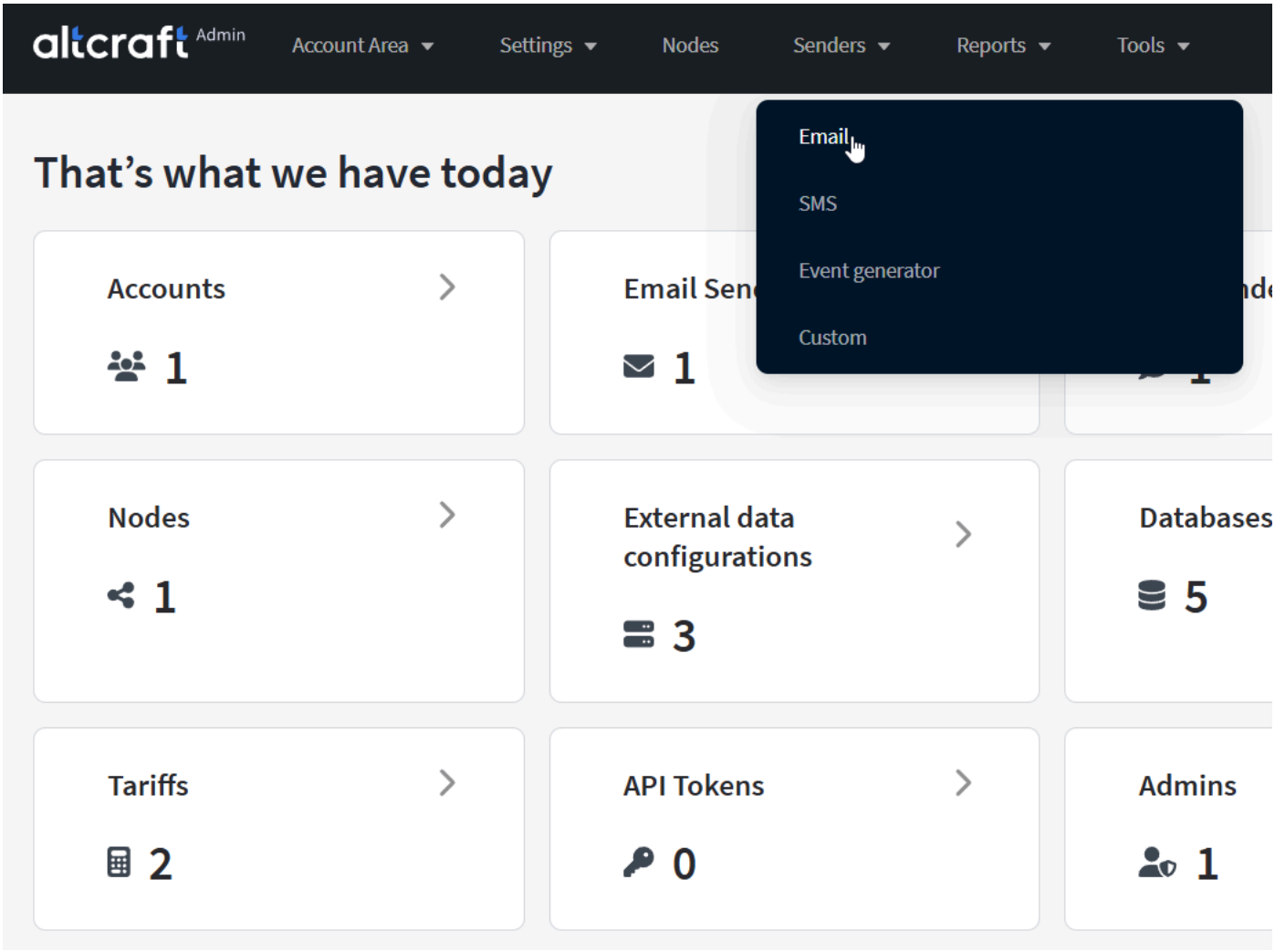
Email

Description

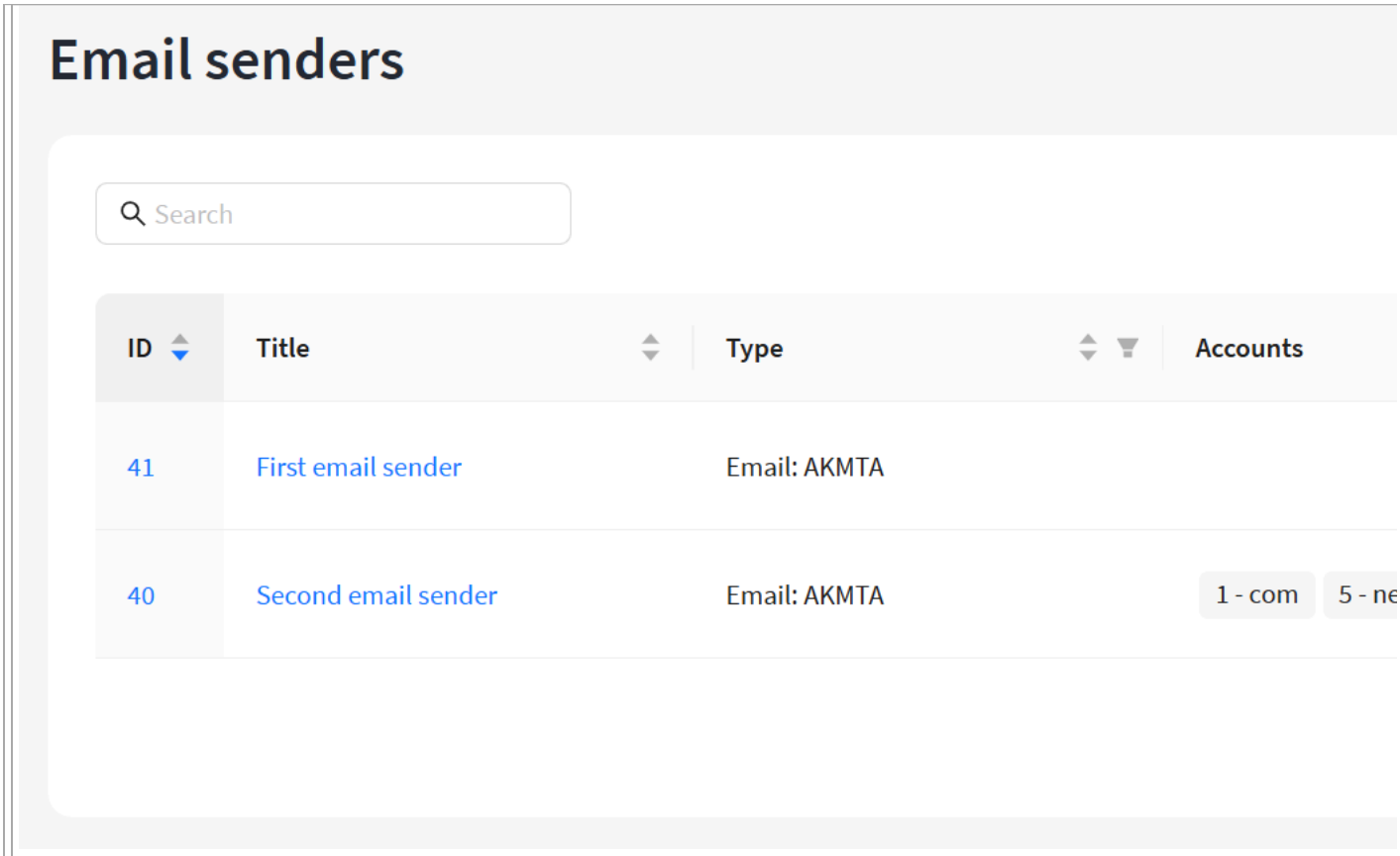
Sender in Altcraft is a configuration of the message delivery service. Senders are configured for email an

Creating, Editing, Deleting

To manage email senders, in the main menu select Senders — Email:



The opened sender list displays existing senders. On the right side of the table, there are edit and delete



For a new sender, in the Name field specify a unique name, and in the Assigned accounts menu select pla

Email Senders

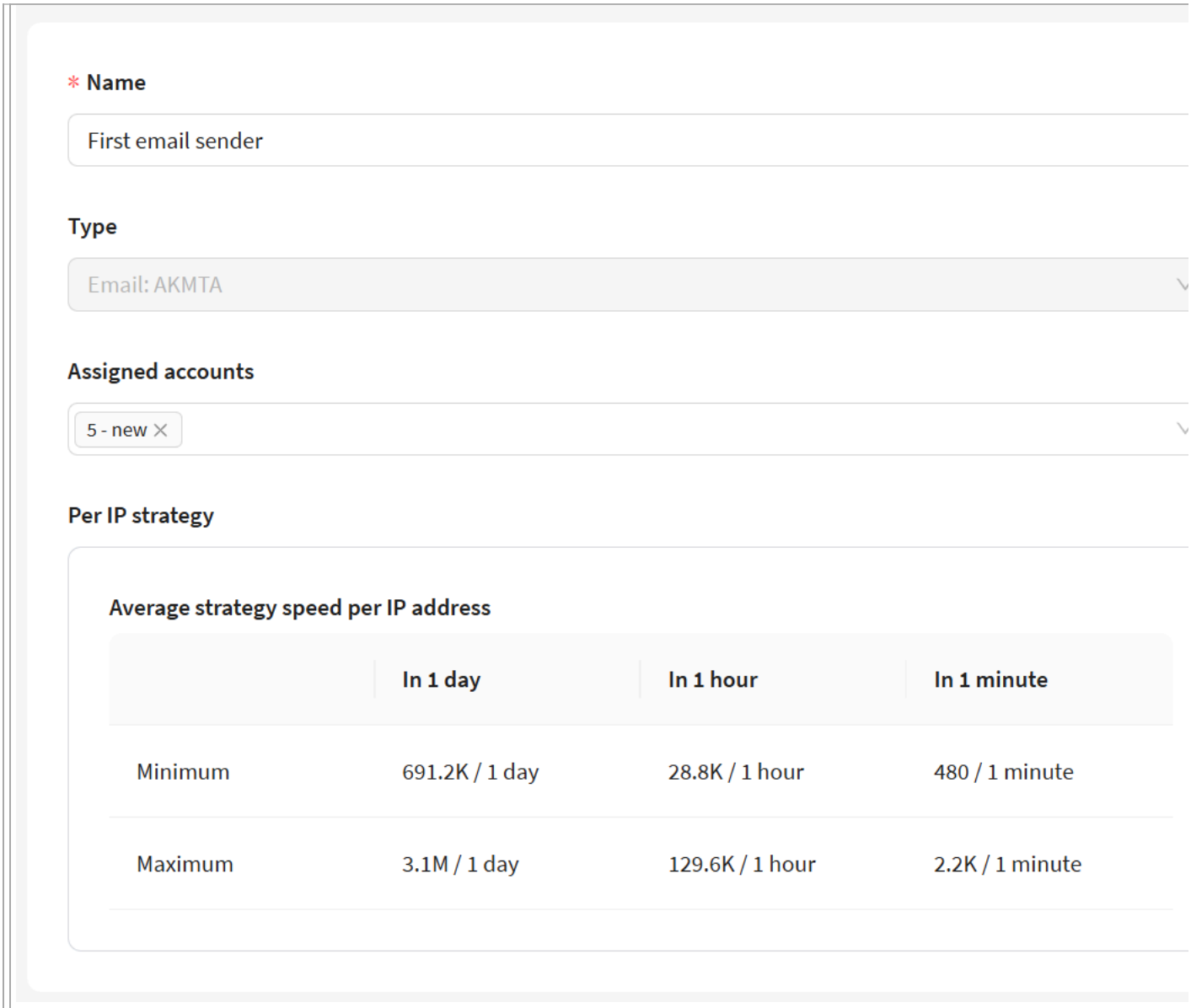
The following solutions are used for sending, processing, and receiving emails:

- **AKMTA (Altcraft Mail Transfer Agent) — the built-in Altcraft sending module. Its setup is described in t**
- **Amazon SES (Simple Email Service) — integration with Amazon's transport module. Documentation lin**
- **Sendsay — integration with Sendsay SMTP relay. Webhooks are used to obtain delivery statuses. Docu**

NOTE

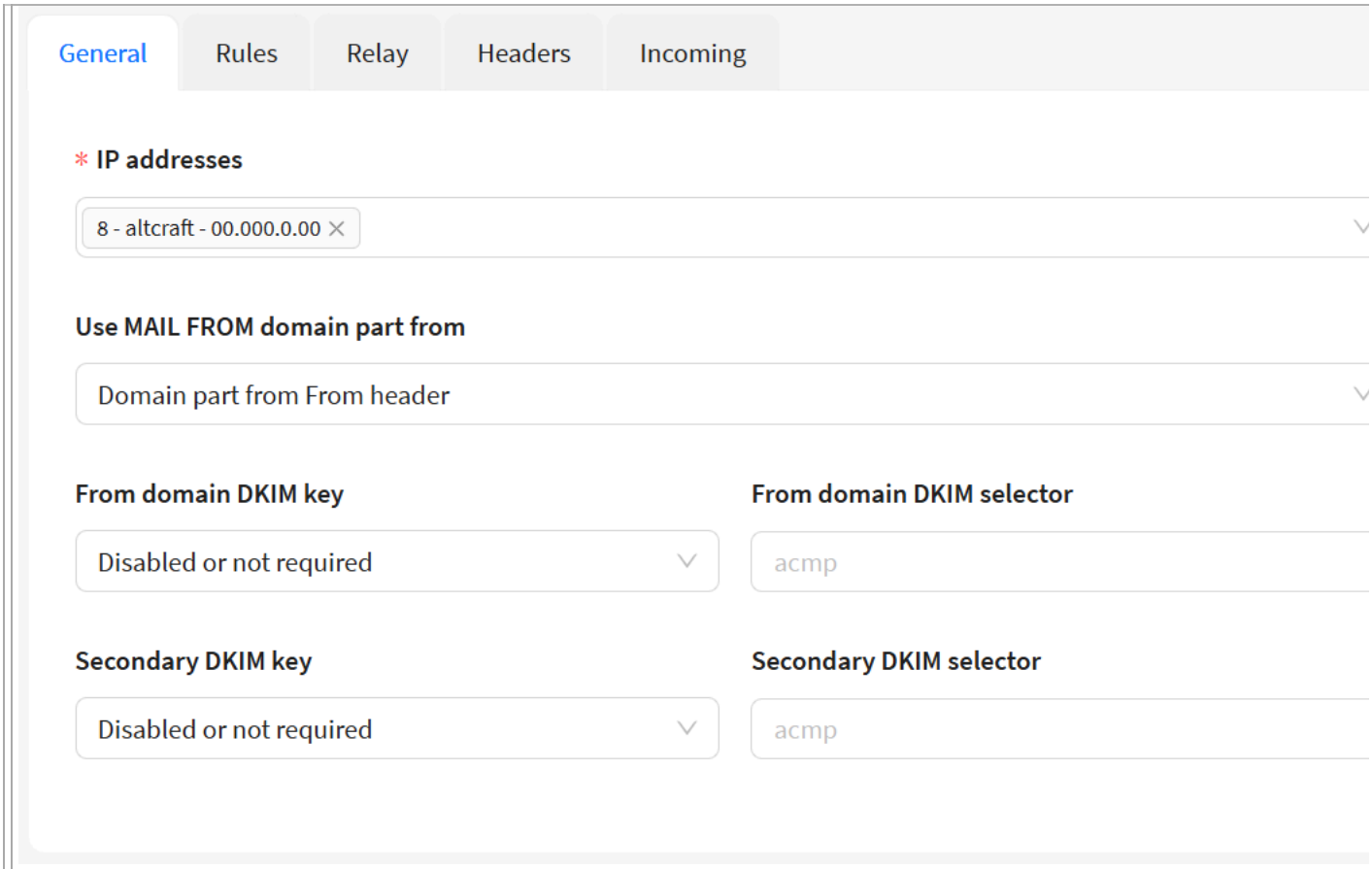
The platform also provides integration with the Get-n-post sender. Contact our support team to learn mo

AKMTA Sender Configuration



When selecting the built-in delivery module (Type - Email: AKMTA), the MTA configuration settings for the

General Settings



In the General block, you configure sender domains and IP addresses. The following parameters are avail

- **IP addresses — sender IP addresses. Must be available on the [**sending node**].**

> **TIP**
> **Instructions for adding IP addresses to the sender are available here.**

- **Sender domain type — select the domain the sending module will represent. This can be:**
  - **Domain from the **From** header of outgoing messages (**Domain part from "From" header**)**
  - **Domain from the PTR record of the sending IP (**Reverse record from sender IP (PTR)**)**
  - **Custom domain (**Specify custom domain**)**
  - **Aligned prefix and **From** header (**Aligned prefix and From header**)**
- **From domain DKIM key — DKIM key of the sender domain.**
- **From domain DKIM selector — DKIM selector of the sender domain.**
- **Secondary DKIM key — secondary DKIM key.**
- **Secondary DKIM selector — secondary DKIM selector.**
- **MAIL FROM subdomain — subdomain of the main domain. Used when choosing the **Aligned prefix ar**
- **Pool — resource pool needed for sending emails from a specific domain. Used when choosing the **Al**

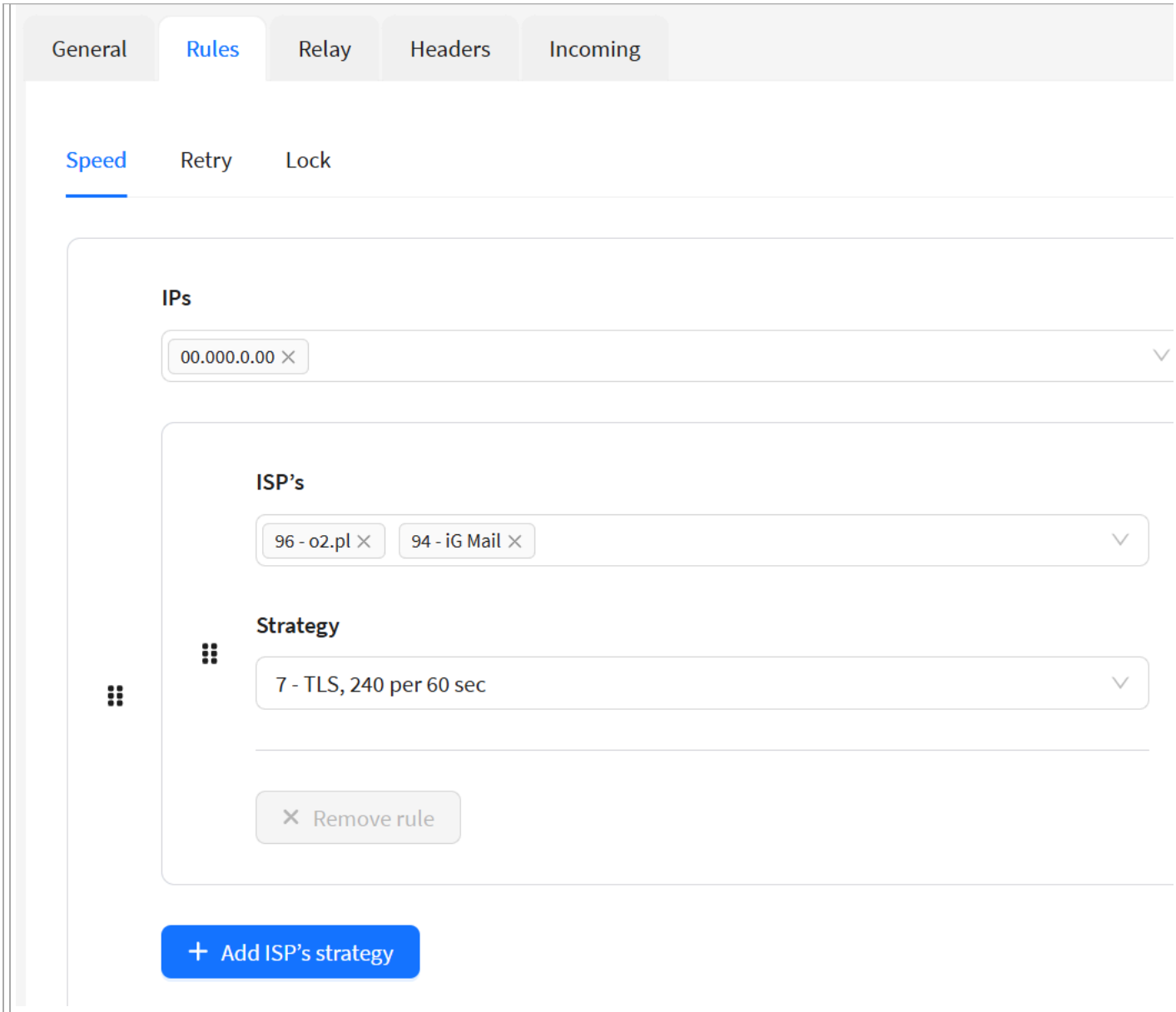
Rules

> **TIP**
> **AKMTA settings are covered in a separate section of the documentation. It describes ISP management, s**

Each AKMTA sender can be finely tuned to deliver messages depending on the number of available IP add

- **Speed — defines the sending speed from sender IP addresses.**
- **Retry rules — retry sending rules.**
- **Lock rules — sending lock rules.**

> **DANGER**
> **Retry rules and Lock rules fully override default AKMTA sending scenarios. Configure them for the sende**



Speed — managing delivery speed

You can separately control the sending speed for different IPs, different email providers, and combination

- **IPs — sender IPs for which you want to set a separate speed.**
- **ISPs — email providers for which you want to set a separate speed within an IP.**

> DANGER
> **Sending speed for major providers like Gmail, Yandex, Mail.ru, Outlook.com, and others must be spec**

- **Strategy — sending strategy parameters. Created in [AKMTA settings](/en/admin-guide/admin-panel/se**

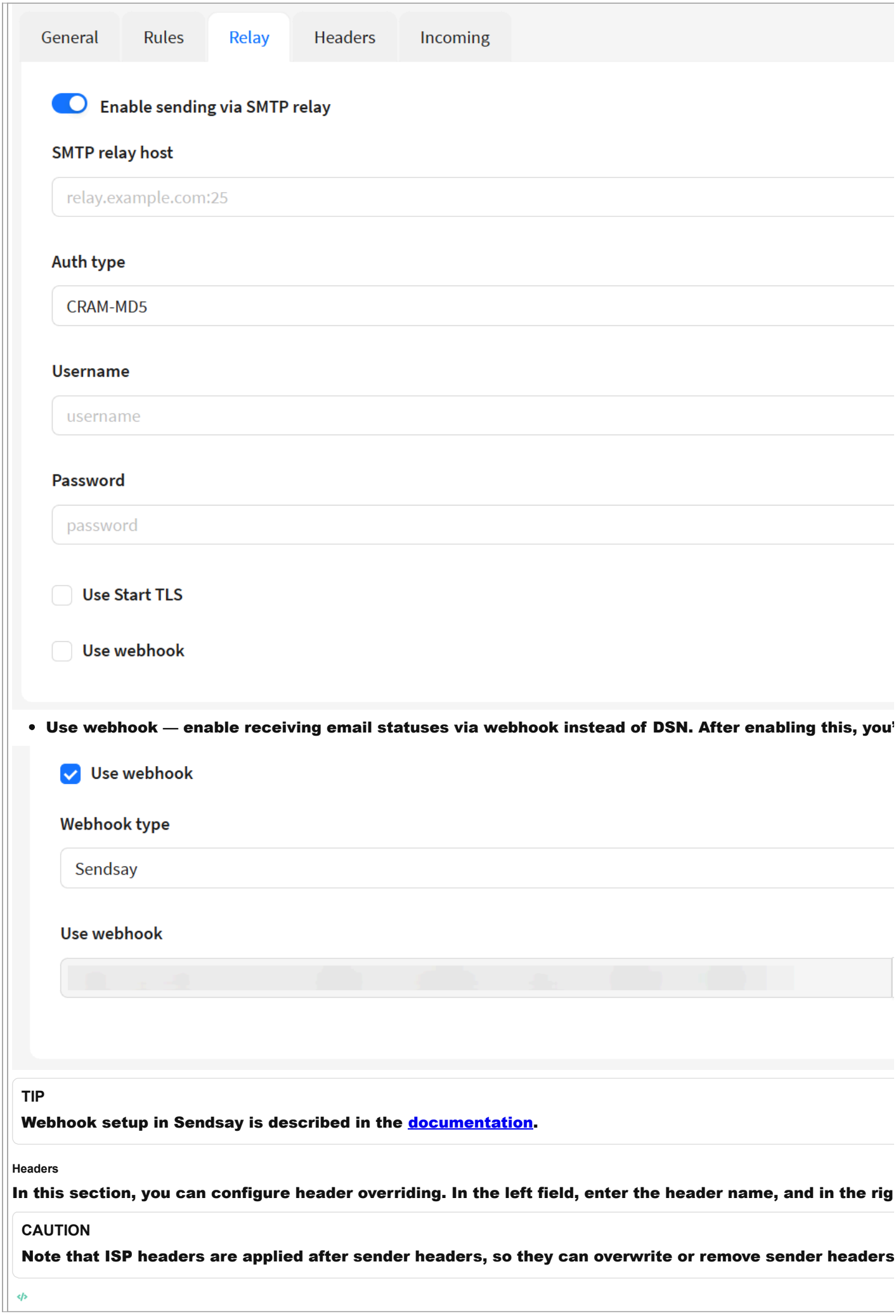
The Add ISPs strategy button adds special delivery rules from sender IP addresses to selected ISPs.

SMTP Relay

The AKMTA sender can deliver messages not directly to ISPs but via an SMTP relay as an additional forwa

To connect to the relay, specify the following:

- **SMTP relay host — relay network address and port.**
- **SMTP auth user — username for relay server authorization.**
- **SMTP auth password — password for relay server authorization.**
- **SMTP auth type — relay authentication protocol. Available options:**
  - **Without authorization**
  - **Text Plain**
  - **Login**
  - **CRAM-MD5**
- **Start TLS — enable connection encryption.**



- **Use webhook — enable receiving email statuses via webhook instead of DSN. After enabling this, you'**

TIP

Webhook setup in Sendsay is described in the documentation.

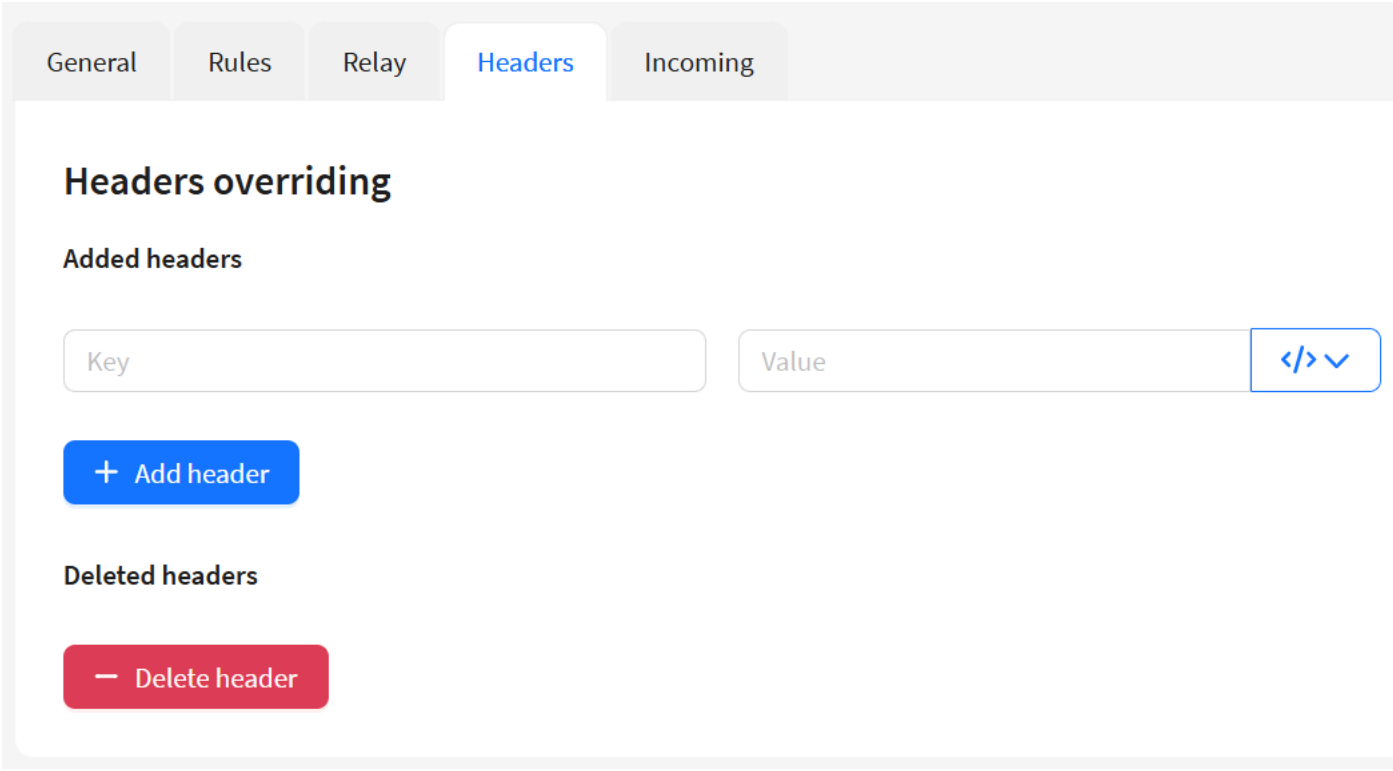
Headers

In this section, you can configure header overriding. In the left field, enter the header name, and in the rig

CAUTION

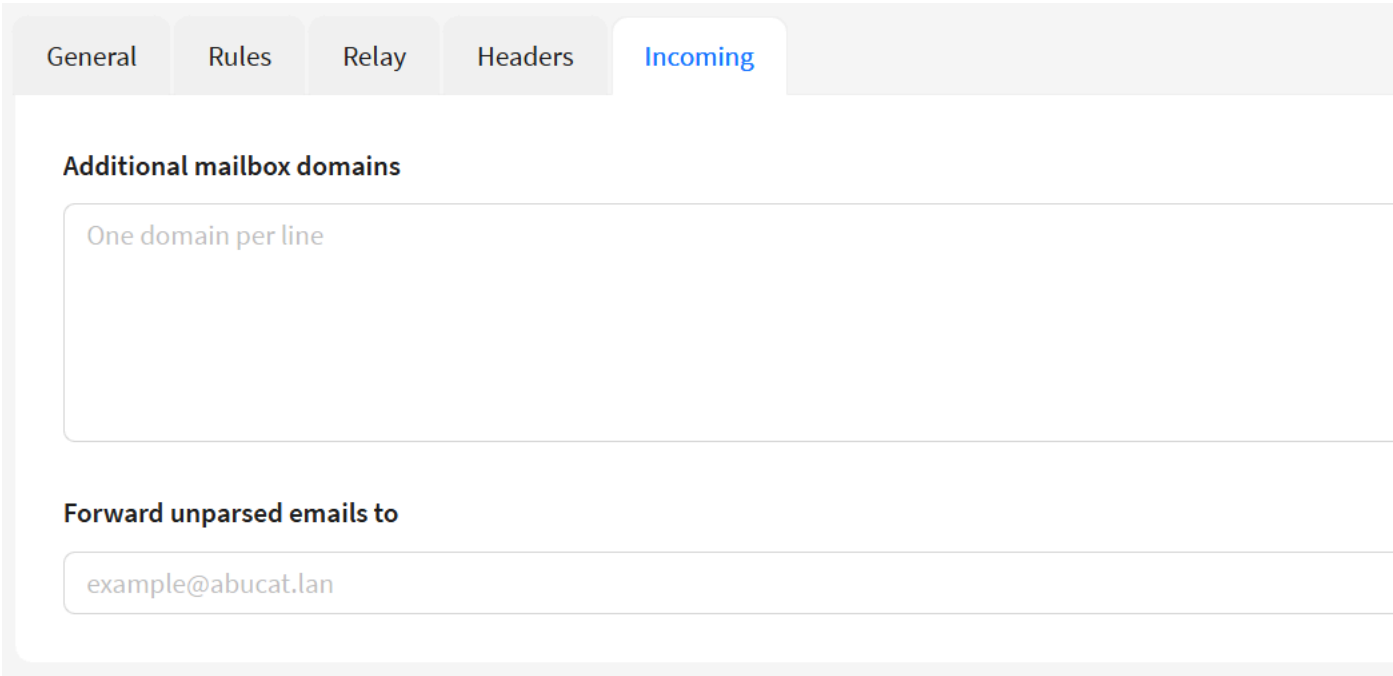
Note that ISP headers are applied after sender headers, so they can overwrite or remove sender headers

— you can use variables in headers. Select the desired variable from the dropdown menu.
Mark for deletion — enabling this allows you to remove the specified header from the email.



Incoming

In the Incoming section, you can specify additional mail domains and email addresses for unparsed emails
Additional mailbox domains — specify additional domains for which your server should accept and proces
Forward unparsed emails to — email address for messages the server failed to process correctly.

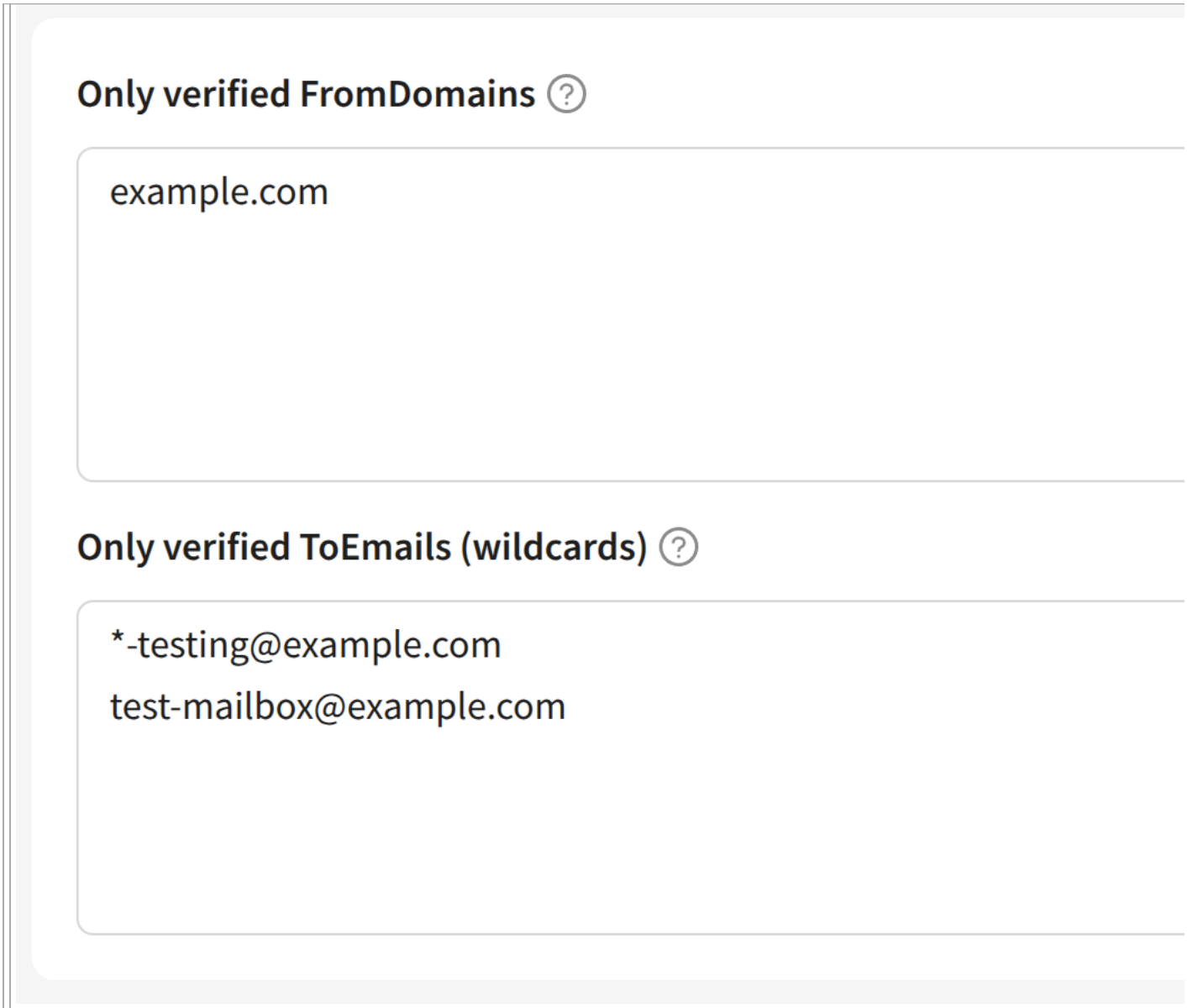


Sending to allowed addresses

The Allowed only on specified FROM domains and Allowed only on specified TO emails options restrict se

- **Allowed only on specified FROM domains — sending will only occur to addresses with the specified Fr**
- **Allowed only on specified TO emails — sending will only occur to the specified email addresses. You c**

When sending, the domain is checked first; then, if the domain matches, the specific recipient address is



Connecting to virtual sender

After creating and configuring a sender, you need to add it to a virtual sender through the platform's user
You can learn more about virtual senders in this article.

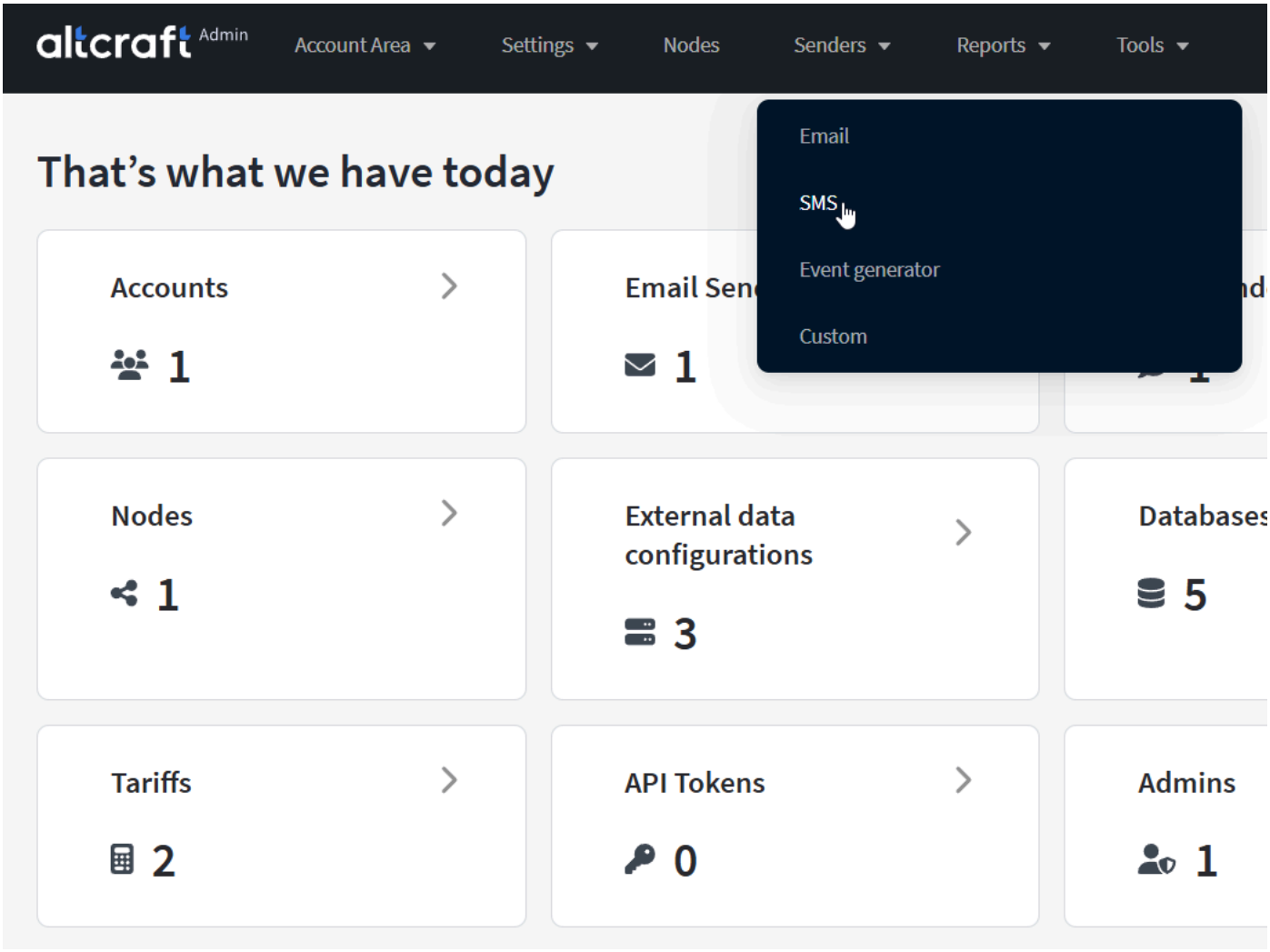
SMS

Description

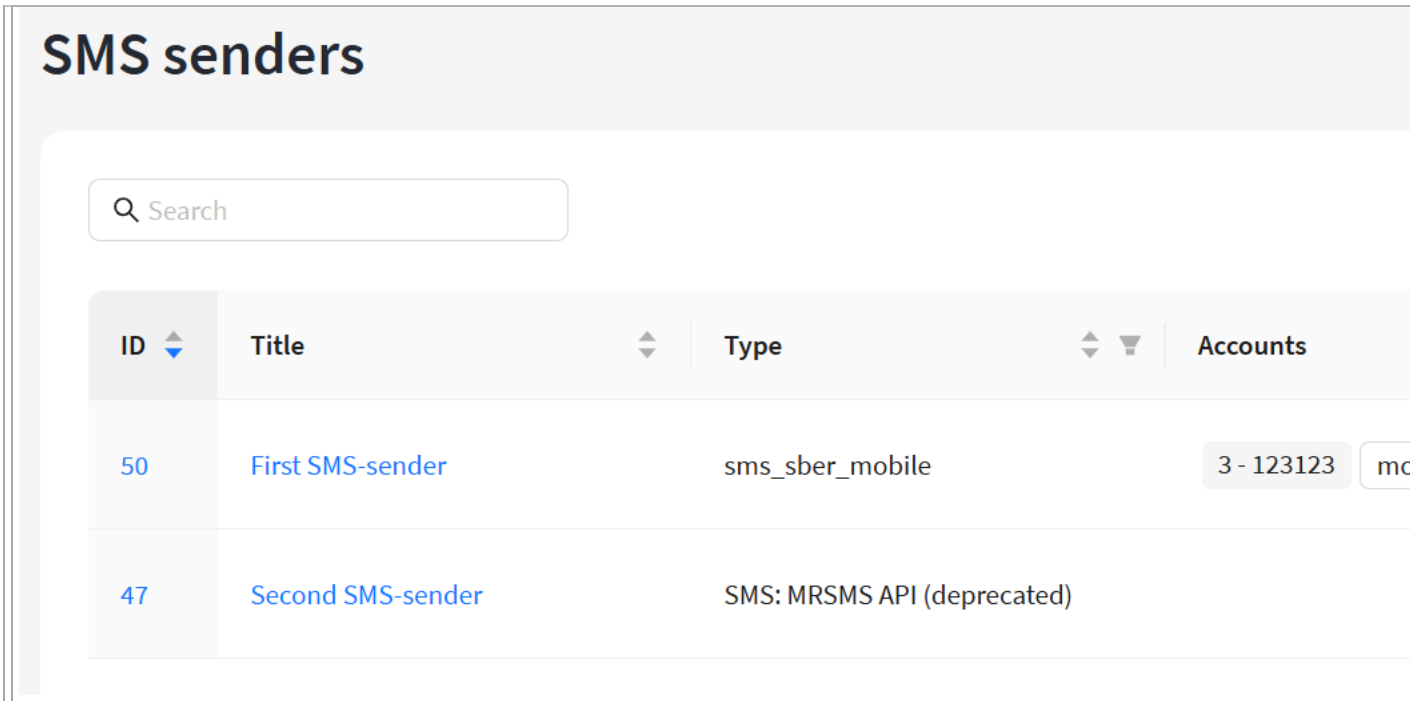
Sender in Altcraft is a configuration of the message delivery service. Senders are configured for email an

Creating, Editing, Deleting

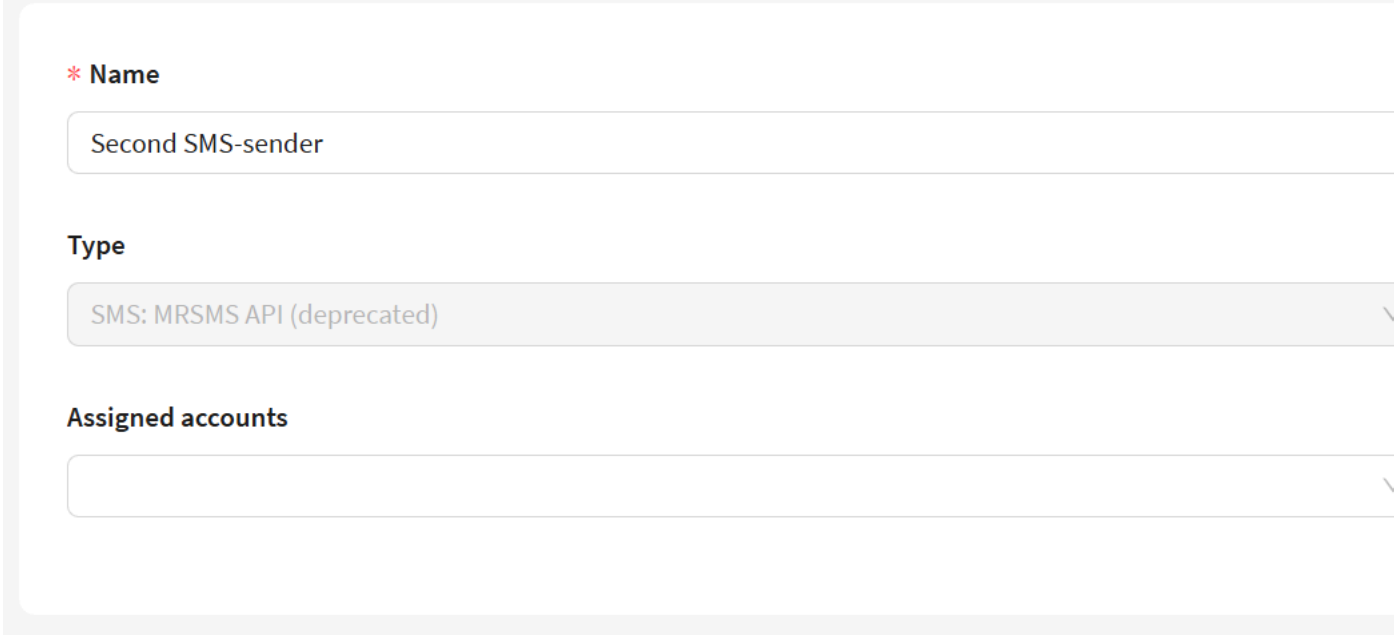
To manage SMS senders, in the main menu select Senders — SMS:



The opened sender list displays existing senders. On the right side of the table, there are edit and delete



For a new sender, in the Name field specify a unique name, and in the Assigned accounts menu select pla



SMS and telecom gateways

The following ready-made integrations are available for SMS:

- **SMSc.ru**
- **SMSimple.ru**
- **BulkSMS.com**
- **Devino.Online SMS**
- **SMSTraffic**
- **Infobip**
- **MTS**
- **MFMS EDNA**
- **Sky Media**
- **Play Mobile**
- **Rapporto**
- **SberMobile**
- **Stream telecom**
- **Intellin**
- **Seven Tech**

When selecting one of the SMS gateways as the delivery service, you must fill in the configuration fields (

BulkSMS.com

Documentation

- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**
- **Max SMS per message — limit on the number of SMS in one message.**
- **Await delivery (seconds) — waiting time after which the delivery status is checked from the gateway.**

Devino.Online SMS

Documentation

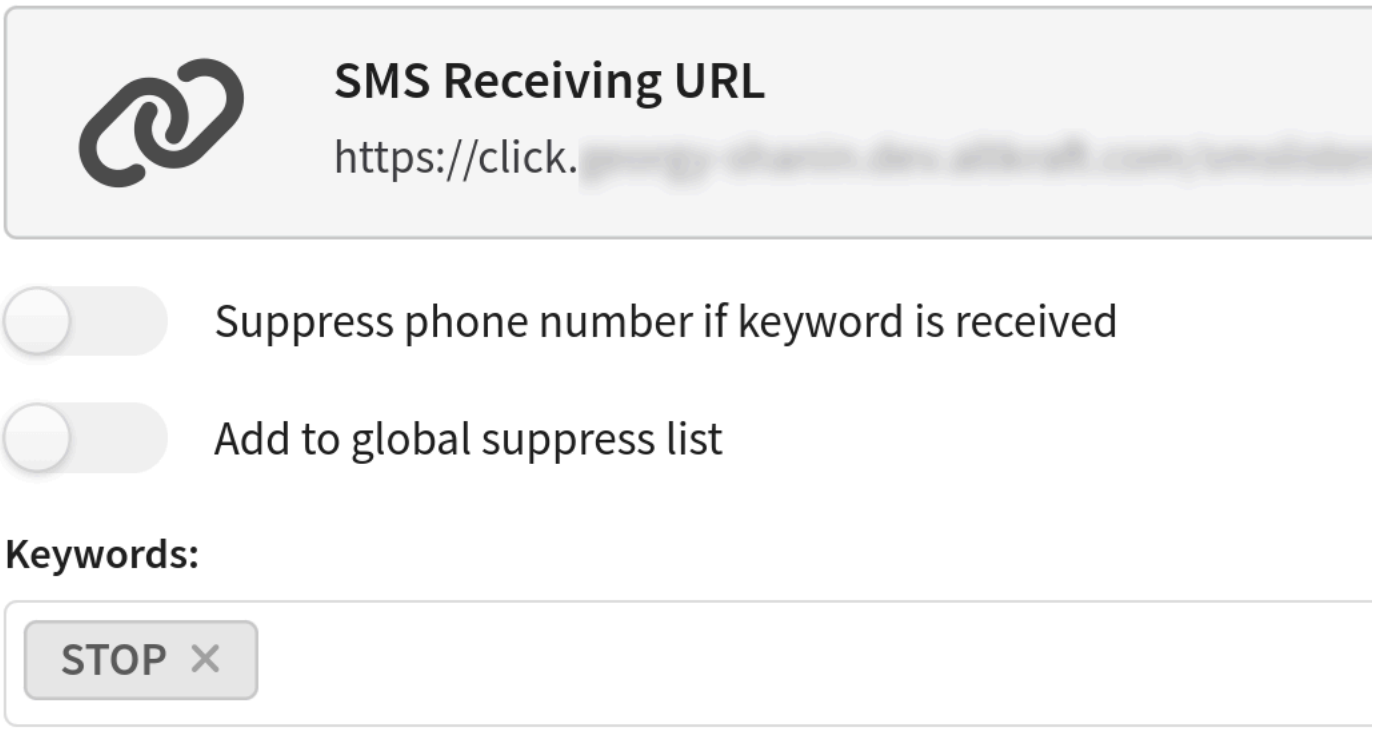
- **SMS service sending script/URL — server URL used to send SMS.**
- **API-key**
- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**

Infobip

Documentation

- **SMS service sending script/URL — server URL used to send SMS.**
- **API token — token for API authentication.**

Infobip provides a feature to stop sending messages to a phone number that replies with a stop word. Afte
After that, in the Keywords field, you can add stop words. When received, the platform will add the phone
The Add to global suppress list option also adds the unsubscribed phone number to the platform's global l



IDigital Direct API

Documentation

- **Token — token for API authorization (Basic Auth).**

Intellin

Documentation

- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**

MFMS EDNA

Documentation

- **SMS service URL — server URL used to send SMS.**

- **Login — client credentials for authorization in the SMS gateway.**
- **Login field name — name of the field in the request that contains the login.**
- **Password — client credentials for authorization in the SMS gateway.**
- **Password field name — name of the field in the request that contains the password.**

In some configurations, the login and password fields для EDNA могут иметь names different from `login` и are not specified, the platform will pass the standard names (`login` и `password`).

MTS

Documentation

- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**

Play Mobile

Documentation

- **User ID — user ID for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**

When integrating with the Play Mobile provider for sending SMS messages, it is necessary to consider the

```
PLAY_MOBILE_LIMIT_REQUESTS_PER_SEC: 49 // Maximum number of SMS send requests per second (default - 49)
```

Rapporto

Documentation

- **Service ID — service id for authorization.**
- **Password — client credentials for authorization in the SMS gateway.**
- **SMS service sending URL — server URL used to send SMS.**

To receive the statuses of sent SMS messages, you need to provide the Rapporto service with a URL for r
URL: `http://{trk_domain}/smslistener/listen_type/sms_rapporto`, где `{trk_domain}` — this is the tracking domain.
You can specify this URL in your Rapporto personal account or send it to their support team by email:

SMS Traffic

Documentation

- **SMS service sending script — The URL or path to the script that will trigger the SMS sending when ac**
- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**

SMSc.ru

Documentation

- **Login — client credentials for authorization in the SMS gateway.**
- **Password or MD5 hash — password or its MD5 hash for authorization in the SMS gateway.**
- **Await delivery (seconds) — the time allowed for delivery from the gateway. Once this period elapses, t**

SMSSimple.ru

Documentation

- **User — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**
- **Original identifie — The original message identifier.**
- **Max SMS per message number — limit on the number of SMS in one message.**

SberMobile

Documentation

- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**
- **Client ID — a unique client identifier in the system.**

Seven Tech

Documentation

- **SMS service sending URL — server URL used to send SMS.**
- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**
- **Shorten links in messages — when the option is enabled, links in the message will be automatically s**

Sky Media

Documentation

- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**

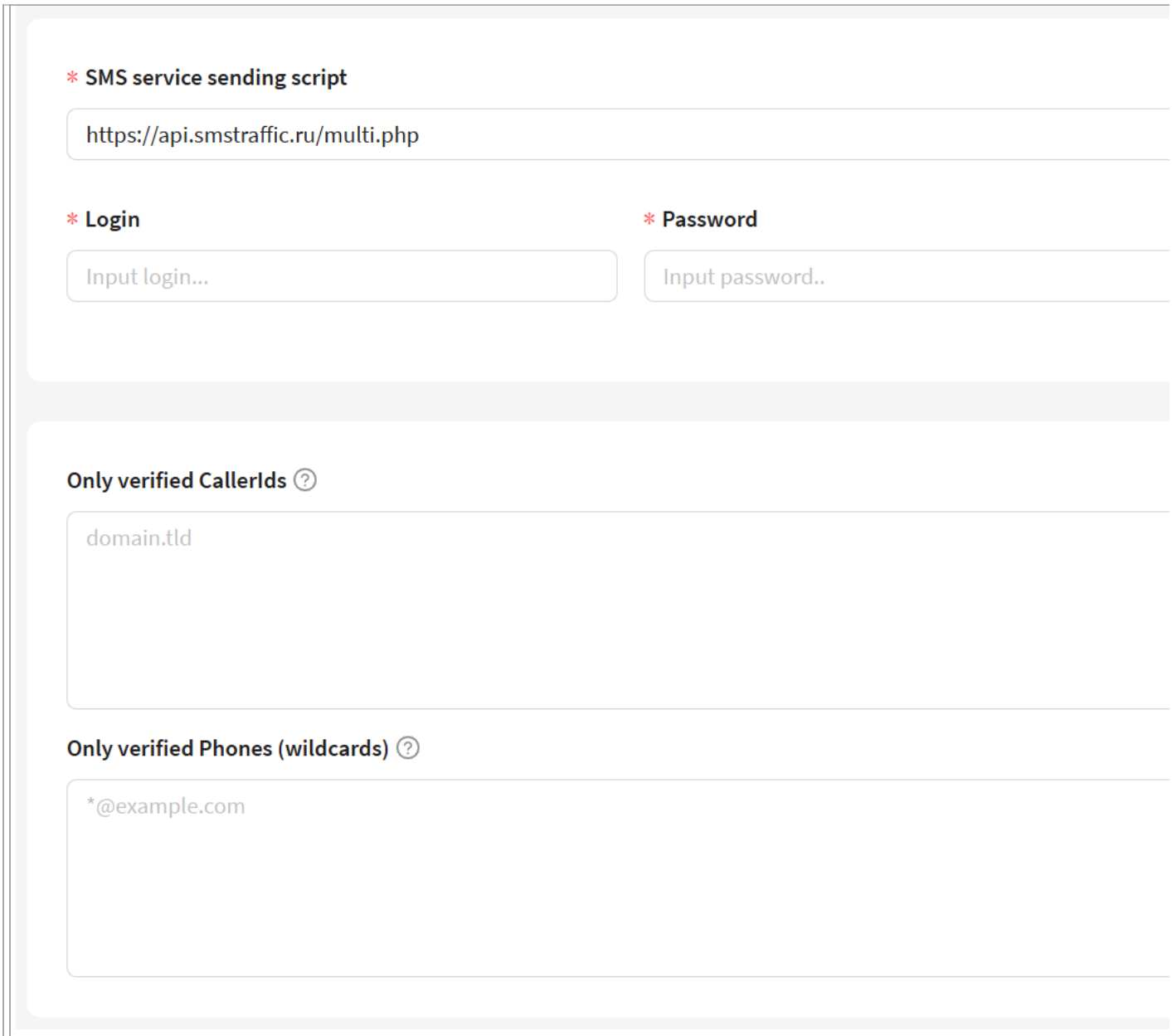
Stream Telecom

Documentation

- **Login — client credentials for authorization in the SMS gateway.**
- **Password — client credentials for authorization in the SMS gateway.**

Also can be configured:

- **Only verified CallerIDs — list of sender names purchased from the SMS provider (specified if you need**
- **Only verified Phones — list of numbers to which sending is allowed (used for test gateways).**



Connecting to virtual sender

After creating and configuring a sender, you need to add it to a virtual sender through the platform's user
You can learn more about virtual senders in this article.

Event generator

Description

Event generator is a sender designed to simulate user activity and events in the system. It can be used to

Main capabilities

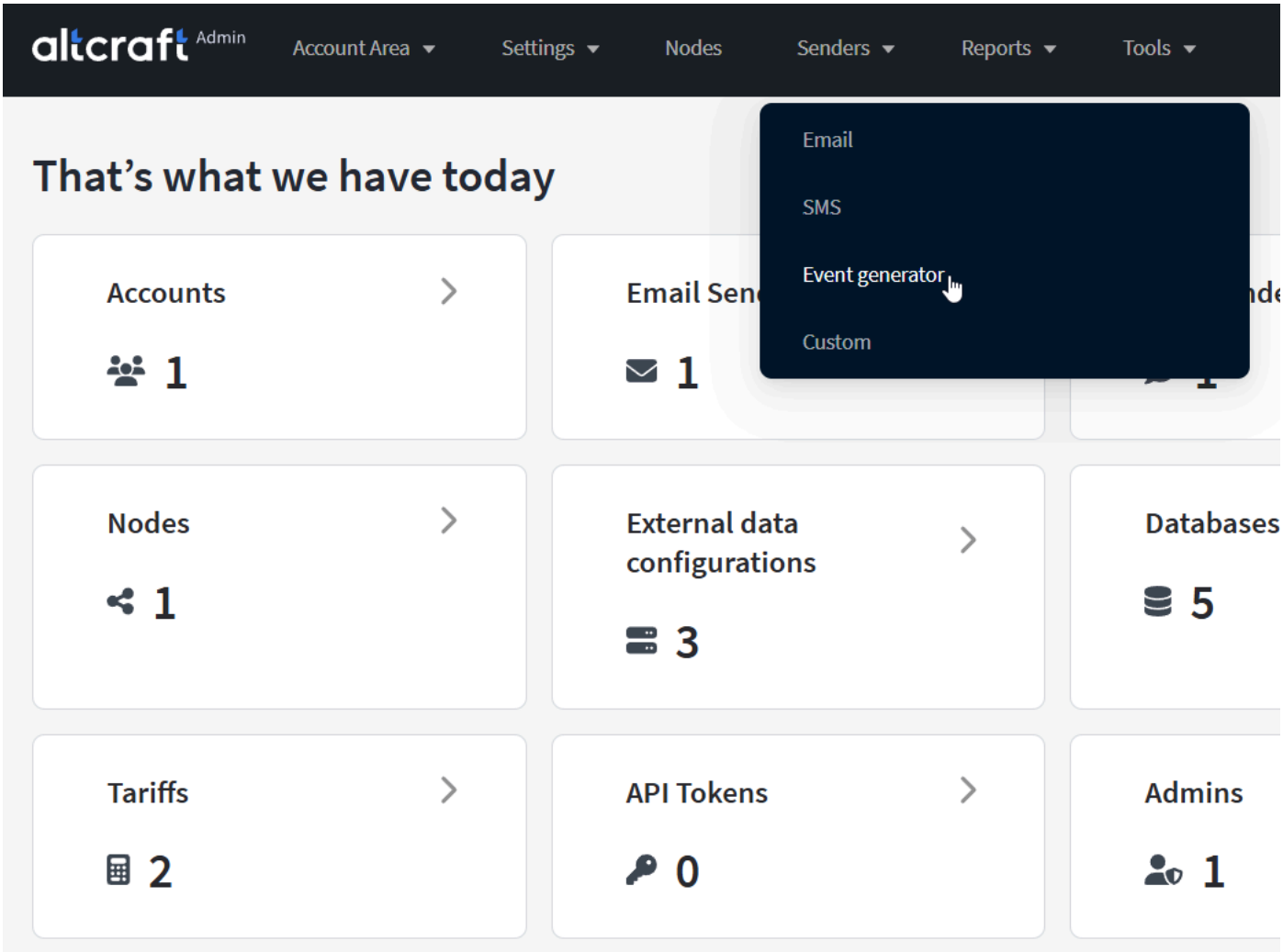
- **Creating events that simulate user behavior.**
- **Generating and recording goal achievements, views, and other metrics.**

Using the event generator, you can emulate:

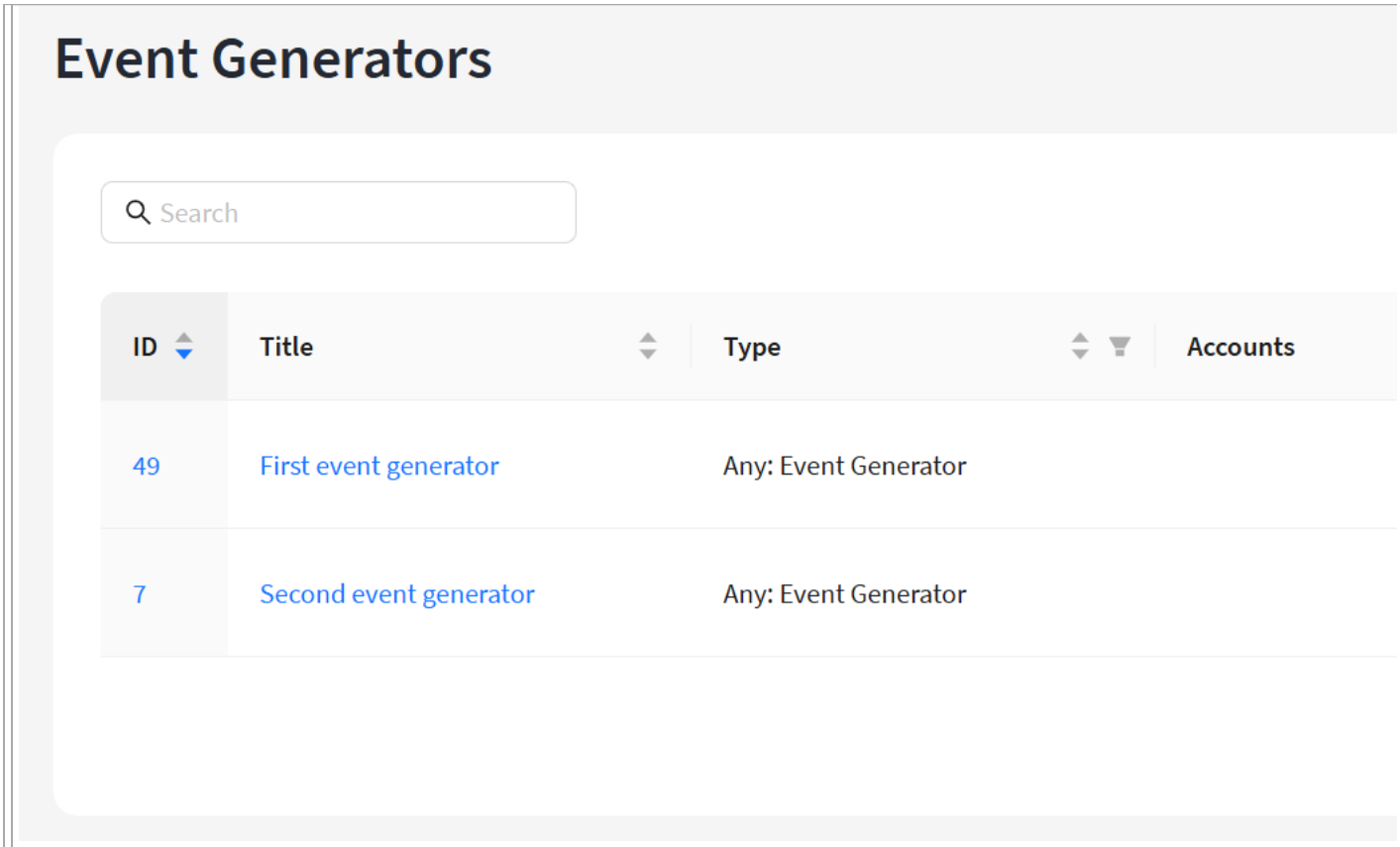
- **User IP address**
- **User Agent**
- **Soft-bounce text**

Creating, Editing, Deleting

To manage Event generator senders, in the main menu select Senders — Event generator



The opened sender list displays existing senders. On the right side of the table, there are edit and delete



Connecting to virtual sender

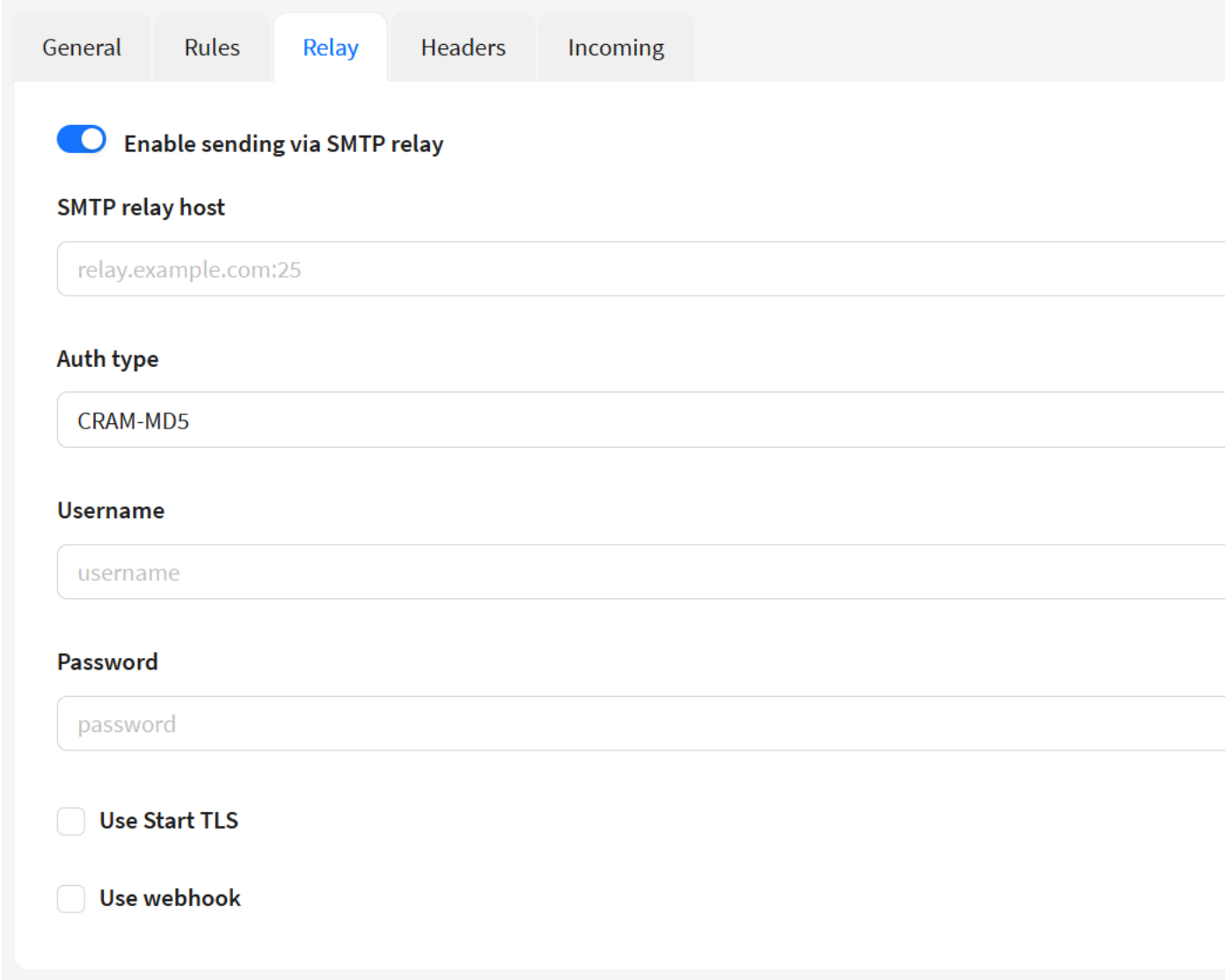
After creating and configuring a sender, you need to add it to a virtual sender through the platform's user
You can learn more about virtual senders in this article.

Integration with Sendsay

Configuring the Sendsay SMTP relay

Sender configuration

- **Go to Senders —> Email**
- **Create a new sender or edit an existing one**
- **In the Type field, select Email: AKMTA**
- **Enable the Send via SMTP relay option**



Connection parameters

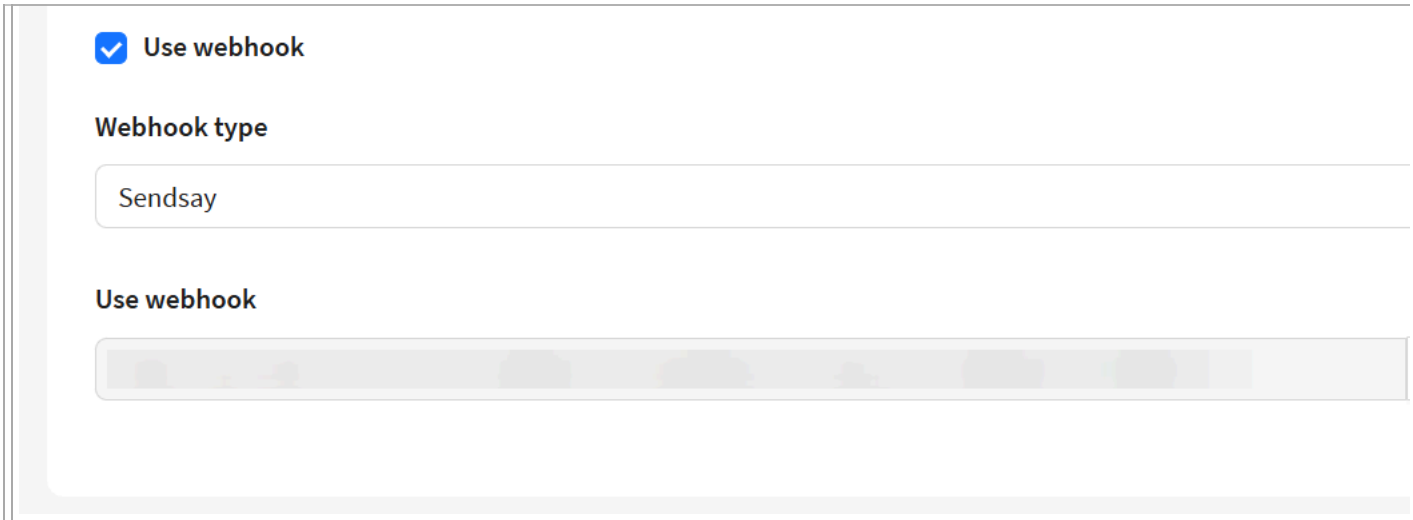
To connect to the Sendsay relay, specify the following data:

- **SMTP relay host:** `smtp.gate.sendsay.ru:587`
- **Username: Your login in the Sendsay system**
- **Password: Your password in the Sendsay system**
- **SMTP auth type: Login**
- **Use Start TLS: Enable**

Webhook configuration

To receive delivery statuses, enable the Use webhook option:

- **Select a gateway from the drop-down list**
- **Copy the generated webhook URL**
- **Configure webhook reception in your Sendsay account according to the official documentation**



Checking relay operation

Analyzing email headers

To make sure that the message went through the Sendsay relay:

- **Open the source code of the email (the “View source code” menu)**
- **Find the lines in the headers that start with** `Received:`
- **The presence of the following entry will confirm that the relay is working:**

```
Received: from gatoNNN.sndsy.ru (gatoNNN.sndsy.ru [185.138.182.XXX])
    by your-server.com with ESMTPS id XXXXXXXXXX
```

```
## Specifics of statistics when working with Sendsay
### Comparison of key indicators

| Event | Altcraft | Sendsay |
|---------|----------|---------|
| **Sends** | Counts each send | Counts each send |
| **Opens** | All opens (including repeats) | All opens (including repeats) |
| **[Unique opens](/user-guide/reports-and-analytics/channels-report/#channel-metrics)** | First open | Not supported |
| **Deliveries** | Successful deliveries | Successful deliveries |
| **Clicks** | All clicks (including repeat clicks) | All clicks (including repeat clicks) |
| **[Unique clicks](/user-guide/reports-and-analytics/channels-report/#channel-metrics)** | First click | Not supported |
| **Unsubscribes** | Special unsubscribe event | Counted as regular clicks |

:::caution
**Key difference:** Clicks on unsubscribe links in Altcraft are recorded as unsubscribe events, while in Sendsay they are counted as regular cl
:::

:::note
**Altcraft reports** are recommended as the primary source of analytics, as the platform provides more detailed statistics and correctly identi
:::

## Connecting to virtual sender
After creating and configuring a sender, you need to add it to a **virtual sender** through the platform's user interface. To do this, open the

You can learn more about virtual senders in [this article](/en/user-guide/settings/9405938).
```

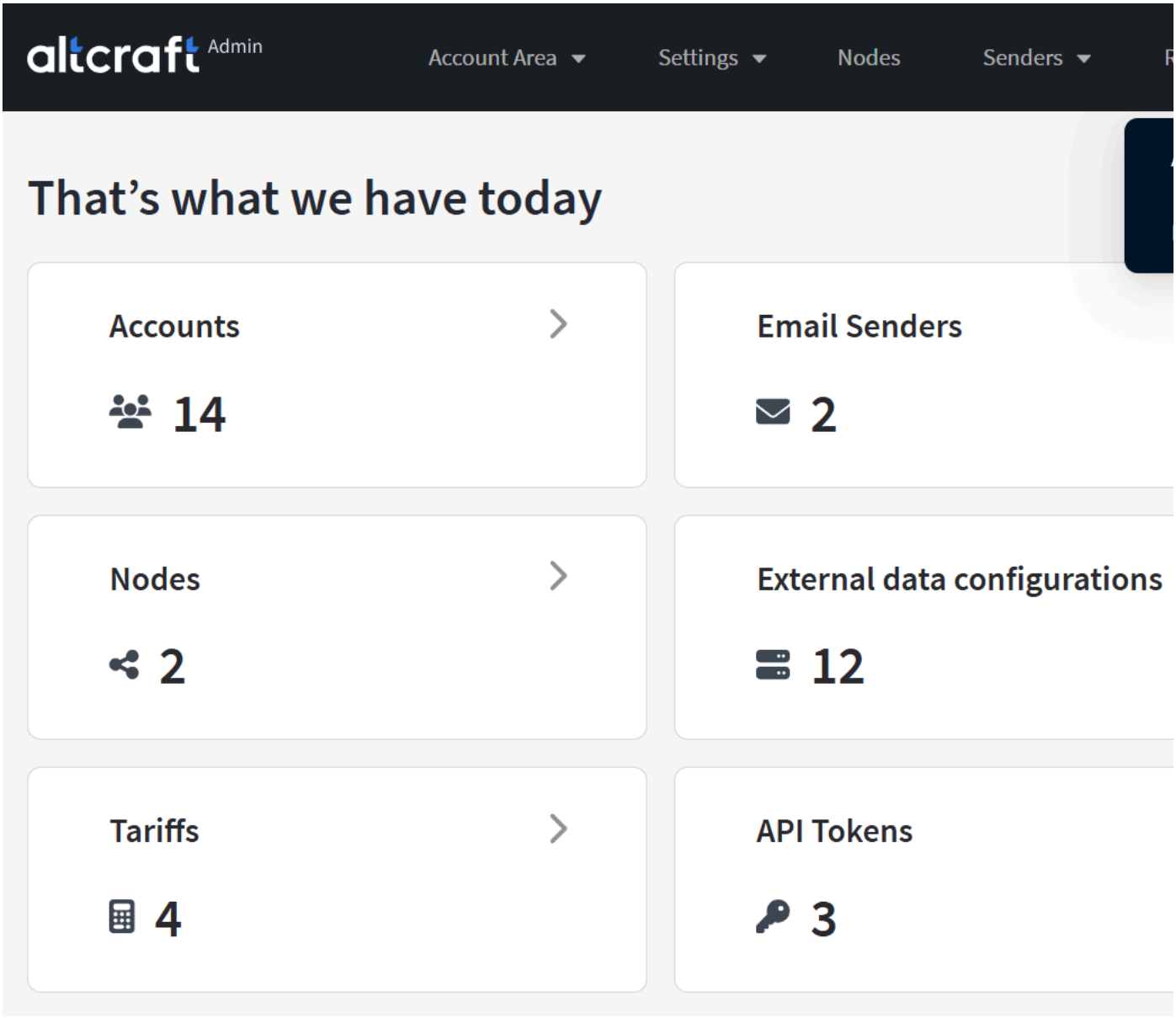
Audit Journal

Description

The Audit Journal is an important security component of the Altcraft platform. It collects information abo

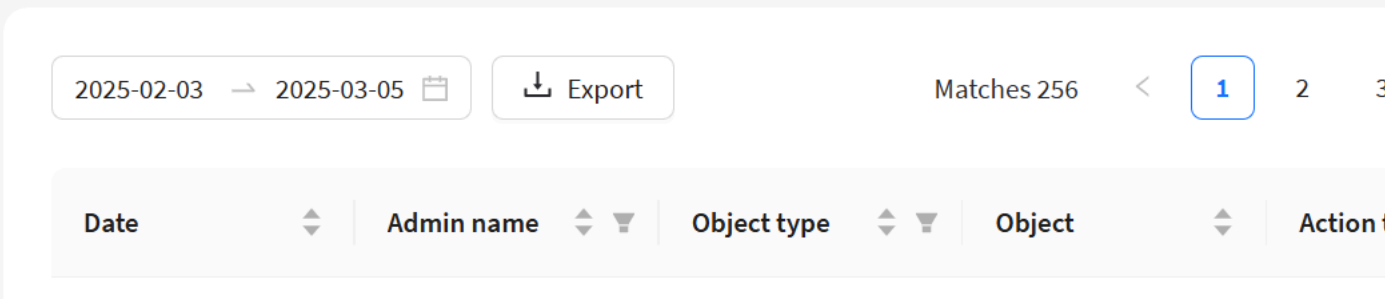
Working with the Audit Journal

To start working with the audit journal, in the main menu select Reports —> Audit journal.



Administrator actions can be filtered by several parameters. From left to right, top to bottom:

- **Time period for the report.**
- **Type of action performed by the administrator.**
- **Name of the administrator account.**
- **Type of object on which the actions were performed.**
- **ID and account name — only for account objects such as users or virtual senders.**
- **ID and name of the specific object in the Administrative Panel.**



After selecting the desired audit parameters, click Apply to generate the report. To display information ab
The report displays the date of the action, administrator account, IP address, action type, and the ID and

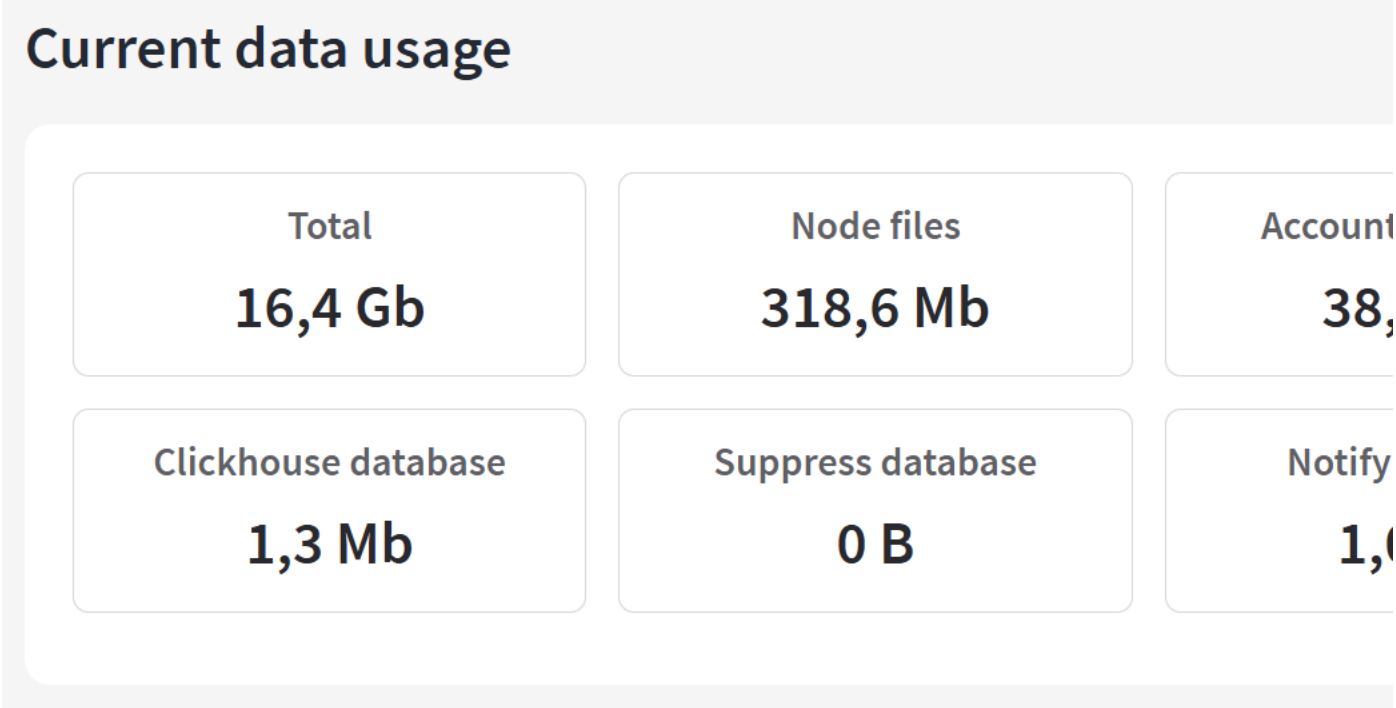
Date	Admin name	Object type	Object	Action typ
2025-02-03 14:17:09	admin	System	-	Login
2025-02-03 16:04:47	admin	System	-	Login
2025-02-03 18:41:53	admin	Sender	50 New sender	Create
2025-02-03 18:45:14	admin	Sender	50 New sender	Save
2025-02-03 18:45:19	admin	Sender	50 New sender	Save
2025-02-04 12:39:28	admin	System	-	Login

Data Usage

Description

Data Usage is a section designed to view the volume of data used in the platform. It contains information

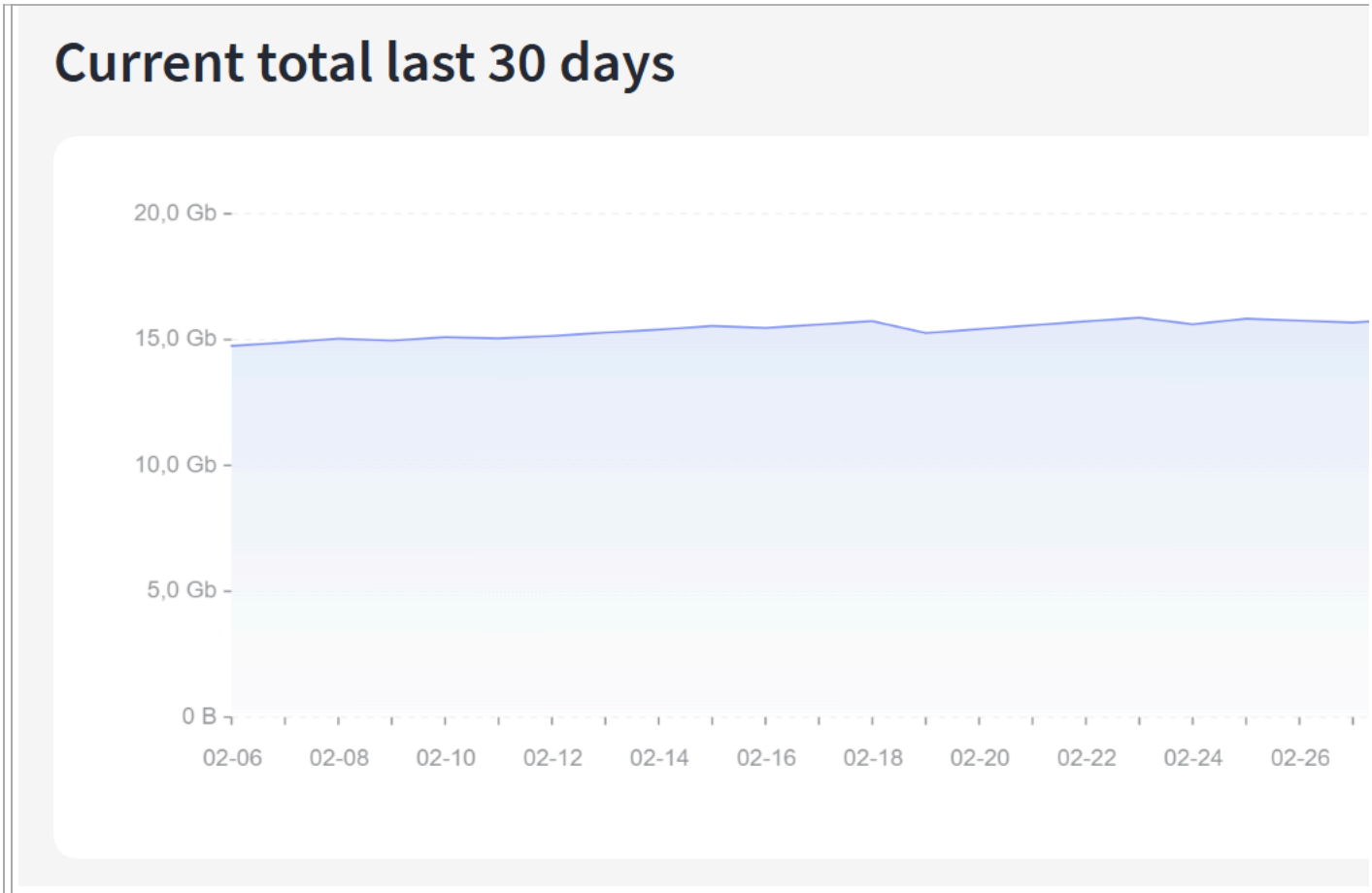
Current Data Usage Statistics



This block displays a summary of the current volume of used data, including:

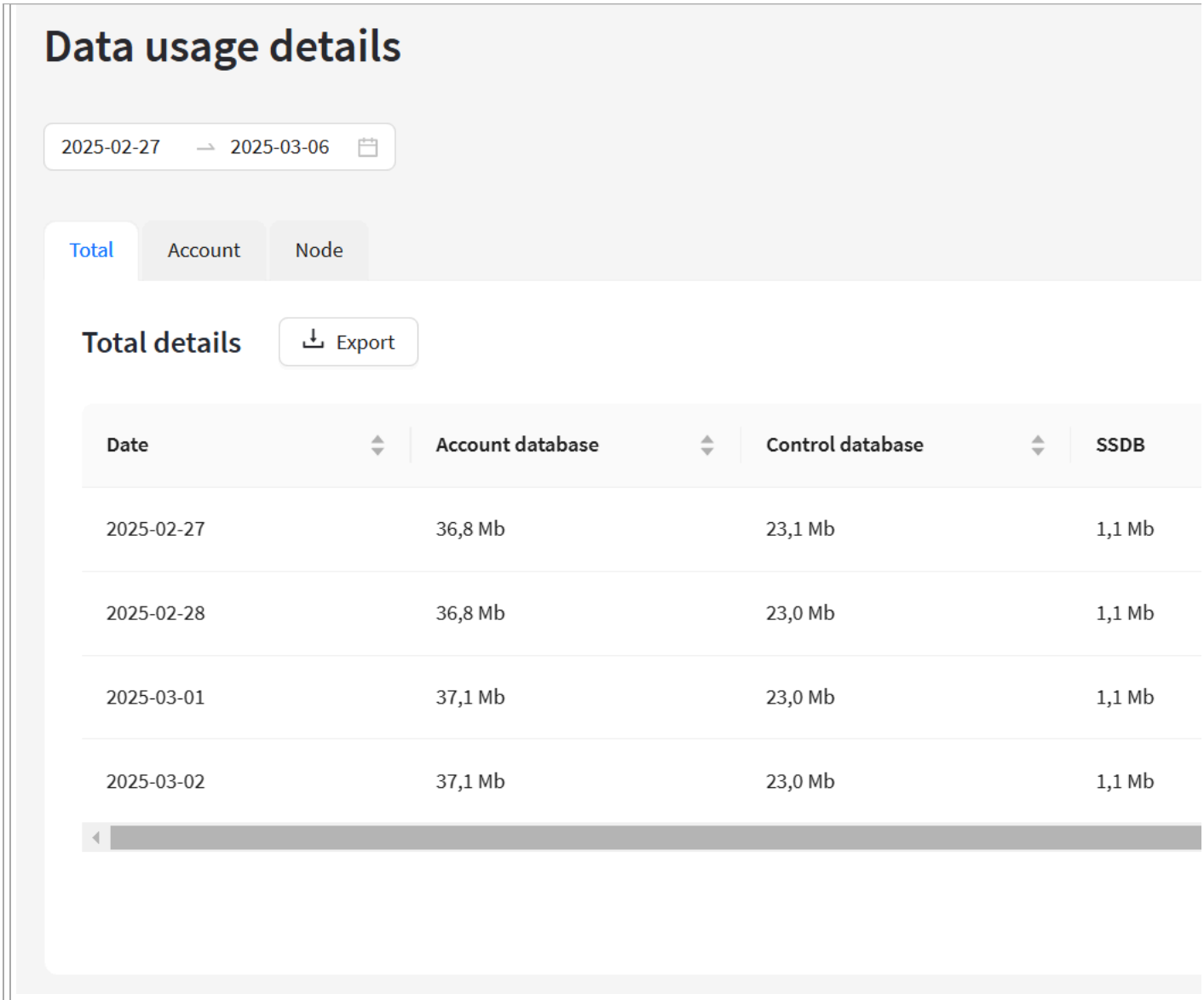
- **Total – total volume of data used by the system.**
- **Node files – volume of node files stored in the system.**
- **Account databases – volume of data related to account databases.**
- **Control database – volume of the control database.**
- **Clickhouse database – volume of the ClickHouse database.**
- **Suppress database – volume of the suppression database (in this case, 0 B means it is not used).**
- **Notify database – volume of the Notify database.**
- **AKD files – volume of AKD files (archived or cached data).**

Data Volume Change Graph for 30 Days



The graph shows the dynamics of changes in data volume over the past month. It displays the total volum

Data Usage Details



The Data Usage Details section provides detailed information about the distribution of data across differe

- **Date – the day for which the data is shown.**
- **Account database – volume of account database data.**
- **Control database – volume of the control database.**
- **SSDB – volume of the SSDB database.**
- **AKD Files – volume of AKD files.**
- **Node Files – volume of node files.**

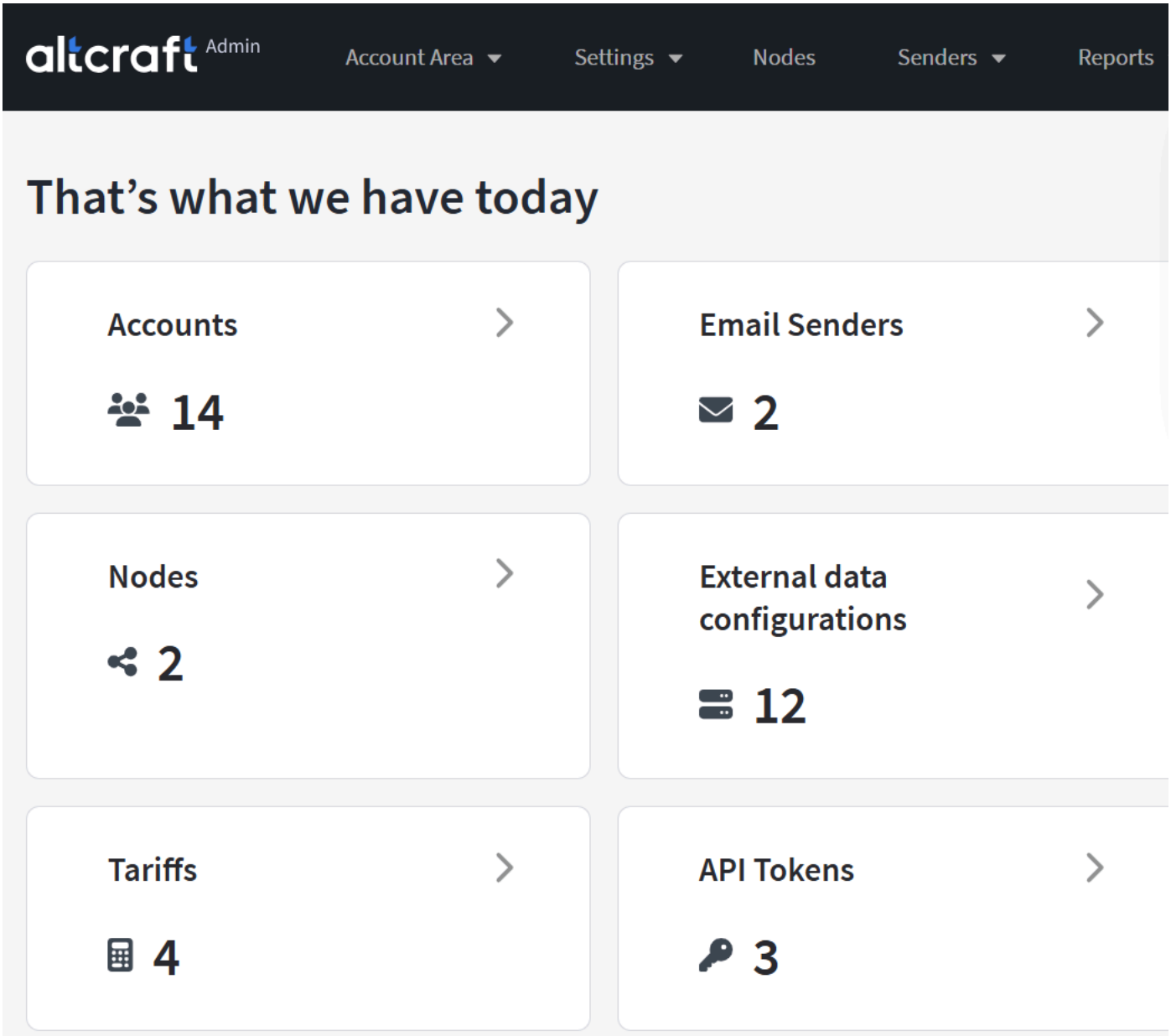
ARF decoder

Description

ARF Decoder is a tool used to decode messages in the ARF (Abuse Report Format), providing information

Connection Setup

In the main menu, select Tools — ARF decoder:



Paste the message with service headers into the corresponding field and click Decode. The result will co

- **Email — the recipient's email address.**
- **Message ID — the identifier of the message sent to the recipient.**
- **SenderID — the identifier of the sender that sent the message.**

* **Paste the FBL header or the whole complaint email**

```
X-Spam-Status: Yes, score=12.516 required=6.6 tests=[BAYES_50=0.8,
    DMARC_FAIL_REJECT=9, FSL_HELO_NON_FQDN_1=0.001, HELO_NO_DOMAIN=0.465,
    HTML_FONT_TINY_NORDNS=0.001, HTML_MESSAGE=0.001, MPART_ALT_DIFF=0.79,
    RDNS_NONE=0.793, SPF_SOFTFAIL=0.665] autolearn=no autolearn_force=no
Received: from mail.altcraft.com ([127.0.0.1])
```

Decode

Result

Email :	altcraft-documentation@example.com
MessageID :	fH8cmmbi13SIFFbalHnkOMXAAAQMQUQEQIQEAABARAQGqOTVbnkOMXZ…
SenderID :	1

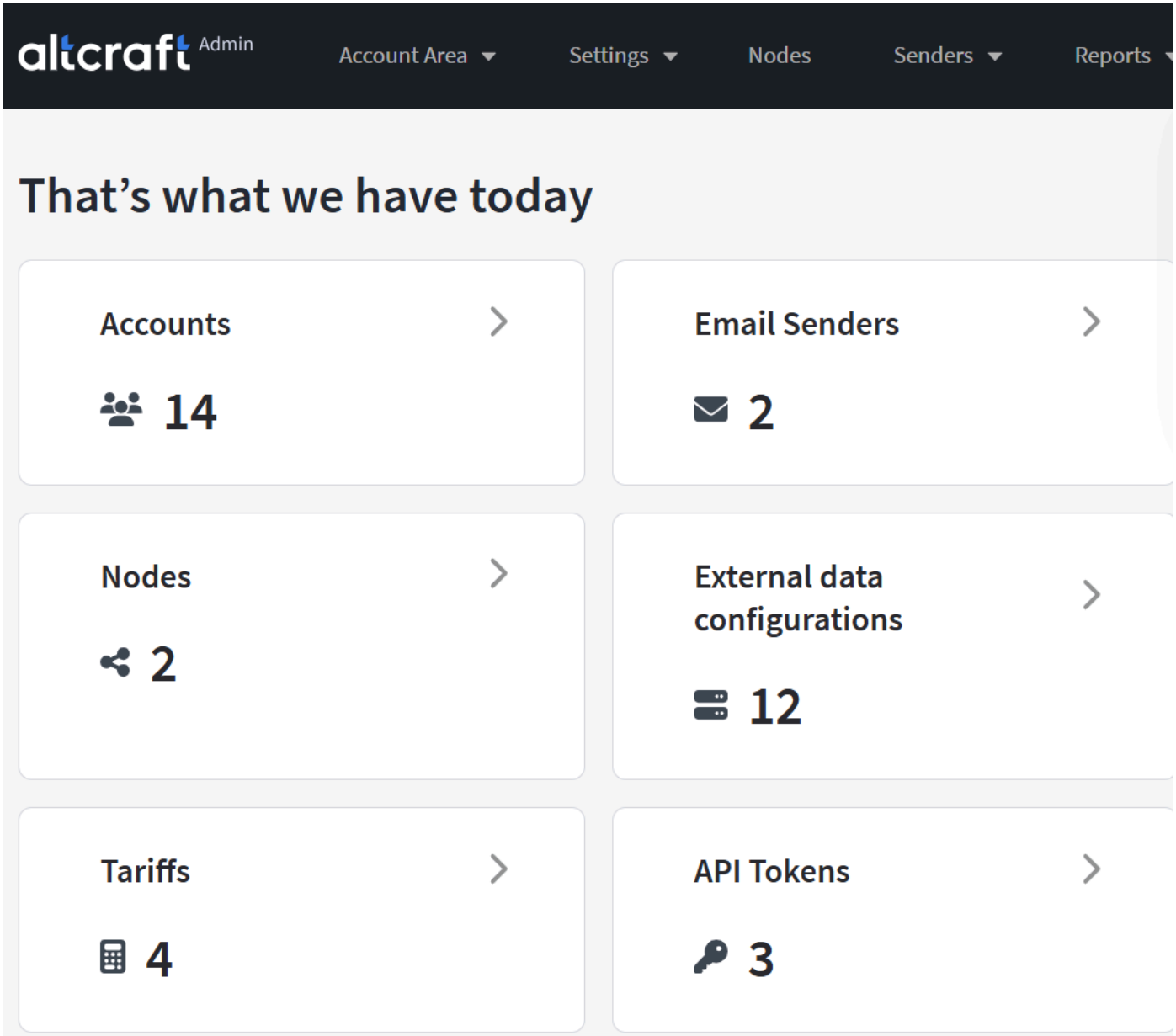
URL decoder

Description

URL Decoder is a tool used to decode links containing information about the recipient and click tracking i

Connection Setup

The links you add to message templates redirect recipients to the tracking server to collect click tracking



Paste the tracking link into the corresponding field and click Decode. The result will contain:

- **List ID — the recipient's database identifier.**
- **Account ID — the platform account identifier.**
- **Lead key — the profile ID (unique identifier) of the recipient of the message with the link.**

/10A9903/dSPdZq8Dcw0alCMzYQXvgwgUfCNp_97xmSnjf9mWdrk5DUDr-_j-kxjpq2mVGy-K1DiypwL7WhOp9ylbK1mfXFN32GwJ19i1GWIF-HnaSKyiYn15Yy3qLlvcqgtf2e_vEtF6LgT5a3gJ3RMOjtXyYRGKb6khKKNvHHFqqTc_gJsGtMbwHRVIKIYAA2z-1vXU4Duap4e3Z2ZYIbnn45G5vQ

Profile ID:	123443545asd657ghf7
Account ID:	3 - New account
Database ID:	2
Campaign ID:	5
Template:	1
Action:	click link
Message ID:	fAAAAAAAAAAAAAAAAAAAAAAAAAAQMQUQEQIAAAEBEBAao5NVueQ4xdmmidVXJEZufQnTfxIZmM0Y...
Resource ID:	1
Channel:	email channel
Launch ID:	4_6_4ZKDyhqwiMc
URL:	https://templating.net/?utm_source=email&utm_medium=&utm_campaign=5&utm_term=new...
Time stamp:	Invalid date
Profile:	No result

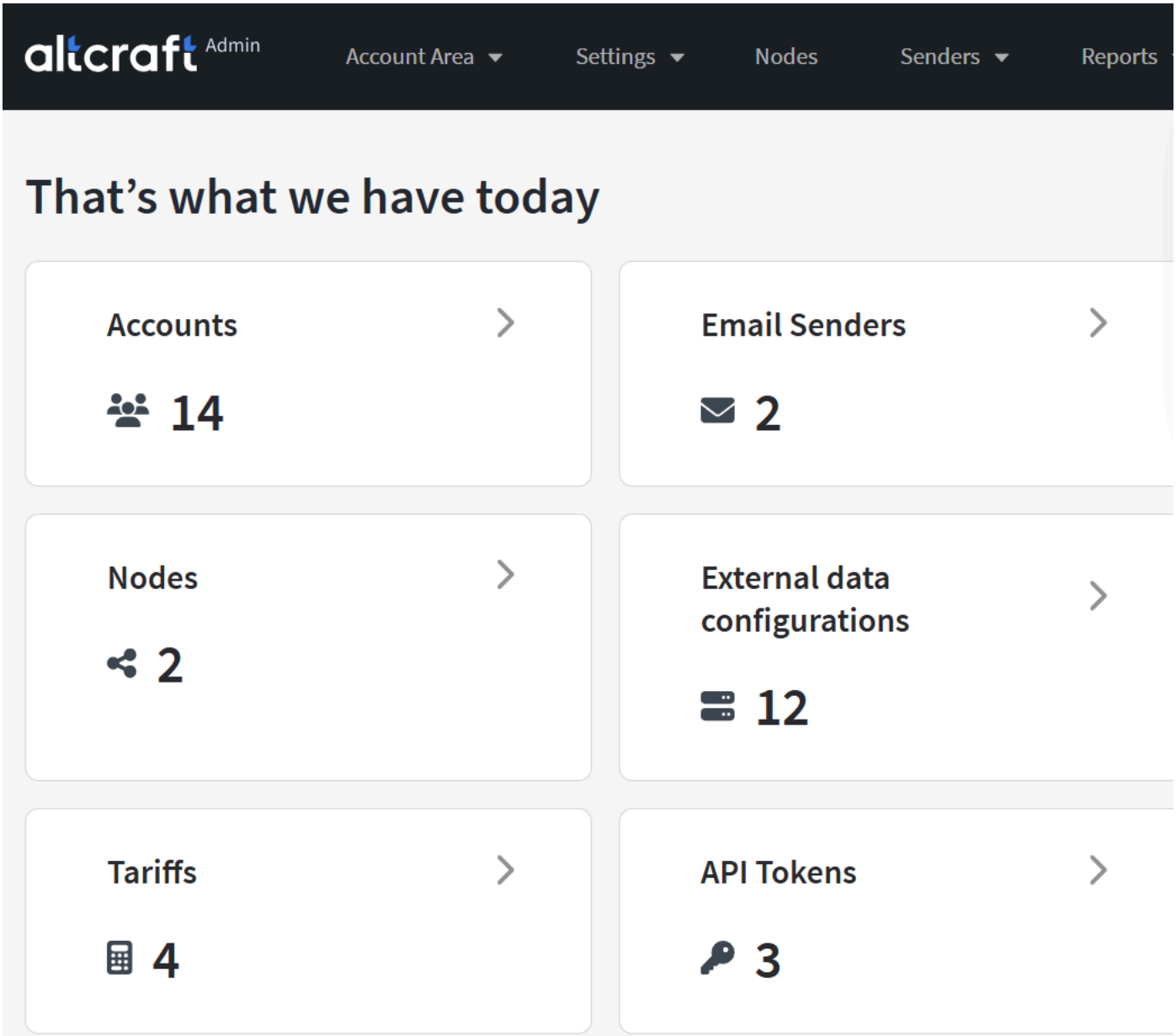
SMID decoder

Description

SMID Decoder is a tool used to decode message identifiers (SMID, FSMID, ESMID, or USMID) sent to speci

Connection Setup

By using the identifier of a message sent to a particular recipient, you can retrieve detailed information al



Paste the message identifier (SMID, FSMID, ESMID, or USMID) into the corresponding field and click Deco

- **date and time of sending,**
- **mailing and template,**
- **recipient database and segment,**
- **profile data,**
- **resource subscriptions, and other details.**

Clicking an entity name will open the corresponding page in the Altcraft User Interface or Admin Panel.

NOTE

If the web version of the message is available, clicking Saved will open a preview and provide a permane

Message ID decoder

* Paste SMID, FSMID, ESMID or USMID here

fNZiyGcQc8uGNR4AMY57ykYAAAQEQMQEQEQEAABChAQCxSOMY57ykYZ6HOj1Tvqh_-6t5DZDJmMWM0ODmA

Decode

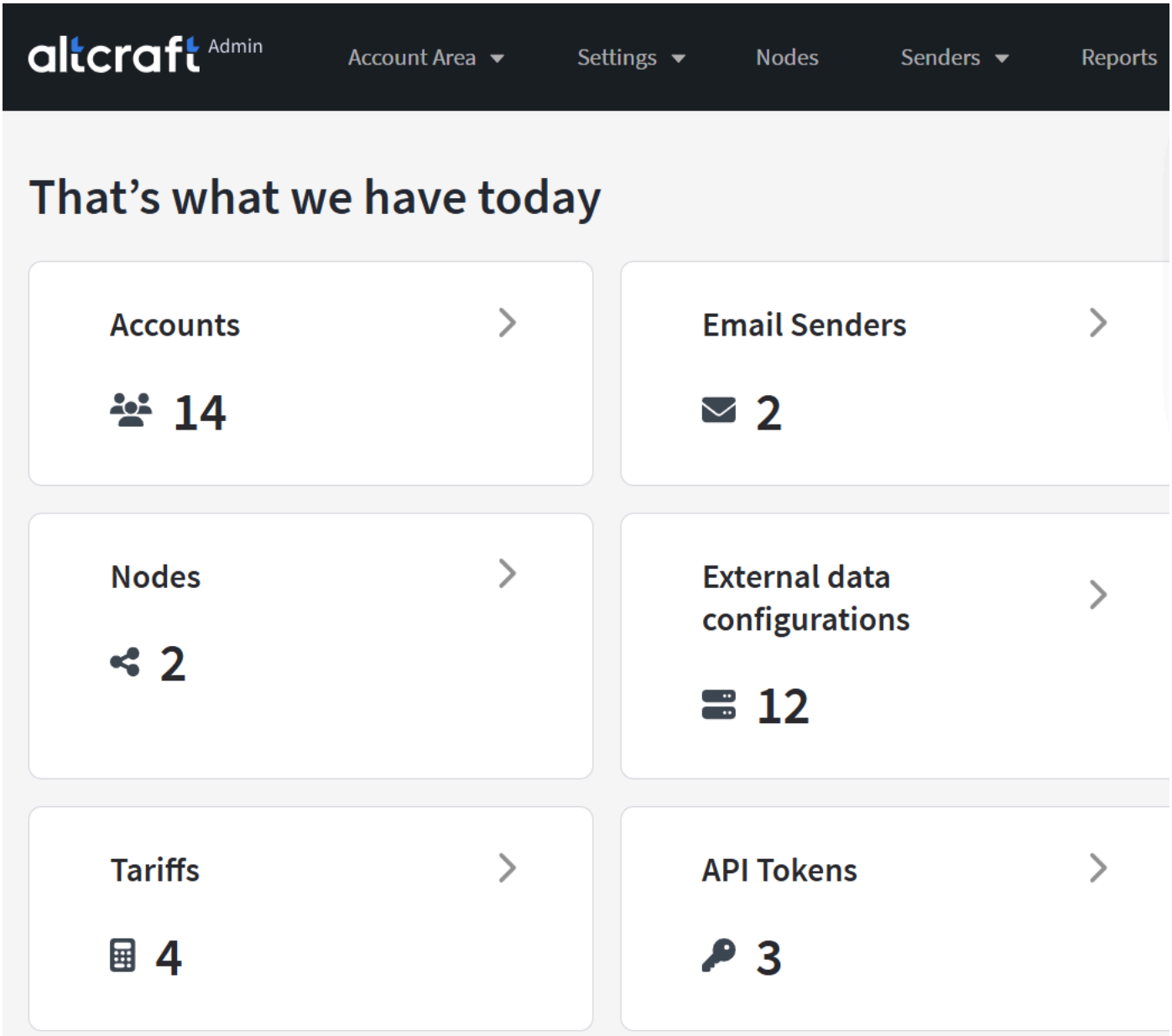
Send Date:	2025-03-05 05:42:42	
Launch Date:	2025-03-05 05:42:42	
Test Flag:	false	
Indirect Flag:	false	
Account:	1 - demo	
Campaign:	3 - Welcome email	
Template:	10 - Welcome template	
Series ID:	No result	
Sender:	1 - AKMTA	
Scenario:	No result	
Scenario element:	No result	
Database:	1 - Client database	
Segment:	No result	
Profile:	**Email:**	john@example.com
	IP:	198.16.74.224
	Identifier:	67a1ce8f54efaa1ffeeade43
	Registration date:	2025-02-04 11:23:43

License

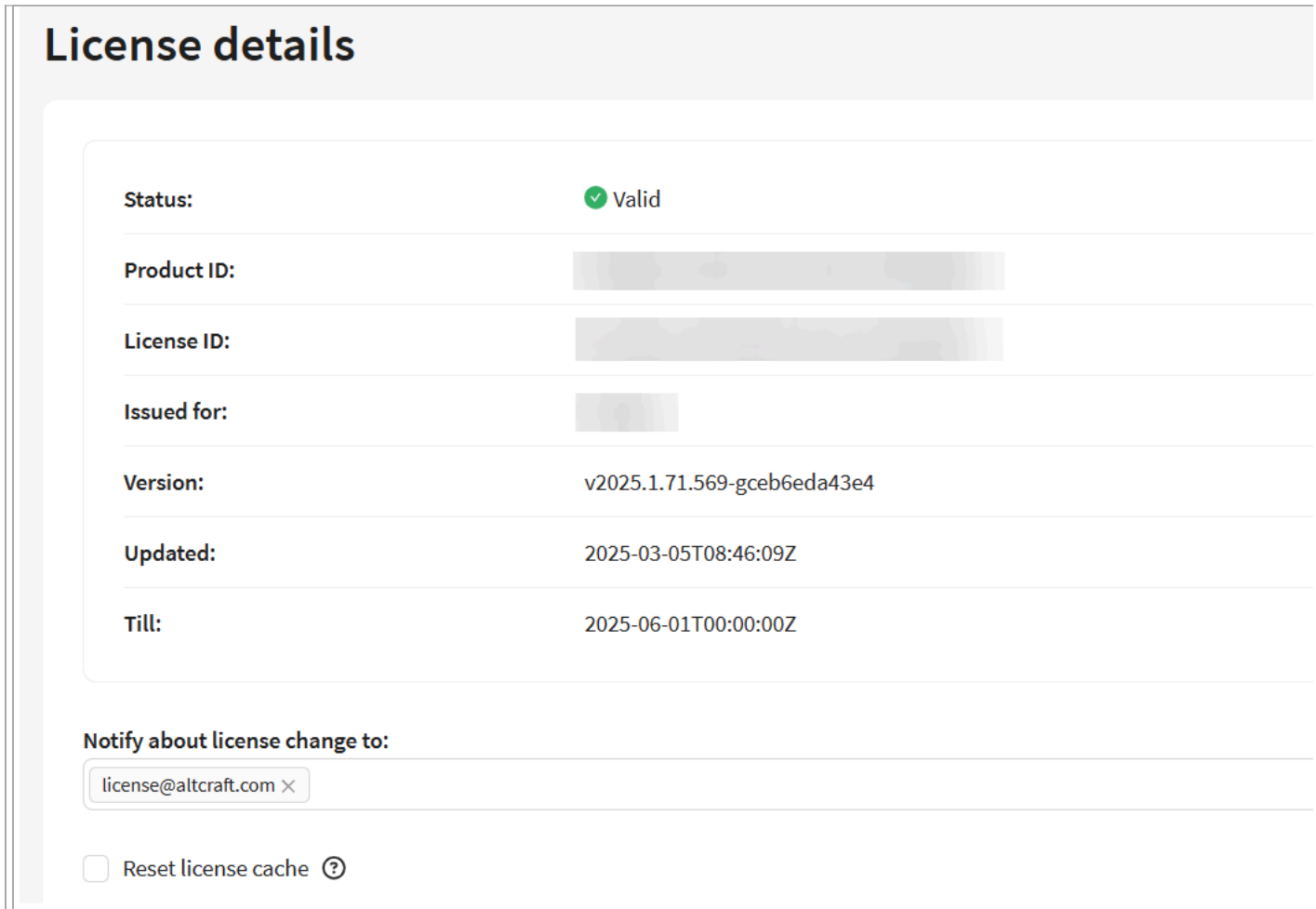
Description

License is an official document that grants rights to use the platform and defines the conditions of its use automatically updating license information via the aktool utility.

Connection Setup



In the License section, you can view detailed information about the platform version, license, and its expi



In the Notify about license changes to field, you can specify the email address where notifications will be

INFO

You can manually check or update license information using the `aktool` utility:

```
# ./aktool help license
This command is used for license management automation.

Usage:
  aktool license [command]

Available Commands:
  cache-clean Clean license cache
  info        Show cached license information
  update      Update license information
```

Manual installation

Preparation

Introduction

Before starting the installation, it is strongly recommended to familiarize yourself with the architecture, t

> **TIP**
>
> **All further actions require `root` user privileges.**

> **INFO**
>
> **Instructions for configuring replications can be found on the Cluster Configuration, Replication page.**

> **WARNING**
>
> **For passwords and secret combinations, a set of characters is used - abcdefghijklmnopqrstuvwxyzABCD**

Archive Loading

All clients are provided with a compressed archive with the extension `.tar.gz`. Unpack the archive and mo

```
{
tar xzf <archive_name>.tar.gz
mv ak /opt/akd
}
```

Checking the Server Time

Time synchronization between servers is critical: discrepancies can lead to malfunction and incorrect sta

```
timedatectl status
```

Setting System Limits

Add the following limits in the `/etc/security/limits.conf` file:

```
*               -       nofile          1048576
root            -       nofile          1048576
root            -       memlock         unlimited
```

Then reconnect to the server and verify that the limits have been updated:

```
ulimit -n
```

Installing Dependencies

Installate the followibg dependencies:

```
{
  apt update
  apt install --no-install-recommends xvfb cutycapt unzip gcc g++ libsnappy-dev make autoconf libpcre3-dev libgomp1
}
```

Installation of MongoDB or Percona Server for MongoDB

MongoDB is used to store system settings and user data.

> **INFO**
>
> **Additional materials:**
>
> - **Install MongoDB Community Edition on Ubuntu**
> - **Configuration file options**
> - **Database configuration**

The platform supports two MongoDB options:

- **Standard MongoDB Community Edition — the basic version from MongoDB Inc.**
- **Percona Server for MongoDB — a compatible replacement from Percona with additional features: enha**

> TIP: Recommendation
>
> **For production environments, Percona Server for MongoDB is recommended as a more secure and perfor**

Choose one of the installation options:

MongoDB Community Edition

Go to the download page, select version 8.0, your OS version, and the `.tgz` archive. Copy the link and dow

```
wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu2404-8.0.10.tgz
```

Extract the archive to `/usr/local` and create a symbolic link:

```
{
  tar xzf mongodb-linux-x86_64-ubuntu2404-8.0.10.tgz -C /usr/local
  ln -s /usr/local/mongodb-linux-x86_64-ubuntu2404-8.0.10 /usr/local/mongodb
}
```

Install the dependencies:

```
{
  apt update
  apt install -y gnupg curl
}
```

Create the user, group, and working directories:

```
{
  groupadd mongodb
  useradd -r -g mongodb -s /bin/false mongodb
  mkdir -p /var/{lib,log}/mongodb
  chown -R mongodb:mongodb /var/{lib,log}/mongodb
}
```

Create the configuration file `/etc/mongod.conf`:

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

storage:
  dbPath: /var/lib/mongodb
  wiredTiger:
    engineConfig:
      directoryForIndexes: true

systemLog:
  destination: file
  logAppend: true
  logRotate: reopen
  path: /var/log/mongodb/mongod.log

net:
  port: 27017
  bindIp: 127.0.0.1
```

Create the file `/etc/systemd/system/mongod.service`:

```
[Unit]
Description=MongoDB Database Server
Documentation=https://docs.mongodb.org/manual
After=network.target

[Service]
User=mongodb
Group=mongodb
ExecStart=/usr/local/mongodb/bin/mongod --config /etc/mongod.conf
PIDFile=/run/mongod.pid
Restart=always
LimitFSIZE=infinity
LimitCPU=infinity
LimitAS=infinity
LimitNOFILE=256000
LimitNPROC=256000
LimitMEMLOCK=infinity
TasksMax=infinity
TasksAccounting=false

[Install]
WantedBy=multi-user.target
```

Enable the service at startup, start it, and check its status:

```
{
  systemctl enable mongod.service
  systemctl start mongod.service
  systemctl status mongod.service
}
```

Download and install MongoDB Shell.

Percona Server for MongoDB

Add the official Percona repository and update the package list:

```
{
  wget https://repo.percona.com/apt/percona-release_latest.generic_all.deb
  dpkg -i percona-release_latest.generic_all.deb
  percona-release setup psmdb80
  apt update
}
```

Install Percona Server for MongoDB. The package will create the required directories (`/var/lib/mongodb` , `/va`

```
{
  apt install -y percona-server-mongodb
}
```

Installing a specific version of Percona Server for MongoDB

If you need to install a specific release instead of the latest available version, first list the available versi

```
apt-cache madison percona-server-mongodb
```

Example output:

```
percona-server-mongodb | 8.0.10-5.jammy | http://repo.percona.com/psmdb-80/apt jammy/main amd64 Packages
percona-server-mongodb | 8.0.9-4.jammy  | http://repo.percona.com/psmdb-80/apt jammy/main amd64 Packages
```

Then install the required version by specifying the package version explicitly:

```
apt install \
  percona-server-mongodb=<version> \
  percona-server-mongodb-server=<version> \
  percona-server-mongodb-shell=<version> \
  percona-server-mongodb-mongos=<version> \
  percona-server-mongodb-tools=<version>
```

Create or edit the configuration file `/etc/mongod.conf` :

```
storage:
  dbPath: /var/lib/mongodb
  engine: wiredTiger
  wiredTiger:
    engineConfig:
      directoryForIndexes: true

systemLog:
  destination: file
  logAppend: true
  logRotate: reopen
  path: /var/log/mongodb/mongod.log

net:
  port: 27017
  bindIp: 127.0.0.1
```

Percona Server for MongoDB supports data-at-rest encryption. To enable it, set the `enableEncryption` param

Encryption can only be enabled on an empty database. If the database already contains data, use a separ

Generate a key file:

```
openssl rand -base64 32 > /etc/mongodb-encryption-keyfile
chown mongod:mongod /etc/mongodb-encryption-keyfile
chmod 600 /etc/mongodb-encryption-keyfile
```

The key file must contain a 32-byte base64 string. Access permissions to the file must be restricted.

Edit `/etc/mongod.conf` :

```
storage:
  dbPath: /var/lib/mongodb
  engine: wiredTiger
  wiredTiger:
    engineConfig:
      directoryForIndexes: true

systemLog:
  destination: file
  logAppend: true
  logRotate: reopen
  path: /var/log/mongodb/mongod.log

net:
  port: 27017
  bindIp: 127.0.0.1

security:
  enableEncryption: true
  encryptionKeyFile: /etc/mongodb-encryption-keyfile
```

Using HashiCorp Vault instead of a local key file

HashiCorp Vault is the preferred way to store the master key. Percona Server for MongoDB uses KV Secre

In this case, use the following configuration instead of the `encryptionKeyFile` parameter:

```
security:
  enableEncryption: true
  vault:
    serverName: VAULT_HOST
    port: 8200
    tokenFile: /etc/mongodb-vault-token
    secret: secret/data/mongodb/node1
    serverCAFile: /etc/ssl/certs/vault-ca.crt
```

The `tokenFile` must contain the Vault token without any extra lines or parameters.

```
chown mongod:mongod /etc/mongodb-vault-token
chmod 600 /etc/mongodb-vault-token
```

Restart MongoDB and check the service status:

```
{
  systemctl enable mongod.service
  systemctl restart mongod.service
  systemctl status mongod.service
}
```

Verify that encryption is enabled:

```
mongosh --eval "db.serverCmdLineOpts()"
```

If a local key file is used, verify the file permissions:

```
ls -la /etc/mongodb-encryption-keyfile
```

If HashiCorp Vault is used, verify the Vault parameters in the runtime configuration:

```
mongosh --eval "db.serverCmdLineOpts()" | grep -A 10 vault
```

Connect to the database using `mongosh` and create a user:

```
mongosh
```

Run the following commands in the MongoDB shell:

```
use admin;
db.createUser({
    "user": "altcraft",
    "pwd": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "roles": [{
        "role": "root",
        "db": "admin"
    }]
})
exit
```

> CAUTION: Important
>
> **The password `abcdefghijklmnopqrstuvwxyzABCDEF` is provided only as an example. Be sure to generate your ow**

Configuring the platform connection

Update the connection parameters in main.json:

```
{
  "MONGO_AUTH_DB": "admin",
  "CONTROLDB_IP": "127.0.0.1",
  "CONTROLDB_PORT": 27017,
  "CONTROLDB_NAME": "control",
  "CONTROLDB_USER": "altcraft",
  "CONTROLDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "FILEDB_ENABLE": true,
  "FILEDB_IP": "127.0.0.1",
  "FILEDB_NAME": "filedb",
  "FILEDB_USER": "altcraft",
  "FILEDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "FILEDB_PORT": 27017
}
```

> TIP: Additional features
>
> - **Authentication: By default, authentication is disabled in MongoDB. To improve security, you can use**
> - **TLS: To enable secure connections, add the following parameters to the configuration:**
>
>   ```
>   net:
>     tls:
>       mode: requireTLS
>       certificateKeyFile: <path to the PEM file>
>       CAFile: <path to the CA certificate>
>   ```

Installation of ClickHouse

> NOTE

ClickHouse database is used for storing history, statistics, and forming user reports. Below is the install

Install dependencies and add the official repository:

```
{
  apt update
  apt install -y apt-transport-https ca-certificates dirmngr gnupg
  apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 8919F6BD2B48D754
  echo "deb https://packages.clickhouse.com/deb stable main" | tee /etc/apt/sources.list.d/clickhouse.list
}
```

Install Clickhouse:

```
{
  apt update
  apt install -y clickhouse-server clickhouse-client
}

Add the service to autostart and start it:

```bash
{
  systemctl enable clickhouse-server.service
  systemctl start clickhouse-server.service
  systemctl status clickhouse-server.service
}
```

Create the user configuration file `altcraft` at the path `/etc/clickhouse-server/users.d/altcraft.xml`:

```
<yandex>
    <users>
        <altcraft>
            <password>abcdefghijklmnopqrstuvwxyzABCDEF</password>
            <networks>
                <ip>::/0</ip>
            </networks>
            <profile>default</profile>
            <quota>default</quota>
            <access_management>1</access_management>
            <database>
                <name>altcraft_system</name>
                <access>
                </access>
            </database>
        </altcraft>
    </users>
</yandex>
```

Update the connection parameters in [the configuration file main.json]:

```
{
"CLICKHOUSE_SYSTEM": {
  "HOST": "localhost",
  "PORT": 9000,
  "USER": "altcraft",
  "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "DATABASE_NAME": "altcraft_system"
  }
}
```

Installation of SSDB

NOTE

SSDB databases are used for system notifications and storing hardbounce (non-existent) email addresse

Install dependencies, clone the Git repository, then build and install:

```
{
  apt update
  apt install -y git build-essential cmake autoconf
  git clone https://github.com/ideawu/ssdb.git /usr/local/ssdb
  cd /usr/local/ssdb
  make
}
```

Create the working directories:

```
mkdir -p /var/lib/ssdb/{hardbounces,notifications} /var/{log,lib}/ssdb /etc/ssdb
```

Create the configuration file `/etc/ssdb/hardbounces.conf`:

TIP

Note: The configuration file should use a tab character for indentation instead of spaces.
```

```
# hardbounces.conf
# MUST indent by TAB!

work_dir = /var/lib/ssdb/hardbounces
pidfile = /run/ssdb@hardbounces.pid

server:
         ip: 127.0.0.1
         port: 4420
         auth: abcdefghijklmnopqrstuvwxyzABCDEF

leveldb:
         cache_size: 500
         write_buffer_size: 64
         compaction_speed: 1000
         compression: yes

logger:
         level: error
         output: /var/log/ssdb/hardbounces.log
         rotate:
                  size: 1000000000
```

Create the configuration file `/etc/ssdb/notifications.conf`:

> **TIP**
>
> **Note: The configuration file should use a tab character for indentation instead of spaces.**

```
# notifications.conf
# MUST indent by TAB!

work_dir = /var/lib/ssdb/notifications
pidfile = /run/ssdb@notifications.pid

server:
         ip: 127.0.0.1
         port: 4430
         auth: abcdefghijklmnopqrstuvwxyzABCDEF

leveldb:
         cache_size: 500
         write_buffer_size: 64
         compaction_speed: 1000
         compression: yes

logger:
         level: error
         output: /var/log/ssdb/notifications.log
         rotate:
                  size: 1000000000
```

Create a systemd service at the path `/etc/systemd/system/ssdb@.service`:

```
[Unit]
Description=High performance NoSQL database
After=network.target

[Service]
User=root
Group=root
WorkingDirectory=/var/lib/ssdb/%i
ExecStart=/usr/local/ssdb/ssdb-server /etc/ssdb/%i.conf -s start -d
ExecStop=/usr/local/ssdb/ssdb-server /etc/ssdb/%i.conf -s stop -d
PIDFile=/run/ssdb@%i.pid
Restart=always
LimitFSIZE=infinity
LimitCPU=infinity
LimitAS=infinity
LimitNOFILE=256000
LimitNPROC=256000
Type=forking

[Install]
WantedBy=multi-user.target
```

Add the services to autostart and start them:

```
{
systemctl enable ssdb@hardbounces
systemctl enable ssdb@notifications
systemctl start ssdb@hardbounces
systemctl start ssdb@notifications
systemctl status ssdb@hardbounces
systemctl status ssdb@notifications
}
```

Update the connection parameters in the configuration file main.json:

```
{
"SSDB_HBSUPP_IP": "127.0.0.1",
"SSDB_HBSUPP_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
"SSDB_HBSUPP_PORT": 4420,
"SSDB_NOTIFY_IP": "127.0.0.1",
"SSDB_NOTIFY_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
"SSDB_NOTIFY_PORT": 4430
}
```

Installation of RabbitMQ

> **NOTE**
>
> **RabbitMQ message broker is used for various interactions between platform services. Below is the instru**

Add the required repositories to the system:

```
{
curl -1sLf https://dl.cloudsmith.io/public/rabbitmq/rabbitmq-server/setup.deb.sh | bash
curl -1sLf https://dl.cloudsmith.io/public/rabbitmq/rabbitmq-erlang/setup.deb.sh | bash
}
```

Install RabbitMQ:

```
{
apt update
apt install -y --fix-missing rabbitmq-server
}
```

Add the service to autostart and start it:

```
{
systemctl enable rabbitmq-server.service
systemctl start rabbitmq-server.service
systemctl status rabbitmq-server.service
}
```

Increase the default disk space limit (Learn more):

```
rabbitmqctl set_disk_free_limit 5GB
```

Create a user, virtual host, and assign permissions:

```
{
  rabbitmqctl add_user "altcraft" "abcdefghijklmnopqrstuvwxyzABCDEF"
  rabbitmqctl set_user_tags "altcraft" administrator
  rabbitmqctl add_vhost "altcraft"
  rabbitmqctl set_permissions -p "altcraft" "altcraft" ".*" ".*" ".*"
}
```

Update the connection parameters in the configuration file main.json:

```
{
  "RABBITMQ_HOST": "127.0.0.1",
  "RABBITMQ_VHOST": "altcraft",
  "RABBITMQ_USER": "altcraft",
  "RABBITMQ_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF"
}
```

Installing Kvrocks

> **INFO**
>
> **Kvrocks is an alternative to Redis, built on RocksDB. Unlike Redis, it stores data on disk, making it suital**

```
{
  apt update
  apt install -y git build-essential cmake libtool python3
}
```

Go to the project's releases page on GitHub, download the latest version, and extract the archive:

```
{
  wget https://github.com/apache/kvrocks/archive/refs/tags/v2.12.1.tar.gz
  tar xzf v2.12.1.tar.gz
}
```

Build and install:

```
{
  cd kvrocks-2.12.1
  ./x.py build -DPORTABLE=1 -DCMAKE_BUILD_TYPE=Release -j $(nproc)
  mv build/kvrocks /usr/local/bin/
}
```

Create the user and directories:

```
{
  groupadd kvrocks
  useradd -r -g kvrocks -s /bin/false kvrocks
  mkdir -p /var/{lib,log}/kvrocks /etc/kvrocks
  chown -R kvrocks:kvrocks /var/{lib,log}/kvrocks
}
```

Create the configuration file `/etc/kvrocks/kvrocks.conf`:

```
# kvrocks.conf
bind 0.0.0.0
port 6666
db-name altcraft.db
dir /var/lib/kvrocks
log-dir /var/log/kvrocks
log-retention-days 7
backup-dir /var/lib/kvrocks/backup
requirepass abcdefghijklmnopqrstuvwxyzABCDEF
supervised systemd
workers 8
```

Create a systemd service file named `kvrocks.service` in the `/etc/systemd/system` directory:

```
[Unit]
Description=kvrocks SSD key-value database
Documentation=https://github.com/apache/kvrocks
Wants=network-online.target
After=network-online.target

[Service]
User=kvrocks
Group=kvrocks
Type=notify
ExecStart=/usr/local/bin/kvrocks -c /etc/kvrocks/kvrocks.conf
WorkingDirectory=/var/lib/kvrocks
Restart=on-failure
ExecStop=/bin/kill -s TERM $MAINPID
RestartSec=10s
LimitNOFILE=100000
LimitNPROC=4096
TimeoutSec=300
NoNewPrivileges=yes

[Install]
WantedBy=multi-user.target
Alias=kvrocks.service
```

Enable the service to start automatically and launch it:

```
{
  systemctl enable kvrocks.service
  systemctl start kvrocks.service
  systemctl status kvrocks.service
}
```

Update the connection parameters in the configuration file main.json:

```
{
  "CAMP_DUPLICATESDB": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  },
  "PROCWORKFLOW_DISTRIBUTED_CACHE_DB": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  }
}
```

Platform Launch

Before launching the platform, make sure you have completed the `main.json` configuration file. A descriptio

```
{
  "BASEDIR": "/opt/akd",
  "MONGO_AUTH_DB": "admin",
  "CONTROLDB_IP": "127.0.0.1",
  "CONTROLDB_PORT": 27017,
  "CONTROLDB_NAME": "control",
  "CONTROLDB_USER": "altcraft",
  "CONTROLDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "FILEDB_ENABLE": true,
  "FILEDB_IP": "127.0.0.1",
  "FILEDB_PORT": 27017,
  "FILEDB_USER": "altcraft",
  "FILEDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "FILEDB_NAME": "filedb",
  "DEFAULT_LOG_LEVEL": "INFO",
  "PRIVATE_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "PRODUCT_NAME": "Altcraft Platform",

  "SYSTEM_REDIS": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  },

  "CAMP_DUPLICATESDB": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  },

  "PROCWORKFLOW_DISTRIBUTED_CACHE_DB": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  },

  "LOYALTY_CACHE": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  },

  "CLICKHOUSE_SYSTEM": {
    "HOST": "127.0.0.1",
    "PORT": 9000,
    "USER": "altcraft",
    "DATABASE_NAME": "altcraft_system",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "IS_DEBUG": false
  },

  "SSDB_HBSUPP_IP": "127.0.0.1",
  "SSDB_HBSUPP_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "SSDB_HBSUPP_PORT": 4420,
  "SSDB_NOTIFY_IP": "127.0.0.1",
  "SSDB_NOTIFY_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "SSDB_NOTIFY_PORT": 4430,

  "RABBITMQ_HOST": "127.0.0.1",
  "RABBITMQ_VHOST": "altcraft",
```

```
  "RABBITMQ_USER": "altcraft",
  "RABBITMQ_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",

  "WEBAPI_PRIVATE_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "WEBCONTROL_HOSTNAME": "altcraft.domain.com",
  "WEBCONTROL_PUBLIC_IP": "192.168.0.1",
  "WEBCONTROL_PUBLIC_PORT": 80,

  "TRACKING_CRYPT_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "TRACKING_HOSTNAME": "click.domain.com",
  "TRACKING_PREFIX": "click",
  "TRACKING_PUBLIC_IP": "192.168.0.2",
  "TRACKING_PUBLIC_PORT": 80,

  "COOKIESAVER_PUBLIC_HOSTNAME": "pxl.domain.com",
  "COOKIESAVER_PUBLIC_IP": "192.168.0.3",
  "COOKIESAVER_PUBLIC_PORT": 80,
  "COOKIESAVER_SSL_ON": false
}
```

To manage platform services, use the `akd` executable. Command reference:

```
Usage:
  akd [flags]
  akd [command]

Available Commands:
  list          List platform information
  start         Start platform services
  stop          Stop platform services
  restart       Restart platform services
  reload-nginx  Reload nginx configuration
  help          Help about any command
  completion    Generate the autocompletion script for the specified shell

Flags:
      --appname         show application full name
      --config string   config path (default "/opt/akd/config/main.json")
      --debug           debug mode (for development purposes)
  -h, --help            help for akd
  -v, --version         version for akd

Use "akd [command] --help" for more information about a command.
```

his completes the installation. The platform is ready to launch and further configuration. To continue setup, follow the guide on the [Getting

Running the platform in a Docker container

The Altcraft image is a monolithic image that runs all the necessary processes in one container.

> NOTE
>
> **Databases (MongoDB, SSDB, ClickHouse) and RabbitMQ must be installed separately on the host or in Do**

Installation

Minimum requirements

The table contains the minimum software versions that must be installed on the target host to run the pla

Software	Version	
Docker Engine	**>= 25.x**	**See the Docker Engine**
Docker Compose	**>= 1.25.0**	**See the Docker Compos**

Volume locations

The Altcraft container uses volumes to store persistent data. You can customize their location according

Location in Docker container	
/altcraft/account_files	**Message previews and task files (export, import) inside accounts. By default, an anonymous volume will be used.**
/altcraft/data	**Temporary files, storage and cache of various jobs**
/altcraft/logs	**Logs Note: the output of logs to the `STDOUT` of the container is configured by the LOG_STDOUT envi**

Minimal main.json configuration

Before starting the platform, you need to configure the minimum configuration `$ALTCRAFT_HOME/config/main.jso`

> CAUTION
>
> **Please generate a stronger password. Do not use `abcdefghijklmnopqrstuvwxyzABCDEF` from the example.**

```
{
    "BASEDIR": "/altcraft",
    "DEFAULT_LOG_LEVEL": "WARN",

    "WEBAPI_PRIVATE_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "WEBCONTROL_HOSTNAME": "altcraft.example.com",
    "WEBCONTROL_PUBLIC_IP": "0.0.0.0",
    "WEBCONTROL_PUBLIC_PORT": 8080,

    "TRACKING_CRYPT_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "TRACKING_PREFIX": "click",
    "TRACKING_PUBLIC_IP": "0.0.0.0",
    "TRACKING_PUBLIC_PORT": 8085,

    "COOKIESAVER_PUBLIC_HOSTNAME": "pixel.altcraft.example.com",
    "COOKIESAVER_PUBLIC_IP": "0.0.0.0",
    "COOKIESAVER_PUBLIC_PORT": 8090,
    "COOKIESAVER_SSL_ON": false,

    "CONTROLDB_IP": "192.168.20.1",
    "CONTROLDB_NAME": "control",
    "CONTROLDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "CONTROLDB_PORT": 27017,
    "CONTROLDB_USER": "altcraft",

    "CLICKHOUSE_SYSTEM": {
        "HOST": "192.168.20.1",
        "PORT": 9000,
        "USER": "altcraft",
        "MAX_CONNECTION": 100,
        "CONN_TIMEOUT": 15,
        "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
        "IS_DEBUG": false
    },

    "RABBITMQ_HOST": "192.168.20.1",
    "RABBITMQ_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "RABBITMQ_USER": "altcraft",

    "SSDB_HBSUPP_IP": "192.168.20.1",
    "SSDB_HBSUPP_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "SSDB_HBSUPP_PORT": 4420,

    "SSDB_NOTIFY_IP": "192.168.20.1",
    "SSDB_NOTIFY_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "SSDB_NOTIFY_PORT": 4430,

    "PRIVATE_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "PRODUCT_NAME": "Altcraft Marketing Platform",

    "NOREPLY_MAILER": {
        "server": "mail.example.com",
        "port": 587,
        "email": "altcraft@example.com",
        "login": "altcraft@example.com",
        "pass": "abcdefghijklmnopqrstuvwxyzABCDEF"
    }
}
```

Running the platform using Docker Engine

After preparing the directories according to your requirements, you can run the image:

```
sudo docker run --detach \
  --name altcraft \
  --publish <WEBCONTROL_PUBLIC_PORT>:<WEBCONTROL_PUBLIC_PORT> \
  --publish <TRACKING_PUBLIC_PORT>:<TRACKING_PUBLIC_PORT> \
  --publish <COOKIESAVER_PUBLIC_PORT>:<COOKIESAVER_PUBLIC_PORT> \
  --volume $ALTCRAFT_HOME/config/main.json:/altcraft/config/main.json \
  --restart always \
  altcraft:<TAG>
```

If you are using SELinux, run this instead:

```
sudo docker run --detach \
  --name altcraft \
  --publish <WEBCONTROL_PUBLIC_PORT>:<WEBCONTROL_PUBLIC_PORT> \
  --publish <TRACKING_PUBLIC_PORT>:<TRACKING_PUBLIC_PORT> \
  --publish <COOKIESAVER_PUBLIC_PORT>:<COOKIESAVER_PUBLIC_PORT> \
  --volume $ALTCRAFT_HOME/config/main.json:/altcraft/config/main.json:Z \
  --restart always \
  altcraft:<TAG>
```

This will ensure that the Docker process has sufficient rights to create files on the mounted volumes.

Optional: If you want to collect logs from the container's `STDOUT`, set the `LOG_STDOUT` environment variable t

```
sudo docker run --detach \
  --name altcraft \
  --publish <WEBCONTROL_PUBLIC_PORT>:<WEBCONTROL_PUBLIC_PORT> \
  --publish <TRACKING_PUBLIC_PORT>:<TRACKING_PUBLIC_PORT> \
  --publish <COOKIESAVER_PUBLIC_PORT>:<COOKIESAVER_PUBLIC_PORT> \
  --volume $ALTCRAFT_HOME/config/main.json:/altcraft/config/main.json:Z \
  --env LOG_STDOUT=true \
  --restart always \
  altcraft:<TAG>
```

Running the platform using Docker Compose

1. Install Docker Compose following the instructions from the minimum requirements section.

2. Prepare `docker-compose.yml` file, example:

```
services:
  altcraft:
    image: altcraft:<TAG>
    container_name: altcraft
    restart: always
    environment:
      - GIN_MODE=release
      - LOG_STDOUT=true
    volumes:
      - /opt/altcraft/account_files:/altcraft/account_files:Z
      - /opt/altcraft/data:/altcraft/data:Z
      - /opt/altcraft/logs:/altcraft/logs:Z
      - /opt/altcraft/config/main.json:/altcraft/config/main.json:Z
    networks:
      altcraft:
        aliases:
          - altcraft
    ulimits:
      nofile:
        soft: 256000
        hard: 256000
    expose:
      - <WEBCONTROL_PUBLIC_PORT>
      - <TRACKING_PUBLIC_PORT>
      - <COOKIESAVER_PUBLIC_PORT>
    ports:
      - <WEBCONTROL_PUBLIC_PORT>:<WEBCONTROL_PUBLIC_PORT>
      - <TRACKING_PUBLIC_PORT>:<TRACKING_PUBLIC_PORT>
      - <COOKIESAVER_PUBLIC_PORT>:<COOKIESAVER_PUBLIC_PORT>

networks:
  altcraft:
      driver: bridge
```

3. Run Altcraft Platform:

```
sudo docker-compose up --detach
```

Changing main.json configuration

After making the necessary changes, you must restart the container to reconfigure Altcraft Platform:

```
sudo docker restart altcraft
```

Update

Update using Docker Engine

1. Download the new version of the image:

```
sudo docker load --input altcraft*.tar.gz
```

2. Remove the existing container:

```
sudo docker rm altcraft
```

3. Create the container again with the parameters specified earlier:

```
sudo docker run --detach \
  --name altcraft \
  --publish <WEBCONTROL_PUBLIC_PORT>:<WEBCONTROL_PUBLIC_PORT> \
  --publish <TRACKING_PUBLIC_PORT>:<TRACKING_PUBLIC_PORT> \
  --publish <COOKIESAVER_PUBLIC_PORT>:<COOKIESAVER_PUBLIC_PORT> \
  --volume $ALTCRAFT_HOME/config/main.json:/altcraft/config/main.json:Z \
  --restart always \
  altcraft:<NEW_TAG>
```

Update using Docker Compose

1. Download the new version of the image:

```
sudo docker load --input altcraft*.tar.gz
```

2. Remove the previous container:

```
sudo docker-compose down
```

3. In the `docker-compose.yml` file set the new version of the image:

```
version: '3.4'

services: altcraft:
  image: altcraft:<NEW_TAG>
  container_name: altcraft
  ...
```

3. Restart Altcraft Platform:

```
sudo docker-compose up --detach
```

Configuration file

The main platform configuration file `main.json` is located by default in the `/opt/akd/config/` folder.
It contains parameters for placing control server components, access settings, and other configuration op

Basic configuration

Below is an example of a standard configuration with local component placement.

```
{
  "BASEDIR": "/opt/akd",
  "MONGO_AUTH_DB": "admin",
  "CONTROLDB_IP": "127.0.0.1",
  "CONTROLDB_PORT": 27017,
  "CONTROLDB_NAME": "control",
  "CONTROLDB_USER": "altcraft",
  "CONTROLDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "FILEDB_ENABLE": true,
  "FILEDB_IP": "127.0.0.1",
  "FILEDB_PORT": 27017,
  "FILEDB_USER": "altcraft",
  "FILEDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "FILEDB_NAME": "filedb",
  "DEFAULT_LOG_LEVEL": "INFO",
  "PRIVATE_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "PRODUCT_NAME": "Altcraft Platform",
  "CLICKHOUSE_SYSTEM": {
    "HOST": "127.0.0.1",
    "PORT": 9000,
    "USER": "altcraft",
    "DATABASE_NAME": "altcraft_system",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "IS_DEBUG": false
  },
  "SSDB_HBSUPP_IP": "127.0.0.1",
  "SSDB_HBSUPP_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "SSDB_HBSUPP_PORT": 4420,
  "SSDB_NOTIFY_IP": "127.0.0.1",
  "SSDB_NOTIFY_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "SSDB_NOTIFY_PORT": 4430,
  "RABBITMQ_HOST": "127.0.0.1",
  "RABBITMQ_VHOST": "altcraft",
  "RABBITMQ_USER": "altcraft",
  "RABBITMQ_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "CAMP_DUPLICATESDB": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  },
  "PROCWORKFLOW_DISTRIBUTED_CACHE_DB": {
    "MODE": "standalone",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "127.0.0.1",
        "PORT": 6666
      }
    ]
  },
  "WEBAPI_PRIVATE_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "WEBCONTROL_HOSTNAME": "altcraft.domain.com",
  "WEBCONTROL_PUBLIC_IP": "192.168.0.1",
  "WEBCONTROL_PUBLIC_PORT": 80,
  "TRACKING_CRYPT_KEY": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "TRACKING_HOSTNAME": "click.domain.com",
  "TRACKING_PREFIX": "click",
  "TRACKING_PUBLIC_IP": "192.168.0.2",
  "TRACKING_PUBLIC_PORT": 80,
  "COOKIESAVER_PUBLIC_HOSTNAME": "pxl.domain.com",
  "COOKIESAVER_PUBLIC_IP": "192.168.0.3",
  "COOKIESAVER_PUBLIC_PORT": 80,
  "COOKIESAVER_SSL_ON": false
}
```

General settings

General parameters

PRODUCT_NAME `string`
Description: Product name
Default: “ALTCRAFT”
Required: Yes
INSTANCE_NAME `string`
Description: Platform instance name
Default: “MAIN”
Required: No
TARIF_CHECK_PERIOD_SEC `int`
Description: Interval to check expiration of limits for tariffs
Default: 60
Required: No

Network

DEFAULT_HTTP_RETRY_COUNT `int`
Description: Number of HTTP reconnection attempts
Default: 3
Required: No
DEFAULT_HTTP_RETRY_DELAYSEC `int`
Description: Interval between HTTP reconnection attempts
Default: 5
Required: No
HTTP_PROXY `string`
Description: Proxy server for HTTP connections
Default: None
Required: No
LICENSE_PROXY `string`
Description: Proxy server for HTTP connections with the License server
Default: “HTTP_PROXY”
Required: No
LONG_REQUEST_TIMEOUT_SEC `int`
Description: Timeout for long-running requests
Default: 840
Required: No
SSH_CONNECTION_TIMEOUT `int`
Description: SSH connection timeout in seconds, 0 = infinite
Default: 60
Required: No
SSH_CACHE_CONNECTION_TIMEOUT `int`
Description: Timeout for cached SSH connections, 0 = infinite
Default: 60
Required: No

File paths

BASEDIR `string`
Description: Platform installation directory
Default: “/opt/akd/”
Required: Yes
UPLOAD_PATH `string`
Description: Directory for temporary storage of uploaded files
Default: “{BASED}/tmp”
Required: No
PIDS_PATH `string`
Description: Directory for storing PID files
Default: “{BASED}/pids”
Required: No
LOGS_PATH `string`

Description: Directory for storing log files
Default: “{BASED}/logs”
Required: No

GEODB_PATH `string`

Description: Directory for storing geodata database
Default: “{BASED}/geo”
Required: No

JSON_CONSTS_PATH `string`

Description: Directory for storing system constants
Default: “{BASED}/json_consts”
Required: No

TEMPLATES_PATH `string`

Description: Directory for storing web interface templates
Default: “{BASED}/templates”
Required: No

LANGUAGE_PATH `string`

Description: Directory for storing localization files
Default: “{BASED}/klukva/languages”
Required: No

UNIX_SOCKETS_PATH `string`

Description: Directory for storing UNIX sockets
Default: “{BASED}/socks”
Required: No

CUTYCAPT_PATH `string`

Description: Directory for the Cutycapt utility
Default: The platform will try to find the path via the PATH variable, searching for “cutycapt” or “CutyCapt
Required: No

Temporary file cleanup

AUTOCLEAN_TMP_DAYS `int`

Description: Frequency of automatic cleanup of temporary files
Default: 30 (days)
Required: No

AUTOCLEAN_EXPORT_API_DAYS `int`

Description: Frequency of automatic cleanup of API export files
Default: 7 (days)
Required: No

Unix domain sockets

PREFER_UNIX_SOCKETS `bool`

Description: Enable UNIX sockets for all available processes
Default: false
Required: No

UNIX_SOCKETS_USER `string`

Description: Username for UNIX sockets
Default: “nginx”
Required: No

UNIX_SOCKETS_GROUP `string`

Description: Access group for UNIX sockets
Default: “nginx”
Required: No

Password requirements

PASSWORDS_MIN_LENGTH `int`

Description: Minimum password length
Default: 8
Required: No

PASSWORDS_MAX_LENGTH `int`

Description: Maximum password length
Default: 32

Required: No

PASSWORDS_STRENGTH `int`

Description: Password complexity level:

- **0 — any characters**
- **1 — excludes dictionary words and usernames**
- **2 — all of the above, plus must contain digits and special characters**

Default: 2
Required: No

PASSWORDS_MAX_AGE `int`

Description: Maximum password lifetime (in seconds), 0 means no limit
Default: 0
Required: No

RPC connection settings

RPC_TOKEN `string`

Description: RPC server authorization token
Default: `"abcdefghiasdwqedwqjklmnopqrstuvwxyzABCDEF"`
Required: No

RPC_HOST `string`

Description: RPC server address
Default: “127.0.0.1”
Required: No

RPC_PORT `string`

Description: RPC server port
Default: “8972”
Required: No

RPC_SELECT_MODE `int`

Description: RPC connection selection mode:

- **0 — Random**
- **1 — Round Robin**
- **2 — Weighted Round Robin**
- **3 — Weighted ICMP (selection by weighted ping time)**
- **4 — Consistent Hash**
- **5 — Closest**

Default: 0
Required: No

Task server settings

PROC_TASK_RPC_TIMEOUT `int`

Description: RPC connection timeout with the task handler
Default: 5
Required: No

PROC_TASK_COUNT_WORKERS `int`

Description: Number of handlers in each group. No more than 1 per CPU core
Default: 2
Required: No

PROCTASK_READER_GROUP_SIZE `int`

Description: Number of handler groups
Default: 4
Required: No

PROCTASK_TASK_READING_PERIOD_MS `int`

Description: Interval between task requests from MongoDB. To remove pauses between requests, set this
Default: 3000
Required: No

PROC_TASK_TTL_HOUR `int`

Description: Task processing timeout
Default: 720 (hours)
Required: No

PROC_TASK_PROGRESS_PERIOD `int`

Description: Task progress update interval
Default: 5 (seconds)
Required: No

PROC_TASK_RPC_CONNECTION_TIMEOUT `int`

Description: Connection timeout duration
Default: 15 (seconds)
Required: No

PROC_TASK_RPC_READ_TIMEOUT `int`

Description: Request wait time after which the connection closes (0 = no timeout)
Default: 60 (seconds)
Required: No

PROC_TASK_SELECT_MODE `string`

Description: Handler operation mode:

- **RoundRobin — cyclic operation**
- **RandomSelect — random handler selection**

Default: “RoundRobin”
Required: No

PROC_TASK_COPY_HISTORY_RANGE_MODE `string`

Description: Profile history copy mode:

- **full_period_mode — copies history for the entire period**
- **range_period_mode — copies history for a specified range (in days)**

Default: “full_period_mode”
Required: No

PROC_TASK_COPY_HISTORY_RANGE_DAYS `int`

Description: Number of days for `range_period_mode`
Default: 3
Required: No

PROC_TASK_RESERVED_TASK_IDLE_TIME_LIMIT_MIN `int`

Description: Max idle time in `reserved` **status Default: 5**
Required: No

“CLUSTER”:{} — clustered deployment of the platform control server

HOST_ID `string`

Description: ID of the current cluster replica
Default: None
Required: Yes

PROC_TASKS `array`

Description: List of system task handlers on cluster replicas:

```
"PROC_TASKS": [{
  "HOST_ID": "", // unique replica ID
  "RPCHOST": "", // RPC replica address
  "RPCPORT": "", // RPC replica port
  "FORCE_MASTER_ROLE": false // use as master server
}]
```

Default: None
Required: Yes

PROC_WORKFLOW `array`

Description: List of automation Scenario handlers on cluster replicas:

```
"PROC_WORKFLOW": [{
  "HOST_ID": "", // unique replica ID
  "RPCHOST": "", // RPC replica address
  "RPCPORT": "", // RPC replica port
  "FORCE_MASTER_ROLE": false // use as master server
}]
```

Default: None
Required: Yes

ATTEMPTS_FOR_EACH_CONNECTION `int`

Description: Number of attempts to establish a connection
Default: 2
Required: No

CONNECTION_FAIL_SLEEP_SEC `int`

Description: Interval between connection attempts
Default: 5
Required: No

Audit Log

AUDIT_LOG_ENABLE `bool`
Description: Enables the audit log for user actions
Default: true
Required: No
AUDIT_LOG_TTL `int`
Description: Retention time for audit log records in days
Default: 90
Required: No

Asynchronous audit log processing

AUDIT_RMQ_PROCESSING_ENABLE `bool`
Description: When enabled, audit log events are sent to the RabbitMQ queue `ak_audit_events` **and processed**
Required: No

RabbitMQ audit log consumer settings

AUDIT_RMQ_CONSUMER_MAX_WORKERS `int`
Description: Number of workers started in the consumer to process audit log events from the RabbitMQ q
Default: 8
Required: No
AUDIT_RMQ_CONSUMER_PREFETCH_SIZE `int`
Description: Prefetch queue size for the audit log RabbitMQ consumer
Default: 8000
Required: No

RabbitMQ audit log publisher settings

AUDIT_RMQ_PUBLISHER_MAX_WORKERS `int`
Description: Number of publisher workers used to send audit log events to RabbitMQ
Default: 8
Required: No
AUDIT_RMQ_PUBLISHER_PREFETCH_SIZE `int`
Description: Internal buffer size of the audit log publisher
Default: 8000
Required: No

Batch processing of audit log events

AUDIT_LOG_BATCH_PROCESSOR_MAX_WORKERS `int`
Description: Number of workers processing batches of audit log events
Default: 8
Required: No
AUDIT_LOG_BATCH_SEND_INTERVAL_MS `int`
Description: Wait time during batch assembly. If no new events arrive within this interval, the batch proce
Default: 1000
Required: No
AUDIT_LOG_BATCH_MAX_SEND_INTERVAL_MS `int`
Description: Maximum wait time before sending a batch, even if it is not fully filled
Default: 1000
Required: No
AUDIT_LOG_MAX_BATCH_LENGTH `int`
Description: Maximum number of events in a single batch. Affects how often insert requests are sent to th
Default: 10000
Required: No
AUDIT_LOG_MAX_BATCH_WORKER_BUFFER `int`
Description: Buffer size of a batch processing worker. Used to tune processing for low or high load scenar
Default: 1000

Required: No

HTTP connection settings

DEFAULT_CONN_DIAL_TIMEOUT_SEC `int`
Description: Timeout for calling a remote server
Default: 15
Required: No

DEFAULT_CONN_TIMEOUT_SEC `int`
Description: Connection timeout with a remote server
Default: 120
Required: No

DEFAULT_CONN_IDLE_TIMEOUT_SEC `int`
Description: Lifetime of blocked connections in the pool
Default: 60
Required: No

SENDER_QUEUE_SIZE `int`
Description: Sender queue size
Default: 8 (no more than 8 per CPU core)
Required: No

Sender settings

SENDER_QUEUE_SIZE `int`
Description: Sender queue limit
Default: 2000
Required: No

SENDER_WORKER_SIZE `int`
Description: Number of sender workers
Default: 1
Required: No

IP address and range exclusions

BLOCKIPRANGES `array`
Description: Exclude IP subnets from sending
Default: None
Required: No

TRK_BLOCK_IPS `array`
Description: Exclude IP subnets from sending
Default: None
Required: No

Data storage

With these parameters you can override system settings of platform databases — for deployment on remo

RESCAN_DB_SEC `int`
Description: Frequency of subscriber counter updates in databases
Default: 900
Required: No

RESCAN_DB_COUNT_WORKERS `int`
Description: Number of profile database recalculation workers
Default: 4
Required: No

SQL_ACCOUNT_POOL_SIZE `int`
Description: Limit of connections in the PostgreSQL account database pool
Default: 10
Required: No

CAMPAIGN_LEDIS_DB_PATH `string`
Description: Path to mailing ledis_db database
Default: “{BASED}/data/ledis_db”
Required: No

MongoDB

MONGO_AUTH_DB `string`

Description: The database on which authorization is performed, if specified
Default: None
Required: No

MONGODB_PROCTRIGGER_MAX_POOL_SIZE `int`

Description: Maximum connection pool size for MongoDB for the proctrigger process
Default: 1500
Required: No

MONGODB_PROCTRIGGER_MIN_POOL_SIZE `int`

Description: Minimum connection pool size for MongoDB for the proctrigger process
Default: 10
Required: No

MONGODB_PROCWORKFLOW_MAX_POOL_SIZE `int`

Description: Maximum connection pool size for MongoDB for the procworkflow process
Default: 1500
Required: No

MONGODB_PROCWORKFLOW_MIN_POOL_SIZE `int`

Description: Minimum connection pool size for MongoDB for the procworkflow process
Default: 10
Required: No

MONGODB_MAX_POOL_SIZE `int`

Description: Maximum connection pool size for MongoDB for general processes
Default: 1000
Required: No

MONGODB_MIN_POOL_SIZE `int`

Description: Minimum connection pool size for MongoDB for general processes
Default: 1
Required: No

MONGODB_RECONNECT_TIME_PERIOD `int`

Description: Wait time before attempting to reconnect
Default: 5
Required: No

MONGODB_SOCKET_TIMEOUT_MIN `int`

Description: Socket timeout
Default: 30
Required: No

MongoDB/Control

CONTROLDB_NAME `string`

Description: Database name
Default: None
Required: Yes

CONTROLDB_IP `string`

Description: Database host address
Default: None
Required: Yes

CONTROLDB_PORT `int`

Description: Database port
Default: None
Required: Yes

CONTROLDB_USER `string`

Description: Database username
Default: None
Required: Yes

CONTROLDB_PASS `string`

Description: Password
Default: None
Required: Yes

CONTROLDB_TLS_ON `bool`

Description: Enable TLS mode for connection (requires additional setup of the `mongod.service`)
Default: None
Required: No

CONTROLDB_TLS_KEY_PATH `string`

Description: Path to TLS encryption key
Default: None
Required: No

CONTROLDB_TLS_CRT_PATH `string`

Description: Path to TLS encryption certificate
Default: None
Required: No

CONTROLDB_TLS_CA_CRT_PATH `string`

Description: Path to CA TLS
Default: None
Required: No

ControlDB cluster deployment

(Used instead of CONTROLDB_NAME, CONTROLDB_IP, CONTROLDB_PORT)

CONTROLDB_REPL_SET_NAME `string`

Description: Cluster name
Default: None
Required: No

CONTROLDB `array`

Description: Array with connection data for databases in the cluster
Default: None, must contain

```
[{
  "IP": "", // Host address
  "PORT": "" // Host port
}]
```

Required: No

ClickHouse

CLICKHOUSE_SYSTEM `object`

Description: Object with connection data for ClickHouse databases
Default: None, must contain

```
{
  "HOST": "", // Host address
  "PORT": 9000, // Port
  "USER": "", // Username for connection
  "MAX_CONNECTION": 100, // Maximum number of connections
  "CONN_TIMEOUT": 15, // Connection timeout
  "PASSWORD": "", // User password
  "IS_DEBUG": false, // Enable/disable debug mode
  "DATABASE_NAME": "" // Database name (default - altcraft_system)
}
```

Deployment in ClickHouse replica:

```
{
  "HOST": "", // Host address
  "PORT": 9000, // Port
  "USER": "", // Username for connection
  "MAX_CONNECTION": 100, // Maximum number of connections
  "CONN_TIMEOUT": 15, // Connection timeout
  "PASSWORD": "", // User password
  "IS_DEBUG": false, // Enable/disable debug mode
  "DATABASE_NAME": "", // Database name (default - altcraft_system)
  "ALT_HOSTS": [ // List of hosts for load balancing
    {
      "HOST": "", // Host address
      "PORT": 9000 // Port
    }
  ],
  "REPLICATED": true // Enable support for ReplicatedMergeTree replicated tables
}
```

The full list of parameters for the ClickHouse configuration object can be found below.
Required: Yes

Kvrocks

SYSTEM_REDIS `object`

Description: Main configuration for Redis storage. If no other Redis configuration parameters are specifie
Default: No
Required: Yes

LOYALTY_CACHE `object`

Description: The settings of the distributed cache storage used for loyalty programs.
Default: No
Required: Yes

PROCWORKFLOW_DISTRIBUTED_CACHE_DB `object`

Description: The settings of the distributed cache storage used by the process `procworkflow` **. Default: No**
Required: Yes

CAMP_DUPLICATESDB `object`

Description: Redis config for duplicates database
Default: None
Required: Yes

KVR_MODE `string`

Description: Redis operation mode
Default: None
Required: Yes

KVR_NODES `array`

Description: List of nodes
Default: None
Required: Yes

KVR_USER `string`

Description: Username
Default: None
Required: No

KVR_PASSWORD `string`

Description: Password
Default: None
Required: No

KVR_DATABASE `int`

Description: Database number
Default: 0
Required: No

KVR_TLS_ON `bool`

Description: Use TLS
Default: false
Required: No

KVR_TLS_KEY_PATH `string`

Description: Path to TLS key
Default: None
Required: Yes, if TLS enabled

KVR_TLS_CRT_PATH `string`

Description: Path to TLS certificate
Default: None
Required: Yes, if TLS enabled

KVR_TLS_CA_CRT_PATH `string`

Description: Path to CA certificate
Default: None
Required: Yes, if TLS enabled

KVR_MAX_RETRIES `int`

Description: Maximum number of retries
Default: 3
Required: No

KVR_MIN_RETRY_BACKOFF `int`

Description: Minimum retry delay
Default: 8
Required: No

KVR_MAX_RETRY_BACKOFF `int`

Description: Maximum retry delay
Default: 512
Required: No

KVR_DIAL_TIMEOUT `int`

Description: Connection establishment timeout
Default: 5
Required: No

KVR_READ_TIMEOUT `int`

Description: Read timeout
Default: 3
Required: No

KVR_WRITE_TIMEOUT `int`

Description: Write timeout
Default: 3
Required: No

KVR_POOL_FIFO `bool`

Description: Pool logic
Default: false
Required: No

KVR_POOL_SIZE `int`

Description: Connection pool size
Default: 10
Required: No

KVR_POOL_TIMEOUT `int`

Description: Timeout for obtaining a connection
Default: KVR_READ_TIMEOUT + 1
Required: No

KVR_MIN_IDLE_CONNECTIONS `int`

Description: Minimum number of idle connections
Default: 0
Required: No

KVR_MAX_IDLE_CONNECTIONS `int`

Description: Maximum number of idle connections
Default: 0
Required: No

KVR_MAX_ACTIVE_CONNECTIONS `int`

Description: Maximum number of connections in the pool
Default: 0
Required: No

KVR_CONNECTION_MAX_IDLE_TIME `int`

Description: Maximum idle time
Default: 1800
Required: No

KVR_CONNECTION_MAX_LIFE_TIME `int`

Description: Maximum connection lifetime
Default: -1
Required: No

KVR_MAX_REDIRECTS `int`

Description: Maximum number of redirects
Default: 3
Required: No

KVR_READONLY `bool`

Description: Read-only mode on replicas
Default: false
Required: No

KVR_ROUTE_BY_LATENCY `bool`

Description: Routing by latency mode
Default: false
Required: No

KVR_ROUTE_RANDOMLY `bool`

Description: Random routing mode
Default: false
Required: No

KVR_MASTER_NAME `string`

Description: Master instance name
Default: None
Required: Yes, if Sentinel

KVR_SENTINEL_USERNAME `string`

Description: Sentinel user
Default: None
Required: No

KVR_SENTINEL_PASSWORD `string`

Description: Sentinel password
Default: None
Required: No

Account file storage (without FileDB)

ACCOUNT_FILES_DOWNLOAD_PATH `string`

Description: Directory for storing exported account files
Default: “{BASED}/account_files/%s/download”
Required: No

ACCOUNT_FILES_IMAGES_PATH `string`

Description: Directory for storing images
Default: “{BASED}/account_files/%s/images”
Required: No

ACCOUNT_FILES_THUMBS_PATH `string`

Description: Directory for storing image thumbnails
Default: “{BASED}/account_files/%s/thumbs”
Required: No

ACCOUNT_FILES_AVATARS_PATH `string`

Description: Directory for storing user avatars
Default: “{BASED}/account_files/%s/avatars_v2”
Required: No

ACCOUNT_FILES_DATA_PATH `string`

Description: Directory for storing files from segment conditions
Default: “{BASED}/account_files/%s/data”
Required: No

ACCOUNT_FILES_ATTACH_PATH `string`

Description: Directory for storing files attached to email templates
Default: “{BASED}/account_files/%s/attach”
Required: No

ACCOUNT_FILES_TABLES_PATH `string`

Description: Directory for storing queries to external SQL databases
Default: “{BASED}/account_files/%s/table”
Required: No

ACCOUNT_FILES_PUSH_PATH `string`

Description: Directory for storing archive for Safari Push notifications
Default: “{BASED}/account_files/%s/push”
Required: No

MongoDB / File — database for storing platform account files

FILEDB_NAME `string`

Description: Database name
Default: None
Required: Yes

FILEDB_IP `string`

Description: Database host address
Default: None
Required: Yes

FILEDB_PORT `int`

Description: Database port
Default: None
Required: Yes

FILEDB_USER `string`

Description: Database username
Default: None
Required: Yes

FILEDB_PASS `string`

Description: Password
Default: None
Required: Yes

FILEDB_TLS_ON `bool`

Description: Enable TLS mode for connection (requires additional setup of the `mongod.service`)
Default: None
Required: No

FILEDB_TLS_KEY_PATH `string`

Description: Path to TLS encryption key
Default: None
Required: No

FILEDB_TLS_CRT_PATH `string`

Description: Path to TLS encryption certificate
Default: None
Required: No

FILEDB_TLS_CA_CRT_PATH `string`

Description: Path to CA TLS
Default: None
Required: No

FileDB cluster deployment

(Used instead of FILEDB_NAME, FILEDB_IP, FILEDB_PORT)

FILEDB_REPL_SET_NAME `string`

Description: Cluster name
Default: None
Required: No

FILEDB `array`

Description: Array with connection data for databases in the cluster
Default: None, must contain

```
[{
  "IP": "", // Host address
  "PORT": "" // Host port
}]
```

Required: No

S3 / Web version — storage of web versions (used in version 67 and higher)

OBJECTS3_HOST `string`

Description: Storage host address for web versions
Default: None
Required: No

OBJECTS3_KEY_ID `string`

Description: Storage access key
Default: None
Required: No

OBJECTS3_KEY_SECRET `string`

Description: Secret key for storage access
Default: None
Required: No

OBJECTS3_IS_SSL `bool`

Description: Use encryption for connection
Default: None
Required: No

OBJECTS3_INSECURE_REQUESTS `bool`

Description: Disable mandatory encryption certificate validation
Default: None
Required: No

WEBVER_LEGACY_TTL_DAYS `int`

Description: Number of days to store web versions in Mongo if there are no policies and none are attache
Default: 30
Required: No

MongoDB / Web version — database for storing web versions (deprecated)

WEBVER_MGO_HOSTNAME `string`

Description: Database host address for web versions
Default: None
Required: No

WEBVER_MGO_PORT `int`

Description: Database port for web versions
Default: None
Required: No

WEBVER_MGO_USER `string`

Description: Username
Default: None
Required: No

WEBVER_MGO_PASS `string`

Description: Password
Default: None
Required: No

SSDB

SSDB_RECONNECT_TIME_PERIOD `int`

Description: Waiting time before retrying a connection
Default: 5
Required: No

SSDB/Suppress

SSDB_HBSUPP_IP `string`

Description: Database host address
Default: None
Required: Yes

SSDB_HBSUPP_PORT `int`

Description: Database port
Default: None
Required: Yes

SSDB_HBSUPP_PASS `string`

Description: Password
Default: None
Required: Yes

SuppressDB cluster deployment

Used instead of parameters for a single database

SSDB_HBSUPP `array`

Description: Array with connection data for databases in the cluster
Default: None, must contain

```
[{
  "IP": "", // Host address
  "PORT": "", // Host port
  "PASS": "" // Database password
}]
```

Required: No

SSDB/Notifier

SSDB_NOTIFY_IP `string`

Description: Database host address
Default: None
Required: Yes

SSDB_NOTIFY_PORT `int`

Description: Database port
Default: None
Required: Yes

SSDB_NOTIFY_PASS `string`

Description: Password
Default: None
Required: Yes

NotifierDB cluster deployment

Used instead of parameters for a single database

SSDB_NOTIFY `array`

Description: Array with connection data for databases in the cluster
Default: None, must contain

```
[{
  "IP": "", // Host address
  "PORT": "", // Host port
  "PASS": "" // Database password
}]
```

Required: No

Message queue

**For fault-tolerant operation, the platform connects to a RabbitMQ cluster. If a connection to one of the no
and the failover mode.**

"RABBITMQ_CLUSTER": {}

CONNECTION_TIMEOUT `int`

Description: Connection timeout (in seconds), 0 = infinite
Default: 30
Required: No

HEARTBEAT_PERIOD_SEC `int`

Description: Interval for health check requests to the connected host (in seconds)
Default: 10
Required: No

FAIL_MODE `string`

Description: Reconnection mode in case of failure:

- **failtry — reconnect to the current host. In this case, `STRATEGY` will not be applied.**
- **failover — connect to the next host in the list depending on the strategy.**

Default: "failtry"
Required: No

STRATEGY `string`

Description: Strategy determines to which node in the cluster connections will be switched:

- **in_order (default) — sequentially**
- **weight — to the node with the highest specified weight**

Default: "in_order"
Required: No

VHOST `string`

Description: RabbitMQ virtual host
Default: "akmp"
Required: No

HOSTS `array`

Description: Array with connection data for nodes in the cluster
Default:

```
"HOSTS": [
  {
    "HOST": "", // RabbitMQ host address (string)
    "PORT": ,   // RabbitMQ port (int)
    "WEIGHT": , // Node weight (int)
  }
]
```

Required: Yes

USER `string`

Description: Username
Default: None
Required: Yes

PASSWORD `string`

Description: Password
Default: None
Required: Yes

RECONNECT_TIME_PERIOD `int`

Description: Waiting time before retrying a connection (in seconds)
Default: 5
Required: No

X_MESSAGE_TTL `int`

Description: Message time-to-live in the broker queue
Default: 86400
Required: No

CONFIRM_PUBLISH `bool`

Description: Allows publishing in server confirm mode
Default: true
Required: No

SSL_ON `bool`

Description: Enable data encryption for the connection
Default: false
Required: No

SSL_CRT_PATH `string`

Description: Path to SSL encryption certificate
Default: None
Required: No

SSL_KEY_PATH `string`

Description: Path to SSL encryption key
Default: None
Required: No

INSECURE_SKIP_VERIFY `bool`

Description: Allows an unencrypted connection
Default: None
Required: No

CACHE_SIZE_MB `int`

Description: Cache size for storing consumer identifiers (evicting, designed for 200k events); prevents pro
Default: 10
Required: No

MAX_FAIL_EVENTS_COUNT `int`

Description: Maximum number of events that may remain unprocessed in case of failure. If the number of
Default: 1024
Required: No

CONFIRM_CHAN_SIZE `int`

Description: Confirmation channel size
Default: 256
Required: No

HISTORY_EVENT_SENDER_WORKER_SIZE `int`

Description: Number of senders for the RMQ event queue
Default: 1
Required: No

HISTORY_EVENT_SENDER_STRATEGY `string`

Description: Sender selection strategy for RMQ event:

- **uniform — uniform**
- **random — random**

Default: “uniform”
Required: No

Deprecated configuration (versions 66 and below)

If the `RABBITMQ_CLUSTER` section is not specified in the configuration file, it will be initialized based on previo

RABBITMQ_HOST `string`

Description: RabbitMQ host address
Default: None
Required: Yes

RABBITMQ_PORT `int`

Description: RabbitMQ port
Default: 5672
Required: No

RABBITMQ_USER `string`

Description: Username
Default: None
Required: Yes

RABBITMQ_PASS `string`

Description: Password
Default: None
Required: Yes

RABBITMQ_VHOST `string`

Description: RabbitMQ virtual host
Default: “akmp”
Required: No

RABBITMQ_RECONNECT_TIME_PERIOD `int`

Description: Waiting time before retrying a connection
Default: 5
Required: No

RABBITMQ_X_MESSAGE_TTL `int`

Description: Message time-to-live in the broker queue
Default: 86400
Required: No

API-RMQ

With these parameters you can optimize message handling for API-RMQ:

API_REQ_CONSUMER_COUNT `int`

Description: Number of consumers for the “api_req” queue
Default: 1
Required: No

API_REQ_CONSUMER_WORKER_SIZE `int`

Description: Number of workers to process messages from the “api_req” queue
Default: 16
Required: No

API_REQ_CONSUMER_PREFETCH_SIZE `int`

Description: Number of messages a consumer can request at once before acknowledging previously proc
Default: API_REQ_CONSUMER_WORKER_SIZE * 100
Required: No

API_RESP_PUBLISHER_WORKER_SIZE `int`

Description: Number of workers for publishing responses to the “api_resp” queue
Default: 16
Required: No

API_RESP_PUBLISHER_QUEUE_SIZE `int`

Description: Queue size for publishing responses. Defines how many messages can be temporarily stored
Default: 50000
Required: No

PebbleDB

Processing a large number of Action Hooks can lead to high RabbitMQ load. To offload the broker, you car

PROCHOOK_BUFFER_PEBBLEDB_ENABLE `bool`

Description: Enables buffer database mode
Default: false
Required: No

PROCHOOK_BUFFER_PEBBLEDB_PATH `string`

Description: Path to the buffer database
Default: None
Required: No

PROCHOOK_BUFFER_PEBBLEDB_INSERT_BATCH_SIZE `int`

Description: Insert batch size
Default: 200
Required: No

PROCHOOK_BUFFER_PEBBLEDB_BATCH_ASSEMBLING_TIME_MS `int`

Description: Insert batch assembly time (ms)
Default: false
Required: No

PROCHOOK_BUFFER_PEBBLEDB_SCAN_TIME_MS `int`

Description: Interval between scans (ms)
Default: 1500
Required: No

PROCHOOK_BUFFER_PEBBLEDB_BATCH_COMMIT_TIME_MS `int`

Description: Commit time for batches (ms)
Default: 500
Required: No

User interface and administrative panel

The parameters are responsible for local and network deployment of the platform's user panel:

PRIVATE_KEY `string`

Description: Encryption key for working with the web interface of the platform
Default: None
Required: Yes

ALLOW_INSECURE_REQUESTS `bool`

Description: Disable mandatory certificate verification
Default: false
Required: No

SSL_ON `bool`

Description: Enable encryption for the platform interface
Default: false
Required: No

SSL_CRT_PATH `string`

Description: Path to SSL encryption certificate
Default: None
Required: No

SSL_KEY_PATH `string`

Description: Path to SSL encryption key
Default: None
Required: No

SSL_PUBLIC_PORT `int`

Description: Port for external encrypted connections
Default: 443
Required: No

WEBCONTROL_HOSTNAME `string`

Description: Domain name of the user interface
Default: None
Required: Yes

WEBCONTROL_PUBLIC_IP `string`

Description: External IP address of the platform web interface
Default: None
Required: Yes

WEBCONTROL_PUBLIC_PORT `int`

Description: Port for external connections to the platform web interface
Default: 80
Required: No

WEBCONTROL_LOCAL_IP `string`

Description: Local IP address of the platform web interface
Default: “127.0.0.1”
Required: No

WEBCONTROL_LOCAL_PORT `int`

Description: Port for local connections to the platform web interface
Default: 8080
Required: No

WEBCONTROL_PRIVATE_KEY `string`

Description: Encryption key for working with the platform web interface
Default: Equal to “PRIVATE_KEY”
Required: No

ADMIN_WEBCONTROL_LOCAL_IP `string`

Description: Local IP address of the admin interface
Default: “127.0.0.1”
Required: No

ADMIN_WEBCONTROL_LOCAL_PORT `int`

Description: Port for local connections to the admin interface
Default: 8081
Required: No

INVALID_LOGINS_ATTEMPTS_MAX `int`

Description: Maximum number of unsuccessful login attempts to the web interface
Default: 5
Required: No

INVALID_LOGINS_ATTEMPTS_SEC `int`

Description: Number of seconds to wait before the next login after reaching the limit of unsuccessful atte
Default: 60
Required: No

Tracking platform module

The following parameters are used to configure the external connection of the platform tracking module,

General settings

TRACKING_PUBLIC_IP `string`

Description: Public network address of the tracking module
Default: None
Required: No

TRACKING_PUBLIC_PORT `int`

Description: Public network port of the tracking module
Default: 80
Required: No

TRACKING_HOSTNAME `string`

Description: Domain name of the tracking server
Default: “TRACKING_PREFIX + . + WEBCONTROL_HOSTNAME”
Required: No

TRACKING_TEST_HOST `string`

Description: Test domain name of the tracking server, overrides TRACKING_HOSTNAME. Used to display l
Default: None
Required: No

TRACKING_PREFIX `string`

Description: Subdomain prefix for the tracking module
Default: “trk”
Required: No

TRACKING_STATIC_PATH `string`

Description: Directory for storing files uploaded by users in requests to the tracking server
Default: “{BASED}/ak/static/trk”
Required: No

TRACKING_PUB_SIZE `int`

Description: Number of handlers to be used by tracking services’ RMQ publishers
Default: 1
Required: No

TRACKING_PUB_QUEUE_SIZE `int`

Description: Channel size for tracking services’ RMQ publishers
Default: 50000
Required: No

UNIQUE_ACTIONS_TTL_MIN `int`

Description: Estimated uniqueness period for tracking-detected actions: opens, clicks
Default: 1440
Required: No

PIXEL_COOLDOWN_TIME_MIN `int`

Description: Estimated uniqueness period for target actions captured by the pixel
Default: 43200
Required: No

Data encryption

TRACKING_CRYPT_KEY `string`

Description: Encryption key for the platform tracking module
Default: None
Required: No

TRACKING_SSL_ON `bool`

Description: Enable encryption for the tracking server
Default: false
Required: No

TRACKING_SSL_CRT_PATH `string`

Description: Path to SSL encryption certificate
Default: None
Required: No

TRACKING_SSL_KEY_PATH `string`

Description: Path to SSL encryption key
Default: None
Required: No

TRACKING_SSL_PUBLIC_PORT `int`

Description: Port for external encrypted connections to the platform tracking module
Default: 443
Required: No

Tracking cache settings

FILE_CACHE_SIZE_MB `int`

Description: Size of the tracking file cache
Default: 256
Required: No

MAX_CACHE_FILE_SIZE_MB `int`

Description: Maximum size of a tracking file stored in the cache
Default: 2
Required: No

FILE_CACHE_EXPIRE_SEC `int`

Description: Retention time for tracking files in the cache
Default: 300
Required: No

FILE_CACHE_ACTIVE_TIME_SEC `int`

Description: Interval for checking and cleaning outdated data in tracking
Default: 1
Required: No

Tracking process connections

Connection of tracking processes to the control server

trkaction

TRACKING_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: "UNIX_SOCKETS_PATH/akd-trackingnew.socket"
Required: No
TRACKING_IP `string`
Description: Process host address
Default: "127.0.0.1"
Required: No
TRACKING_PORT `int`
Description: Process port
Default: 8095
Required: No

---

trkpush

TRACKING_PUSH_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: "UNIX_SOCKETS_PATH/akd-trackingpush.socket"
Required: No
TRACKING_PUSH_IP `string`
Description: Process host address
Default: "127.0.0.1"
Required: No
TRACKING_PUSH_PORT `int`
Description: Process port
Default: 8100
Required: No

---

trkread

TRACKING_READ_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: "UNIX_SOCKETS_PATH/akd-trackingnewread.socket"
Required: No
TRACKING_READ_IP `string`
Description: Process host address
Default: "127.0.0.1"
Required: No
TRACKING_READ_PORT `int`
Description: Process port
Default: 8097
Required: No

---

trkimage

TRACKING_IMAGE_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: "UNIX_SOCKETS_PATH/akd-imgtrackingnew.socket"
Required: No
TRACKING_IMAGE_IP `string`
Description: Process host address
Default: "127.0.0.1"
Required: No
TRACKING_IMAGE_PORT `int`

Description: Process port
Default: 8096
Required: No

---

trkwebversion

TRACKING_WEB_VERSION_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: “UNIX_SOCKETS_PATH/akd-trackingwebversionnew.socket”
Required: No
TRACKING_WEB_VERSION_IP `string`
Description: Process host address
Default: “127.0.0.1”
Required: No
TRACKING_WEB_VERSION_PORT `int`
Description: Process port
Default: 8199
Required: No

---

trksms

TRACKING_SMS_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: “UNIX_SOCKETS_PATH/akd-trackingnewsms.socket”
Required: No
TRACKING_SMS_IP `string`
Description: Process host address
Default: “127.0.0.1”
Required: No
TRACKING_SMS_PORT `int`
Description: Process port
Default: 8098
Required: No

---

trkcustom

TRACKING_CUSTOM_CHANNELS_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: “UNIX_SOCKETS_PATH/akd-trackingnewcustomchannels.socket”
Required: No
TRACKING_CUSTOM_CHANNELS_IP `string`
Description: Process host address
Default: “127.0.0.1”
Required: No
TRACKING_CUSTOM_CHANNELS_PORT `int`
Description: Process port
Default: 8097
Required: No
TRACKING_CUSTOM_CHANNELS_PERIOD_SEC `int`
Description: Tracking process call interval
Default: 30
Required: No

---

trkmandrill

Add the “TRACKING_MANDRILL”: {} object with the following properties:
LOCAL_PORT `int`
Description: Local process port
Default: 8094
Required: No

HOSTNAME `string`
Description: Mandrill tracking domain name
Default: “mandrill.{WEBCONTROL_HOSTNAME}”
Required: No

PUBLIC_IP `string`
Description: Public process host address
Default: “TRACKING_PUBLIC_IP”
Required: No

PUBLIC_PORT `int`
Description: Public process port
Default: 80
Required: No

---

trk_amazon_sns

Add the “TRACKING_AMAZON_SNS_HTTP”: {} object with the following properties:

LOCAL_PORT `int`
Description: Process port
Default: 8093
Required: No

LOCAL_IP `string`
Description: Process host address
Default: “127.0.0.1”
Required: No

SSL `int`
Description: Data encryption settings
Default: None, contains two string values:

```
{
  "CRT_PATH": "", // Path to certificate
  "KEY_PATH": ""  // Path to key
}
```

Required: No

Pixel settings

With these parameters you can override system connection settings for the cookie saver and procpush pr

External connection

COOKIESAVER_PUBLIC_IP `string`
Description: Public host address of the cookie handler
Default: None
Required: No

COOKIESAVER_PUBLIC_PORT `int`
Description: Public port of the cookie handler
Default: 443
Required: No

COOKIESAVER_PUBLIC_HOSTNAME `string`
Description: Domain name of the cookie handler
Default: None
Required: No

COOKIESAVER_CACHE_EXPIRATION_SEC `int`
Description: Retention time for cached handler data
Default: 15 (seconds)
Required: No

COOKIESAVER_CACHE_EVENT_LIMIT `int`
Description: Limit on registering goals for one cookie under a specific pixel
Default: 9
Required: No

Internal connection

COOKIESAVER_UNIXSOCK_PATH `string`

Description: Path to the UNIX socket of the process
Default: “UNIX_SOCKETS_PATH/akd-cookiesaver.socket”
Required: No

COOKIESAVER_IP `string`

Description: Process host address
Default: “127.0.0.1”
Required: No

COOKIESAVER_PORT `int`

Description: Process port
Default: 8099
Required: No

Data encryption

COOKIESAVER_SSL_ON `bool`

Description: Enable encryption for the cookie handler
Default: true
Required: No

COOKIESAVER_SSL_CRT_PATH `string`

Description: Path to SSL encryption certificate
Default: None
Required: No

COOKIESAVER_SSL_KEY_PATH `string`

Description: Path to SSL encryption key
Default: None
Required: No

Platform Web API

With these parameters you can override system connection settings for the api process:

WEBAPI_UNIXSOCK_PATH `string`

Description: Path to the UNIX socket of the process
Default: “UNIX_SOCKETS_PATH/akd-api.socket”
Required: No

WEBAPI_LOCAL_IP `string`

Description: Process host address
Default: “127.0.0.1”
Required: No

WEBAPI_LOCAL_PORT `int`

Description: Process port
Default: 8801
Required: No

Deduplication settings

With these parameters you can configure the API request deduplication mechanism in the platform:

API_DEDUPLICATION_ENABLE `bool`

Description: Enable deduplication
Default: true
Required: No

API_DEDUPLICATION_LEVELDB_PATH `string`

Description: Directory for the LevelDB storage
Default: “data/api/deduplication”
Required: No

API_DEDUPLICATION_LIVE_TIME_SEC `int`

Description: Maximum number of seconds between identical requests. If more seconds pass between req
Default: 86400
Required: No

API_DEDUPLICATION_PERIOD_SEC `int`

Description: Deduplication database cleanup period (in seconds). Default – 30 minutes
Default: 1800

Required: No

SMS event listener

With these parameters you can override system connection settings for the procsmslisten process:

SMS_LISTENER_UNIXSOCK_PATH `string`
Description: Path to the UNIX socket of the process
Default: "UNIX_SOCKETS_PATH/akd-smslistener.socket"
Required: No

SMS_LISTENER_IP `string`
Description: Process host address
Default: "127.0.0.1"
Required: No

SMS_LISTENER_PORT `int`
Description: Process port
Default: 8803
Required: No

Automation Scenarios

With these parameters you can override system settings of the procworkflow process, responsible for eve

WORKFLOW_TASK_REQUEUE_SEC `int`
Description: Interval between attempts to send a profile to an inactive chain
Default: 60
Required: No

WORKFLOW_CACHE_EXPIRATION_SEC `int`
Description: Event cache retention time
Default: 15
Required: No

LEADSAVER_LIST_CACHE_EXPIRATION_SEC `int`
Description: Retention time for profile import event cache
Default: 15
Required: No

WORKFLOW_LOOP_LIMIT `int`
Description: Limit on loops in a chain
Default: 20
Required: No

WORKFLOW_LOOP_AUTODELAY_SEC `int`
Description: Delay between loops in a chain
Default: 60
Required: No

WORKFLOW_ITERATION_LIMIT `int`
Description: Number of iterable tasks per call, 0 = no limit
Default: 0
Required: No

WORKFLOW_ITERATION_PERIOD_SEC `int`
Description: Interval between task iteration calls
Default: 3
Required: No

WORKFLOW_ITERATION_TIMEOUT_SEC `int`
Description: Request wait time when reading from the iteration channel
Default: 5
Required: No

WORKFLOW_RPC_TIMEOUT `int`
Description: RPC server response timeout
Default: 15
Required: No

WORKFLOW_KV_EXPIRATION_SEC `int`
Description: Retention time for task identifiers in the KV store
Default: 3600

Required: No

WORKFLOW_ITERATION_BUFFER_SIZE `int`

Description: Buffer size for the iteration channel
Default: 0
Required: No

WORKFLOW_CONNECTION_TIMEOUT `int`

Description: Connection timeout duration. To configure reconnection attempt timeout, slave processes mu
Default: 15
Required: No

WORKFLOW_READ_TIMEOUT `int`

Description: Request wait time after which the connection closes (0 = no timeout)
Default: 60
Required: No

WORKFLOW_TASK_TIMEOUT_SEC `int`

Description: Sets the maximum time a Scenario task may remain in processing. If the task does not comp
Default: 300
Required: No

WORKFLOW_WORKERS_DISTRIBUTION_SIZE `int`

Description: Number of handler groups
Default: 16
Required: No

WORKFLOW_READERS_DISTRIBUTION_SIZE `int`

Description: Number of handlers in each group
Default: 4
Required: No

WORKFLOW_READER_QUERY_HINT `bool`

Description: Forces the use of the `"idx_1_t_1_a_1_c_1"` **index in database queries**
Default: false
Required: No

WORKFLOW_FREEZE_TTL_SEC `int`

Description: Defines the expiration date until which workflow tasks are frozen
Default: 31536000 (365 days)
Required: No

WORKFLOW_SELECTED_MODE `string`

Description: Controls the operating mode of RPC clients in the workflow. Supported values:

- **RandomSelect**
- **RoundRobin**
- **WeightedRoundRobin**
- **WeightedICMP**
- **ConsistentHash**
- **Closest**

Default: “RoundRobin”
Required: No

WORKFLOW_API_NODE_DEBUG_MODE `bool`

Description: Enables/disables debug mode for HTTP requests in the feedback node
Default: false
Required: No

WORKFLOW_CONNECTION_MODE_NEW `bool`

Description: Controls the connection mode to the database and account caches:

- **true — uses a shared cache, accessible by all nodes in the workflow chain**
- **false — each node uses its own connection**

Default: true
Required: No

WORKFLOW_NODE_LOG_ROTATE `int`

Description: Maximum number of errors stored for each node in the database
Default: 10
Required: No

WORKFLOW_MASSCAMP_WORKER_SIZE `int`

Description: Number of workers for trigger mailings in the workflow
Default: Number of logical CPU cores (up to 16)
Required: No

WORKFLOW_MASSCAMP_REFRESH_SEC `int`

Description: Heartbeat interval inside each trigger mailing worker in the workflow
Default: 30
Required: No

WORKFLOW_COUNTER_TIMEOUT_SEC `int`

Description: The interval (in seconds) after which worker counters are automatically reset during scenari
Default: 5
Required: No

WORKFLOW_EVENT_CONTEXT_LENGTH `int`

Description: The limit on the number of events whose context is stored in the scenario. The value is alway
Default: 20
Required: No

COND_NODE_WORKER_LIMIT `int`

Description: Limit on the number of workers for the condition node
Default: 8
Required: No

COND_NODE_BATCH_SIZE `int`

Description: Size of the batch collected by the condition node worker
Default: 256
Required: No

COND_NODE_BATCHS_LIMIT `int`

Description: Limit on the number of batches collected by the condition node worker.
If `COND_NODE_BATCHS_LIMIT` **is less than** `COND_NODE_WORKER_LIMIT`**, then it is set equal to** `COND_NODE_WORKER_LIMIT`**.**
Default: 200
Required: No

COND_NODE_BATCH_ACCUMULATION_TIME_MS `int`

Description: Accumulation time for condition node tasks
Default: 300
Required: No

COND_NODE_WORKERS_OBSERVATION_PERIOD_MS `int`

Description: Interval for collecting reports about the state of the condition node worker pool. If the value
Default: 0
Required: No

COND_NODE_BATCH_EXECUTION_TIMEOUT_MS `int`

Description: Maximum waiting time for processing a batch in the Condition node. If the condition check pr
Default: 15000
Required: No

“Feedback” node — HTTP call to an external service

WORKFLOW_FEEDBACK_NODE_HTTP_TIMEOUT `int`

Description: HTTP request timeout to an external service
Default: 15
Required: No

WORKFLOW_FEEDBACK_NODE_SOCKS_TIMEOUT `int`

Description: Socket timeout
Default: 15
Required: No

WORKFLOW_FEEDBACK_NODE_RETRY_COUNT `int`

Description: Number of request attempts
Default: 3
Required: No

WORKFLOW_FEEDBACK_NODE_RETRY_TIMEOUT `int`

Description: Interval between attempts
Default: 15
Required: No

Segment settings

With these parameters you can fine-tune customer profile segmentation for working with large audiences

SEGMENT_DISK_SORT_PATH `string`

Description: Directory for storing segment cache
Default: “{BASED}/data/segment/”
Required: No

SEGMENT_SORT_IN_MEM `bool`

Description: Enables storing segment data for sorting in memory
Default: false
Required: No

SEGMENT_ITER_CACHE_PATH `string`

Description: Directory for storing cache of nodes of type “Condition” in Scenarios
Default: “{BASED}/data/workflow/segment”
Required: No

SEGMENT_ITER_CACHE_DELAY_SECOND `int`

Description: Time for which external data in segments is cached
Default: 60
Required: No

SEGMENT_ITER_CACHE_BATCH_WRITE_SIZE `int`

Description: The batch size for writing data to the cache. Query results are accumulated in a buffer and w
Required: No

SEGSTATIC_WORKER_SIZE `int`

Description: Number of static segment handlers
Default: 8 (no more than 1 per 2 CPU cores)
Required: No

SEGSTATIC_DAILY_REFRESH_HOUR `int`

Description: Time to start daily recalculation of static segments (hours and minutes) for which the “Updat
Default: 23
Required: No

SEGSTATIC_DAILY_REFRESH_MIN `int`

Description: See description for SEGSTATIC_DAILY_REFRESH_HOUR
Default: 59
Required: No

SEGSTATIC_DAILY_REFRESH_COUNT `int`

Description: Maximum number of parallel runs of the static segment recalculation process
Default: 2
Required: No

Static segment synchronization

PROCINTEGRAS_STATSEG_WORKER_SIZE `int`

Description: Number of RMQ consumers
Default: 1
Required: No

PROCINTEGRAS_STATSEG_INTEG_AMOUNT_OF_WORKERS `int`

Description: Number of synchronization workers
Default: 4
Required: No

PROCINTEGRAS_STATSEG_INTEG_MAX_RETRIES_AMOUNT `int`

Description: Limit on connection attempts to the SQL server
Default: 5
Required: No

PROCINTEGRAS_STATSEG_INTEG_RETRY_TIMEOUT `int`

Description: Connection attempt duration to the SQL server
Default: 15
Required: No

Profile deletion by segment

PROFILES_REMOVAL_MIN_MONTHS `int`

Description: Minimum number of months without interaction after which profiles will be deleted; 0 — delet
Default: 1
Required: No

Caching settings for external data from segmentation queries

CACHE_SERVICE_INSERT_BATCH_SIZE `int`

Description: Insert batch size for the caching service (maximum — 1,000,000)
Default: 1,000,000
Required: No

CACHE_SERVICE_TTL_MULTIPLIER `int`

Description: Cache retention multiplier; the retention period is multiplied by this value
Default: 4
Required: No

CACHE_SERVICE_CLEANER_PERIOD `int`

Description: Frequency of checking unused caches (in minutes)
Default: 60
Required: No

CACHE_SERVICE_EXPIRED_PERIOD `int`

Description: Minimum cache retention time in seconds (default — 1 week)
Default: 10080
Required: No

CACHE_SERVICE_TESTING_MODE `bool`

Description: Testing mode to allow a retention period of less than 1 week
Default: false
Required: No

Segmentation query settings

DISABLE_QUERY_EDITOR `bool`

Description: Prohibits editing queries in the user panel
Default: false
Required: No

INTERNAL_SQL_ROWS_LIMIT `int`

Description: Limits the number of options in the platform (maximum — 10,000)
Default: 1000
Required: No

Yandex Audience settings

EXTERNAL_ID_COLL_EXPIRED_TIME_LIMIT_SECOND `int`

Description: How many seconds are given to check whether the integration has expired
Default: 180
Required: No

Process configuration

With these parameters you can override the configuration of platform processes, for example, system par

PROC_PUB_STAT_SIZE `int`

Description: Number of statistics senders
Default: 1
Required: No

PROC_PUB_STAT_QUEUE_SIZE `int`

Description: Statistics sending queue size
Default: 5000
Required: No

PROC_PUB_HOOK_SIZE `int`

Description: Number of web hook senders
Default: 1
Required: No

PROC_PUB_HOOK_QUEUE_SIZE `int`

Description: Web hook sending queue size
Default: 5000

Required: No

PROC_PREFETCH_SIZE `int`

Description: “prefetch size” setting of the RMQ queue consumer
Default: 5000
Required: No

PROC_OPEN_WORKER_SIZE `int`

Description: Number of handlers for “Message opened” events
Default: 1
Required: No

PROC_CLICK_WORKER_SIZE `int`

Description: Number of handlers for “Link click” events
Default: 1
Required: No

PROC_UNSUB_WORKER_SIZE `int`

Description: Number of handlers for “Unsubscribe” events
Default: 1
Required: No

PROC_APICALL_WORKER_SIZE `int`

Description: Number of API request handlers
Default: 1
Required: No

PROC_CONFIRMSUB_WORKER_SIZE `int`

Description: Number of handlers for “Subscription confirmation to resource” events
Default: 1
Required: No

PROC_CONFIRM_WORKER_SIZE `int`

Description: Number of handlers for “Global subscription confirmation” events
Default: 1
Required: No

PROC_MANDRILL_WORKER_SIZE `int`

Description: Number of handlers for Mandrill MTA events
Default: 1
Required: No

PROC_READ_WORKER_SIZE `int`

Description: Number of handlers for “Email read” events
Default: 1
Required: No

PROC_PIXEL_WORKER_SIZE `int`

Description: Number of handlers for “Pixel activation” events
Default: 16 (no more than 1 per 1 CPU core)
Required: No

PROC_PIPER_WORKER_SIZE `int`

Description: Number of event handlers in the Custom channel
Default: 16 (no more than 1 per 1 CPU core)
Required: No

PROC_PIPER_PERIOD_SEC `int`

Description: Cache update period for Custom channels in procpiper
Default: 15
Required: No

PROC_PIXEL_UNDEFINED_EXPIRATION_SEC `int`

Description: Retention time for data on visits that did not become profiles
Default: 259200 (3 days)
Required: No

PROC_RPC_READ_TIMEOUT `int`

Description: Request wait time after which the connection closes (0 = no timeout)
Default: 60
Required: No

PROC_RPC_CONNECTION_TIMEOUT `int`

Description: RPC connection timeout
Default: 30

Required: No

STAT_WORKER_SIZE `int`

Description: Number of statistics handlers
Default: 4
Required: No

WEBVER_WORKER_SIZE `int`

Description: Number of Email web-version handlers
Default: 4 (no more than 4, and no more than 1 per 1 CPU core)
Required: No

WEBVER_CLEANER_ACCOUNT_WORKER_SIZE `int`

Description: Number of accounts checked simultaneously when cleaning Email web versions
Default: 4 (no more than 4, and no more than 1 per 2 CPU cores)
Required: No

APICONTROL_PERIOD_SEC `int`

Description: API process cache cleanup period
Default: 60
Required: No

PROCWORKFLOW_PERIOD_SEC `int`

Description: procworkflow process cache cleanup period
Default: 15
Required: No

WEBCONTROL_PERIOD_SEC `int`

Description: webcontrol process cache cleanup period
Default: 60
Required: No

WEBCONTROL_PERIOD_SUPERVISOR_SEC `int`

Description: Node status check period
Default: 15
Required: No

CHECK_PLACEMENTS_WORKER_SIZE `int`

Description: Number of connections handlers with advertising placements
Default: 4
Required: No

CHECK_PLACEMENTS_PERIOD_SEC `int`

Description: Cache cleanup period for advertising placement connection handlers
Default: 900
Required: No

STATAGGREGATOR_SYNC_PERIOD_SEC `int`

Description: Statistics aggregator synchronization frequency
Default: 30
Required: No

WEBCONTROL_SESSION_ALLOW_CHNET `bool`

Description: Do not reset the session when the user's IP address changes
Default: false
Required: No

AUTO_RUN_AS_USER `string`

Description: Username under which platform services will be launched
Default: "altcraft"
Required: No

AUTO_RUN_AS_GROUP `string`

Description: Group name under which platform services will be launched
Default: "altcraft"
Required: No

Webhook listener configuration

With these parameters you can override system connection settings for the prochook process:

PROC_HOOK_WORKER_SIZE `int`

Description: Number of web hook event handlers
Default: 16 (no more than 1 per 1 CPU core)

Required: No

PROCHOOK_PERIOD_SEC `int`

Description: Event capture cache cleanup period
Default: 15
Required: No

HOOK_HTTP_CACHE_CLEANUP_SEC `int`

Description: HTTP cache cleanup frequency
Default: 3600
Required: No

HOOK_HTTP_CONN_TIMEOUT_SEC `int`

Description: HTTP hook connection timeout
Default: 120
Required: No

HOOK_HTTP_BATCH_TIMEOUT_SEC `int`

Description: Hook batch accumulation time
Default: 1
Required: No

HOOK_RETRY_DATAPATH `string`

Description: Directory for temporary storage of messages for retry
Default: “{BASED}/hookdata”
Required: No

HOOK_RETRY_PERIOD_SEC `int`

Description: Retry sending frequency
Default: 900
Required: No

HOOK_RETRY_TIMES `int`

Description: Retry attempt limit
Default: 4
Required: No

HOOK_CACHE_EXPIRATION_SEC `int`

Description: Hook cache validity period
Default: 60
Required: No

HOOK_CACHE_CLEANUP_SEC `int`

Description: Hook cache cleanup frequency
Default: 60
Required: No

Trigger listener configuration

With these parameters you can override system connection settings for the proctrigger process:

PROCTRIGGER_PERIOD_SEC `int`

Description: proctrigger process cache cleanup period
Default: 15
Required: No

PROCTRIGGER_LATENCY_MSEC `int`

Description: proctrigger process latency
Default: 500 (milliseconds)
Required: No

TRIGCONTROL_WORKER_SIZE `int`

Description: Number of trigger mailing handlers
Default: 16 (no more than 1 per 1 CPU core)
Required: No

TRIGCONTROL_CONSUMER_WORKER_SIZE `int`

Description: Number of consumers of the “oxy_triggers” queue
Default: 8 (no more than 1 per 2 CPU cores)
Required: No

TRIGCONTROL_PRIOR_CONSUMER_WORKER_SIZE `int`

Description: Number of consumers of the “oxy_triggers_prior” queue
Default: 8 (no more than 1 per 2 CPU cores)

Required: No

Email settings

COUNT_COPY_EMAILS `int`

Description: Limit on copies of sent Email messages
Default: 32
Required: No

MTA_DEFAULT_TTL `int`

Description: Message lifetime on the sender (in seconds). If the message is not sent within this time, it is
Default: 86400
Required: No

MTA_MIN_TTL `int`

Description: Minimum possible TTL for a message
Default: 3600
Required: No

MTA_MAX_TTL `int`

Description: Maximum possible TTL for a message
Default: 259200
Required: No

MTA_ATTACH_MAX_SIZE `int`

Description: Maximum attachment size for Email
Default: 2147483648
Required: No

MTA_STAT_SYNC_TIME `int`

Description: Sender statistics synchronization frequency
Default: 120
Required: No

MTA_LOGLEVEL `string`

Description: Sender logging level
Default: “INFO”
Required: No

Disk space

MTA_FREE_AVAILABLE_SPACE_WARN_PERCENT `float64`

Description: Percentage of free space at which a notification is sent to administrators
Default: 10
Required: No

MTA_FREE_AVAILABLE_SPACE_LOCK_PERCENT `float64`

Description: Percentage of free space at which the MTA stops accepting new messages
Default: 5
Required: No

SMS sending settings

PROC_SMS_EVENTS_PERIOD_SEC `int`

Description: Interval between processing tasks for obtaining SMS message statuses
Default: 900
Required: No

GET_NOTIFICATION_WORKER_SIZE `int`

Description: Number of event handlers in SMS
Default: 4
Required: No

SMS_DEFAULT_TTL `int`

Description: Time-to-live of the task to obtain the final status of an SMS message. If not specified, it is inh
Default: -1
Required: No

SHORTER_STORE_TTL_SEC `int`

Description: Time-to-live of shortened links in storage
Default: 7776000

Required: No

Push notification settings

Retry mechanism configuration

PUSH_SENDER_RETRY_MAX_AMOUNT `int`

Description: Maximum number of push message resend attempts
Default: 3
Required: No

PUSH_SENDER_RETRY_INTERVAL `int`

Description: Time between resend attempts (in seconds)
Default: 1800
Required: No

PUSH_SENDER_RETRY_SCAN_INTERVAL `int`

Description: Frequency of scanning the retry database for new records (in seconds)
Default: 60
Required: No

Yandex AppMetrica push

APPMETRICA_PUSH_BATCH_SIZE `int`

Description: Maximum batch size
Default: 25000
Required: No

APPMETRICA_MAX_PUSH_REQUEST_SIZE_BYTE `int`

Description: Maximum request size; if exceeded, split into several requests
Default: 52428800
Required: No

APPMETRICA_REQUEST_LIMIT_COUNT `int`

Description: Number of allowed requests per day
Default: 5000
Required: No

APPMETRICA_BATCH_WORKER_SIZE `int`

Description: Number of workers to process a batch of messages
Default: By number of cores (no more than 8)
Required: No

Custom channel

These parameters are used to configure Custom channels for sending messages:

Pipe router configuration

PIPE_ROUTER_CHAN_SIZE `int`

Description: Pipe router channel size
Default: 100
Required: No

PIPE_ROUTER_WORKER_SIZE `int`

Description: Number of pipe router handlers
Default: 1
Required: No

PIPE_ROUTER_STOP_WORKER_SIZE `int`

Description: Number of handlers for events placed into the RMQ queue when stopping the platform to avoi
Default: 8
Required: No

PIPE_ROUTER_STOP_CHAN_SIZE `int`

Description: Channel size (see description above)
Default: 500
Required: No

SCHEDULER_PIPE_ROUTER_TTL `int`

Description: Router activity time for scheduled pipes
Default: 86400

Required: No

SCHEDULER_PIPE_ROUTER_PERIOD `int`

Description: Frequency of checking events to start the scheduled pipeline
Default: 10
Required: No

Incoming request settings (listener)

LISTENER_IP `string`

Description: Host address for incoming requests
Default: “127.0.0.1”
Required: No

LISTENER_PORT `int`

Description: Port for incoming requests
Default: 8069
Required: No

LISTENER_CHAN_SIZE `int`

Description: Incoming requests channel size
Default: 100
Required: No

LISTENER_WORKER_SIZE `int`

Description: Number of handlers for incoming requests
Default: 16 (no more than 1 per 1 CPU core)
Required: No

LISTENER_UNIXSOCK_PATH `string`

Description: Path to the UNIX socket of the process
Default: “UNIX_SOCKETS_PATH/akd-trackinglistener.socket”
Required: No

LISTENER_REFRESH_PERIOD `int`

Description: Handler cache cleanup frequency
Default: 30
Required: No

Pipe settings

PIPE_HTTP_MAX_CONN `int`

Description: Maximum number of simultaneous `http_request` **pipe requests**
Default: 128
Required: No

PIPE_HTTP_TIMEOUT `int`

Description: Request timeout for the `http_request` **pipe**
Default: 15
Required: No

PIPE_WORKER_SIZE `int`

Description: Number of pipe handlers
Default: 1
Required: No

PIPE_EVENTER_WORKER_SIZE `int`

Description: Number of eventer pipe handlers
Default: 16 (no more than 1 per 1 CPU core)
Required: No

PIPE_EVENTER_CHAN_SIZE `int`

Description: Eventer pipe channel size
Default: 100
Required: No

PIPE_SCHEDULER_WORKER_SIZE `int`

Description: Number of scheduler pipe handlers
Default: 1
Required: No

PIPE_SCHEDULER_CHAN_SIZE `int`

Description: Scheduler pipe channel size
Default: 100

Required: No

PIPE_SCHEDULER_DELAY `int`

Description: Scheduler pipe delay
Default: 60 (seconds)
Required: No

PIPE_USER_AGENT `string`

Description: User agent for the pipe's HTTP request
Default: "altcraft platform"
Required: No

Integration with Yandex AppMetrica

These parameters are used for fine-tuning Yandex AppMetrica event captures used for importing custome

APPMETRICA_EVENTS_REFRESH_WORKERS_PERIOD_SEC `int`

Description: Event handler cache cleanup frequency
Default: 60
Required: No

APPMETRICA_EVENTS_REQUEST_PERIOD_SEC `int`

Description: Frequency of event retrieval requests
Default: 60
Required: No

APPMETRICA_EVENTS_DELAY_COMPENSATIONS_REQUEST_PERIOD_SEC `int`

Description: If lag occurs, overrides `APPMETRICA_EVENTS_REQUEST_PERIOD_SEC`
Default: 600
Required: No

APPMETRICA_EVENTS_REQUEST_DATE_RANGE_SEC `int`

Description: Event request time range
Default: 60
Required: No

APPMETRICA_PROC_GOALS_WORKER_SIZE `int`

Description: Number of goal registration handlers
Default: 2
Required: No

APPMETRICA_PROC_GOALS_PREFETCH_SIZE `int`

Description: "prefetch_size" setting of the RMQ queue for goal registration
Default: 2000
Required: No

APPMETRICA_PROC_IMPORTS_WORKER_SIZE `int`

Description: Number of profile import handlers
Default: 1
Required: No

APPMETRICA_PROC_IMPORTS_PREFETCH_SIZE `int`

Description: "prefetch_size" setting of the RMQ queue for profile import
Default: 2000
Required: No

APPMETRICA_EVENTS_RESPONSE_LIMIT `int`

Description: Limit on the number of events received from AppMetrica
Default: 10000
Required: No

Message templates settings

MAX_GALLERY_FILE_SIZE `int`

Description: Maximum image size uploaded to the gallery (in bytes)
Default: 2097152 (2 MB)
Required: No

MAX_REMOTE_CONTENT_BYTES `int`

Description: Maximum size of dynamic JSON (in bytes)
Default: 1048576 (1 MB)
Required: No

Profile import and update

These parameters configure the processes responsible for creating and updating customer profiles, inclu

LEADSAVER_WORKER_SIZE `int`

Description: Number of import handlers
Default: 16 (no more than 13 per 10 CPU cores)
Required: No

MASSTASK_DB_POOLSIZE `int`

Description: Connection pool size to MongoDB for dbimport
Default: 8 (no more than 1 per 4 CPU cores)
Required: No

Scheduled import

DB_IMPORT_LAST_RUN_HISTORY `int`

Description: Retention limit for history of last runs
Default: 10
Required: No

DB_IMPORT_BATCH_SIZE `int`

Description: Number of records processed simultaneously
Default: 2500
Required: No

DB_IMPORT_STATS_PERIOD `int`

Description: Statistics update period
Default: 15
Required: No

LDAP access settings

The LDAP object defines the configuration for accessing the admin panel using the company's SSO. You c

HOST `string`

Description: Primary LDAP server
Default: "ALTCRAFT"
Required: Yes

PRIMARY_PORT `int`

Description: Primary LDAP port
Default: "MAIN"
Required: Yes

SECONDARY_HOST `string`

Description: Secondary LDAP server
Default: "60"
Required: Yes

SECONDARY_PORT `int`

Description: Secondary LDAP port
Default: 19888
Required: Yes

RETRY_CONNECTION_COUNT `int`

Description: Limit on reconnection attempts to the directory service
Default: 30
Required: No

BINDLOGIN `string`

Description: Authorization login in the directory service
Default: None
Required: Yes

BINDPASS `string`

Description: Authorization password in the directory service
Default: None
Required: Yes

IS_SSL `bool`

Description: Use encryption for the connection
Default: false
Required: No

CRT_PATH `string`

Description: Path to the encryption certificate
Default: None
Required: No

KEY_PATH `string`

Description: Path to the encryption key
Default: None
Required: No

AUTH_GROUP `string`

Description: DistinguishedName of the access group for Platform Administrators
Default: None
Required: Yes

LDAP_ONLY `bool`

Description: False — use LDAP and regular authorization; True — LDAP authorization only
Default: None
Required: Yes

LDAP_CHECK_USER_TIME_PERIOD_SEC `int`

Description: Time before rechecking user privileges in the directory service
Default: 300
Required: No

Notification settings

The following parameters are used to send notifications and reports by Email and for notifications in Tele

“NOREPLY_MAILER”: {} — Email sending server settings

server `string`

Description: Host address of the Email notifications and reports sending server
Default: None
Required: Yes

port `int`

Description: Port of the Email notifications and reports sending server
Default: None
Required: Yes

email `string`

Description: Sender Email address
Default: None
Required: Yes

login `string`

Description: Authorization login on the server
Default: None
Required: Yes

pass `string`

Description: Authorization password on the server
Default: None
Required: Yes

insecure `bool`

Description: Allows an unencrypted connection
Default: false
Required: No

force_no_ssl `bool`

Description: Force the use of an unencrypted connection
Default: false
Required: No

helo `string`

Description: Value for the HELO greeting command
Default: None
Required: No

“TELEGRAM_NOTIFIER”: {} — Telegram chatbot settings

bot_id `int`

Description: Telegram bot ID
Default: None
Required: Yes

api_key `string`

Description: Bot API key
Default: None
Required: Yes

chat_id `int`

Description: Telegram chat ID
Default: None
Required: Yes

"NOTIFICATION": {} — notification settings

HARDBOUNCES_WARNING_PERCENT `float32`

Description: Threshold percentage of hard bounces — 500x Email sending errors
Default: 0.5
Required: No

SOFTBOUNCES_WARNING_PERCENT `float32`

Description: Threshold percentage of soft bounces — 400x Email sending errors
Default: 10
Required: No

COMPLAINS_WARNING_PERCENT `float32`

Description: Threshold percentage of Email complaints
Default: 0.07
Required: No

SELECT_INTERVAL_HOURS `int`

Description: Frequency of statistics analysis for notifications
Default: 1
Required: No

NOTIFY_INTERVAL_HOURS `int`

Description: Frequency of sending identical notifications
Default: 12
Required: No

ClickHouse configuration

The list below contains all possible parameters of the `CLICKHOUSE_SYSTEM` **object. You can read more in the C**

HOST `string`

Description: Host address
Default: None
Required: Yes

PORT `int`

Description: Host port
Default: None
Required: Yes

USER `string`

Description: Username for connection
Default: None
Required: Yes

MAX_CONNECTION `int`

Description: Maximum number of connections
Default: 100
Required: No

CONN_TIMEOUT `int`

Description: Connection timeout (in seconds)
Default: 15
Required: No

READ_CONN_TIMEOUT_SEC `int`

Description: Request wait time after which the connection closes (in seconds)
Default: 300
Required: No

PASSWORD `string`

Description: User password
Default: None
Required: Yes

IS_DEBUG `bool`

Description: Enable debug mode
Default: false
Required: No

DATABASE_NAME `string`

Description: Database name
Default: “altcraft_system”
Required: Yes

REPLICATED `bool`

Description: Support for ReplicatedMergeTree replicated tables
Default: false
Required: No

ALT_HOSTS `array<object>`

Description: List of hosts for load balancing
Default: None
Required: No

STRATEGY `string`

Description: Host handling strategy (“in_order”, “random”)
Default: “in_order”
Required: No

CLICKHOUSE_STORNER_ENABLE `bool`

Description: Deduplication mode
Default: No
Required: No

CLICKHOUSE_STORNER_DUP_EVENET_MAX `int`

Description: Processed duplicate count
Default: 1000
Required: No

CLICKHOUSE_STORNER_TESTING_PERIOD_SEC `int`

Description: Testing period override
Default: No
Required: No

COMPRESSION_METHOD `string`

Description: Data compression mode (“lz4_frame”, “zstd”, “none”)
Default: “none”
Required: No

COMPRESSION_LEVEL `int`

Description: Data compression level
Default: 3
Required: No

ENABLE_WORKFLOW_CH_QUERY_OPTIMIZATION `bool`

Description: Query optimization mode, sets the parameter `settings max_threads=1`
Default: None
Required: No

DISTRIBUTED_DDL_OUTPUT_MODE `string`

Description: Behavior when a ClickHouse node is unavailable (“throw”, “never_throw”, “null_status_on_tim
Default: “never_throw”
Required: No

DISTRIBUTED_DDL_TASK_TIMEOUT `int`

Description: Timeout for responses to requests from all hosts in the ClickHouse cluster (in seconds)
Default: 3
Required: No

SSL_ON `bool`

Description: Enable SSL encryption for the connection
Default: false
Required: No

SSL_CRT_PATH `string`

Description: Path to the SSL encryption certificate
Default: None
Required: No

SSL_KEY_PATH `string`

Description: Path to the SSL encryption key
Default: None
Required: No

SSL_CA_CERT_PATH `string`

Description: Path to CA SSL
Default: None
Required: No

INSECURE_SKIP_VERIFY `bool`

Description: Disable mandatory verification of encryption certificates
Default: false
Required: No

Process logging

With these parameters you can enable collection of platform logs into the operating system's syslog, and

LOG_SYSLOG_ENABLE `bool`

Description: Enable collection of platform logs into syslog
Default: false
Required: No

LOG_ROTATE_WEEK_DAY `int`

Description: Log rotation day of the week
Default: 0
Required: No

LOG_ROTATE_HOUR `int`

Description: Log rotation hour
Default: 0
Required: No

LOG_ROTATE_MINUTE `int`

Description: Log rotation minute
Default: 0
Required: No

LOG_ROTATE_SECOND `int`

Description: Log rotation second
Default: 0
Required: No

LOG_ROTATE_MAX_MB `int`

Description: Maximum log file size
Default: 100
Required: No

LOG_ROTATE_MAX_FILES `int`

Description: Maximum number of log files
Default: 7
Required: No

LOG_ROTATE_ZIP_FILES `int`

Description: Number of archived log files
Default: 2
Required: No

DEFAULT_LOG_LEVEL `string`

Description: Default logging level
Default: "INFO"
Required: No

LOG_LEVEL `string`

Description: Logging level
Default: None
Required: No

JSON_SYSLOG `bool`

Description: Write information to syslog in JSON format
Default: false
Required: No

Override settings for an account

Some settings can be overridden within a specific account. Use the `ACCOUNT_OVERRIDES` **field:**
"ACCOUNT_OVERRIDES": { "‹account number›": {} }

FIREFOX_CUSTOM_SENDING_URL `string`

Description: Overriding API URL for the Mozilla Firefox provider
Default: None
Required: No

SAFARI_CUSTOM_SENDING_URL `string`

Description: Overriding API URL for the Safari provider
Default: None
Required: No

FIREBASE_LEGACY_CUSTOM_SENDING_URL `string`

Description: Overriding API URL for providers in the Firebase (Legacy) group
Default: None
Required: No

FIREBASE_APIv1_CUSTOM_SENDING_URL `string`

Description: Overriding API URL for providers in the Firebase (API v1) group
Default: None
Required: No

HUAWEI_CUSTOM_SENDING_URL `string`

Description: Overriding API URL for providers in the Huawei group
Default: None
Required: No

RUSTORE_CUSTOM_SENDING_URL `string`

Description: Overriding API URL for providers in the RuStore group
Default: None
Required: No

YM_APPMETRICA_CUSTOM_SENDING_URL `string`

Description: Overriding API URL for providers in the Appmetrica group
Default: None
Required: No

APN_TEST_HOST `string`

Description: Overriding API URL for the iOS APNs provider
Default: None
Required: No

FIREBASE_APIv1_DISABLE_OAUTH `string`

Description: Disables OAUTH authorization for providers in the Firebase (API v1) group
Default: None
Required: No

Firebase settings

FIREBASE_APIv1_DISABLE_OAUTH `bool`

Description: Disables OAUTH authorization for providers in the Firebase (API v1) group
Default: false
Required: No

FIREBASE_PUSH_PUBLISHER_WORKER_SIZE `int`

Description: Number of workers for publishing responses to messages in the RabbitMQ queue
Default: 8
Required: No

FIREBASE_PUSH_PUBLISHER_QUEUE_SIZE `int`

Description: Response publishing queue size. Defines how many messages can be temporarily stored in th
Default: 500
Required: No

FIREBASE_PUSH_CONSUMER_WORKER_SIZE `int`

Description: Number of RabbitMQ queue consumers
Default: 8
Required: No

FIREBASE_PUSH_CONSUMER_QUEUE_SIZE `int`

Description: RabbitMQ queue limit
Default: 500 Required: No

FIREBASE_PUSH_SENDER_WORKER_SIZE `int`

Description: Number of send workers
Default: 32
Required: No

FIREBASE_PUSH_RETRY_WORKER_SIZE `int`

Description: Number of retry workers
Default: 32
Required: No

FIREBASE_PUSH_BATCH_SIZE `int`

Description: Number of messages to send in a Firebase batch (range 0–500)
Default: 500
Required: No

FIREBASE_PUSH_BATCH_TIMEOUT_SEC `int`

Description: Frequency of sending a batch of size `FIREBASE_PUSH_BATCH_SIZE` **(in seconds)**
Default: 5
Required: No

FIREBASE_BODY_TRACE_LOG_ENABLED `bool`

Description: Prints the request body sent to Firebase and outputs it to the trace log
Default: false
Required: No

FIREBASE_PUSH_BATCH_WORKERS `int`

Description: Number of batch workers (range 10–500)
Default: 200 Required: No

Setting limits on the number of requests

With these parameters you can override system settings that set limits on the number of the following req

- `/api/v1.1/profiles/import_batch`
- `/api/v1.1/profiles/update_batch`
- `/api/v1.1/profiles/add_batch`

If `API_MAX_REQUESTS_COUNT` **is exceeded, the server will start responding with status code 429.**

API_REQUEST_LIMITER_ENABLE `bool`

Description: Enable the limiter
Default: false
Required: No

API_MAX_REQUESTS_COUNT `int`

Description: Maximum number of requests the server can accept for this group of requests
Default: 1000
Required: No

API_EXECUTING_REQUEST_COUNT `int`

Description: Maximum number of identical requests executed simultaneously
Default: 100
Required: No

Machine learning service

With these parameters you can configure the machine learning service that determines the best time to s

ML_REFRESH_TIME_HOURS `$$int`

Description: Hours to start recalculating profile data (by the account's time zone)
Default: 0

Required: No

ML_NUMBER_OF_MONTHS_TO_PREDICT `int`

Description: Number of months of a profile's activity sample. By default, the ML model uses activity data f
Default: 6
Required: No

ML_WORKERS_COUNT_FOR_WRITE `int`

Description: Number of concurrent handlers for calculating the best sending time
Default: 5
Required: No

ML_BATCH_SIZE_FOR_WRITE `int`

Description: Number of parallel processing jobs for the ML service calculation
Default: 1000
Required: No

ML_MODELS_FILE `string`

Description: Direct path to the ML model file in .cbm format if the model needs to be overridden. By defaul
Default: None
Required: No

MAX_PARALLEL_ML_ENRICH_JOBS `int`

Description: Maximum number of parallel runs of the best send time calculation
Default: Inherited from `MAX_PARALLEL_ANY_JOBS`
Required: No

MAX_PARALLEL_ANY_JOBS `int`

Description: Maximum number of parallel runs for any job type if the parameters above are not set. The de
Default: -1
Required: No

Configuring parallel job run limits

With these parameters you can override the maximum number of parallel runs for various processes (jobs

MAX_PARALLEL_SEGSTAT_RECALC_JOBS `int`

Description: Maximum number of parallel runs of the static segment recalculation process
Default: None
Required: No

MAX_PARALLEL_SEGSTAT_SYNC_JOBS `int`

Description: Maximum number of parallel runs of the process that sends static segment data to external S
Default: None
Required: No

MAX_PARALLEL_ML_ENRICH_JOBS `int`

Description: Maximum number of parallel runs of the machine-learning-based profile data recalculation pr
Default: None
Required: No

MAX_PARALLEL_DBIMPORT_JOBS `int`

Description: Maximum number of parallel runs of the scheduled import from external databases
Default: None
Required: No

MAX_PARALLEL_RESCAN_DATABASES_JOBS `int`

Description: Maximum number of parallel runs of the profile recount process in the database
Default: None
Required: No

MAX_PARALLEL_WF_CAPT_JOBS `int`

Description: Maximum number of parallel runs of the message preview generation process in the platform
Default: None
Required: No

MAX_PARALLEL_CAMPAIGN_JOBS `int`

Description: Maximum number of parallel runs of the mailing start process
Default: None
Required: No

MAX_PARALLEL_ML_ENRICH_JOBS `int`

Description: Maximum number of parallel runs of the best send time calculation process
Default: Inherited from `MAX_PARALLEL_ANY_JOBS`

Required: No

MAX_PARALLEL_ANY_JOBS `int`

Description: Maximum number of parallel runs for any job type if the parameters above are not set. Defaul
Default: -1
Required: No

Global control groups configuration

With these parameters you can override GCG settings:

GCG_MIN_GCG_SIZE `int`

Description: Minimum size of a global control group
Default: 1000
Required: No

GCG_MAX_SOURCE_SIZE `int`

Description: Maximum source data size for a global control group
Default: 50000000
Required: No

GCG_MAX_PERCENT `int`

Description: Maximum size of a global control group as a percentage of the data source
Default: 25
Required: No

GCG_RUN_HOUR `[]int`

Description: Overrides the start time for calculating global control groups (default 00:00 in the account tir
Default: 00.00
Required: No

Other settings

USER_CACHE_EXPIRATION_SEC `int`

Description: User cache expiration time
Default: 15
Required: No

USER_CACHE_CLEANUP_SEC `int`

Description: User cache cleanup interval
Default: 60
Required: No

PROMOCODES_EXPIRE_PERIOD `int`

Description: Promo code validity check period
Default: 15
Required: No

CUSTOM_DATA_TTL `int`

Description: Retention time for user data from event captures
Default: 30 (days)
Required: No

SKIP_URL_VALIDATION `bool`

Description: Disables URL validation
Default: true
Required: No

ENABLE_PROCEVENTGEN `bool`

Description: Enables demo data generator
Default: false
Required: No

ALLOW_SKIP_UNSUBS `bool`

Description: Allows saving and sending messages without an unsubscribe link
Default: false
Required: No

SKIP_PROCESSES `array`

Description: Do not start the specified platform processes at startup
Default: []
Required: No

BROWSER_CACHE_CONTROL_SEC `int`

Description: Cache duration for `ak_container.js` **and** `service_worker.js` **in the browser during dynamic connec**
Default: 900
Required: No

WHATSAPP_SEND_LIMIT_PER_SECOND `int`

Description: Limits the sending rate via the WhatsApp* channel
Default: 20 (messages per second)
Required: No

NGINX_TRUSTED_PROXIES `array`

Description: List of trusted proxy IP addresses
Default: [“127.0.0.1”]
Required: No

NGINX_REALIP_HEADER `string`

Description: Header from which the real client IP address will be obtained
Default: “X-Forwarded-For”
Required: No

ENABLE_REMOVE_RELATIONS_DATA_FROM_CH `bool`

Description: Enables deletion of Relation data from ClickHouse
Default: false
Required: No

USE_LIGHTWEIGHT_DELETE_FOR_REMOVE_RELATIONS_DATA `bool`

Description: Enables lightweight delete mode for deleting Relation data from ClickHouse
Default: true
Required: No

NOTIFICATION_LICENSE_DEFAULT_EMAIL `string`

Description: Email address to which license-related notifications will be sent
Default: “license@altcraft.com”
Required: No

NOTIFICATION_LICENSE_WARN_DAYS `int`

Description: Number of days before license expiration after which an email notification will be sent
Default: 14
Required: No

DIE_WHEN_LICENSE_FAILS `bool`

Description: If set to “true”, when license errors are present `proctask` **will not start**
Default: false
Required: No

PROCAUTH_WELCOME_LINK_TTL_SEC `int`

Description: Lifetime of the registration completion and password setup link
Default: 7200 (2 hours)
Required: No

MIN_FORM_BATCH_SIZE `int`

Description: Minimum number of form submissions required to send a notification email (batch mode)
Default: 3
Required: No

MIN_POPUP_FORM_EMAIL_ACTION_SEND_BATCH_SIZE `int`

Description: Minimum number of popup form submissions required to send a notification email (batch mod
Default: 3
Required: No

FORM_EMAIL_SENDING_CACHE_RESCAN_SEC `int`

Description: Time the form data is stored in cache before being sent by email (in seconds)
Default: 60

POPUP_FORM_EMAIL_SENDING_CACHE_RESCAN_SEC `int`

Description: Time the popup form submission data is stored in cache before being sent by email (in secon
Default: 60

The Meta organization, which owns Instagram, Facebook, and WhatsApp, is recognized as extremist and

Domain settings

This guide contains examples of records that should be added to your domains for better deliverability.

The following IP addresses are used as an example:

- **192.168.0.1 — IPv4-address for web interface address;**
- **192.168.0.2 — IPv4-address for message tracking;**
- **192.168.0.3 — IPv4-address for website tracking;**
- **192.168.0.4 — IPv4-address for sender (AKMTA).**

Web interface domain settings

Choose a subdomain and add the following A record.

Domain
altcraft.example.com.

Message tracking domain settings

Choose a subdomain for your from-domain and add the following A record.

Domain
trk.example.com.

You can use unlimited additional tracking domains. The settings will be the same - or you can use CNAME

Web site tracking domain settings

Choose a subdomain and add the following A record.

Domain
pxl.example.com.

You can use unlimited additional tracking domains. The settings will be the same - or you can use CNAME

Sender domain settings (AKMTA)

Choose a domain or a subdomain and add the following records.

Domain
mail.example.com.
4.0.168.192.in-addr.arpa.
example.com.

SPF settings

Sender Policy Framework (SPF) technology allows to verify that sender domain has not been faked. To ma

We recommend to add records for different SPF versions:

Domain	Record type
example.com.	TXT
example.com.	TXT

> CAUTION
>
> **Note that we recommend to use restricting rules where only the enlisted resources are allowed!**

DKIM settings

DomainKeys Identified Mail (DKIM) technology adds to your message a digital signature of your from-dom infrastructure and the public part should be added to your domain as a special subdomain TXT record.

In Altcraft Administrative panel open Setup —> Keys, then paste or generate a new DKIM:

Domain	Record type	
altcraft._domainkey.example.com.	TXT	v=DKIM1; k=rsa; p=AAAAB3NzaC1yc2EAAAADAQABAAABAQDLWvAVr5k+4nneMRKhUeb/yGjRBIpGu9CLXx4LJJb1dUZeNWBCbjpjN0P5hlXD3tymI8aBJ
_domainkey.example.com.	TXT	o=-;

> CAUTION
>
> **Please note, that you can use one DKIM for one sender only.**

DMARC settings

DMARC technology allows mailing servers to decide what to do if SPF or DKIM verification fails.

Here we recommend to restrict messages sent from unknown IP addresses or signed incorrectly.

Domain	Record type
_dmarc.example.com.	TXT

You will get replies to `report@example.com`.

BIMI settings

BIMI technology (Brand Indicators for Message Identification) is used along with SPF, DKIM and DMARC o

To start using BIMI you will need an svg image, that is rectangular and has no extra layers.

Add the following record to your domain settings, specifying the svg file path:

Domain	Record typ
default._bimi.example.com	TXT

You can also use BIMI record generator at https://bimigroup.org/bimi-generator/

LDAP access configuration

Overview

You can configure LDAP access to your Altcraft MP account and Administrative panel. This allows your en
LDAP can be used for both Administrative panel and for account User interface.
A `DistinguishedName` for a group of users includes nested groups of directory service hierarchy.

LDAP access to Administrative panel

LDAP access to platform Administrative panel is configured in `MarketingPlatform/config/main.json` configuratio

```
"LDAP": {
  "LDAP_ONLY": false,
  "PRIMARY_HOST": "ldap.primary.example.com",
  "PRIMARY_PORT": 389,
  "SECONDARY_HOST": "ldap.secondary.example.com",
  "SECONDARY_PORT": 389,
  "BINDLOGIN": "cn=admin,dc=example,dc=com",
  "BINDPASS": "your_password",
  "BASE_DN": "dc=example,dc=com",
  "AUTH_GROUP": "CN=AdminGroup,CN=Users,DC=example,DC=com",
  "IS_SSL": false,
  "START_TLS": false,
  "CRT_PATH": "/etc/ssl/certs/ldap_client.crt",
  "CA_CRT_PATH": "/etc/ssl/certs/ca.crt",
  "KEY_PATH": "/etc/ssl/private/ldap_client.key"
},
"LDAP_CHECK_USER_TIME_PERIOD_SEC": 300
```

Parameter	Type	
Main parameters		
`LDAP_ONLY`	**bool**	**Prohibits access to the admin panel using login/password created within t**
`PRIMARY_HOST`	**string**	**Network address of the primary LDAP server.**
`PRIMARY_PORT`	**int**	**Port of the primary LDAP server (usually 389 for LDAP, 636 for LDAPS).**
`SECONDARY_HOST`	**string**	**Network address of the backup LDAP server for fault tolerance.**
`SECONDARY_PORT`	**int**	**Port of the backup LDAP server.**
`BINDLOGIN`	**string**	**Login of the account for connecting to the LDAP server (e.g., `cn=admin,dc=ex`**
`BINDPASS`	**string**	**Password of the account for connecting to the LDAP server.**
`BASE_DN`	**string**	**The base DN (Distinguished Name) from which the directory search begins**
`AUTH_GROUP`	**string**	**The distinguished name of the LDAP access group that determines employ**
Secure connection parameters		
`IS_SSL`	**bool**	**Enables LDAPS mode (LDAP over SSL). The connection is encrypted from t**
`START_TLS`	**bool**	**Enables the STARTTLS extension. Encryption is established on top of the**
`CRT_PATH`	**string**	**Path to the client certificate file (if mutual authentication is required).**
`CA_CRT_PATH`	**string**	**Path to the Certificate Authority (CA) certificate file.**
`KEY_PATH`	**string**	**Path to the client's private key file.**
Additional parameters		
`LDAP_CHECK_USER_TIME_PERIOD_SEC`	**int**	**Time interval (in seconds) for checking the compliance of user rights with**

INFO

- **The account for connecting to the server (`BINDLOGIN` / `BINDPASS`) must have sufficient rights to read the**
- **The `IS_SSL` and `START_TLS` parameters are mutually exclusive — only one encryption mode can be ena**

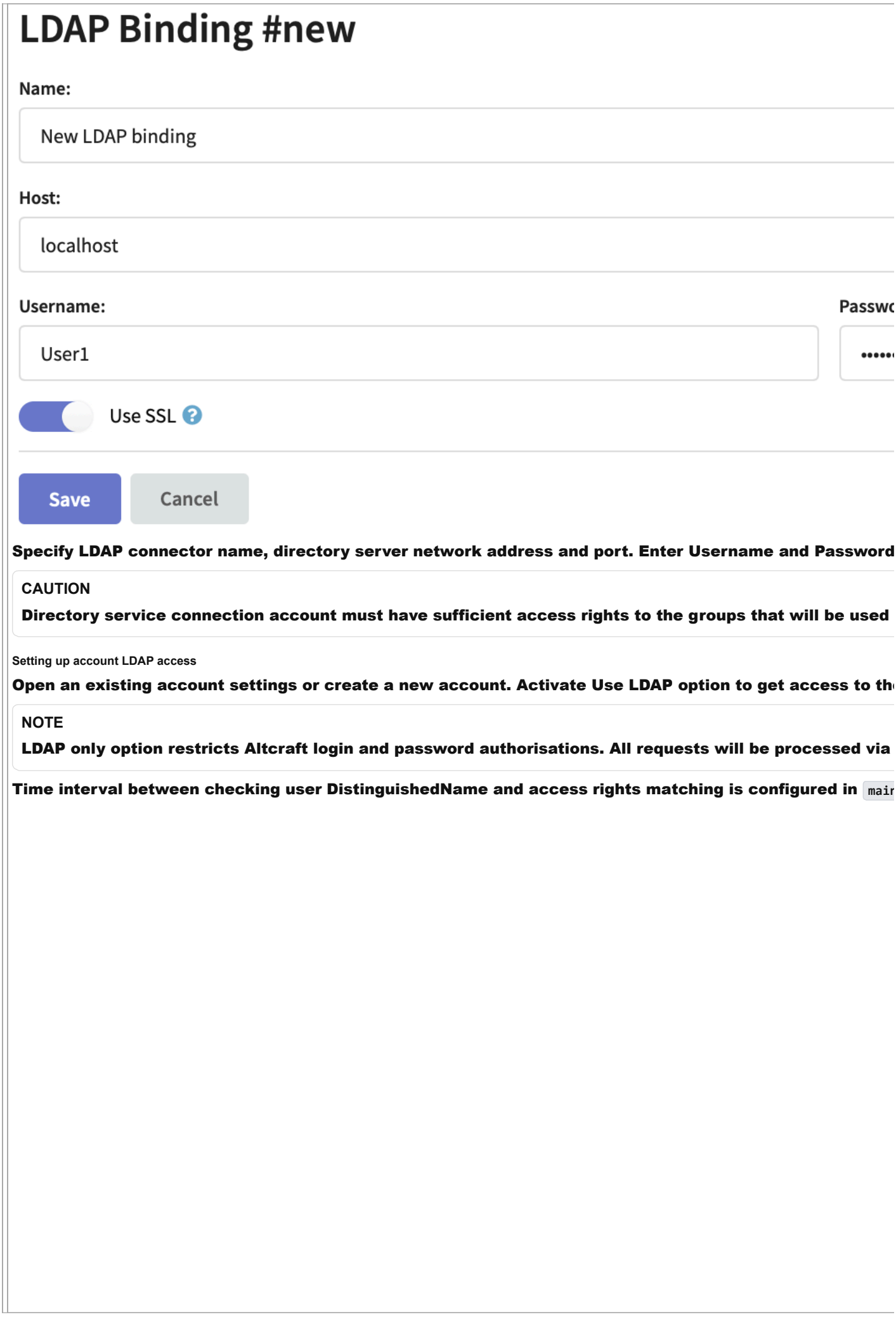
After making changes to the `main.json` configuration file, restart Altcraft MP (`./akd restart`).

LDAP User interface access

LDAP access to account User interface is set up for every account separately in Altcraft Administrative p

LDAP binding creation

Enter Altcraft MP Administrative panel and open Create —> LDAP binding or Setup —> LDAP binding and p



Specify LDAP connector name, directory server network address and port. Enter Username and Password

CAUTION

Directory service connection account must have sufficient access rights to the groups that will be used

Setting up account LDAP access

Open an existing account settings or create a new account. Activate Use LDAP option to get access to th

NOTE

LDAP only option restricts Altcraft login and password authorisations. All requests will be processed via

Time interval between checking user DistinguishedName and access rights matching is configured in `mair`

Use LDAP

LDAP options

LDAP only

Assigned LDAP connector:

new LDAP binding #1

Authorized group:

CN=Altkraft-Users,OU=AltkraftMP,DC=ad,DC=ldap-server,DC=devel

Master group:

CN=Altkraft-Master_user,OU=AltkraftMP,DC=ad,DC=ldap-server,DC=devel

Default groups:

Corporate clients project ✖ default ✖

Default roles:

default ✖

Groups matching:

CN=East_coast,OU=AltkraftMP,DC=ad,DC=ldap-server,DC=devel

CN=East_cowst,OU=AltkraftMP,DC=ad,DC=ldap-server,DC=devel

+ Add Group

Roles matching:

CN=East_coast,OU=AltkraftMP,DC=ad,DC=ldap-server,DC=devel

+ Add Role

In a dropdown select an Assigned LDAP connector: LDAP binding, you are going to use for this account ac

Basic access settings

For configuring basic account access you will need directory service DistinguishedNames for the followin

- **Auth Group — allows platform access with a default set of user groups and roles.**
- **Master Group — allows master access to account settings, main group containing all account objects**

Both LDAP groups are required for a user to have master access.
Specify these groups DistinguishedNames in the corresponding fields. As DN examples the following are u
Select or create Default Groupsand Default Roles to be accessible by default for all users with Authorized

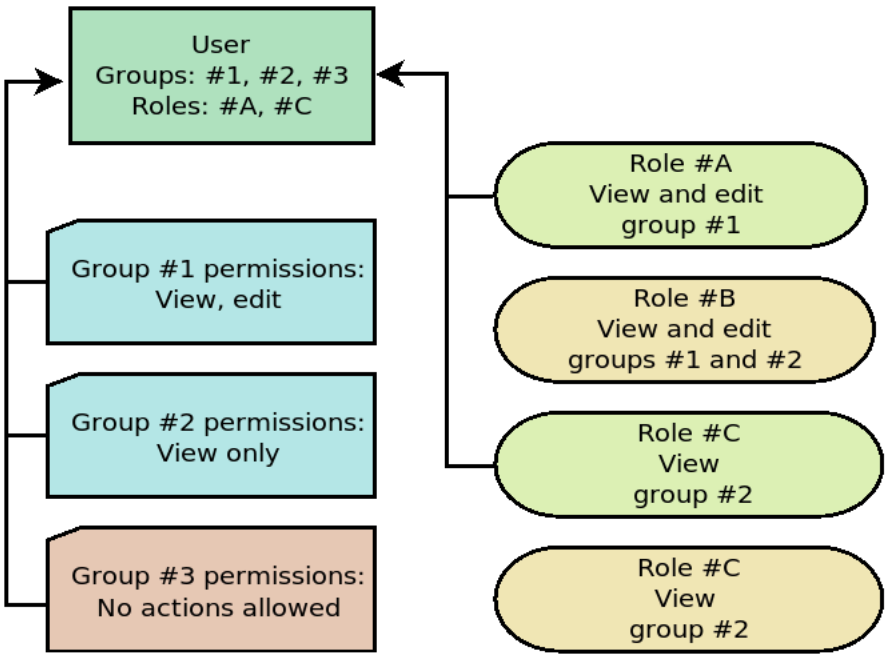
NOTE

WARNING! Users with no assigned groups or roles cannot log into account.
In case a user role allows to interact with a certain group make sure the corresponding group is assigne

Access differentiation

You can create directory service DistinguishedNames matching custom Altcraft MP groups and roles.
Enter the DistinguishedNames into these fields:

- **Groups matching — to set access groups matching.**
- **Roles matching — to set access roles matching.**



Authorizing with LDAP

To enter Administrative panel use LDAP username and password.
To enter account interface use LDAP username with account alias: `ldapuser@domain.com@account1`**, and LDAP p**
If LDAP only option is disabled, both Altcraft MP and LDAP login credentials can be used.

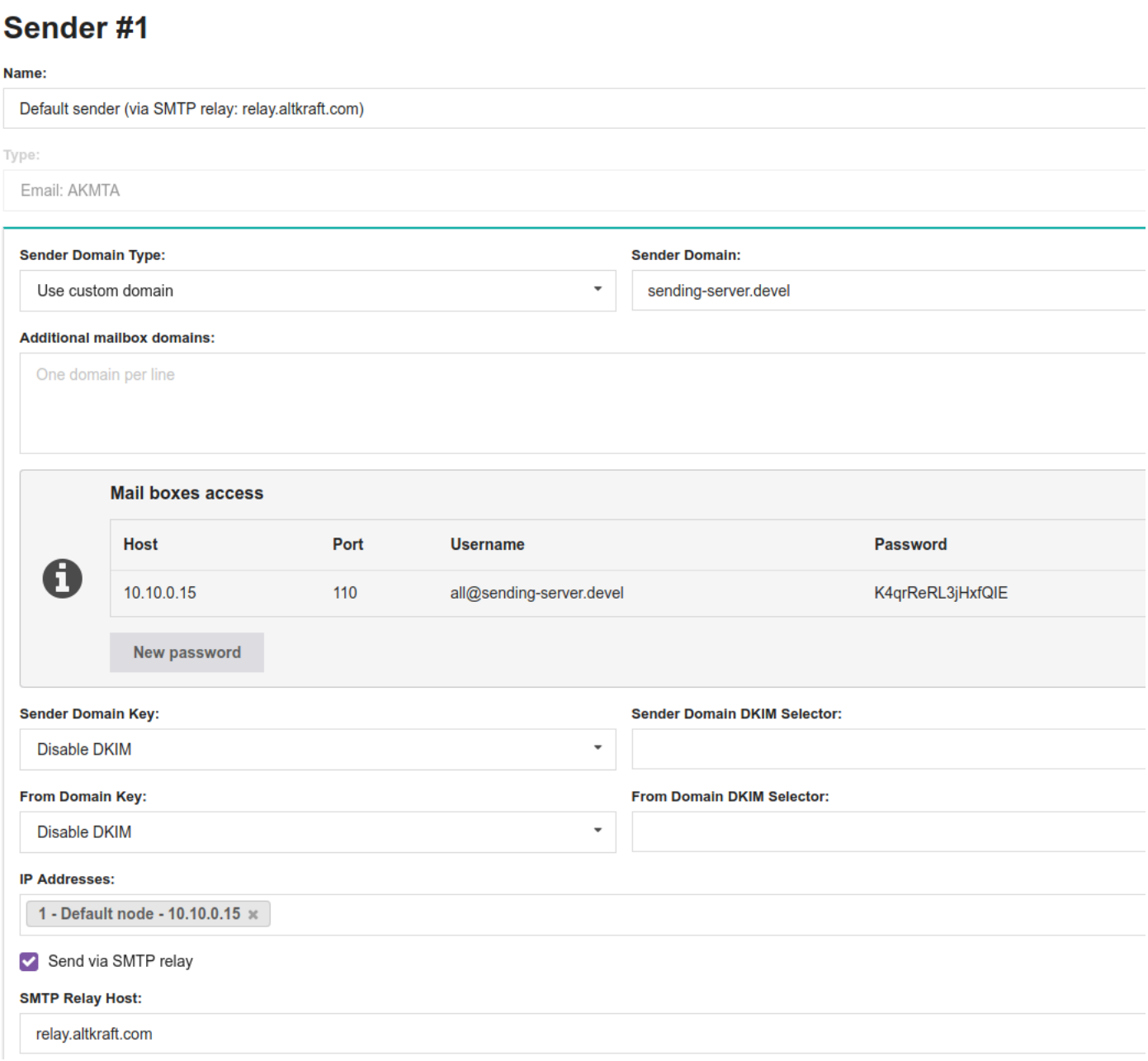
Sending Email via SMTP relay

Sometimes Emails cannot be simply sent from the platform's MTA — your company can have regulations a

NOTE

To collect information about delivery and customer activity, your relay must support DSN.
Also check PTR record for AKMTA and whether it is accessible from the SMTP relay server.

In Altcraft Marketing Administrative panel create a sender and enable "Send via SMTP relay".



NOTE

If you do not use DKIM, add it for your SMTP relay.

Postfix

By default Postfix is configured to notify on message delivery only in case the receiving server does not s

Settings

In Postfix configuration file (`/etc/postfix/main.cf`) add this parameter:

```
smtp_discard_ehlo_keyword_address_maps = cidr:/etc/postfix/esmtp_access
```

Create file `/etc/postfix/esmtp_access` with a list of receiving servers supporting DSN:

```
10.10.0.0/16  silent-discard
0.0.0.0/0     dsn
::/0          dsn
```

Documentation:

- **http://www.postfix.org/DSN_README.html**
- **http://www.postfix.org/postconf.5.html**

> NOTE
>
> **Be careful not to mistake `smtp_discard_ehlo_keyword_address_maps` for `smtpd_discard_ehlo_keyword_address_maps`.**

Pixel and push domain configuration

Overview

This domain is used for website visitors behavior tracking and handling push notifications subscription an
This standalone domain is required in any infrastructure design where you want to use behavior tracking
There are two ways of pixel handling in Altcraft MP, see [Infrastucture design variations] for details.
The following terms are used in the article below:

- **Control server — the main server or server group of Altcraft MP, processing control panel and databas**
- **cookiesaver — Altcraft MP service processing pixel and push domain**
- **main.json — the main Altcraft MP configuration file, placed in config directory of Altcraft MP installati**
- **pxl.example.com — pixel, push and cookies domain**
- **10.20.10.0/24 — *Control Server* public IPs**
- **10.10.10.0/24 — *Control Server* private IPs**

An A record is made for the pixel domain - pointing cookiesaver's IP address or load balancer's IP address

```
pxl.example.com IN A 10.20.10.3
```

Mind that SSL usage is mandatory for the pixel domain. Examples below use letsencrypt SSL certificates.

Without proxy

When using no proxy or balance loader — the domain's IP points to any of the Control server's network int
The following changes should be made in `main.json` configuration file:

```
"COOKIESAVER_PUBLIC_IP": "10.20.10.3",
"COOKIESAVER_PUBLIC_HOSTNAME": "pxl.example.com",
"COOKIESAVER_PUBLIC_PORT": 443,
"COOKIESAVER_SSL_CRT_PATH": "/etc/letsencrypt/live/pxl.example.com/fullchain.pem",
"COOKIESAVER_SSL_KEY_PATH": "/etc/letsencrypt/live/pxl.example.com/privkey.pem",
```

With external proxy

A local network IP address is provided for proxying needs. SSL termination is mandatory.
The following changes should be made in `main.json` configuration file:

```
"COOKIESAVER_PUBLIC_IP": "10.10.10.3",
"COOKIESAVER_PUBLIC_HOSTNAME": "pxl.example.com",
"COOKIESAVER_PUBLIC_PORT": 443,
"COOKIESAVER_SSL_ON": false,
```

Note that even when SSL termination is disabled - port 443 is used. That is a system restriction.
The following configuration depends on the proxy you are going to use. Here is an example for Nginx:

```
server {
 listen 10.20.10.3:443 ssl;
 server_name pxl.example.com;
 ssl on;
 ssl_certificate /etc/letsencrypt/live/pxl.example.com/fullchain.pem;
 ssl_certificate_key /etc/letsencrypt/live/pxl.example.com/privkey.pem;
 upstream cookiesaver {
  least_conn;
  server 10.10.10.3:443;
  keepalive 1024;
 }
 location / {
  proxy_http_version 1.1;
  proxy_set_header Connection "";
  proxy_pass_header Server;
  proxy_set_header Host $http_host;
  proxy_redirect off;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Scheme $scheme;
  proxy_pass http://cookiesaver;
 }
}
```

Testing and launching

After configuring `main.json` restart cookiesaver process by `akd` utility from Altcraft MP installation directo

```
./akd onerestart cookiesaver
```

If no errors occur during restarting cookiesaver, the next step is to restart the platform.

```
./akd restart
```

Then you can enable Pixel and Push in Altcraft administrative panel for any service instance account.

- [x] Enable Pixel
- [x] Enable Pushes

To make sure the pixel domain is functional, check if hyperlinks in sent messages are processed within th

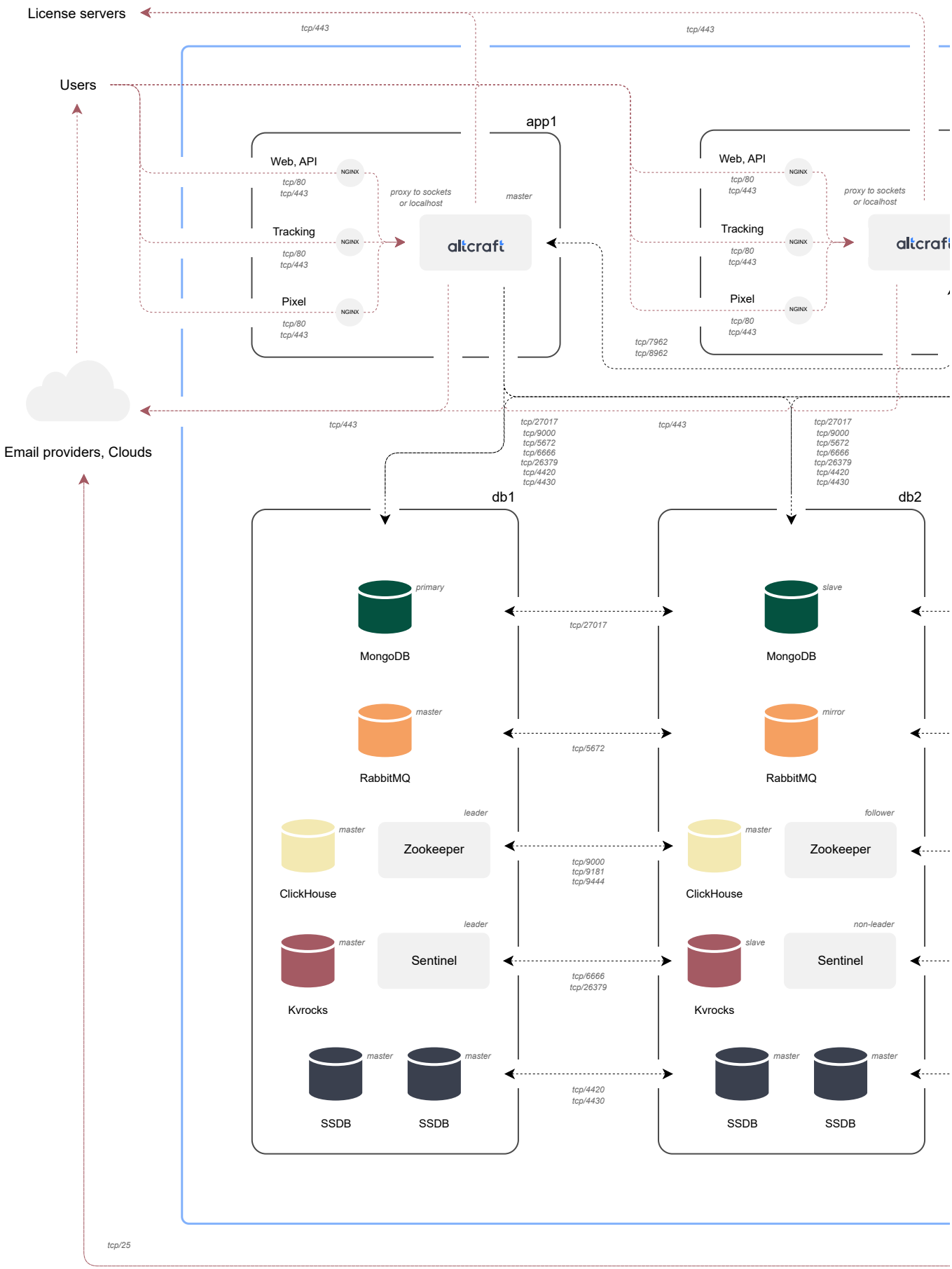
```
wget http://trk.example.com/0f1d00e4d93978fc/TdguHgCa<cut>w 2>&1 | grep Location:
Location: https://pxl.example.com:443/pixel?_open_pix=1&k=7E<cut>P5 [following]
Location: https://example.com/ [following]
```

Mind that pixel tracking is disabled for mailing testing. Thus you will need a segment of testing subscribe

The following pixel and push configuration is set up in Altcraft MP user interface: pixel settings, web push

Cluster and Replication Setup

Deployment Scheme



> **DANGER**
>
> **For passwords and secret combinations, the character set abcdefghijklmnopqrstuvwxyzABCDEF is used.**

MongoDB Replica Set Configuration

According to the deployment scheme, the replica consists of nodes on hosts db1, db2, and db3.

> **INFO**
>
> **Read more about the replication mechanism in the official documentation: https://www.mongodb.com/do**

1. Add a new parameter with the replica name to the configuration file [db1, db2, db3]:

```
replication:
  replSetName: rs0
```

Example full configuration file for db1:

```
storage:
  dbPath: /var/lib/mongodb
  journal:
    enabled: true
  wiredTiger:
    engineConfig:
      directoryForIndexes: true

replication:
  replSetName: rs0

systemLog:
  destination: file
  logAppend: true
  logRotate: reopen
  path: /var/log/mongodb/mongod.log

net:
  port: 27017
  bindIp: "127.0.0.1,db1"
```

2. After restarting MongoDB, connect to the database via MongoDB Shell and initialize the replica [db1]:

> **TIP**
>
> **Make sure that the instance you are connecting to has the `Primary` status.**

```
> rs.initiate({
  _id: "rs0",
  version: 1,
  members: [
    {
      _id: 0,
      host: "db1:27017"
    },
    {
      _id: 1,
      host: "db2:27017"
    },
    {
      _id: 2,
      host: "db3:27017"
    }
  ]
});
```

3. Check the replication status [db1]:

```
rs0: PRIMARY > rs.status();
```

4. Update the connection parameters in main.json configuration [app1, app2]:

```
{
  "MONGO_AUTH_DB": "admin",
  "CONTROLDB_NAME": "control",
  "CONTROLDB_USER": "altcraft",
  "CONTROLDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "CONTROLDB_REPL_SET_NAME": "rs0",
  "CONTROLDB": [
    {
      "IP": "db1",
      "PORT": 27017
    },
    {
      "IP": "db2",
      "PORT": 27017
    },
    {
      "IP": "db3",
      "PORT": 27017
    }
  ],
  "FILEDB_ENABLE": true,
  "FILEDB_NAME": "filedb",
  "FILEDB_USER": "altcraft",
  "FILEDB_PASS": "abcdefghijklmnopqrstuvwxyzABCDEF",
  "FILEDB_REPL_SET_NAME": "rs0",
  "FILEDB": [
    {
      "IP": "db1",
      "PORT": 27017
    },
    {
      "IP": "db2",
      "PORT": 27017
    },
    {
      "IP": "db3",
      "PORT": 27017
    }
  ]
}
```

ClickHouse Cluster Setup

According to the deployment diagram, the replica (a single replicated shard) and Zookeeper are hosted or

INFO

In this guide, ClickHouse Keeper is used instead of Zookeeper.

INFO

Read more about the replication mechanism in the official documentation:
https://clickhouse.com/docs/engines/table-engines/mergetree-family/replication/

1. **Add the cluster configuration with replicas [db1, db2, db3]:**

```
cat <<EOF > /etc/clickhouse-server/config.d/clusters.xml
<?xml version="1.0" ?>
<clickhouse>
    <remote_servers>
        <altcraft>
            <shard>
                <internal_replication>true</internal_replication>
                <replica>
                    <host>db1</host>
                    <port>9000</port>
                </replica>
                <replica>
                    <host>db2</host>
                    <port>9000</port>
                </replica>
                <replica>
                    <host>db3</host>
                    <port>9000</port>
                </replica>
            </shard>
        </altcraft>
    </remote_servers>
</clickhouse>
EOF
```

2. Add the macros configuration [db1, db2, db3]:

TIP

Macros are used in ClickHouse Keeper paths. The `replica` parameter on each host must match the curre

Example macros configuration for db1:

```
cat <<EOF > /etc/clickhouse-server/config.d/macros.xml
<?xml version="1.0" ?>
<clickhouse>
    <macros>
        <cluster>altcraft</cluster>
        <replica>db1</replica>
        <shard>01</shard>
    </macros>
</clickhouse>
EOF
```

Example macros configuration for db2:

```
cat <<EOF > /etc/clickhouse-server/config.d/macros.xml
<?xml version="1.0" ?>
<clickhouse>
    <macros>
        <cluster>altcraft</cluster>
        <replica>db2</replica>
        <shard>01</shard>
    </macros>
</clickhouse>
EOF
```

Example macros configuration for db3:

```
cat <<EOF > /etc/clickhouse-server/config.d/macros.xml
<?xml version="1.0" ?>
<clickhouse>
    <macros>
        <cluster>altcraft</cluster>
        <replica>db3</replica>
        <shard>01</shard>
    </macros>
</clickhouse>
EOF
```

3. Add ClickHouse Keeper configuration [db1, db2, db3].

TIP

On each host, the `server.id` parameter must match the unique numeric identifier of that host.

Example configuration for db1:

```
cat <<EOF > /etc/clickhouse-server/config.d/keeper.xml
<?xml version="1.0" ?>
<clickhouse>
    <keeper_server>
        <tcp_port>9181</tcp_port>
        <server_id>1</server_id>
        <log_storage_path>/var/lib/clickhouse/coordination/log</log_storage_path>
        <snapshot_storage_path>/var/lib/clickhouse/coordination/snapshots</snapshot_storage_path>

        <coordination_settings>
            <operation_timeout_ms>10000</operation_timeout_ms>
            <session_timeout_ms>30000</session_timeout_ms>
            <rotate_log_storage_interval>10000</rotate_log_storage_interval>
        </coordination_settings>

      <raft_configuration>
            <server>
               <id>1</id>
               <hostname>db1</hostname>
               <port>9444</port>
          </server>
          <server>
               <id>2</id>
               <hostname>db2</hostname>
               <port>9444</port>
          </server>
          <server>
               <id>3</id>
               <hostname>db3</hostname>
               <port>9444</port>
          </server>
      </raft_configuration>
    </keeper_server>

    <zookeeper>
        <node>
            <host>db1</host>
            <port>9181</port>
        </node>
        <node>
            <host>db2</host>
            <port>9181</port>
        </node>
        <node>
            <host>db3</host>
            <port>9181</port>
        </node>
    </zookeeper>

    <distributed_ddl>
        <path>/clickhouse/altcraft/task_queue/ddl</path>
    </distributed_ddl>
</clickhouse>
EOF
```

4. By default, ClickHouse and ClickHouse Keeper listen on localhost. To listen on all addresses, override

```
cat <<EOF > /etc/clickhouse-server/config.d/settings.xml
<?xml version="1.0" ?>
<clickhouse>
   <listen_host>0.0.0.0</listen_host>
</clickhouse>
EOF
```

4. Update connection parameters in [main.json configuration] [app1, app2]:

```
{
  "CLICKHOUSE_SYSTEM": {
    "HOST": "db1",
    "PORT": 9000,
    "USER": "altcraft",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "ALT_HOSTS": [
      {
        "HOST": "db2",
        "PORT": 9000
      },
      {
        "HOST": "db3",
        "PORT": 9000
      }
    ],
    "REPLICATED": true,
    "STRATEGY": "random"
  }
}
```

SSDB Replication Configuration

According to the deployment scheme, the replica consists of nodes on hosts db1, db2, and db3.

> **INFO**
>
> **Read more about the replication mechanism in the official documentation: https://ideawu.github.io/ssdb-d**

1. **To configure replication, add the `replication` parameter to the configuration file.**

> **TIP**
>
> **Note: in the configuration file, you must use the tab character for indentation, not spaces.**

Example configuration for db1:

```
replication:
	binlog: yes
		capacity: 100000000
	sync_speed: -1
	slaveof:
		id: db2
		type: mirror
		host: db2
		port: 4420
		auth: abcdefghijklmnopqrstuvwxyzABCDEF
		recv_timeout: 900
	slaveof:
		id: db3
		type: mirror
		host: db3
		port: 4420
		auth: abcdefghijklmnopqrstuvwxyzABCDEF
		recv_timeout: 900
```

Example configuration for db2:

```
replication:
	binlog: yes
		capacity: 100000000
	sync_speed: -1
	slaveof:
		id: db1
		type: mirror
		host: db1
		port: 4420
		auth: abcdefghijklmnopqrstuvwxyzABCDEF
		recv_timeout: 900
	slaveof:
		id: db3
		type: mirror
		host: db3
		port: 4420
		auth: abcdefghijklmnopqrstuvwxyzABCDEF
		recv_timeout: 900
```

Example configuration for db3:

```
replication:
        binlog: yes
                capacity: 100000000
        sync_speed: -1
        slaveof:
                id: db1
                type: mirror
                host: db1
                port: 4420
                auth: abcdefghijklmnopqrstuvwxyzABCDEF
                recv_timeout: 900
        slaveof:
                id: db2
                type: mirror
                host: db2
                port: 4420
                auth: abcdefghijklmnopqrstuvwxyzABCDEF
                recv_timeout: 900
```

> **INFO**
>
> **Repeat the configuration changes for the other SSDB instance, changing the password and port.**

Repeat the configuration changes for the other SSDB instance, changing the password and port.

2. After restarting SSDB, verify the replication status by running the command on [db1]:

```
echo info | /usr/local/ssdb/tools/ssdb-cli -h db1 -p 4420 -a abcdefghijklmnopqrstuvwxyzABCDEF
```

The values of `binlogs.max_seq` and `replication.slaveof.last_seq` must match. Example output for db1:

```
binlogs
        capacity : 100000000
        min_seq  : 0
        max_seq  : 2
replication
        client db2:34526
                type     : mirror
                status   : SYNC
                last_seq : 2
        client db3:34526
                type     : mirror
                status   : SYNC
                last_seq : 2
replication
        slaveof db2:4420
                id         : db2
                type       : mirror
                status     : SYNC
                last_seq   : 2
                copy_count : 0
                sync_count : 1
replication
        slaveof db3:4420
                id         : db3
                type       : mirror
                status     : SYNC
                last_seq   : 2
                copy_count : 0
                sync_count : 1
```

3. Update the connection parameters in the main.json configuration [app1, app2]:

```
{
  "SSDB_HBSUPP": [
    {
      "IP": "db1",
      "PORT": 4420,
      "PASS": "abcdefghijklmnopqrstuvwxyzABCDEF"
    },
    {
      "IP": "db2",
      "PORT": 4420,
      "PASS": "abcdefghijklmnopqrstuvwxyzABCDEF"
    },
    {
      "IP": "db3",
      "PORT": 4420,
      "PASS": "abcdefghijklmnopqrstuvwxyzABCDEF"
    }
  ],
  "SSDB_NOTIFY": [
    {
      "IP": "db1",
      "PORT": 4430,
      "PASS": "abcdefghijklmnopqrstuvwxyzABCDEF"
    },
    {
      "IP": "db2",
      "PORT": 4430,
      "PASS": "abcdefghijklmnopqrstuvwxyzABCDEF"
    },
    {
      "IP": "db3",
      "PORT": 4430,
      "PASS": "abcdefghijklmnopqrstuvwxyzABCDEF"
    }
  ]
}
```

Configuring the RabbitMQ Cluster

According to the deployment scheme, the cluster consists of nodes on hosts db1, db2, and db3.

> **INFO**
>
> **Read more about how the cluster works on this page: https://www.rabbitmq.com/clustering.html**

1. Update the contents of the `/var/lib/rabbitmq/.erlang.cookie` file [db1, db2, db3]:

```
printf "ABCDEFGHIJKLMN" > /var/lib/rabbitmq/.erlang.cookie
```

2. After restarting RabbitMQ, join the cluster on [db2, db3]:

```
{
  rabbitmqctl stop_app
  rabbitmqctl reset
  rabbitmqctl join_cluster rabbit@db1
  rabbitmqctl start_app
}
```

3. Check the replication status [db1, db2, db3]:

```
rabbitmqctl cluster_status
```

Example output:

```
Cluster status of node rabbit@db1 ...
Basics

Cluster name: rabbit@db1

Disk Nodes

rabbit@db1
rabbit@db2
rabbit@db3

Running Nodes

rabbit@db1
rabbit@db2
rabbit@db3

Versions

rabbit@db1: RabbitMQ 3.10.6 on Erlang 24.3.4
rabbit@db2: RabbitMQ 3.10.6 on Erlang 24.3.4
rabbit@db3: RabbitMQ 3.10.6 on Erlang 24.3.4

Maintenance status

Node: rabbit@db1, status: not under maintenance
Node: rabbit@db2, status: not under maintenance
Node: rabbit@db3, status: not under maintenance

Alarms

(none)

Network Partitions

(none)

Listeners

Node: rabbit@db1, interface: [::], port: 15672, protocol: http, purpose: HTTP API
Node: rabbit@db1, interface: [::], port: 25672, protocol: clustering, purpose: inter-node and CLI tool communication
Node: rabbit@db1, interface: [::], port: 5672, protocol: amqp, purpose: AMQP 0-9-1 and AMQP 1.0
Node: rabbit@db2, interface: [::], port: 15672, protocol: http, purpose: HTTP API
Node: rabbit@db2, interface: [::], port: 25672, protocol: clustering, purpose: inter-node and CLI tool communication
Node: rabbit@db2, interface: [::], port: 5672, protocol: amqp, purpose: AMQP 0-9-1 and AMQP 1.0
Node: rabbit@db3, interface: [::], port: 15672, protocol: http, purpose: HTTP API
Node: rabbit@db3, interface: [::], port: 25672, protocol: clustering, purpose: inter-node and CLI tool communication
Node: rabbit@db3, interface: [::], port: 5672, protocol: amqp, purpose: AMQP 0-9-1 and AMQP 1.0

Feature flags

Flag: classic_mirrored_queue_version, state: enabled
Flag: drop_unroutable_metric, state: disabled
Flag: empty_basic_get_metric, state: disabled
Flag: implicit_default_bindings, state: enabled
Flag: maintenance_mode_status, state: enabled
Flag: quorum_queue, state: enabled
Flag: stream_queue, state: enabled
Flag: user_limits, state: enabled
Flag: virtual_host_metadata, state: enabled
```

4. To enable mirroring of all queues in all virtual hosts, run the following script [db1]:

```
vhosts=($(rabbitmqctl list_vhosts -q)); for item in "${vhosts[@]}"; do rabbitmqctl set_policy -p "$item" ha-all ".*" '{"ha-mode":"all","ha-sync
```

5. Update the connection parameters in main.json configuration [app1, app2]:

```
{
  "RABBITMQ_CLUSTER": {
    "USER": "altcraft",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "FAIL_MODE": "failover",
    "STRATEGY": "in_order",
    "RECONNECT_TIME_PERIOD": 5,
    "HOSTS": [
      {
        "HOST": "db1",
        "PORT": 5672,
        "WEIGHT": 1
      },
      {
        "HOST": "db2",
        "PORT": 5672,
        "WEIGHT": 2
      },
      {
        "HOST": "db3",
        "PORT": 5672,
        "WEIGHT": 3
      }
    ]
  }
}
```

Kvrocks Replication Configuration

According to the deployment scheme, the replica consists of nodes on hosts db1, db2, and db3.

1. Update the Kvrocks configuration by adding the `masterauth` parameter [db1, db2, db3]:

```
# kvrocks.conf
masterauth abcdefghijklmnopqrstuvwxyzABCDEF
```

2. On the slave nodes, add the `slaveof` parameter specifying the master's address [db2, db3]:

```
# kvrocks.conf
slaveof 192.168.0.10 6666
```

Where 192.168.0.10 is the address of `db1`.

3. Install Redis Sentinel:

```
{
  apt update
  apt install -y redis-sentinel
}
```

4. Add the Redis Sentinel configuration [db1, db2, db3]:

```
# sentinel.conf
bind 0.0.0.0
port 26379

daemonize yes
pidfile "/var/run/sentinel/redis-sentinel.pid"
logfile "/var/log/redis/redis-sentinel.log"

sentinel deny-scripts-reconfig yes

dir "/var/lib/redis"

sentinel monitor altcraft db1 6666 2
sentinel auth-pass altcraft abcdefghijklmnopqrstuvwxyzABCDEF
```

5. After restarting Kvrocks and Redis Sentinel, connect to one of the nodes and check the replication st

```
redis-cli -h db1 -p 6666 -a abcdefghijklmnopqrstuvwxyzABCDEF
```

Example output:

```
192.168.0.10:6666> INFO REPLICATION
# Replication
role:master
connected_slaves:2
slave0:ip=192.168.0.10,port=6666,offset=12068788,lag=0
slave1:ip=192.168.0.11,port=6666,offset=12068788,lag=0
master_repl_offset:12068788
```

6. Update connection parameters in the main.json configuration [app1, app2]:

```
{
  "CAMP_DUPLICATESDB": {
    "MODE": "sentinel_replica",
    "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
    "NODES": [
      {
        "HOST": "db1",
        "PORT": 26379
      },
      {
        "HOST": "db2",
        "PORT": 26379
      },
      {
        "HOST": "db3",
        "PORT": 26379
      }
    ],
    "SENTINEL": {
      "MASTER_NAME": "altcraft"
    }
  }
}
```

Altcraft Platform Cluster Setup

According to the deployment scheme, the application is deployed on the hosts app1 and app2.

1. In cluster mode, only two services operate: `proctask` and `procworkflow`. Update the parameters in main.

```
{
  "CLUSTER": {
    "HOST_ID": "app1",
    "PROC_TASKS": [
      {
        "HOST_ID": "app1",
        "RPC_HOST": "0.0.0.0",
        "RPC_HOST_CLI": "app1",
        "RPC_PORT": 8962,
        "FORCE_MASTER_ROLE": true
      },
      {
        "HOST_ID": "app2",
        "RPC_HOST": "0.0.0.0",
        "RPC_HOST_CLI": "app2",
        "RPC_PORT": 8962,
        "FORCE_MASTER_ROLE": false
      }
    ],
    "PROC_WORKFLOW": [
      {
        "HOST_ID": "app1",
        "RPC_HOST": "0.0.0.0",
        "RPC_HOST_CLI": "app1",
        "RPC_PORT": 7962,
        "FORCE_MASTER_ROLE": true
      },
      {
        "HOST_ID": "app2",
        "RPC_HOST": "0.0.0.0",
        "RPC_HOST_CLI": "app2",
        "RPC_PORT": 7962,
        "FORCE_MASTER_ROLE": false
      }
    ]
  }
}
```

Here:

- `HOST_ID` — **unique host identifier,**
- `RPC_HOST` — **the address on which the service will accept connections,**
- `RPC_HOST_CLI` — **the address clients will use to connect,**
- `FORCE_MASTER_ROLE` — **parameter for forcing the master role.**

2. After restarting the services, the logs on [app1] should display the following lines:

```
{"level":"info","proc":"proctask","pid":306390,"hid":"app1","ver":"v2023.3.65.2060","file":"server/task_server.go:314","time":"2023-10-17T07:0
{"level":"info","proc":"procworkflow","pid":306827,"hid":"app1","ver":"v2023.3.65.2060","file":"workflow/controlworkflow.go:108","time":"2023-
```

Configuring Custom Reader Groups

You can configure custom reader groups that will process tasks for specific accounts only. By default, the

Groups are configured in the master host configuration using the `READERS_GROUPS` parameter, for example:

```
{
  "PROC_WORKFLOW": [
    {
      "HOST_ID": "app1",
      "RPC_HOST": "0.0.0.0",
      "RPC_HOST_CLI": "app1",
      "RPC_PORT": 7962,
      "FORCE_MASTER_ROLE": true,
      "READERS_GROUPS": {
        "acc2": {
          "INCLUDE": "2",
          "EXCLUDE": ""
        },
        "acc1-5": {
          "INCLUDE": "1-5",
          "EXCLUDE": "2"
        }
      }
    },
    {
      "HOST_ID": "app2",
      "RPC_HOST": "0.0.0.0",
      "RPC_HOST_CLI": "app2",
      "RPC_PORT": 7962,
      "FORCE_MASTER_ROLE": false,
      "READERS_GROUP": "acc2"
    }
  ]
}
```

The `INCLUDE` parameter specifies the accounts to process, and `EXCLUDE` specifies accounts to exclude.

> **INFO**
>
> **Group names must be unique. Accounts must not appear in multiple groups. For example, if you don't exc**
> **If no group parameters are set for a slave host, it will process tasks from the default group.**

System notifications configuration

Mailbox configuration

To allow Altcraft MP to send weekly and daily reports, credentials changes and other email notifications,
This configuration consists of SMTP access credentials for notifications mailbox. If you do not have a ded
This configuration may be different depending on your email server type:

```
{
    "NOREPLY_MAILER": {
        "email": "altcraft@example.com",
        "login": "altcraft@example.com",
        "pass": "xeiT9oL8We",
        "port": 587,
        "server": "mail.example.com"
    }
}
```

Mailbox parameters:

Parameter	Type	Required
server	**string**	**Yes**
port	**int**	**No**
email	**string**	**Yes**
login	**string**	*
pass	**string**	**No**
helo	**string**	**No**
insecure	**bool**	**No**
force_no_ssl	**bool**	**No**

You can ignore `"login"` parameter if you are sending to an open mail relay.

Notifications configuration

System notifications are configured by several `main.json` parameters.
The default values are the following:

```
"NOTIFICATION" {
  "HARDBOUNCES_WARNING_PERCENT": 1,
  "SOFTBOUNCES_WARNING_PERCENT": 30,
  "COMPLAINS_WARNING_PERCENT": 0.07,
  "PERFOMANCE_WARNING_PERCENT": 90,
  "SELECT_INTERVAL_HOURS": 1,
  "NOTIFY_INTERVAL_HOURS": 12,
  "MINIMAL_DIVIDER": 1000
}
```

Notification parameters

Parameter	
HARDBOUNCES_WARNING_PERCENT **SOFTBOUNCES_WARNING_PERCENT** **COMPLAINS_WARNING_PERCENT**	**Define Hard bounces, Soft bounces and Com** **By default – 1, 30 and 0.07.**
PERFOMANCE_WARNING_PERCENT	**Defines the threshold for triggering a notific** **By default – 90.**
SELECT_INTERVAL_HOURS	**Assessment time interval for data aggregati** **By default – 1 hour. Maximum – 24 hours.**
NOTIFY_INTERVAL_HOURS	**Duplicated notifications time interval.** **By default – 12 hours (If the same problem o**
MINIMAL_DIVIDER	**Minimum deliveries number to aggregate st** **By default – 1000.**

Processes UNIX sockets configuration

To enable UNIX-sockets support add `"PREFER_UNIX_SOCKETS": true` parameter into `main.json` configuration file.
By default enabling this option forces most of the platform processes to use UNIX sockets.
To make a specific process use UNIX socket UNIX-сокета:

Parameter
“COOKIESAVER_UNIXSOCK_PATH”
“PUSH_HTTP_SERVER_UNIXSOCK_PATH”
“SMS_LISTENER_UNIXSOCK_PATH”
“TRACKING_IMAGE_UNIXSOCK_PATH”
“TRACKING_PUSH_UNIXSOCK_PATH”
“TRACKING_READ_UNIXSOCK_PATH”
“TRACKING_SMS_UNIXSOCK_PATH”
“TRACKING_UNIXSOCK_PATH”
“TRACKING_WEB_VERSION_UNIXSOCK_PATH”
“WEBAPI_UNIXSOCK_PATH”

TIP

To test the UNIX socket functionality, use the utility `socat`:

```
socat - "UNIX-CONNECT:/opt/akd/socks/akd-api.socket"
```

HTTPS Configuration

> **NOTE**
>
> **All parameters are configured in the main.json configuration file. A full restart is required after making a**

Web Panel and API

Add the following parameters:

```
{
  "SSL_ON": true,
  "SSL_PUBLIC_PORT": 443,
  "SSL_CRT_PATH": "/etc/ssl/altcraft.domain.com/fullchain.pem",
  "SSL_KEY_PATH": "/etc/ssl/altcraft.domain.com/privkey.pem"
}
```

Tracking

Add the following parameters:

```
{
  "TRACKING_SSL_ON": true,
  "TRACKING_SSL_PUBLIC_PORT": 443,
  "TRACKING_SSL_CRT_PATH": "/etc/ssl/click.domain.com/fullchain.pem",
  "TRACKING_SSL_KEY_PATH": "/etc/ssl/click.domain.com/privkey.pem"
}
```

Pixels

Add the following parameters:

```
{
  "COOKIESAVER_SSL_ON": true,
  "COOKIESAVER_PUBLIC_PORT": 443,
  "COOKIESAVER_SSL_CRT_PATH": "/etc/ssl/pxl.domain.com/fullchain.pem",
  "COOKIESAVER_SSL_KEY_PATH": "/etc/ssl/pxl.domain.com/privkey.pem"
}
```

Migrating from MongoDB to Percona Server

This article describes how to replace MongoDB Community Edition with Percona Server for MongoDB with
The standard in-place upgrade method (replacing binary files on the existing server) is not suitable when
The procedure involves minimal platform downtime. Most of the migration is performed without stopping A
For more details, refer to the official Percona documentation.

> CAUTION
>
> **Before starting, make a backup of all MongoDB data.**

Migration Overview

The migration follows this process:

1. **On the existing MongoDB Community server, replication mode (`replSet`) is enabled, and the server bec**
2. **A second clean server is provisioned with Percona Server for MongoDB installed and encryption enabl**
3. **The second server is added to the replication set and performs an initial sync. During synchronization**
4. **After the initial sync completes and replication status is verified, the PRIMARY role is switched to the**
5. **Altcraft Platform is reconfigured to point to the new server.**
6. **The old server is decommissioned.**

Infrastructure Preparation

The second server must meet the following requirements:

- **Same Ubuntu version as the primary server**
- **Disk space at least equal to the current MongoDB server**
- **Network connectivity between servers on port 27017 in both directions**

The second server must be clean, with no MongoDB installed before starting.
It is also recommended to check the `oplog` size on the current server in advance. It must be large enough

Check MongoDB Version

On the current server, ensure that the MongoDB Community version matches the Percona Server for Mong

```
/usr/local/mongodb/bin/mongod --version | head -1
```

The Percona Server for MongoDB version must be from the same major branch as MongoDB Community. F

Configure Replication on the Current Server

Edit the configuration file `/etc/mongod.conf`:

```
replication:
  replSetName: rs0

net:
  port: 27017
  bindIp: 0.0.0.0   # required for the second server to connect
```

Restart MongoDB:

```
systemctl restart mongod
```

Initialize replication:

```
/usr/local/mongodb/bin/mongosh --eval "rs.initiate()"
```

Check the node status:

```
/usr/local/mongodb/bin/mongosh --eval "rs.status()"
```

Confirm that the current server has `PRIMARY` status.

Install and Configure Percona Server on the Second Server

On the second server, install Percona Server for MongoDB of the same version as the first server.

Add the Percona repository:

```
wget https://repo.percona.com/apt/percona-release_latest.generic_all.deb
dpkg -i percona-release_latest.generic_all.deb
percona-release setup psmdb80   # for version 8.0
apt update
apt install percona-server-mongodb
```

Installing a Specific Version of Percona Server for MongoDB

If you need to install a specific version rather than the latest, first list available versions:

```
apt-cache madison percona-server-mongodb
```

Example output:

```
percona-server-mongodb | 8.0.10-5.jammy | http://repo.percona.com/psmdb-80/apt jammy/main amd64 Packages
percona-server-mongodb | 8.0.9-4.jammy  | http://repo.percona.com/psmdb-80/apt jammy/main amd64 Packages
```

Then install the desired version by specifying the version number for the Percona Server for MongoDB pac

```
apt install \
  percona-server-mongodb=<version> \
  percona-server-mongodb-server=<version> \
  percona-server-mongodb-shell=<version> \
  percona-server-mongodb-mongos=<version> \
  percona-server-mongodb-tools=<version>
```

Generate an encryption key:

```
openssl rand -base64 32 > /etc/mongodb-encryption-keyfile
chmod 600 /etc/mongodb-encryption-keyfile
chown mongod:mongod /etc/mongodb-encryption-keyfile
```

Edit `/etc/mongod.conf` on the second server:

```
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

net:
  port: 27017
  bindIp: 0.0.0.0

storage:
  dbPath: /var/lib/mongodb
  wiredTiger:
    engineConfig:
      directoryForIndexes: true

security:
  enableEncryption: true
  encryptionKeyFile: /etc/mongodb-encryption-keyfile

replication:
  replSetName: rs0
```

> **INFO**
>
> **The preferred method for storing the master key is HashiCorp Vault.**
> **If HashiCorp Vault is not used, the `mongodb-encryption-keyfile` should be stored in a secure location, such as**
> **Disk encryption reduces the risk of data exposure in case of media compromise, such as disk theft or im**

Using HashiCorp Vault Instead of a Local Key File

Instead of a local `encryptionKeyFile`, Percona Server for MongoDB can retrieve the master key from HashiCo

Before starting the second server, ensure the following are prepared:

- **A running HashiCorp Vault server**
- **KV v2 secrets engine**
- **Vault token with access to the required secret path**
- **Vault CA certificate if using a custom certificate authority**

In this case, replace the `encryptionKeyFile` parameter with the following configuration:

```
security:
  enableEncryption: true
  vault:
    serverName: VAULT_HOST
    port: 8200
    tokenFile: /etc/mongodb-vault-token
    secret: secret/data/mongodb/node2
    serverCAFile: /etc/ssl/certs/vault-ca.crt
```

The `tokenFile` must contain the Vault token.

```
chown mongod:mongod /etc/mongodb-vault-token
chmod 600 /etc/mongodb-vault-token
```

> **INFO**
>
> **It is recommended to store the `tokenFile` in a secure location, such as an encrypted partition or containe**

Ensure the `processManagement` section is not present in the file.

Stop MongoDB and clear the data directory. It must be empty for initial sync:

```
systemctl stop mongod
rm -rf /var/lib/mongodb/*
```

Start MongoDB:

```
systemctl start mongod
```

Check the service status and MongoDB logs for any startup errors:

```
systemctl status mongod
tail -n 50 /var/log/mongodb/mongod.log
```

Add the Second Server to the Replica Set

On the first server (current PRIMARY), connect to MongoDB:

```
/usr/local/mongodb/bin/mongosh
```

Add the second server:

```
rs.add("SECOND_SERVER_IP:27017")
```

Monitor the synchronization process:

```
rs.status()
```

The new server typically goes through `STARTUP` and `STARTUP2` states before transitioning to `SECONDARY`.

Wait until the second server reaches `SECONDARY` status.

Verify Replication Status

Before switching over, ensure that the initial sync is complete and the new server does not have significa

Check the replication status:

```
/usr/local/mongodb/bin/mongosh --eval "rs.status()"
```

Verify that the second server:

- **is in `SECONDARY` state**
- **is not performing initial sync**
- **has no significant lag from the current PRIMARY**

Only proceed to the platform switchover after these checks.

Verify Encryption

On the second server, perform the following checks.

Startup parameters should show `enableEncryption: true`:

```
mongosh --eval "db.serverCmdLineOpts()" | grep -A 5 encryption
```

If using a local key file, check permissions:

```
ls -la /etc/mongodb-encryption-keyfile
```

If using HashiCorp Vault, verify Vault parameters in the startup configuration:

```
mongosh --eval "db.serverCmdLineOpts()" | grep -A 10 vault
```

The log should contain a record indicating startup with encryption enabled:

```
grep -i encryption /var/log/mongodb/mongod.log
```

Data files on disk should not contain readable fragments of user data in plaintext:

```
strings /var/lib/mongodb/*.wt | head -20
```

Switch Altcraft Platform to the New Server

During the final switchover, Altcraft Platform must be stopped briefly to prevent new writes while changin

Stop the platform services:

```
/opt/akd/akd stop -fk
```

Connect to the current PRIMARY and step it down:

```
/usr/local/mongodb/bin/mongosh
rs.stepDown()
```

After this, check the replication status:

```
rs.status()
```

Verify that the second server has become `PRIMARY` and the first server has transitioned to `SECONDARY`.

Update Altcraft Platform Configuration

On the Altcraft Platform server, edit `/opt/akd/config/main.json`:

```
{
  "CONTROLDB_IP": "SECOND_SERVER_IP",
  "FILEDB_IP": "SECOND_SERVER_IP"
}
```

Start the platform:

```
/opt/akd/akd start -f
```

Check Altcraft Platform logs for any MongoDB connection errors.

Decommission the First Server

After confirming that the platform is stable with the new server, the old MongoDB server can be decommi

Stop and disable MongoDB on the first server:

```
systemctl stop mongod
systemctl disable mongod
```

If desired, remove MongoDB Community packages:

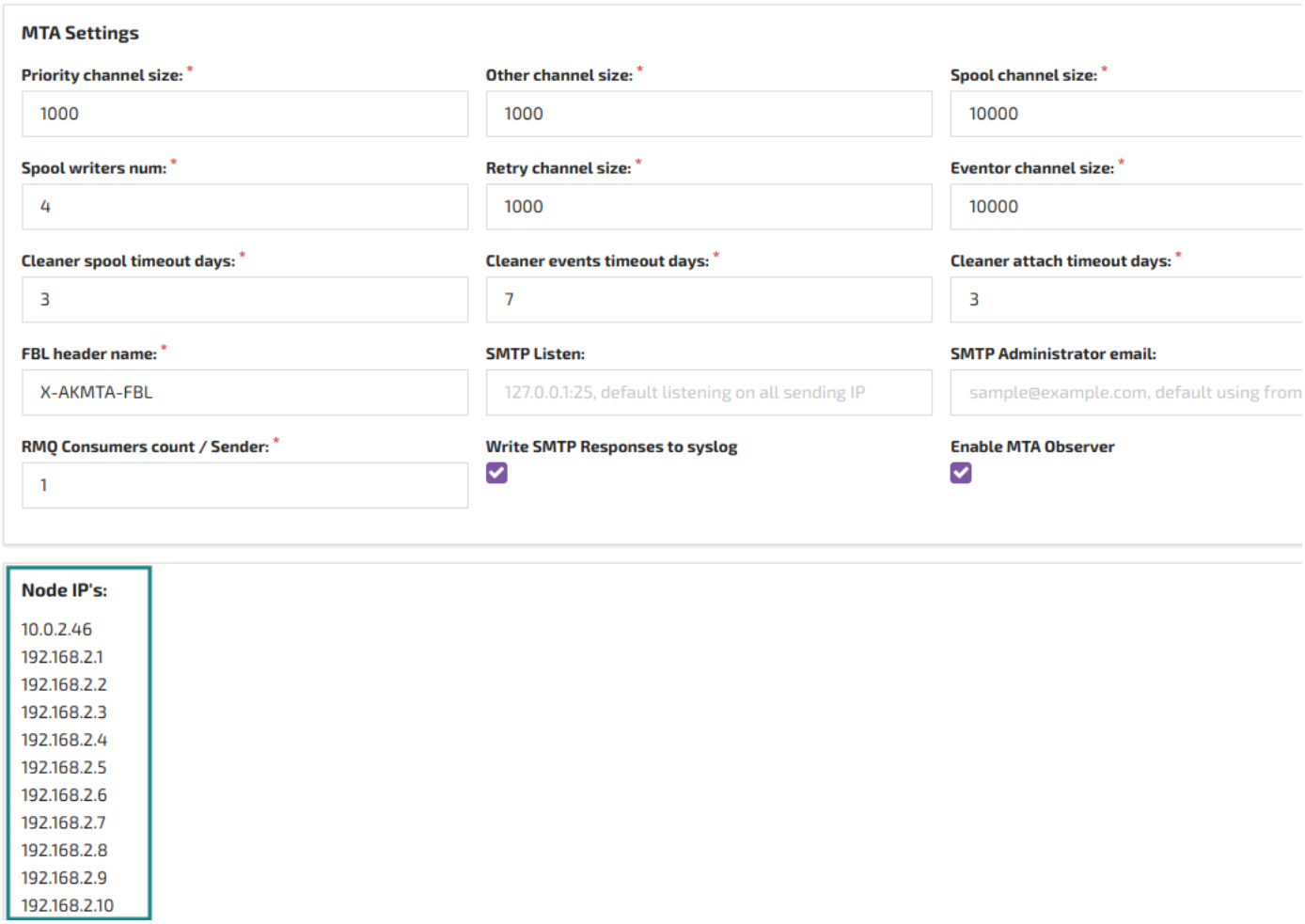
```
dpkg -l | grep mongo
apt remove mongodb-org mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
```

To free up disk space, remove data and log directories. Do this only after confirming data integrity on the

```
rm -rf /var/lib/mongodb/*
rm -rf /var/log/mongodb/*
```

Adding sender IP addresses

1. Add new IP addresses to any interface of your operating system.
2. In Altcraft MP administrative panel open node settings and save it to enable new IPs.
3. Reopen settings to make sure the new IP addresses are available:



4. Add the new IPs to your sender and restart the node to sync the settings.

Deduplication request settings

Repeat requests are API requests that contain the same headers and the same data as the original reques
The platform will not accept a repeated request if it changes the data in order to avoid duplicate events. T
To configure deduplication, specify the following parameters in the main.json configuration file:

```
"API_DEDUPLICATION_ENABLE": true,
"API_DEDUPLICATION_LEVELDB_PATH": "data/api/deduplication",
"API_DEDUPLICATION_LIVE_TIME_SEC": 86400,
"API_DEDUPLICATION_PERIOD_SEC": 1800
```

Parameter	Type	Default value	Required
API_DEDUPLICATION_ENABLE	**bool**	**true**	**No**
API_DEDUPLICATION_LEVELDB_PATH	**string**	**data/api/deduplication**	**No**
API_DEDUPLICATION_LIVE_TIME_SEC	**int**	**86400**	**No**
API_DEDUPLICATION_PERIOD_SEC	**int**	**1800**	**No**

PostgreSQL database for account data

PostgreSQL installation

You can find PostgreSQL installation instruction for Ubuntu here: https://www.postgresql.org/download/lin

> **INFO**
>
> **PostgreSQL must be connected to enable the Market module. For correct operation, PostgreSQL version**

User and database configuration

> **CAUTION**
>
> **A standard variant with local database installation is described. In this example we use `altcraft` as the u**

After you have PostgreSQL installed, make the following preparations:

- **Re-login as the newly created user postgres:**

```
su - postgres
```

- **Run the PostgreSQL command shell:**

```
psql
```

- **Create a user through which we will connect to PostgreSQL:**

```
CREATE USER altcraft WITH PASSWORD 'myPassword';
```

- **Check if the user has been created:**

```
\du
```

- **Create empty databases:**

```
CREATE DATABASE ak_init_db OWNER altcraft;
CREATE DATABASE ak_account_1 OWNER altcraft;
```

- **Add the following lines to the end of the `/etc/postgresql/<version_postgresql>/main/pg_hba.conf` file:**

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host      all         altcraft       localhost              password
```

Then go to the administrative panel for further customization.

Platform settings

In the Admin panel, go to the Account Area — External data configuration section. It is necessary to creat
Select the connection type PostgreSQL. Specify the database name, host, port and credentials of the use

External data configurations > Edit

* **Name**

PostgreSQL Market connector

Type

PostgreSQL

* **Database name**

default_market

* **Connection timeout**

1

* **Query timeout**

1

* **Protocol**

TCP

* **Host**

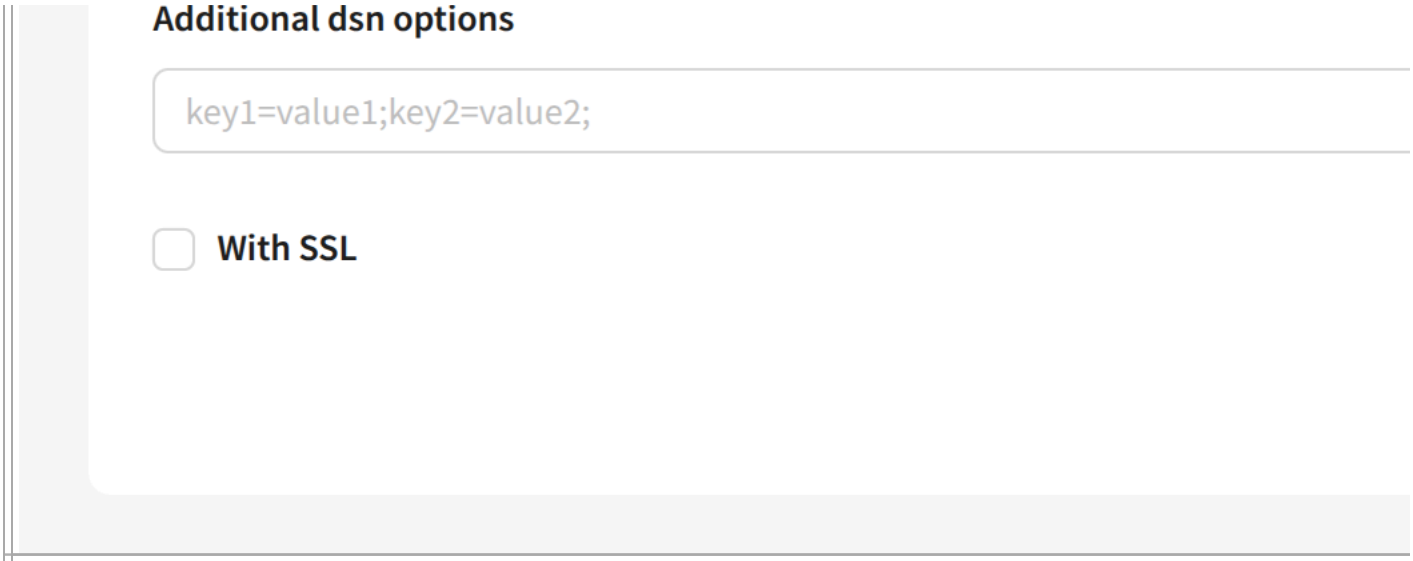
5432

* **Port**

3306

* **Username**

altcraft



Add the created configuration to your account. To do this, edit your account and select the configuration i

INFO

You can learn more about setting up connectors to databases here.

Proxy server settings

Main proxy server parameters

To configure the proxy, there are several parameters for different connections in the configuration file. If r proxy will not be used for this connection.

Parameter	
HTTP_PROXY	Proxy s
PREVIEWS_PROXY	Proxy f
LICENSE_PROXY	Proxy s
YM_PROMOCODES_PROXY	Proxy s
NET_GOOGLE_PROXY	Proxy s
NET_FACEBOOK_PROXY	Proxy s
NET_VK_ADS_PROXY	Proxy f

Exceptions for proxy server

The platform may access the user's internal resources for some information (for example, data for templa is stored in the `INTERNAL_RESOURCES` object:

```
"INTERNAL_RESOURCES": [string]
```

Inside this object, you can list local addresses and domains that the proxy server will not process. The va

1. **Check if the address exists in `INTERNAL_RESOURCES`.**
2. **Checking the parameter corresponding to the connection (e.g., `PREVIEW_PROXY`)**
3. **If no parameter is specified, the value from the main parameter `HTTP_PROXY` is used.**

Example configuration

```
"HTTP_PROXY":"http://proxy.example.ru:3128",
"PREVIEWS_PROXY":"http://proxy.example.ru:3117",
"YM_PROMOCODES_PROXY":"noproxy",
"LICENSE_PROXY":"http://proxy.example.ru:3089",
"NET_GOOGLE_PROXY":"noproxy",
"NET_FACEBOOK_PROXY":"noproxy",
"NET_VK_ADS_PROXY":"noproxy",
"INTERNAL_RESOURCES": ["10.10.10.10", "internal.lan"]
```

***The Meta organization, which owns Instagram, Facebook and WhatsApp products, has been declared as**

Keycloak Integration with Altcraft

Access to Altcraft can be provided via the Keycloak centralized authentication system. This allows using Keycloak supports modern authentication and authorization protocols (OpenID Connect, OAuth 2.0, SAML)

Keycloak Installation

Install Keycloak. For a quick start, you can use the official Docker image.

Option 1. Running a container

```
docker run -p 8080:8080 \
  -e KC_BOOTSTRAP_ADMIN_USERNAME=admin \
  -e KC_BOOTSTRAP_ADMIN_PASSWORD=admin \
  -v keycloak_data:/opt/keycloak/data \
  quay.io/keycloak/keycloak:26.0.6 start-dev
```

Option 2. Creating a container

```
docker create --name keycloak \
  -p 8080:8080 \
  -e KC_BOOTSTRAP_ADMIN_USERNAME=admin \
  -e KC_BOOTSTRAP_ADMIN_PASSWORD=admin \
  -e KC_LOG_LEVEL=DEBUG \
  -e KC_LOG_LEVEL_ORG_KEYCLOAK_PROTOCOL_OIDC=TRACE \
  -v keycloak_data:/opt/keycloak/data \
  quay.io/keycloak/keycloak:26.0.6 start-dev
```

The Keycloak administrative console is available at the following addresses:

`http://localhost:8080` **(locally)**

`http://10.200.0.100:8080` **(via VM IP address)**

Realm Configuration

Create a new realm named `oauth_app`

Create realm

A realm manages a set of users, credentials, roles, and groups. A user b
only manage and authenticate the users that they control.

Resource file

Drag a file here or browse to upload

1

Upload a JSON file

Realm name *

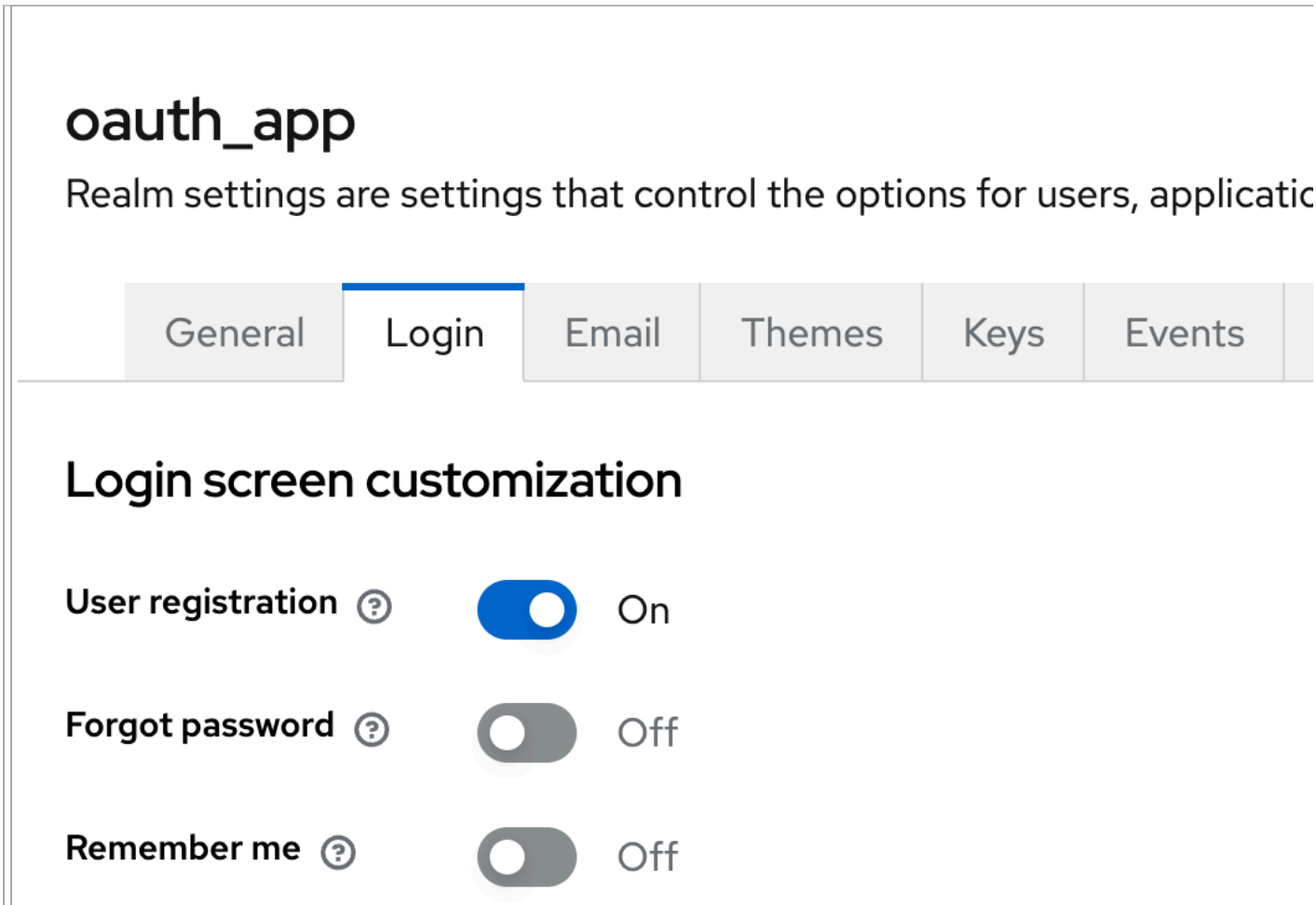
oauth_app

Enabled

On

Create Cancel

For isolated testing, enable user registration. In Realm Settings → Login, enable the User Registration par



Create a user via the realm account page: `http://localhost:8080/realms/oauth_app/account`

Client Configuration

Altcraft uses two separate clients:

`altcraft_admin` **— administrator panel**

`altcraft_platform` **— platform**

Create a client with the following parameters:

- **Client ID:** `altcraft_admin` **or** `altcraft_platform`
- **Client Protocol:** `openid-connect`
- **Enable the** `Client authentication` **parameter:**

Configure the dependent fields accordingly (for the platform client, use the platform domain).

Roles and Groups

The following role types are available in Keycloak:

- **Realm Roles — global roles within the realm**
- **Client Roles — roles associated with a specific client (not used in the current integration)**
- **Composite Roles — roles that include other roles (not used in the current integration)**
- **Default Roles — roles automatically assigned to new users**

Assign roles to a user:

Option 1. Using Realm Roles

1. **Create a role in the Realm Roles section.**
2. **Open the user.**
3. **In the Role mapping section, select Realm roles in the Filter by clients parameter and assign the requ**

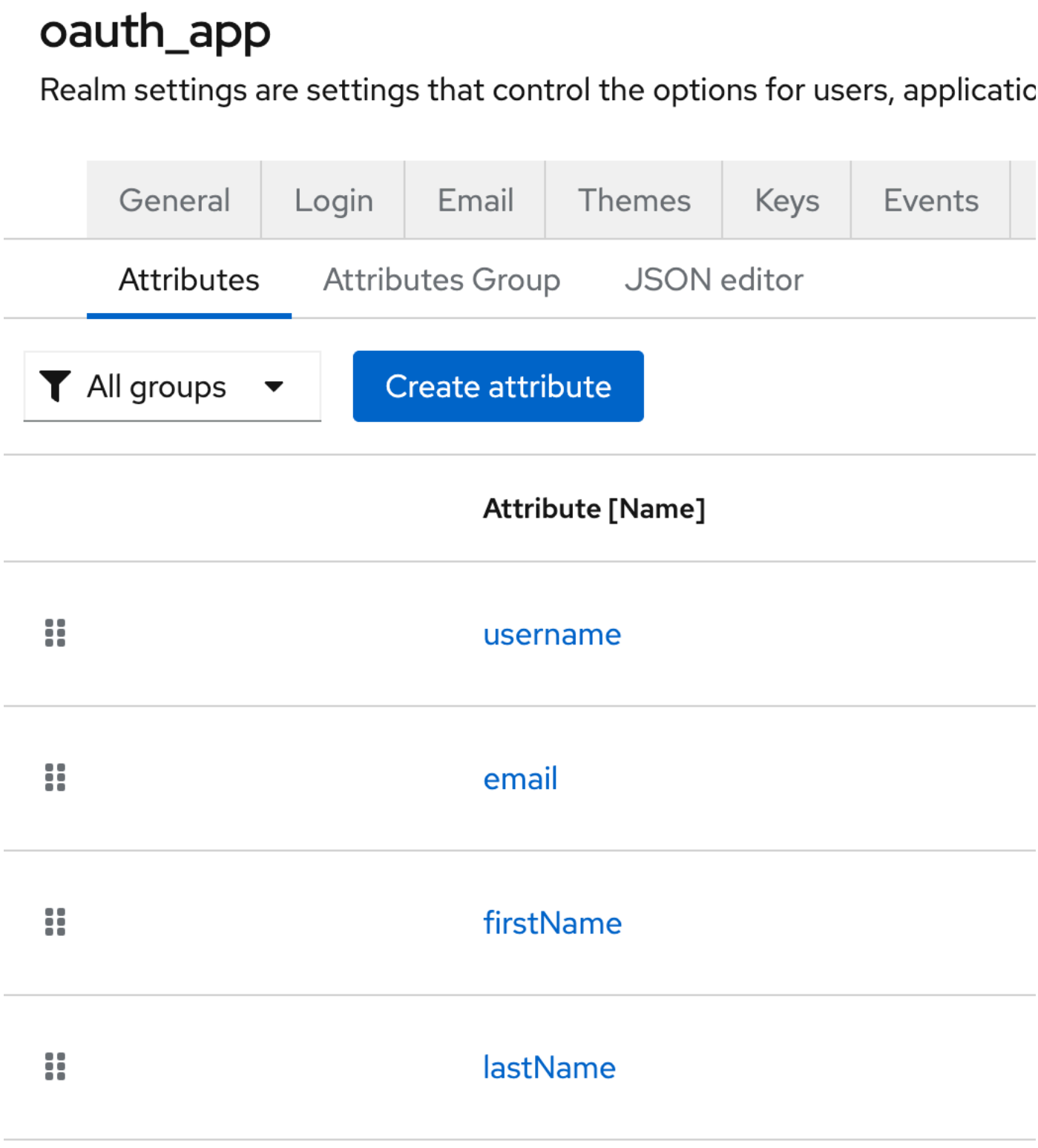
Option 2. Using Groups

1. **Create a group.**
2. **Assign a Realm Role to the group.**
3. **Add the user to the group.**

Client Scopes

Altcraft requires the groups parameter to be present in the token. This parameter is used to determine wh
To add this parameter to the token, it is recommended to use the predefined dedicated client scope (`<clie`

User Attributes



An attribute is created (or imported) at the realm level: Realm Settings → User profile → Attributes

- **Attribute (Name) — the technical name of the attribute. Used in code, tokens, and APIs. Must be uniqu**
- **Display name — the human-readable name of the attribute (optional);**
- **Multivalued — enables support for multiple values;**
- **Attribute group — the group used to display the attribute in the UI (for example, Contact, Personal Inf**
- **Who can edit / Who can view — defines which user categories are allowed to view and modify the attr**
- **Validations — a set of rules for validating the attribute value when saving data;**

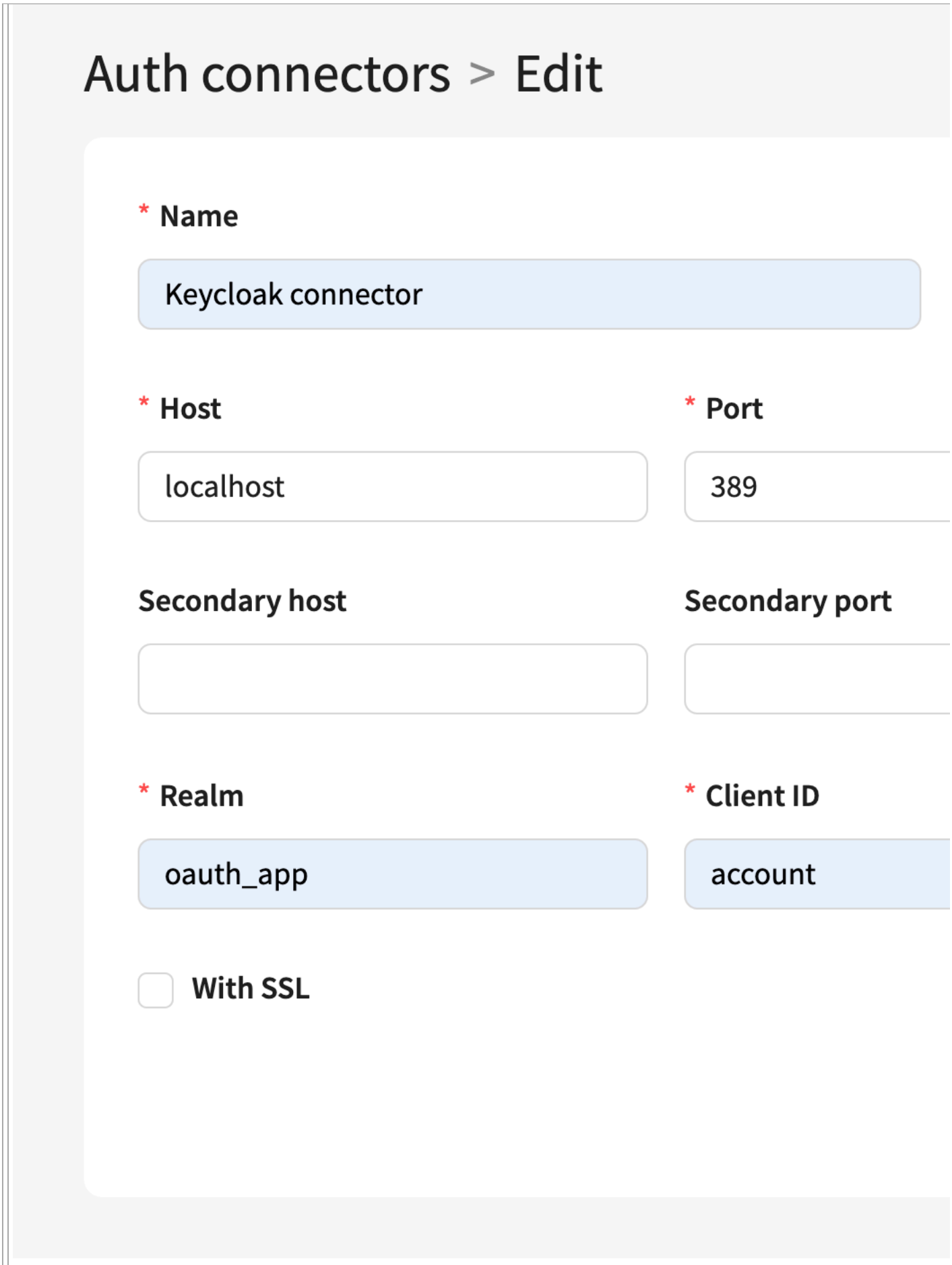
- **Annotations — arbitrary metadata.**

After creating a user attribute, it must be added to a client scope so that its value is included in the token
To pass the attribute value into the token, add a mapper with the following parameters:

- **Mapper type: User Attribute — maps a custom user attribute into tokens;**
- **Name — a unique mapper name in the configuration;**
- **User Attribute — the attribute key from the user profile (for example, “phone”, “department”);**
- **Token Claim Name — the name of the claim in the JWT token where the attribute value will be placed**
- **Claim JSON Type — the data type of the value in the token (String, int, boolean, JSON);**
- **Add to ID token: On — include the attribute in the ID token (for client applications);**
- **Add to access token: On — include the attribute in the access token (for backend APIs);**
- **Add to userinfo: On — include the attribute in the /userinfo endpoint response;**
- **Add to token introspection: On — include the attribute in the token introspection response;**
- **Multivalued: Off — the attribute contains a single value, not an array;**
- **Aggregate attribute values: Off — do not aggregate values from multiple sources.**

Configuring Keycloak Access in the Administrator Panel

Configuring access via Keycloak in Altcraft is performed similarly to [configuring access via LDAP] and is d
To configure access, log in to the Altcraft Administrator Panel and navigate to Account Area → Auth Conn
When creating the connector, specify the following parameters: connector name, Keycloak host and port,



After saving the connector, it becomes available for selection in the account settings and can be used for

User Authorization Flow via Keycloak

User authorization via Keycloak in Altcraft is implemented using the standard OIDC redirect flow.

When signing in, the user enters their login in the Altcraft authentication form. At this stage, the platform redirected to the Keycloak login page.

On the Keycloak side, the user enters their credentials and completes authentication. After successful ve

Altcraft validates the received token, extracts the user's groups and attributes, and determines access ri

Getting HTTP service statuses

Correct balancing of web services requires the user to constantly and accurately transmit the status of p
The status requests are as follows:
`/${AK-SERVICE-NAME}/health`
`/${AK-SERVICE-NAME}/readiness`
where `${AK-SERVICE-NAME}` is a name of the platform service: `webcontrol`, `webadmin`, `webapi`, `trkaction`, `trkwebctr`

List of requests

Request	Name of a service
GET `/webcontrol/health`	`webcontrol`
GET `/webadmin/health`	`webadmin`
GET `/webapi/health`	`webapiapi`
GET `/trkaction/health`	`trkaction`
GET `/trkwebctrl/health`	`trkwebctrl`
GET `/cookiesaver/health`	`cookiesaver`
GET `/webcontrol/readiness`	`webcontrol`
GET `/webadmin/readiness`	`webadmin`
GET `/webapi/readiness`	`webapi`
GET `/trkaction/readiness`	`trkaction`
GET `/trkwebctrl/readiness`	`trkwebctrl`
GET `/cookiesaver/readiness`	`cookiesaver`

Response to request

Requests of type `/health` always return the code 200 with the text “OK” in the response.
If there are no problems, `/readiness` requests return the code 200 and the following response:

```
{
        "status": "OK"
}
```

If there are any problems, the code 503 (Service not available) and the response will be returned:

```
{
        "status": "FAIL",
        "failed": ["<name of an error>"]
}
```

The following errors can be specified in the `"failed"` array:

- **started - the service startup has not been completed yet**
- **mongo - the service does not have access to Mongo**
- **rmq-alive - the service does not have access to RabbitMQ**
- **rpc-procrpc - no RPC access to ProcRPC service or ProcRPC service does not have access to Mongo**
- **rpc-proctask - no RPC access to ProcTask service or ProcTask service does not have access to Mong**

INFO

More information about the occured errors can be found in the logs of services.

Configuration MongoDB logs rotation

Log files are critical for analyzing errors encountered when executing platform processes. However, outda for MongoDB.

To configure rotation using the `logrotate` utility, create the `/etc/logrotate.d/mongod` file with the following co

```
/var/log/mongodb/*.log {
    size 100M
    rotate 7
    compress
    missingok
    notifempty
    sharedscripts
    copytruncate
    postrotate
        /bin/kill -SIGUSR1 $(cat /var/lib/mongodb/mongod.lock 2> /dev/null) 2> /dev/null || true
    endscript
}
```

> **NOTE**
>
> **When installing MongoDB from a deb package, the above file is created automatically. You can edit it to**

The main parameters of the script:

- **`size` - the size above which the file will be rotated at the next `logrotate` startup**
- **`rotate` - number of saved file rotations**
- **`compress` - old versions of log files will be compressed with `gzip`**
- **`notifempty` - do not save the log file in rotation if it is empty**

Above is the recommended configuration of the utility, you can customize it as you wish. All possible para

Configuration of system constants and directories

The system settings of the platform are regulated in configuration files. Such JSON files contain various p

> CAUTION
>
> **If you are not sure that you will be able to create and customize a custom configuration yourself, we stro the platform operation will be disrupted.**

Creating a custom configuration

In order to change the JSON configuration, you need to create a copy of the original file and rename it to
For example, you want to add new cities to the `"cities.json"` config. To do this, create a copy of the `"citie` after saving the new config.
Settings from custom files like `custom.` will be loaded in priority. If critical errors are detected in this confi platform will load the default configuration.
To configure custom communication channels from the "channels " folder, it is not necessary to add the `c`

List of JSON configurations

File	
bots_ip.json	Contains a list of IP addresses to be identified as bots, a
bots_ua.json	Contains a list of UserAgents that will be identified as bo
browsers.json	Dictionary of major browsers
cities.json	cities.json
countries.json	Dictionary of countries
default_eventgen.json	Event Generator default settings
devices.json	Dictionary of device types
domain_groups.json	Dictionary of domain groups for MTA ISPs
genders.json	Dictionary of sex and genders
languages.json	Available languages for the platform interface
oses.json	Dictionary of Operating Systems
platforms.json	Dictionary of major user platforms
preset_reports.json	Standard metrics for the summary report
sender_types.json	Sender types supported by the platform
sql_requests.json	SQL query templates
tlds.json	Dictionary of top-level domains
tzones.json	Platform-supported time zones
reports/channels.json	Channel report parameters
reports/traffic.json	Traffic report parameters
tag_manager/built_in_vars.json	Tag manager system variables
platforms.yaml	Additional config file to define the user's platform and OS

Configurations reconfigurable in Admin Panel

File	
mta_globals.json	Ge
mta_isps.json	MT
mta_lock_rules.json	MT
mta_retry_rules.json	St
mta_strategies.json	St

Filtering bot actions

The Altcraft platform can detect bot activity and modified proxy settings and is able to filter out these eve
The platform checks each connection and compares its pattern or subnet with those in the directory. Set
Validation can be performed on the following events:

Event	
import_form	Import from a form with a web layer
import_popup	Import from pop-up
open	Opening a message
click	Clicks and confirms for all channels and redirect to the subscription manager
click_sms	Clicks on the SMS handler
read	Checking the read pixel in an email
unsub	Unsubscription of channels
click_push	Clicks on the push handler
pixel_open	Registration of target and redirection through pixel

The following actions are available for the identified bot/proxy connection:

Action	
event_ignore	These events are completely ignored. Used to ignore events that were not committed by the user, but the target action (for example, antivirus
ua_ignore	Ignores UserAgent events and all associated data. Applies when a user performs an action, but because there was a data corruption (e.g., usin
ip_ignore	Ignores the IP address of the event and all associated data, including `geo_ip`. Applies when a user performs an action, but because there has b
block	The request is blocked, opening an error page. Used when you want to protect against malicious traffic (e.g., botnets or crawlers).
log	Regardless of other actions, additionally records all details of the event in the info level of the logging (Bot detector action)
none	Used if you want to temporarily disable actions for a particular rule

CAUTION

If you need to change the configuration, create a copy of the config file and name it "custom.bots_ua.jso

Structure of bots_ua.json

```
{
    "pattern": "MTRobot",  // Pattern
    "events": ["click", "open"],  // List of events to check ([] - all supported events)
    "actions": [ // List of actions to take (default [] - ua_ignore and ip_ignore)
        "ua_ignore",
        "ip_ignore"
    ],

    // Information fields (optional)
    "added": "2023-09-08",
    "url": "https://metrics-tools.de/robot.html",
    "instances": [
        "MTRobot/0.2 (Metrics Tools Analytics Crawler; https://metrics-tools.de/robot.html; crawler@metrics-tools.de)"
    ],
    "source": [
        "https://raw.githubusercontent.com/monperrus/crawler-user-agents/master/crawler-user-agents.json"
    ]
},
```

Structure of bots_ip.json

```
{
    "network": "87.240.128.0/18", // Subnets
    "events": ["click", "open"],  // List of events to check ([] - all supported events)
    "actions": [ // List of actions to take (default [] - ua_ignore and ip_ignore)
        "ua_ignore",
        "ip_ignore"
    ],

    // Information fields (optional)
    "added": "2023-09-08",
    "url": "https://metrics-tools.de/robot.html",
    "instances": [
        "MTRobot/0.2 (Metrics Tools Analytics Crawler; https://metrics-tools.de/robot.html; crawler@metrics-tools.de)"
    ],
    "source": [
        "https://raw.githubusercontent.com/monperrus/crawler-user-agents/master/crawler-user-agents.json"
    ]
},
```

TIP

If you want to temporarily disable a rule, you should specify “none” in the "actions " parameter. If you sp

Directory of gender markers

Gender markers for the platform are contained in the “genders.json” directory file. By default, it contains
If you need to change the configuration of the directory, for example, to add other designations, create a

CAUTION

Do not make any changes to the original files! They should remain unchanged in case there is an error in

Structure of genders.json


```
[
    {
        "id": 1,
        "title": "Male",
        "values": [
            "male",
            "1",
            "Male",
            "MASCULINE",
            "M",
            "MAN",
            "MÄNNLICH",
            "♂",
            "МУЖ",
            "МУЖСК",
            "МУЖСКОЙ",
            "МУЖЧИНА"
        ]
    },
    {
        "id": 2,
        "title": "Female",
        "values": [
            "female",
            "2",
            "Female",
            "FEM",
            "FEMININE",
            "F",
            "WOMAN",
            "W",
            "WEIBLICH",
            "♀",
            "Ж",
            "ЖЕН",
            "ЖЕНСК",
            "ЖЕНСКИЙ",
            "ЖЕНЩИНА"
        ]
    }
]
```

Personnel requirements

Requirements for platform support specialists from the customer’s side

Specialists who monitor platform health, administer and support the customer’s infrastructure must have

- **technical education;**
- **knowledge of the TCP / IP protocols;**
- **knowledge of the basic principles of the database;**
- **Linux OS administration skills for at least 1 year;**
- **knowledge of the functionality of the platform;**
- **skills in working with MongoDB;**
- **knowledge of database recovery tools and server performance monitoring.**

NOTE

Elimination of software errors in the platform operation is carried out only by specialists of Altcraft LLC,

Platform maintenance processes

Consulting users and administrators on operation issues

The users and administrators of the platform are consulted by Altcraft LLC.
The consultation can take place in a personal chat, via email or phone, depending on the level of support

Regular platform updates

For the entire period of the right to use the platform, the customer is granted the right to receive regular f
The number of updates is at least 4 major updates per year.
One update is released at least once a quarter.

Eliminate software errors

Elimination of software errors in the platform operation is carried out only by specialists of Altcraft LLC, a
All diagnostic information can be shared via a personal support chat or by email to team@altcraft.com.

Platform modification

Creation, modification, modernization of platform components / services is carried out only by specialists
The platform development team has the necessary knowledge to work with all the components.
The customer has the opportunity to agree on the possibility of finalizing the platform's functionality to so
To agree on specifications, please contact customers.department@altkraft.com.

Platform updating

> INFO
>
> **After the update, be sure to check the node synchronization status and update its settings if necessary.**

Installer

Verify the installer archive integrity with `md5sum`:

```
md5sum AKD_Auto_Installer | grep -q $(<AKD_Auto_Installer.md5) && echo "OK" || echo "Incorrect checksum"
```

Launching the installer with `-u` or `-update` attributes starts the updating process instantly.

```
bash AKD_Auto_Installer --update
```

You can specify your Altcraft MP installation folder as well:

```
bash AKD_Auto_Installer --update /opt/akd
```

After you launch the update your system will be checked for a running Altcraft MP instance. If it is found,

If any configuration changes occur, the installer will notify you before the platform starts over.

Manual update

Updates are provided in archive file: `AKD.tar.gz`

All you need is to unpack the file in Altcraft MP base directory.

```
tar -xzf AKD.tar.gz
```

No configuration changes are required.

Platform service monitoring

Depending on the monitoring system used in the organization, a specialist must choose a method for moni
The page describes all the processes that need to be kept in working order for the platform to fully operat
We recommend checking for the existence of a process by name using the tools built into the monitoring s

> NOTE
>
> **The following information is necessary :**
>
> - **RAM usage;**
> - **CPU usage;**
> - **disks ad free space;**
> - **opened files and connections.**

Components

Name
MongoDB (control) **Single or cluster**
MongoDB (accounts) **Single or cluster**
SSDB (actions)
SSDB (hb)
SSDB (notify)
SSDB (stats)
RabbitMQ

Processes and services

Platform executable files are located at `{BASEDIR}/ak/bin`**.**
PID-files are located at `{BASEDIR}/pids`**.**
Logs are located at `{BASEDIR}/logs`**.**

> TIP
>
> **For complete list of processes you can use** `{BASEDIR}/akd --processes`**.**

Binary file name	Process name	Default IP	
adminwebcontrol	**AK:adminwebcontrol**	**127.0.0.1**	**8081**
akmtad ***Can be moved to a remote host***	**AK:akmtad**	**127.0.0.1**	**13444**
api	**AK:api**	**127.0.0.1**	**8801**
cookie_saver ***Can be moved to a remote host***	**AK:cookie_saver**	**127.0.0.1**	**8099**
proctask ***Can be moved to a remote host***	**AK:proctask**	**127.0.0.1**	**8962**
procactions	**AK:procactions**	**-**	**-**
proceventgen	**AK:proceventgen**	**-**	**-**
prochook	**AK:prochook**	**-**	**-**
procpixel	**AK:procpixel**	**-**	**-**
procintegras	**AK:procintegras**	**-**	**-**
procleadsaver	**AK:procleadsaver**	**-**	**-**
procnotify	**AK:procnotify**	**-**	**-**
procpush	**AK:procpush**	**-**	**-**
procresume	**AK:procresume**	**-**	**-**
procrpc	**AK:procrpc**	**127.0.0.1**	**8972**
procsenderev	**AK:procsenderev**	**-**	**-**
procsmsev	**AK:procsmsev**	**-**	**-**
procsmslisten	**AK:procsmslisten**	**127.0.0.1**	**8803**
proctrigger	**AK:proctrigger**	**-**	**-**
procwebver	**AK:procwebver**	**-**	**-**
procworkflow	**AK:procworkflow**	**127.0.0.1**	**6060**
pywebcontrol	**AK:pywebcontrol**	**127.0.0.1**	**8080**
stataggregator	**AK:stataggregator**	**-**	**-**
tariffcontroller	**AK:tariffcontroller**	**-**	**-**

Binary file name	Process name	Default IP	
trkaction ***Can be moved to a remote host***	**AK:trkaction**	**127.0.0.1**	**8095**
trk_amazon_sns ***Can be moved to a remote host***	**AK:trk_amazon_sns**	**127.0.0.1**	**8093**
trkmandrill ***Can be moved to a remote host***	**AK:trkmandrill**	**127.0.0.1**	**8094**
trkimage ***Can be moved to a remote host***	**AK:trkimage**	**127.0.0.1**	**8096**
trkpush ***Can be moved to a remote host***	**AK:trkpush**	**127.0.0.1**	**8100**
trkread ***Can be moved to a remote host***	**AK:trkread**	**127.0.0.1**	**8097**
trksms ***Can be moved to a remote host***	**AK:trksms**	**127.0.0.1**	**8098**
trkwebversion ***Can be moved to a remote host***	**AK:trkwebversion**	**127.0.0.1**	**8199**
webcontrol	**AK:webcontrol**	**127.0.0.1**	**8082**

Problem diagnostics

Additionaly you can gather the required information and ask our team to help.

This page will give you the instructions.

RabbitMQ migration

If your RabbitMQ broker starts to consume too much hardware resources (CPU or memory) it is a good pra
This guide will show you how to transfer RabbitMQ messages to another instance with the help of Shovel

> **TIP**
>
> **For minimum latency the new RabbitMQ server should be located close to controlling server.**

Steps

1 Configure new server for RabbitMQ

Create virtual hosts and install RabbitMQ Management Plugin:

```
#!/usr/bin/env bash

user="rabbit"
password="password"

# Configure
rabbitmqctl add_user "$user" "$password"
rabbitmqctl set_user_tags "$user" administrator
rabbitmqctl set_permissions -p / "$user" ".*" ".*" ".*"

vhosts=(
  "akmta_events"
  "akmta_senders"
  "akmta_stat"
  "amazon_sns_events"
  "dmta_events"
  "dmta_pools"
  "dmta_stat"
  "emaildirect_events"
  "eshark_events"
  "http_api_events"
  "mandrill_events"
  "wz_events"
)

for vhost in "${vhosts[@]}"; do
  rabbitmqctl add_vhost "$vhost"
  rabbitmqctl set_permissions -p "$vhost" "$user" ".*" ".*" ".*"
done

rabbitmq-plugins enable rabbitmq_management
systemctl enable rabbitmq-server.service
systemctl restart rabbitmq-server.service
```

2 Configure Controling server

Update RabbitMQ location and credentials in `main.json` configuration file and restart the platform with `./a`

```
"RABBITMQ_HOST": "192.168.*.*",
"RABBITMQ_PASS": "abcdefghijklmnqrstuvwxyzABCDEFGH",
"RABBITMQ_USER": "rabbit",
```

3 Transfer messages from RabbitMQ instance

Install Management Plugin and Shovel Plugin.

```
rabbitmq-plugins enable rabbitmq_management
rabbitmq-plugins enable rabbitmq_shovel
rabbitmq-plugins enable rabbitmq_shovel_management
```

Open RabbitMQ web interface and go to Admin —> Shovel ManagementS` and create shovels for all queue

```
Name: deliv
Source: amqp://rabbit:password@rabbitmq-1:5672/akmta_events
Source queue: deliv
Destination: amqp://rabbit:password@rabbitmq-2:5672/akmta_events
Destination queue: deliv

Name: undeliv
Source: amqp://rabbit:password@rabbitmq-1:5672/akmta_events
Source queue: undeliv
Destination: amqp://rabbit:password@rabbitmq-2:5672/akmta_events
Destination queue: undeliv

...
```

NOTE

If you import messages directly into RabbitMQ create a shovel for `"database_import"` and `"database_import_re`

Use these commands to delete the shovels:

```
rabbitmqctl clear_parameter shovel -p "queue"
rabbitmqctl clear_parameter shovel -p "vhost" "queue"
```

Messages in all the queues for which you create shovels will be transferred to the new RabbitMQ instance

TIP

You can also use Shovel to transfer messages from old queues to new ones when upgrading RabbitMQ.

Backup and recovery

For full backup and recovery to work you will need the following data:

- **control module**
- **MongoDB databases (`account`, `control`)**
- **SSDB databases (`actions`, `hb`, `notify`, `stat`)**

> NOTE
>
> **All of the backup methods create an archive in a user's current directory. Before executing the commanc**

Control module

Backup

Archive control module data except `*.pid` и `*.log` files.

```
#!/usr/bin/env bash

tar -czf /backup/altcraft-$(date +%F).tar.gz /opt/altcraft --exclude=*.pid --exclude=*.log
```

Recovery

Unpack the archive back on its place and restart control module.

```
#!/usr/bin/env bash

/opt/altcraft/akd restart
```

> NOTE
>
> **To restore the control module from a backup copy on a new server, you must first configure the environm**

MongoDB

Data backup methods have their own documentation.
https://docs.mongodb.com/v3.4/core/backups/

> TIP
>
> **We recommend backing up using snapshots (see Back Up and Restore with Filesystem Snapshots). The r**
>
> Backup

Dump all databases and collections with the help of mongodump utility.
Use compression (`--gzip`) if you need.
mongodump utility description.
https://docs.mongodb.com/manual/reference/program/mongodump/

```
#!/usr/bin/env bash

mongodump --host 127.0.0.1 --port 27017 --gzip --archive=/backup/mongodb-control-$(date +%F).tar.gz
mongodump --host 127.0.0.1 --port 27018 --gzip --archive=/backup/mongodb-account-$(date +%F).tar.gz
```

Recovery

Recower the data with `mongorestore` and restart systemd services.
mongorestore utility description.
https://docs.mongodb.com/manual/reference/program/mongorestore/

```
#!/usr/bin/env bash

mongorestore --host 127.0.0.1 --port 27017 --gzip --arcdhive=/backup/mongodb-control-2019-06-01.tar.gz
mongorestore --host 127.0.0.1 --port 27018 --gzip --arcdhive=/backup/mongodb-account-2019-06-01.tar.gz
systemctl restart mongod-control
systemctl restart mongod-account
```

SSDB

> TIP
>
> **We recommend backing up using snapshots and then moving all data from the snapshot to an archive. Im**

```
/var/lib/ssdb
├── actions
│   ├── data
│   └── meta
├── hb
│   ├── data
│   └── meta
├── notify
│   ├── data
│   └── meta
└── stat
    ├── data
    └── meta
```

Backup

Archive catalogues, containing `data` and `meta`.

```
#!/usr/bin/env bash

tar -czf /backup/ssdb-actions-$(date +%F).tar.gz /var/lib/ssdb/actions
tar -czf /backup/ssdb-hb-$(date +%F).tar.gz /var/lib/ssdb/hb
tar -czf /backup/ssdb-notify-$(date +%F).tar.gz /var/lib/ssdb/notify
tar -czf /backup/ssdb-stat-$(date +%F).tar.gz /var/lib/ssdb/stat
```

Recovery

Unpack the archive and restart systemd services.

```
#!/usr/bin/env bash

systemctl restart ssdb-actions
systemctl restart ssdb-hb
systemctl restart ssdb-notify
systemctl restart ssdb-stat
```

Transferring the platform to a new server

Step 1. Install the necessary components for the platform to work on the new server in accordance with t
Step 2. Switch RabbitMQ by correcting the connection string in main.json to the new host.
Step 3. Make a backup according to the instructions:
For a full backup, the following data is required:

- **AKD control module;**
- **MongoDB databases (account and control can be separated or combined into one database);**
- **6SSDB databases (actions, hardbounces, nofitications and stats).**

> NOTE
> **All of the backup methods create an archive in a user's current directory. Before executing the commanc**

Control module

Backup

The recommended way to create a backup is to place the directory with the control module in an archive,

```
tar -czvf akd-backup.tar.gz /opt/akd --exclude=*.log --exclude=*.sock --exclude=*.pid
```

Recovery

Unpack the archive and move its contents:

```
tar -xzvf akd-backup.tar.gz
mkdir -p /opt/akd
cp -R akd/* /opt/akd/
```

> NOTE
> **To restore the control module from a backup copy on a new server, you must first configure the environm**

MongoDB

All backup methods are described in more detail in the official documentation::
https://docs.mongodb.com/manual/core/backups

> TIP
> **We recommend backing up using snapshots ([Back Up and Restore with Filesystem Snapshots]) (https://d large collections with a large number of indexes.**

Backup

For small instances, you can use the `utilitymongodump` https://docs.mongodb.com/database-tools/mongodum

```
mongodump --host 127.0.0.1 --port 27017 --gzip --archive=mongodb-backup.tar.gz
```

Recovery

Restore a backup using the utility `mongorestore` (https://docs.mongodb.com/database-tools/mongorestore):

```
mongorestore --host 127.0.0.1 --port 27017 --gzip --archive=mongodb-backup.tar.gz
```

ClickHouse

Backup methods are described in the official documentation:
https://clickhouse.com/docs/ru/operations/backup/

Using the utility as an example clickhouse-backup

Minimum configuration file

```
clickhouse:
  username: default
  password: "abcdefghijklmnopqrstuvwxyzABCDEF"
  host: localhost
  port: 9000
data_path: "/var/lib/clickhouse"
general:
  remote_storage: none
  backups_to_keep_local: 7
  backups_to_keep_remote: 31
```

Backup

```
clickhouse-backup create $BACKUP_NAME >> /var/log/clickhouse-backup.log
```

Recovery

```
clickhouse-backup restore $BACKUP_NAME >> /var/log/clickhouse-restore.log
```

SSDB

All backup methods are described in more detail in the official documentation:
https://ideawu.github.io/ssdb-docs/backup.html

> **TIP**
>
> **We recommend backing up using snapshots and then moving all data from the snapshot to an archive. Im**

Backup

For small instances you can use utility `ssdb-dump` (см. https://ideawu.github.io/ssdb-docs/backup.html):

```
/usr/local/ssdb/tools/ssdb-dump -h 127.0.0.1 -p 4420 -a abcdefghijklmnopqrstuvwxyzABCDEF ./ssdb-hardbounces-dump
/usr/local/ssdb/tools/ssdb-dump -h 127.0.0.1 -p 4430 -a abcdefghijklmnopqrstuvwxyzABCDEF ./ssdb-notifications-dump
```

Recovery

Clear data from previous instances and move backup contents:

```
rm -rf /var/lib/ssdb/hardbounces
mv ssdb-hardbounces-dump /var/lib/ssdb/hardbounces

rm -rf /var/lib/ssdb/notifications
mv ssdb-notifications-dump /var/lib/ssdb/notifications
```

Step 4. Check that the platform is working correctly.

- **Check that processes are running:**

```
AK:api
AK:proctask
AK:procrpc
AK:procactions
AK:procresume
AK:procsenderev
AK:procsmsev
AK:proctrigger
AK:procworkflow
AK:prochook
AK:procnotify
AK:procpixel
AK:procwebver
AK:proctrkweb
AK:procpush
AK:procpiper
AK:stataggregator
AK:tariffcontroller
AK:webcontrol
AK:procleadsaver
AK:procevent
AK:proceventgen
AK:procintegras
AK:procplmt
AK:trkaction
AK:trkimage
AK:trkread
AK:trksms
AK:trkwebctrl
AK:trkpush
AK:cookie_saver
AK:trkcustom
AK:trklisten
AK:procsmslisten
AK:trk_amazon_sns
AK:health
AK:webadmin
```

- **Check the logs of these processes for errors:**

> **NOTE**
>
> **Default path:** `/opt/akd/logs`

Step 5. Start tracking on the new host manually:

```
/opt/akd/ak/bin/proctrkweb --config /opt/akd/config/main.json
/opt/akd/ak/bin/trkaction --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trkimage --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trkread --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trksms --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trkwebctrl --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trkpush --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trkcustom --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trklisten --rpcAddrs <old_host_ip>:8972
/opt/akd/ak/bin/trk_amazon_sns --rpcAddrs <old_host_ip>:8972
```

Step 6. Stop the platform on the current host:

```
/opt/akd/akd stop
```

Step 7. Launch the platform on the new host.

> **NOTE**
>
> `restart` **used to ensure that tracking switches to the current host.**

```
/opt/akd/akd restart
```

Creating, deleting, and populating tables for statistics in ClickHouse

**In some cases, such as when setting up a replicated cluster, you may need the function to create empty t
utility.**

Command reference

aktool utility commands:

```
Init stat table.

Usage:
  aktool stats init [summary|pixel|engagement|click_map|forms|undelive|mailings|notify|channel|all] [flags]

Flags:
  -h, --help   help for init

Global Flags:
      --config string   config path (default "/opt/akd/config/main.json")
  -o, --output string   file path to output in the result of the command (default "stdout")
      --truncate        whether to truncate the file specified by --output
```

Prameters of commands

List of parameters (tables) for `delete`, `fill`, `init` commands:

Parameter	
all	**All tables for storing statistics**
summary	**summary_report** **summary_MV_%версия**
pixel	**pixel_report** **pixelUndefMV_%версия** **pixelMV_%версия**
engagement	**engagement_report** **engagement_MV**
click_map	**click_map_report** **click_mapMV**
forms	**total_form_report** **formMV** **form_pages_view_report** **form_pages_viewMV** **group_form_report** **group_formMV**
undelive	**undelive_report** **undeliveMV**
campaigns	**camp_stats**
notify	**notify_sender_stats**

Create table

Create tables for the summary report:

```
./aktool stats init summary
```

Create all necessary tables for storing statistics:

```
./aktool stats init all
```

Population table

Populate tables for forms:

```
./aktool stats fill forms
```

Populate all necessary tables for storing statistics:

```
./aktool stats fill all
```

Delete table

> NOTE
>
> **To fully delete a replicated table, you need to wait until the table data is removed from ZooKeeper. The w**

Delete tables for undelivered report:

```
./aktool stats delete undelive
```

Delete all necessary tables for storing statistics:

```
./aktool stats delete all
```

Usage of the aktool utility

Description

aktool — is a command tool that gives administrators access to various platform management functions t

To run `aktool`, start the terminal and, while in the root directory of the platform, execute the command:

`./aktool --help`

This command will display the available commands and flags.

> **NOTE**
>
> **In some cases, upgrading the platform may require you to perform a migration using `aktool`. This allows**

List of aktool commands

For more information about a specific command, type the following:

`./aktool [command] --help`

Command	
accounts	**Managing accounts**
admins	**Managing administ**
akmta	**AKMTA manageme**
cleanup	**Cleaning up data fo**
completion	**Generate an autoc**
configs	**Configuration mana**
field_migrator	**Migrating one field**
help	**Get help on any co**
history_storage	**Actions with histor**
indexes	**Creating indexes in**
keys	**Managing keys**
migrate	**Perform data migra**
nodes	**Managing nodes**
printconf	**Display current val**
profile_history	**Manage profile hist**
rabbitmq	**Manage RabbitMQ**
resource_system_fields	**Adding system field**
roles	**Actions with roles**
smid	**[SMID] decoder(/en**
stats	**Manage statistics t**
tariffs	**Manage tariffs**
users	**Manage account us**
workflow	**Scenarios manager**

accounts

`./aktool accounts`

```
The accounts command allows you to interact with accounts

Usage:
  aktool accounts [command]

Available Commands:
  create      Create account
  delete      Delete account
  disable     Disable account
  enable      Enable account
  list        Show list of accounts
  update      Update account

Flags:
  -h, --help   help for accounts

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

admins

`./aktool admins`

```
The accounts command allows you to interact with accounts

Usage:
  aktool accounts [command]

Available Commands:
  create      Create account
  delete      Delete account
  disable     Disable account
  enable      Enable account
  list        Show list of accounts
  update      Update account

Flags:
  -h, --help   help for accounts

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

akmta

`./aktool akmta`

```
The accounts command allows you to interact with accounts

Usage:
  aktool accounts [command]

Available Commands:
  create      Create account
  delete      Delete account
  disable     Disable account
  enable      Enable account
  list        Show list of accounts
  update      Update account

Flags:
  -h, --help   help for accounts

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

cleanup

`./aktool cleanup`

```
The accounts command allows you to interact with accounts

Usage:
  aktool accounts [command]

Available Commands:
  create      Create account
  delete      Delete account
  disable     Disable account
  enable      Enable account
  list        Show list of accounts
  update      Update account

Flags:
  -h, --help   help for accounts

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

completion

`./aktool completion`

```
The accounts command allows you to interact with accounts

Usage:
  aktool accounts [command]

Available Commands:
  create      Create account
  delete      Delete account
  disable     Disable account
  enable      Enable account
  list        Show list of accounts
  update      Update account

Flags:
  -h, --help   help for accounts

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

configs

`./aktool configs`

```
The accounts command allows you to interact with accounts

Usage:
  aktool accounts [command]

Available Commands:
  create      Create account
  delete      Delete account
  disable     Disable account
  enable      Enable account
  list        Show list of accounts
  update      Update account

Flags:
  -h, --help   help for accounts

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

field_migrator

`./aktool field_migrator`

```
Migrates value from old field to new. Now support ONLY phone field
                  If new field is not exists it will be created.
                  If new field is exists, value from old field (if it not empty) will override new field

Usage:
  aktool field_migrator --accID=int --dbID=int --old_field=string --new_field=string [flags]

Examples:
field_migrator --accID=1 --dbID=2 --old_field=phone --new_field=mbl_phone --config=path/to/main.json

Flags:
      --accID int          account id where placed database
      --dbID int           database with field to be migrated
  -h, --help               help for field_migrator
      --new_field string   new field to migrate to
      --old_field string   field name to be migrated

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

help

`./aktool help`

```
Help provides help for any command in the application.
Simply type aktool help [path to command] for full details.

Usage:
  aktool help [command] [flags]

Flags:
  -h, --help   help for help

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

history_storage

`./aktool history_storage`

```
The command allows you to manipulate with history storage tables.

Usage:
  aktool history_storage [command]

Available Commands:
  init        Init history tables

Flags:
  -h, --help   help for history_storage

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

indexes

`./aktool indexes`

```
The commands allows you to create all platform database indexes

Usage:
  aktool indexes [command]

Available Commands:
  ensure      Ensure indexes [account_id_1,account_id_2,...]

Flags:
  -h, --help   help for indexes

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

keys

`./aktool keys`

```
The keys command allows you to interact with keys.

Usage:
  aktool keys [command]

Available Commands:
  list        Show all keys
  remove      Remove a key

Flags:
  -h, --help   help for keys

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

migrate

`./aktool migrate`

```
Perform some data migration

Usage:
  aktool migrate [command]

Available Commands:
  importempty Mountiong checkboxes for scheduled imports for account(s)
  importfeed  Update column for import feed
  log         Migrate old logs for account(s)
  ssdbhbsupp  Transfer data from ssdb hbsupp to mongodb
  ssdbnotify  Transfer 30 days data from ssdb notify to mongodb
  webversion  Transfer web version to s3like storage

Flags:
  -h, --help   help for migrate

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

nodes

`./aktool nodes`

```
The nodes command allows you to interact with nodes

Usage:
  aktool nodes [command]

Available Commands:
  create      Create a node
  list        Show all nodes
  remove      Remove a node
  update      Update a node

Flags:
  -h, --help   help for nodes

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

printconf

`./aktool printconf`

```
Print full config parameters static or dynamic in JSON

Usage:
  aktool printconf [type=base|trk] [flags]

Flags:
  -h, --help   help for printconf

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

profile_history

`./aktool profile_history`

```
The command allows you to interact with profile history tables

Usage:
  aktool profile_history [command]

Available Commands:
  delete      Delete profile history table
  fill        Fill profile history table
  init        Init profile history table

Flags:
  -h, --help   help for profile_history

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

profiles

`./aktool profiles`

```
Usage:
  aktool profiles [command]

Available Commands:
  fix               Fixes legacy profiles
  migrate-gcg-flags migrate profiles: adding gcg flags
  subscriptions     Manage profiles subscriptions

Flags:
  -h, --help   help for profiles

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

profiles_stat

`./aktool profiles_stat`

```
Usage:
  aktool profiles_stat [command]

Available Commands:
  deduplicate Deduplicate profile history table
  delete      Delete profile stat table
  fill        Fill stat table
  init        Init profile stat table

Flags:
  -h, --help   help for profiles_stat

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

rabbitmq

`./aktool rabbitmq`

```
The command allows you to interact with profile history tables

Usage:
  aktool profile_history [command]

Available Commands:
  delete      Delete profile history table
  fill        Fill profile history table
  init        Init profile history table

Flags:
  -h, --help   help for profile_history

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

resource_system_fields

`./aktool resource_system_fields`

```
Add system fields in resource

Usage:
  aktool resource_system_fields [flags]

Flags:
  -h, --help   help for resource_system_fields

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

roles

`./aktool roles`

```
Actions with roles in accounts

Usage:
  aktool roles [command]

Available Commands:
  tag_manager_def_role Fix rights for tag_manager object in default roles

Flags:
  -h, --help   help for roles

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

smid

`./aktool smid`

```
Decode send message id

Usage:
  aktool smid [flags]

Flags:
  -h, --help        help for smid
      --json        pretty json
      --to string   convert send message id to ESMID|USMID

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

stats

`./aktool stats`

```
The stats command allows you to interact with stat tables

Usage:
  aktool stats [command]

Available Commands:
  delete      Delete stat table
  fill        Fill stats table
  init        Init stat table

Flags:
  -h, --help   help for stats

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

tariffs

`./aktool tariffs`

```
The tariffs command allows you to interact with tariffs

Usage:
  aktool tariffs [command]

Available Commands:
  create      Create a tariff
  list        Show all tariffs
  remove      Remove a tariff.
  update      Update a tariff

Flags:
  -h, --help   help for tariffs

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

users

`./aktool users`

```
The users command allows you to interact with account users

Usage:
  aktool users [command]

Available Commands:
  create      Create user
  list        Show all users
  password    Change password
  remove      Remove user
  update      Update user

Flags:
  -h, --help   help for users

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

workflow

`./aktool workflow`

```
The workflows command allows you to interact with workflows

Usage:
  aktool workflow [command]

Available Commands:
  api_nodes_convert convert api_nodes struct

Flags:
  -h, --help   help for workflow

Global Flags:
      --clickhouse_database_name string   usage database for clickhouse
      --config string                     config path (default "/opt/akd/config/main.json")
  -o, --output string                     file path to output in the result of the command (default "stdout")
      --truncate                          whether to truncate the file specified by --output
```

Users and directories engaged by the platform

Users and access rights

Platform services will be launched on behalf of the user and group specified in `AUTO_RUN_AS_USER` and `AUTO_RU` configuration file main.json. The user must have basic rights to read, write and edit files.

> **DANGER**
>
> **To allow the `akmtad` process to listen on port 25 when running as a non-root user, you need to run the foll**
> `setcap "cap_net_bind_service=+ep" akmtad`

Directories

All platform services work with their working directory, which is usually located at `/opt/akd` path. Some se
To create screenshots of templates (for example, to display a preview of a template in the interface), the

Overview

A channel in AKMP is an entity that defines interaction with a subscriber (profile) through external integra
Such interaction could be, for example, sending an SMS, a message to a CRM or to a social network comn
There are three types of channels:

- **Built-in (Email, SMS, Push) — channels are strictly described in the system code and cannot be config**
- **System — channels whose settings are supplied with the system;**
- **Custom — systems configured by the user (administrator).**

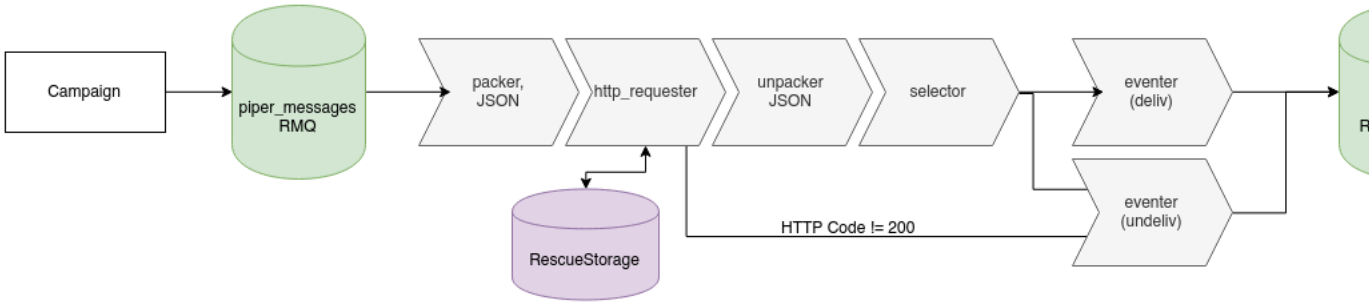
System channels are specified in the `json_consts/channels directory`.
A custom channel is configured in the platform admin panel, in the *Setup* > *Custom Channels* menu.
The channel contains:

- **Channel name;**
- **Unique numeric and unique string identifiers designed to store the channel attribute in events, data st**
- **Type of communication: personalized or non-personalized;**
- **Identifier of the system module that controls the channel or configures pipelines for interaction with e**
- **Settings for events that occur in the channel;**
- **Settings for data fields and their displays in platform entities (content).**

Non-personalized or indirect channels are used in cases where interaction occurs with an indefinite set of
For a personalized channel (direct), the mailing will create messages for each profile. Accordingly, they ca
The channel has the ability to set events that are reflected in AKMP reports, profile history, segments, etc reflect events not provided for by the platform, the custom type (custom event) is used.
Content and channel settings are specified through fields. Fields can be set for the following AKMP entitie

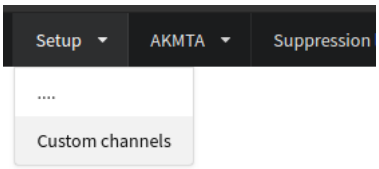
- **Template define message fields, such as the body of a text message or an image;**
- **Resource define customization fields in the resource, for example, an API key available to the user, or**
- **Mailing define message fields and settings, such as an identifier for reporting;**
- **Sender define settings on the sender, for example, an API key that is not available to the user, or an A**
- **Subscriptions define identifiers that associate a resource with a profile in a channel, for example, a us**

Interaction with external services when sending messages and receiving events on them occurs via pipel
Each pipeline consists of a JSON configuration in a special format that describes interaction with service



Creating and configuring a channel

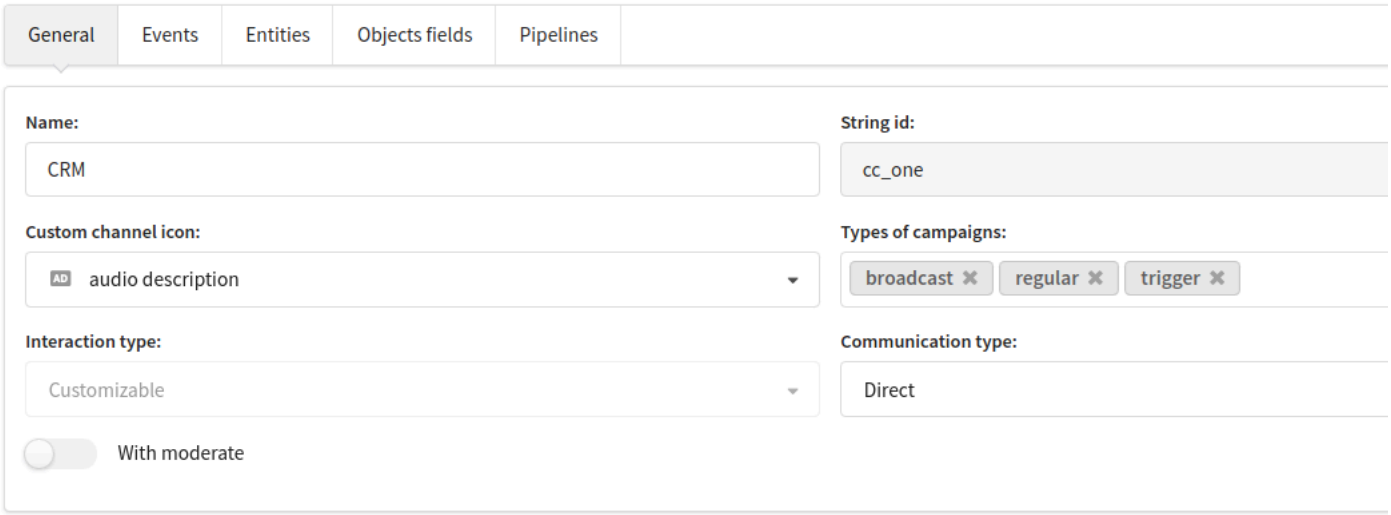
Channels are created in the system administration panel in the Custom channels section:



Before creating a channel, you will need the following information:

- **Documentation of the channel API service and interactions with it. So request to send a message, req**
- **Fields which are used when sending a message;**
- **Fields which are used to identify the subscription;**
- **Fields that will be configured by the user and which by the administrator;**
- **What events will occur on the channel and what events in the platform they correspond to.**

Basic settings



Name — channel's name.
String id - the string identifier is used in uploads, in the API and within the platform.
Custom channel's icon - channel's icon in the interface
Types of mailings - types of mailings where this channel will be available.
Communication type - direct or indirect.

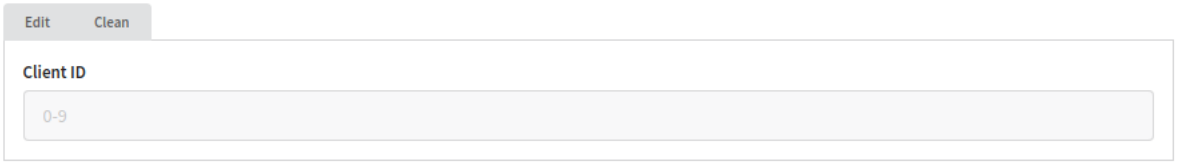
- **direct - sending content personally to each profile (example Email, SMS, Push, etc.). Requires setting**
- **indirect - sending content not to a specific profile, but to a group (example VKGroup, a banner on a we**

With moderate is responsible for enabling mailing moderation for the indirect channel. Opens the Moderat

Fields

For each of the platform objects associated with channels, you can set the following fields:

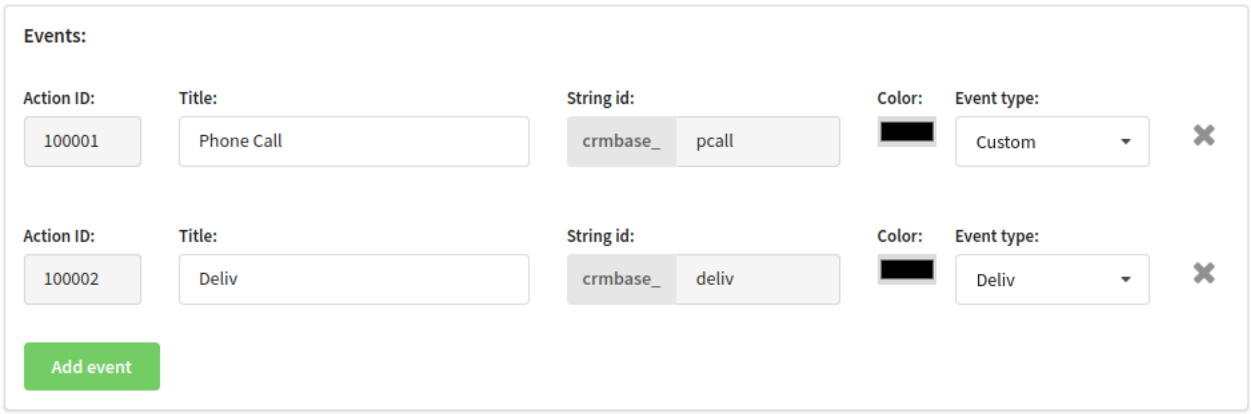
The minimum requirement is to set the *Template* fields and the fields of entities that depend on the Comm
For a direct channel, you need to set a field linking the profile to the resource - subscription:



Such a field, for example, will be available in the pipeline as `$subscription.client_id`.

Events

We create events in the channel, define a event name, string id (will be used in the API and uploads).



If the event is reflected in the platform (for example, a delivery event), then the correspondence must be i
The color is set for presentation in reports and graphs.

External Entities

The Entities configuration is described in detail here.

Pipelines

The pipeline configuration that describes the interaction with external services is specified in the Pipelin
Types of pipelines:

- **Message**
- **Schedule**
- **Listener**
- **Stop**
- **Moderate**

The configuration is described in more detail in this article.

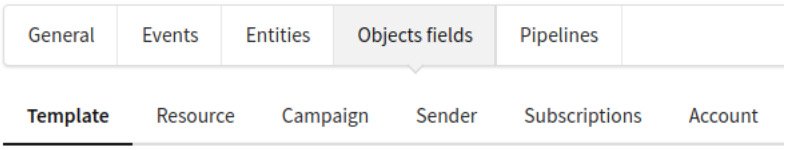
Technical limitations in custom user channels

1. No ability to further process arrays of objects in the pipeline. For example, if an external service sends attachments in the content field, the data for this field can only be sent to the external service in its unch
2. The ability to send media attachments is limited. The channel is not intended for sending attachments I sent instead of the image itself.

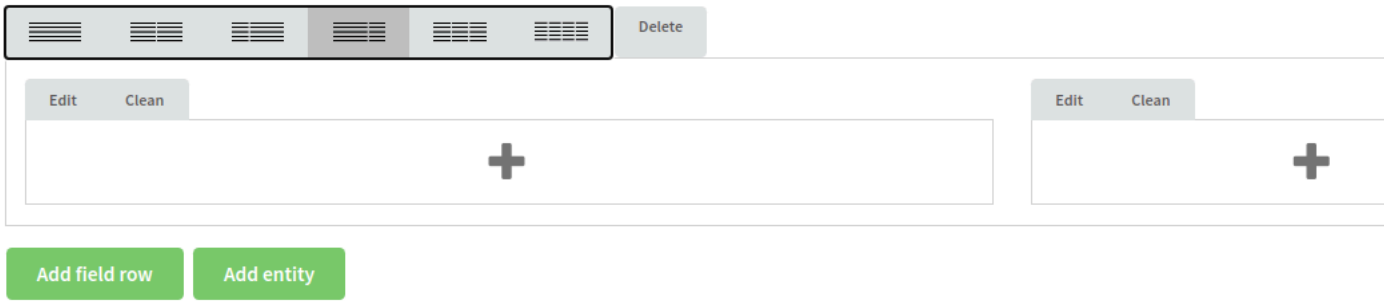
Entity field configuration

Description

For each platform object associated with channels, you can set the following fields:

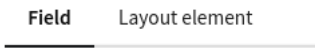


Layout is responsible for the arrangement of fields on forms. You can select several options for one-, two-



Layout elements

When you click to add an element to Layout, you can select its type.

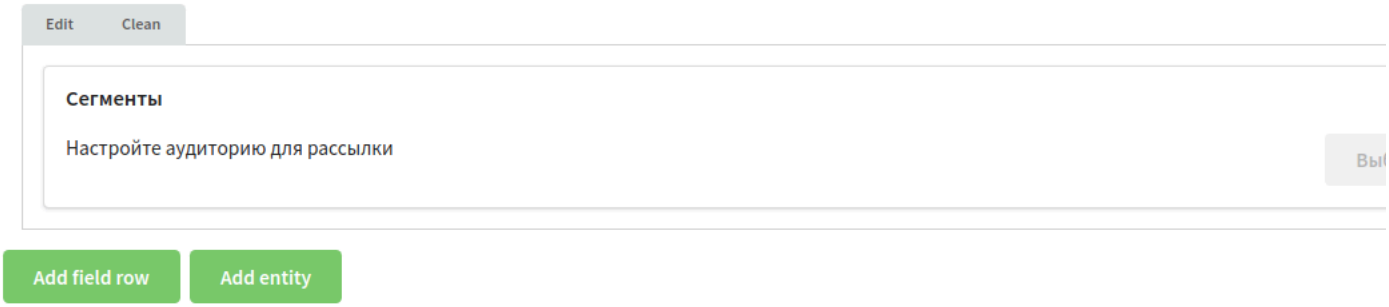


1. **Layout element is a field of this type does not store data, but simply helps to group or describe the ma**
   - **message;**
   - **header;**
   - **divider.**
2. **Field that directly store data. They are accessed through the keyword of a specific entity using a tem**
   - **int;**
   - **string;**
   - **bool;**
   - **json;**
   - **text;**
   - **html;**
   - **file;**
   - **enum;**
   - **color.**

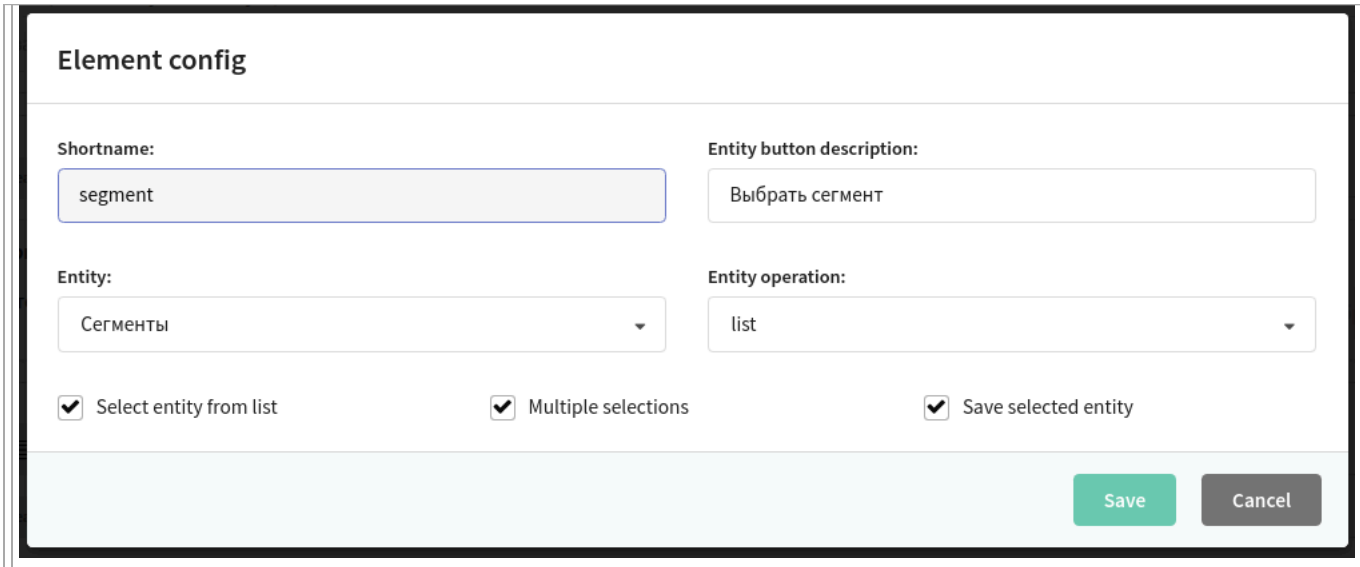
CAUTION

The set of available field types may vary depending on the entity.

The Entity field is configured separately. *Layout* has an *Add entity* button. It allows you to add a field with



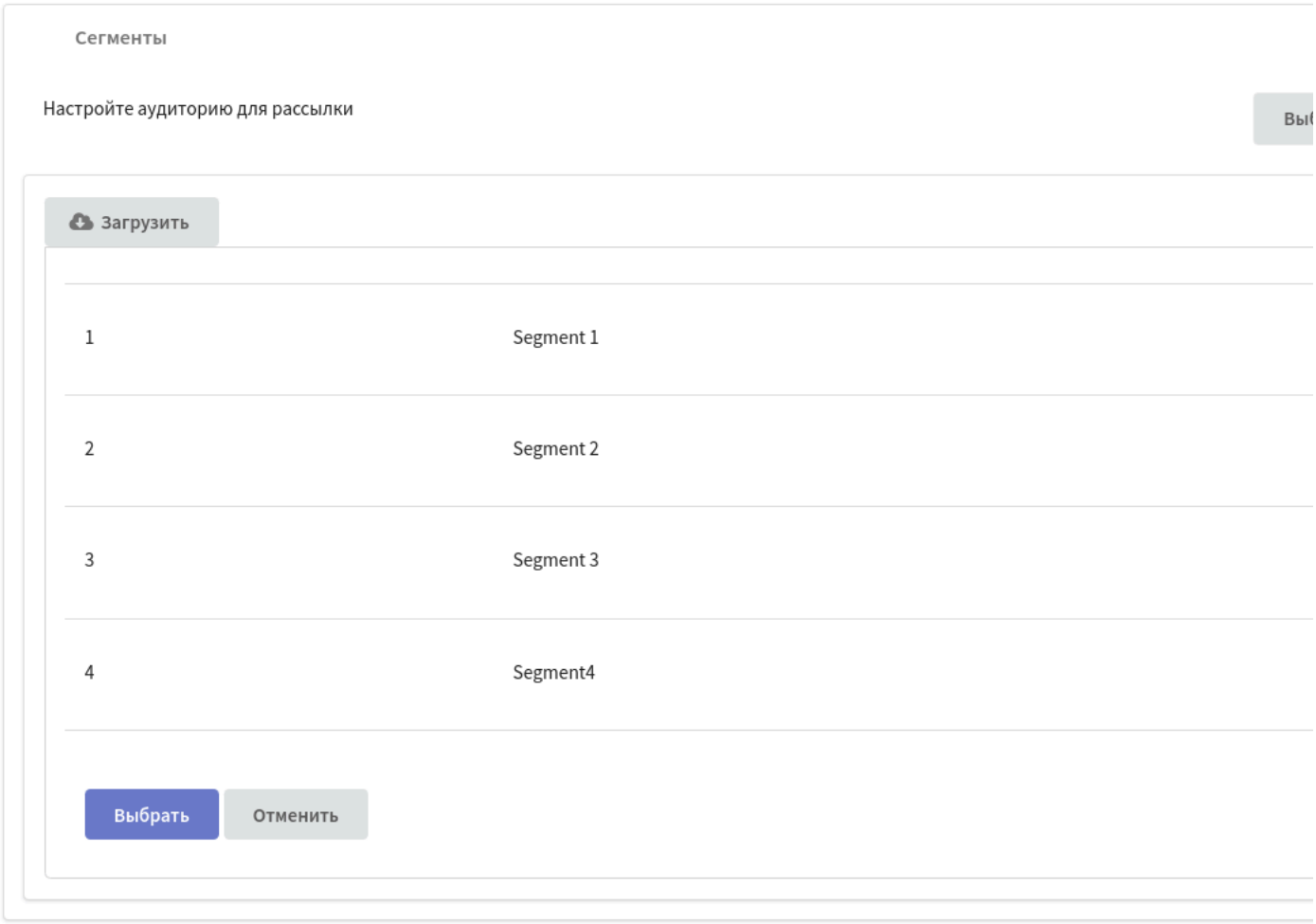
Setting up a field with an associated entity



Shortname is the name by which the data of the external entity in the pipeline will be accessed. (templat
Entity button description is the name of the button that opens editing of the external entity in the interfac
Entity is an entity that is bound to a field.
Entity operation is a selection of entity operation (Entities), the result of which will be written to this field
Select entity from list is available only for the List operation. If set, you will need to select from the list w
Multiple selection is available only for the List operation. Turns multi-select mode on or off. If disabled, on
Save selected entity is available only for the List operation. Sets whether to save data in a field while setti
For example, for the field configuration in the screenshot above, let's assume that the List operation retur

```
[
  {
    "id": 1,
    "name": "Segment 1"
  },
  {
    "id": 2,
    "name": "Segment 2"
  },
  {
    "id": 3,
    "name": "Segment 3"
  },
  {
    "id": 4,
    "name": "Segment 4"
  }
]
```

In the interface with this setting, the field will look like this (provided that the output form of the *List* oper



The screenshot shows that 2 segments are selected: Segment 1 and Segment 3. Then the following data a

```
[
  {
    "id": 1,
    "name": "Segment 1"
  },
  {
    "id": 3,
    "name": "Segment 3"
  }
]
```

In a Pipelines, for example, if a field is configured in Mailing, it will be possible to access the stored data (

```
Templates:

"$mailing.segment.%1.name"
or
"$mailing.segment"
or
"$$mailing.segment[some_field]": ["[$]id"]

Result:

"Segment 3"

[{ "id": 1, "name": "Segment 1" }, { "id": 3, "name": "Segment 3" }]

["13"]
```

If the Select entity from list option is disabled, the data list cannot be edited. All data will be recorded in t

Template language

Description

Templates are used to access event fields. For example, when generating json, you need to add a identifie

Keywords

Keywords begin with one of the following prefixes:

Parameter	
$	Inserts a value as a string.
$#	Allows you to use the original field type.
$$	Allows you to unpack an array into several repeating structures. It should only be used with a template field of the “file” type, since you often need to add several files to a message.

For the `$` and `$#` prefixes, there are two options for writing the pattern:

```
$template.field
```

And

```
${template.field}
```

> CAUTION
>
> **The $$ prefix only supports a template’s notation without curly braces.**

A pattern prefixed with $ can be freely combined with strings, i.e. a pipe field containing such a template

For example:

```
qwerty$in.a_id
```

or

```
qwerty${in.a_id}ytsuken
```

> CAUTION
>
> **Patterns prefixed with `$#` and `$$` cannot be combined with strings.**

Keywords:

Parameter	Type	
$in	Object	Data that came into the pipe fron
$in.send_message_id	string	Send Message ID. The unique ide
$in.launch_id	string	Unique submission identifier. On
$in.a_id	int	Account ID
$in.c_id	int	Mailing ID.
$origin	Object	Data received in the pipeline.
$template	Object	Data configured in the message t
$sender	Object	Data configured in the sender for
$resource	Object	The data configured in the resou
$campaign	Object	Data configured in the mailing fo
$subscription	Object	Data configured in the subscripti
$account	Object	Data configured in the account fo
$camp_schedule	Объект	Mailing start and stop settings.
$camp_schedule.start_time	time	Scheduled mailing start time.
$camp_schedule.stop_time	time	Scheduled mailing stop time.
$error	Object	Signs of a previous pipe error
$error.code	int	Error code, for example HTTP co
$error.message	string	Text content of the error.

Accessing object fields

All keywords that have an Object type can have nested fields, which can also be objects or arrays of obje

- **An object field is accessed via `'.'` (dot). There are no restrictions on the amount of nesting.**

```
$template.field_name.inner_field_name
```

- **An array element is accessed via `'.%<element index>'`. There are no restrictions on the amount of nestir**

```
$template.field_attach.%0.name
```

Functions

Function	Description
`$base64`	Returns base64 from the value.
`$gen_uniq_int_id`	Returns a unique 10-digit number.
`$escapeCharacters`	Removes the symbols.

Function	Description
`$getWebhookURL`	**Provides a webhook.**
`$removeCodeFromPhone`	**Removes the area code from the phone number.**
`$#toObject`	**Turns JSON from a string into an object.**

Examples

1. General example

At the entrance:

```
{
     "attribute as string": "$sender.id",
     "attribute as original type": "$#sender.id",
     "attribute as part of string": "test $sender.name"
     "$$template.field_files_template[array]": {
         "content": "[$]content",
         "filename": "[$]name",
         "type": "[$]mime",
         "size": "[$]size",
         "constant": 1,
         "sender_id": "$sender.id"
     }
}
```

At the exit:

```
{
     "attribute as string": "145",
     "attribute as original type": 145,
     "attribute as part of string": "test interstellar sender"
     "array": [
         {
             "content": "base64string",
             "filename": "file1.txt",
             "type": "text/plain; charset=utf-8",
             "size": "10",
             "constant": 1,
             "sender_id": "145"
         },
         {
             "content": "base64string",
             "filename": "file2.txt",
             "type": "text/plain; charset=utf-8",
             "size": "100",
             "constant": 1,
             "sender_id": "145"
         }
     ]
}
```

2. Let's assume that the `params.url` field in the HTTP Request pipe is configured as follows:

```
https://example.com/handler/v${sender.version}/test
```

then after processing the template the `params.url` field will be rendered as:

```
https://example.com/handler/v2/test
```

3. Let's assume that in the Pack the `params.template.test_field` field is configured as follows:

```
qwerty$in.a_id
```

then after processing the template the `params.url` field will be rendered as:

```
qwerty5
```

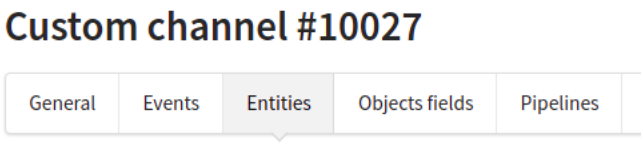
but the following option will return an error because you can't concatenate a string and an int:

```
qwerty$#in.a_id
```

Entities

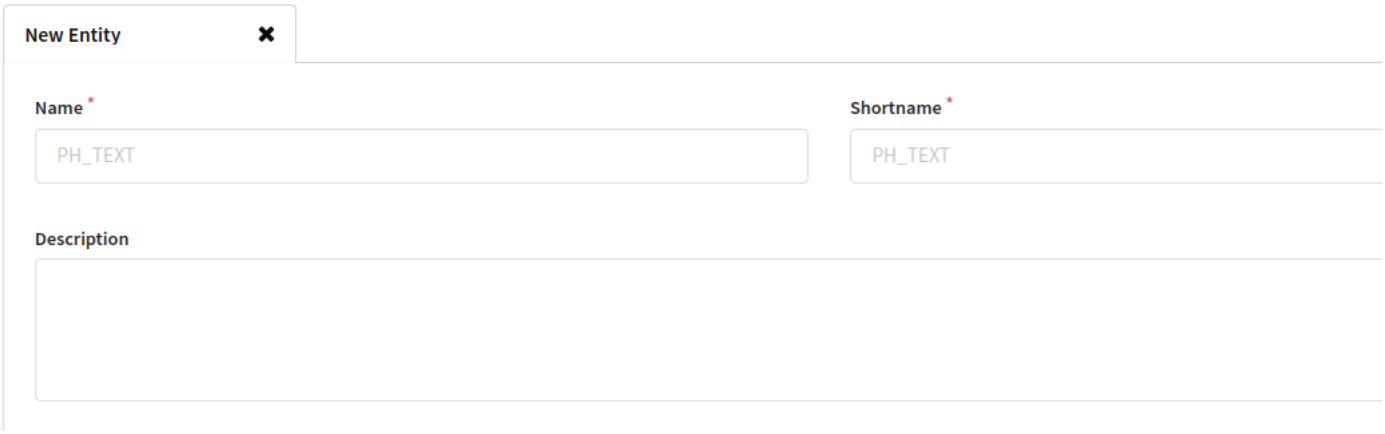
Description

Entity is an external entity with which a custom channel can interact via pipelines. It makes sense to use
External entities are configured on the Entities tab in the custom channel menu.



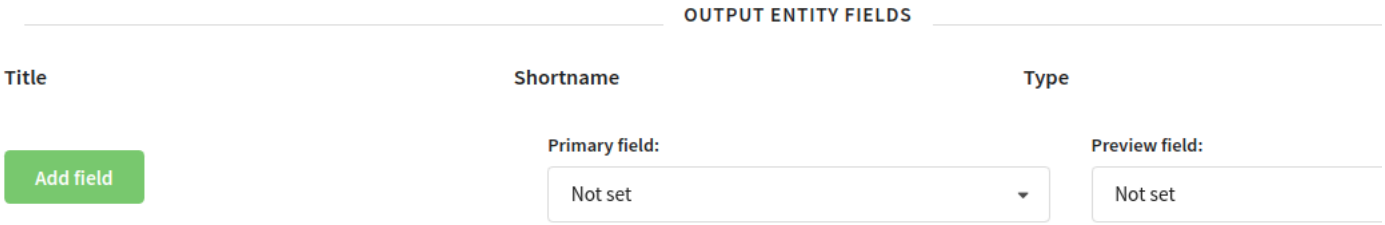
Basic settings

- **Name is the name of the entity (will be displayed in the field to which the entity will be bound);**
- **Shortname is a unique string identifier of the entity. According to it, the entity fields will be available**
- **Description - brief description.**



Structure

The structure of the external entity is specified in the Output Entity Fields section. This is a list of fields c



Operations

Operations are methods of interacting with an entity (create, delete, get a list, get one object, etc.).
Each operation consists of:

1. **Input data – is a set of fields that the user can fill in before performing an operation. The values from t places in the channel.**
2. **Pipeline – JSON array of pipes that will be executed when the user launches an operation in the interf will wait for the result. Based on this fact, you need to design the pipeline accordingly. Therefore, onl**
   - **Result – returns the final result of the operation. Must be present in the pipeline.**
   - **Error**
   - **Log**
   - **Pack**
   - **Unpack**
   - **HTTP Request**
   - **Store Set**
   - **Store Get**
   - **Selector**
   - **SQL**

CAUTION

An event must get to the result pipe before exiting the pipeline, if it does not exit through Serror pipe

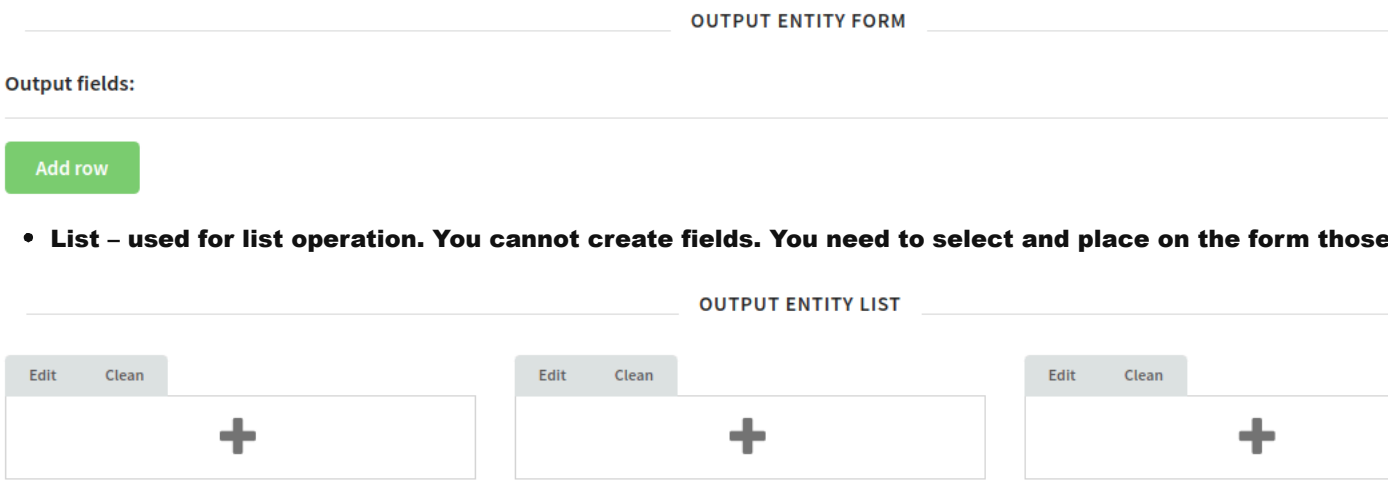
Currently, two types of operations are available for configuration:

- **list – should return an array of entities.**
- **read – must return one entity.**

Output forms

Output forms are a way that how a user will see the result of a operations. Will a table with entities be re
There are currently two forms available:

- **Single form - used for *read* operation. You cannot create fields. You need to select and place on the fo**



- **List – used for list operation. You cannot create fields. You need to select and place on the form those**

Pipe: Basic Concepts

Description

Pipe is an element that performs one action. For example, the log pipe writes a log, and the http_request p
Each pipe in the pipeline is an array of workers, the number of which is configured in the configuration fil

General structure

| Field | Example | Required | Description | | ------ | ---------- | -------- | ------------------------ | ------ | — | — | ----------

Field	Example	
success	**3**	**no**
error	**4**	**no**

Template language

Fields with this label support templating

. The templating language is described [in this article](#).

Result

CAUTION

Can only be used in *Entities* pipelines (see Entities).

Description

The Result pipe sets the result of synchronous pipelines to Entities (see Entities). If `params.path` is not spe
Peculiarities:

- **An empty result is not an error;**
- **The Result pipe does not have to be the last one in the chain. After recording the result, the pipeline c**
- **A pipeline can have several Result pipes, but not a single such pipe should have parent pipes with Res**
- **Must return an object or an array of objects**

Structure

Field	Type Value
id	**number**
type	**string**
params	**ResultParamsObject**
outs	**OutsObject**

Field	Type	Example
path	**string** ★	**"$#in.data.categories"**

Errors

1. **Error processing params.path – params.path was specified, but the format is incorrect or the field wa**
2. **Error processing $in.data - params.path is not set and there is no *data* key in the $in object.**

Examples

1. **Correct example. $#in.data.categories is supposed to contain an array of objects:**

```
{
  "id": 10,
  "type": "result",
  "params": {
    "path": "$#in.data.categories"
  }
}
```

2. **An example of an error. $#in.data.categories is supposed to contain an array of objects:**

```
{
  "id": 10,
  "type": "result",
  "params": {
    "path": "$in.data.categories"
  }
}
```

The result here is returned as a string, which is incorrect in this case.

Errors

> CAUTION
>
> **Can only be used in *Entities* pipelines (see Entities).**

Description

The *Error* pipe interrupts the execution of an event in the pipeline with an error. Allows you to set the erro
Peculiarities:

- **Has no exits;**
- `params.message` **must be a text.**

Structure

|Field|Value|Type|Example|Required|Description| |id|number|10|yes|Pipe's ID| |type|string|"error"|yes|Pipe's t

Fi	Type	Example
code	**number**	**400**
message	**srtring** ★	**"$in.data.response.error_message"**

Errors

Error processing `params.message` **is your template is incorrect or the field was not found.**

Examples

Example 1:

```
{
  "id": 10,
  "type": "error",
  "params": {
    "code": 500
  }
}
```

Example 2:

```
{
  "id": 10,
  "type": "error",
  "params": {
    "code": 500,
    "message": "$in.data.error"
  }
}
```

Log

Description

The log pipe writes a message to `procpiper.log`. Allows you to set the text of the message.
Peculiarities:

- **`params.message` must be a text**

Structure

Field	Type of value
id	**number**
type	**string**
params	**LogParamsObject**
outs	**OutsObject**

Fi	Type	Example
prefix	**string**	**"Send|"**
message	**string** ★	`"error http request $error.code $error.message"`
level	**string**	**"trace"**

Errors

1. **Error processing params.message - the template is incorrect or the field was not found.**

Examples

```
1.
{
     "id": 10,
     "type": "log",
     "params": {
         "message": "error http request $error.code $error.message"
     },
     "outs": {
         "success": 20
     }
}
{
     "id": 10,
     "type": "log",
     "params": {
         "message": "some message"
         "level": "trace"
     },
     "outs": {
         "success": 15
     }
}
```

Pack

Description

The Pack pipe marshals JSON or XML and writes the result to `$in.body` as a byte array. It only makes sens

Structure

Field	Type of value
id	**number**
type	**string**
params	**PackParamsObject**
outs	**OutsObject**

PackParamsObjectt

Field	Type	Example	Reqiered	Default
type	**string**	**"JSON"**	**no**	**"JSON"**
template	**object**	**{}**	**yes**	**-**

Errors

1. **params.type is not valid.**
2. **Error processing **params.template*** **- the template is not correct or the field was not found.**

Examples

1. **Example of a JSON packer.**

Pipe:

```
{
    "id": 6,
    "type": "pack",
    "params": {
        "type": "JSON",
        "template": {
            "acceptUrl": "$template.acceptUrl",
            "backgroundColor": "$template.background_color",
            "backgroundImage": "$template.background_Image",
            "campaignEnd": "$camp_schedule.stop_time",
            "feeType": "fix",
            "campaignFee": 0,
            "campaignStart": "$camp_schedule.start_time",
            "campaignType": "$campaign.campaignType",
            "categoryId": "$#campaign.category.category",
            "conditions": "$template.conditions",
            "description": "$template.description",
            "discountAmount": "$#template.discountAmount",
            "discountCurrency": "$#template.discountCurrency",
            "discountType": "$template.discountType",
            "fontColor": "$template.font_color",
            "merchantDomain": "$account.merchantDomain",
            "merchantLogoPng": "$account.merchantLogoPng",
            "merchantLogoSvg": "$account.merchantLogoSvg",
            "merchantName": "$account.merchantName",
            "offerEnd": "$camp_schedule.stop_time",
            "partnerCampaignId": "$#origin.c_id",
            "blockIds": "$#campaign.blockIds",
            "$$campaign.segment[segments]": ["[$]id"]
            "test_obj": {
                "field1": 1,
                "field2": ""
            }
        }
    },
    "outs": {
        "success": 7,
        "error": 8
    }
}
```

Result:

```
{
    "acceptUrl": "http://exmaple.com/accept_url",
    "backgroundColor": "#ffffff",
    "backgroundImage": "http://exmaple.com/background_image",
    "campaignEnd": "2021-08-09T18:31:42",
    "feeType": "fix",
    "campaignFee": 0,
    "campaignStart": "2021-05-09T15:22:00",
    "campaignType": "test",
    "categoryId": 50,
    "conditions": "test_condition",
    "description": "",
    "discountAmount": 155,
    "discountCurrency": 482,
    "discountType": "RUB",
    "fontColor": "#ffffff",
    "merchantDomain": "http://exmaple.com/merchantDomain",
    "merchantLogoPng": "http://exmaple.com/merchantLogoPng",
    "merchantLogoSvg": "http://exmaple.com/merchantLogoSvg",
    "merchantName": "merchantName",
    "offerEnd": "2021-08-09T18:31:42",
    "partnerCampaignId": 101,
    "blockIds": ["adfadf", "dadasdasdas", "aaaaaa"],
    "segments": ["aaaa", "bbbb"]
    "test_obj": {
        "field1": 1,
        "field2": ""
    }
}
```


2. Example of an XML packer.

Pipe:

```
{
  "id": 106,
  "type": "pack",
  "params": {
    "type": "XML",
    "template": {
      "event": {
        "-cancel": "false",
        "-type": "MESSAGE",
        "-xmlns": "http://example.com/event.xsd",
        "recipients": {
          "recipient": {
            "addresses": {
              "address": [
                {
                  "-channel": "APL",
                  "-type": "CLIENT_ID",
                  "-value": "$subscription.id"
                },
                {
                  "-channel": "PUSH",
                  "-type": "CLIENT_ID",
                  "-value": "$subscription.id"
                }
              ]
            }
          }
        },
        "templates": {
          "template": [
            {
              "#text": "$template.title",
              "-channel": "APL",
              "-type": "TITLE"
            },
            {
              "#text": "$template.body",
              "-channel": "APL",
              "-type": "BODY"
            },
            {
              "#text": "$template.push_body",
              "-channel": "PUSH",
              "-type": "BODY"
            }
          ]
        }
      }
    }
  },
  "outs": {
    "success": 108,
    "error": 109
  }
}
```

Result:

```
<?xml version="1.0" encoding="UTF-8"?>
<event xmlns="http://example.com/event.xsd" cancel="false" type="MESSAGE">
    <recipients>
        <recipient>
            <addresses>
            </addresses>
        </recipient>
    </recipients>
    <templates>
        <template channel="APL" type="TITLE">some title</template>
        <template channel="APL" type="BODY">some body</template>
        <template channel="PUSH" type="BODY">some push body</template>
    </templates>
</event>
```

Unpack

Description

The Unpack pipe unmarshals JSON or XML from the $in.body field and writes the result to `$in.data`
Peculiarities:
`$in.body` must be a byte array

Structure

Field	Type of value
id	**number**
type	**string**
params	**UnpackParamsObject**
outs	**OutsObject**

UnpackParamsObject

Field	Type	Example	Reqi
type	**string**	**"JSON"**	**no**

Errors

1. **$in.body is not found.**
2. **$in.body is not a byte array.**
3. **params.type is not valid**
4. **Byte unmarshalling error.**

Examples

1.

```
{
  "id": 10,
  "type": "unpack",
  "params": {
    "type": "XML"
  },
  "outs": {
    "success": 20,
    "error": 1000
  }
}
```

2.

```
{
  "id": 10,
  "type": "unpack",
  "outs": {
    "success": 15,
    "error": 1001
  }
}
```

HTTP Request

Description

Http request pipe performs an http request.
Peculiarities:
Depending on *prams.content_type*:

- **form, form-data – data is expected in params.form;**
- **xml, json – data is expected in $in.body (must be a byte array).**

Structure

Field	Type of value
id	**number**
type	**string**
params	**HttpParamsObject**
outs	**OutsObject**

Field	Type	Example	Required	
content_type	**string**	**"json"**	**no**	**"form"**
method	**stringn**	**"POST"**	**no**	**"GET"**
auth_type	**string**	**"basic"**	**no**	**"basic"**
auth_username	**stringa**	**"root"**	**no**	**""**
auth_password	**string**	**"1234"**	**no**	**""**
url	**string** ★	**"http://exmaple.com/send"**	**no**	**""**
timeout	**тгьиук**	**20**	**тщ**	**PIPE_HTTP_TIMEOUT**
success_codes	**array of numbers**	`[200, 201]`	**no**	`[200, 201, 202, 203, 204,`
pool_size	**number**	**10**	**no**	**PIPE_HTTP_MAX_CON**
limit	**LimitObject**	`{ "period_sec": 60, "count_requests": 30, "key": "some key" }`	**no**	**-**
headers	**object** ★	`{ "someHeader": "value" }`	**no**	**-**
form	**object** ★	`{ "test": "123" }`	**no**	**-**

Field	Type	Example	Required	Default	
period_sec	**number**	**1**	**yes**	**-**	**Rang**
count_requests	**number**	**2**	**yes**	**-**	**The** The </td>
key	**string** ★	**"$resource.token"**	**no**	**random uuid**	**Cons** Need ht </st </td>

Errors

1. **Errors in field template processing.**
2. **$in.body not found (missing pack pipe).**
3. **$in.body not a byte array.**
4. **params.content_type not valid.**
5. **params.method not valid.**

6. http request error.

Examples

1. Example request with form.

```
{
  "id": 3,
  "type": "http_request",
  "params": {
    "method": "POST",
    "timeout": 10,
    "content_type": "FORM",
    "form": {
      "login": "$sender.login",
      "password": "$sender.password",
      "externalUserCode": "$sender.login"
    },
    "url": "https://example.com/api/v1.0/sessions",
    "success_codes": [200, 202]
  },
  "outs": {
    "success": 6,
    "error": 7
  }
}
```

2. Example request with JSON. It is assumed that in the chain before the pipe there is a pack pipe that g

```
{
  "id": 2,
  "type": "http_request",
  "params": {
    "method": "POST",
    "timeout": 10,
    "content_type": "JSON",
    "headers": {
      "Authorization": "$in.ow_tp $in.ow_token"
    },
    "url": "https://example.com/api/v1/api/campaign/start",
    "success_codes": [200, 202]
  },
  "outs": {
    "success": 9,
    "error": 10
  }
}
```

Store Set

Description

Store Set a pipe writes the value under the given key to mongodb.

Structure

Field	Type of value
id	**number**
type	**string**
params	**StoreSetParamsObject**
outs	**OutsObject**

Field	Type	Example	Reqiered	Default	
key	**string** ★	**"key_name"**	**yes**	**-**	**Key name**
value	**any type** ★	**"qwerty"**	**yes**	—	**Data that will be saved.**
table	**string** ★	**"some"**	**no**	**""**	**The name of the virtual collection into which the key-value pair will be writt**
ttl	**number** ★	**50**	**yes**	**-**	**After what time the data will be deleted.**

Errors

1. **Error processing params.key, params.value, params.table or params.ttl - the template is invalid or the**
2. **Database query error.**

Example

```
{
  "id": 37,
  "type": "store_set",
  "params": {
    "key": "session_id",
    "value": "$in.data.SessionInfo.%0.SessionToken",
    "ttl": 900
  },
  "outs": {
    "success": 12,
    "error": 13
  }
1. }
```

Store Get

Description

Store Get the pipe takes data from mongodb for a given key and writes it to `$in.<name/>`.

Stucture

Field	Type of value
id	**number**
type	**string**
params	**StoreGetParamsObject**
outs	**OutsObject**

Field	Type	Example	Reqiered	Default	
key	**string** ★	**"key_name"**	**yes**	**-**	**Key name**
name	**string** ★	**"name"**	**no**	**<key>**	**Name of the field where value will be written `($in.<name>)`**
delete	**bool**	**true**	**no**	**false**	**Flag for deleting a record immediately after it is read.**
table	**string** ★	**"some"**	**no**	**""**	**The name of the virtual collection into which the key-value pair will be**
not_found	**number**	**12**	**no**	**-**	**ID of the pipe to which the event will go if a record for the given key is**

Errors

1. **Error processing params.key and params.table - the template is incorrect or the field was not found.**
2. **Error when querying the database, except Not found, if params.not_found is specified**

Example

```
{
  "id": 2,
  "type": "store_get",
  "params": {
    "key": "session_id",
    "not_found": 3
  },
  "outs": {
    "success": 28,
    "error": 14
  }
}
```

Selector

Description

Selector the pipe redirects events to different pipes based on the condition.

Peculiarities:

- **If none of the conditions from `params.outs` are met, the event will go to `outs.success` (if one is specified)**

Structure

Field	Type of value
id	**number**
type	**string**
params	**SelectorParamsObject**
outs	**OutsObject**

Field	Type
outs	**array (SelectorOutsObject)**

Field	Type	Example	
query	**string** ★	**"($template.show_push = true)"**	**yes**
out	**number**	**15**	**yes**

Errors

1. **Error processing params.outs.query.**
2. **Error executing JsonQL query.**

Examples

1.
```
{
  "id": 15,
  "type": "selector",
  "params": {
    "outs": [
      {
        "query": "('$template.show_push' = 'true')",
        "out": 102
      },
      {
        "query": "('$template.show_push' = 'false')",
        "out": 103
      }
    ]
  },
  "outs": {
    "success": 104,
    "error": 105
  }
}
```

2.
```
{
  "id": 15,
  "type": "selector",
  "params": {
    "outs": [
      {
        "query": "('$in.data.status' = 'success')",
        "out": 26
      },
      {
        "query": "('$in.data.status' = 'error')",
        "out": 27
      }
    ]
  }
  "outs": {
    "success": 28,
    "error": 29
  }
}
```

SQL

Description

The SQL pipe executes an SQL query to the database and puts the result as an object or an array of objec
Peculiarities:

- **If none of the conditions from *params.outs are met, the event will go to outs.success (if one is speci**

Structure

Field	Type of value
id	**number**
type	**string**
params	**SQLParamsObject**
outs	**OutsObject**

Fi	Type	Example	Reqiered	
connector_id	**number**	**5**	**yes**	**-**
map	**bool**	**true**	**no**	**false**
query	**string** ★	`"SELECT * FROM Test_Table"`	**yes**	**-**
connection_timeout	**number**	**15**	**no**	**PIPE_SQL**
query_timeout	**numbero**	**10**	**no**	**PIPE_SQL**

Errors

1. **Error processing template params.query.**
2. **Error while executing SQL query.**

Examples

```
{
  "id": 2,
  "type": "sql_pipe",
  "params": {
    "connection_timeout": 10,
    "query_timeout": 10,
    "query": "SELECT CAST(SEGMENT_LIST_ID AS VARCHAR) AS 'id', SEGMENT_LIST_NAME AS 'name' FROM SEGMENT_LIST",
    "connector_id": 2
  },
  "outs": {
    "success": 3,
    "error": 5
  }
}
```

Eventer

Description

Eventer The pipe registers a custom channel event. This pipe is also used to implement the moderation m

Structure

Field	Type of value
id	**number**
type	**string**
params	**EventParamsObject**
outs	**OutsObject**

Field	Type	Example	
event_type	**string**	**send**	**no**
channel_id	**int**	**10001**	**yes, beside even** • **stop** • **moderate_** • **moderate_**
action_type	**int**	**100005**	**yes, beside even** • **stop** • **moderate_** • **moderate_**
event	**EventObject**		**yes**

Field	Type	Example	Reqiered	Default	
send_message_id	**string** ★	**$origin.send_message_id**	**yes**	**-**	**ID of the message in the platform.** CAUTION: If we are talking about the <a href="/admin-guide/communicati **<code>$origin.<name of the field with the identifier from th pipe></code>**

Errors

1. **Error processing template** `params.event.send_message_id`.
2. `params.event.send_message_id format is incorrect.`

Example

```
{
  "id": 100,
  "type": "event",
  "params": {
    "event_type": "send",
    "channel_id": 10001,
    "action_type": 100021,
    "event": {
      "send_message_id": "$origin.send_message_id"
    }
  },
  "outs": {
    "error": 101
  }
}
```

```
{
  "id": 150,
  "type": "event",
  "params": {
    "event_type": "stop",
    "event": {
      "send_message_id": "$origin.send_message_id"
    }
  }
}
```

```
    {
      "id": 18,
      "type": "event",
      "params": {
        "event_type": "moderate_error",
        "event": {
          "send_message_id": "$origin.send_message_id"
        }
      },
      "outs": {
        "success": 19,
        "error": 20
      }
3. }
```

Scheduler

Description

Scheduler The pipe transmits the event for execution to the Schedulepipeline.

> **CAUTION**
>
> **The event passed to the Schedule pipeline changes the data in `$origin` to `params.template`**

Structure

Field	Type of value
id	**int**
type	**string**
params	**ScheduleParamsObject**
outs	**OutsObject**

Field	Type	Example	Reqiered	
pipe_id	**int**	**5**	**yes**	**-**
delay	**int**	**10**	**no**	**PIP**
template	**object** ★	`{ "test": "$origin.send_message_id" }`	**yes**	**-**

Errors

1. **Error processing template params.template.**

Examples

```
{
  "id": 801,
  "type": "scheduled",
  "params": {
    "pipe_id": 1,
    "template": {
      "camp_id": "$origin.c_id",
      "send_message_id": "$origin.send_message_id",
      "ym_cmp_id": "$#in.data.ymCampaignId"
    },
    "delay": 10
  },
  "outs": {
    "success": 802,
    "error": 803
  }
}
```

Pipeline: Basic Concepts

Description

Pipelines are designed to handle submissions across any channel. Can be configured in JSON.

- **There is a set of elements that process data (pipes).**
- **Each element has an input and outputs. Their format has been determined.**
- **If the output type of one element is equal to the input of another, they can be connected in a pipeline.**
- **Inside the pipe, you can access the data and results of the pipe's action through keywords. The synta**
- **In each element, data entering the pipe is accessible through the `$in` keyword, and data entering the**

A list of all available pipes with a description of their configuration can be found here.

Pipeline example:

```
[
  {
    "id": 1,
    "type": "packer",
    "params": {
      "type": "JSON"
    },
    "outs": {
      "success": 2
    }
  },
  {
    "id": 2,
    "type": "http_requester",
    "params": {
      "timeout": 10,
      "content_type": "JSON",
      "success_codes": [200, 202]
    },
    "outs": {
      "success": 3,
      "error": 4
    }
  },
  {
    "id": 3,
    "type": "event",
    "params": {
      "name": "mydeliv",
      "event": {
        "send_message_id": "$origin.send_message_id"
      }
    }
  },
  {
    "id": 4,
    "type": "event",
    "params": {
      "name": "myundeliv",
      "event": {
        "send_message_id": "$origin.send_message_id",
        "message": "$error.code $error.message"
      }
    }
  }
]
```

Message

Description All messages sent by the mailing end up in the Message pipeline.

Schedule

Description

The *Schedule* pipeline is used to process events transmitted through the Scheduler pipe. The mentioned p
Peculiarities:
scans events every `SCHEDULER_PIPE_ROUTER_PERIOD` **seconds (default - 10).**

Listener

Description

The *Listener* pipeline is used to process events received by the `trklistener` tracking service.
Peculiarities:

- **You can send an event to any pipeline in the pipeline. The ID of the pipe into which the event will fall a**
- **In addition to pipe settings, you can set a string with which trklistener will respond to requests. This**

Tracking service

The configuration of the tracking service can be obtained from the platform administrators. By default it r
The tracking service is configured with the parameters `LISTENER_UNIXSOCK_PATH`, `LISTENER_IP` and `LISTENER_PORT`

> **NOTE: e**
> **Default settings: |Parameter|Meaning| |ip|127.0.0.1| |port|8069|**

Data transfer

Data must be submitted via the following link:
`http://<listener_host>/fbp/v1/<custom_channel_sid>/<pipe_id>`
All GET parameters will be written to `$in.form` of the event, which will go to the *listener* pipeline.
The body of the request will be written to `$in.body` as a byte array. To then convert the bytes into an objec

Moderate

Description

The *Moderate* pipeline is used to manage mailing moderation. It becomes available for editing if the *With Moderate*.

Stop

Description

The *Stop* pipeline is used only in placement mailings. If it is set, the mailing will have settings for ending i

System page customization

Custom template of the unsubscribe page

Tracking page templates are located in the directory `ak/static/trk/`. If you need to reassign them, you nee

```
$ cd /opt/akd
$ cp ak/static/trk/unsubscribe.html ak/static/trk/custom.unsubscribe.html
$ nano ak/static/trk/custom.unsubscribe.html
...
$ ./akd onerestart procactions
```

The templates will be applied only when you restart the process `procactions`.

Description of the pages

Template	
`unsubscribe.html`	**Page after unsubscribing, if it isn't specified in the message**
`deprecated.html`	**The page of the expired web message**
`not_found.html`	**Page for invalid links on tracking**

Custom notification templates

Custom notification templates are located in the directory `templates/email/<CountryCode>/`. As they will be ove

You need to add `custom.` to the file name in order to define such a file:

```
$ cd /opt/akd
$ cp templates/email/RU/password.html templates/email/RU/custom.password.html
$ nano templates/email/RU/custom.password.html
...
$ ./akd restart --job_server
```

The templates will be applied only when you restart the process `akd`.

Template description

Template	Level	
`content.json`	**all**	**Applies to all template**
`template_all.html`	**all**	**Applies to all template**
`welcome.html`	**account, admin**	**Appears when the user**
`password.html`	**account, admin**	**Password change by us**
`complains.html`	**account**	**In case of exceeding th**
`hardbounces.html`	**account**	**In case of exceeding th**
`softbounces.html`	**account**	**In case of exceeding th**
`report_weekly.html`	**account**	**Weekly report of the us**
`report_daily.html`	**account**	**Daily report of the user**
`sendlimit.html`	**account**	**In case of exceeding th**
`leadslimit.html`	**account**	**In case ofexceeding th**
`suspended.html`	**account**	**In case the account is**
`sender_problem.html`	**admin**	**During the process of s**

Customization of form pages

All template files are located in the platform installation directory under `/templates/form/v1/EN` and `/template`

File	
button.tpl	**Template for displaying buttons. Controls their appearance and placement in the form.**
content.json	**File containing the form's data and structure — includes field descriptions, transition logic, texts, settings, etc.**
user_code.tpl	**A custom template designed to override the default form template. It is used to insert custom code into the `<head>`, `<body>`, and `<footer>` section**
error.html	**Page displayed when an error occurs while loading the form or if the link is invalid.**
final.tpl	**Template for the final page shown to the user after successful form submission.**
form_fill.tpl	**Template for the main form page where the user fills in the fields.**
form.tpl	**Main container template for the form. Includes basic HTML layout, structure, and links to other templates.**
hidden.html	**Displayed when the form is hidden from public access. Usually contains a message like "Form unavailable".**
redirect.tpl	**Template for the redirection page.**

Similar to other config files, you can create a copy of any of them, rename it to `"custom.<original_name>"`, and

Static tracking files

In some cases it can be useful to place static files on the tracking — the necessary files can be placed in

Send Message ID

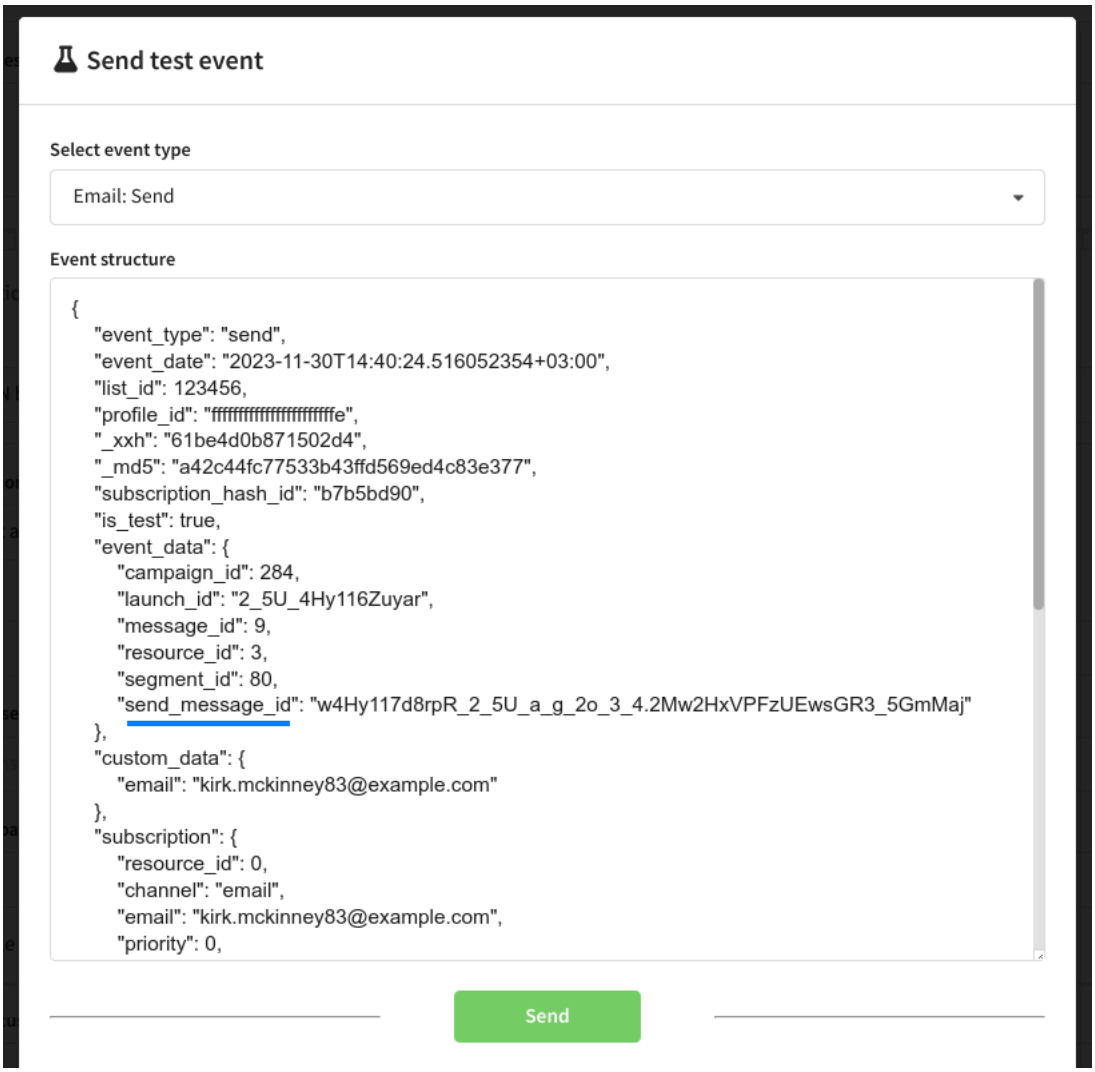
Send Message ID is a unique sending identifier used in the platform. It stores the identifiers necessary to

Identifier types

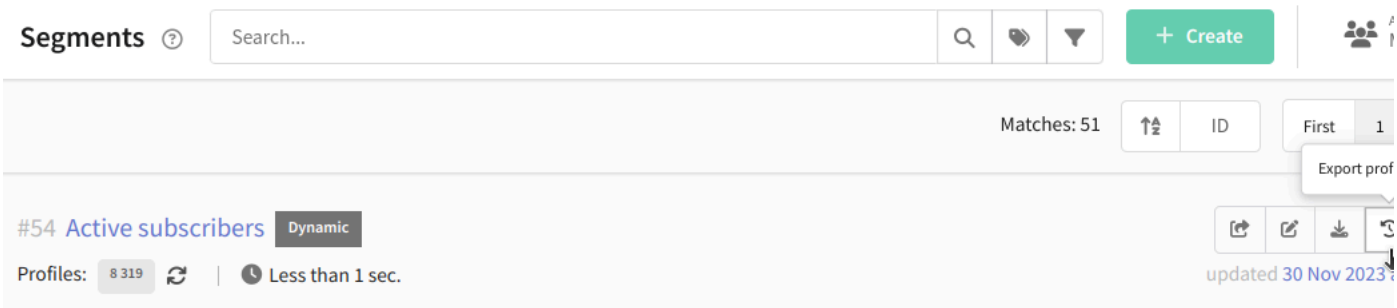
Type	Prefix	
USMID	`u`	**Unique SendMessageID. It is passed in Action Hooks and is used to save the web version.**
ESMID	`m`	**Email-safe SendMessageID. It is added to the `Message-ID` and `Return-Path` headers in email messages.**
FSMID	`f`	**Full SendMessageID. Added to the `X-AKMTA-FSMID` header in email messages. Contains maximum information about the s**
SMID	`w`	**The identifier is used in version 2022.3.66 and below.**

Obtaining SMID

You can get `SendMessageID` using Action Hooks — field `send_message_id`:

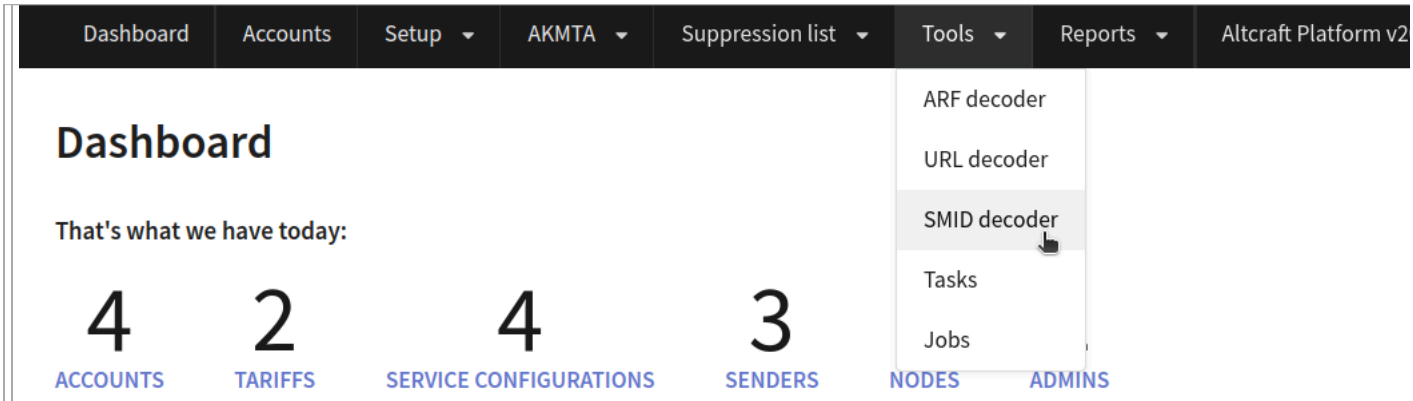


Also `SendMessageID` is available in the profile history. You can export profile history from a database or segm



Decoding SMID

In the main menu select Tools —> Message ID decoder:



Paste the message ID (SMID, FSMID, ESMID или USMID) and click Decode. Depending on the identifier typ

- **date and time of sending,**
- **mailing and template,**
- **database and segment,**
- **profile data,**
- **resource subscriptions and other data.**

Object labels are clickable and will open the corresponding screens of Administrative panel or User interf

SMID Decoder

Paste SMID, FSMID, ESMID or USMID here:

fDbrZmUOCeN1LUPcHsiCJwXAAAQEh4CEfEPEBAAAkUBEBEpIA8-oUiCJwXYN836j9i_HE3BfDjZDJmMWM0ODlbQ8doA

Decode

Send Date	2023-11-29T10:41:43.608884352+03:00	
Launch Date	2023-11-29T10:41:43.432056757+03:00	
Test Flag	false	
Indirect Flag	false	
Account	1 - account	
Campaign	542 - New collection	
Template	325 - New collection (December)	
Series ID	No information	
Sender	1 - AKMTA	
Workflow	No information	
Workflow Node	No information	
Database	15 - B2C customers	
Segment	41 - Active subscribers	
Profile	Email	john.doe@example.com
	Identifier	60df37ea3f62fc713705f0e3
	ip	192.168.0.1
	Registration date	2019-03-14T21:00:00Z
Altcraft Client ID	No information	
External ID	No information	
Segment	41 - Active subscribers	
Subscription	Channel	email
	Contact	john.doe@example.com
	Subscription ID	d2f1c489
	Resource	Website
	Subscription status	subscribed
Web version	Saved	

INFO

If the message web version is available, you can open the message preview by clicking Saved.

You can also use the aktool utility for decoding. Run the command `./aktool smid "<Any SMID>"` in the terminal More detailed information about the utility and command reference is available in this article.

Instructions for migrating history to ClickHouse

Step 1. Install ClickHouse DBMS following the official instructions.

> **NOTE**
>
> **The minimum required version for the platform is `>21.4.6`.**

Step 2. Create a new user `/etc/clickhouse-server/users.d/altcraft.xml` (see the documentation for details — Us

```
<yandex>
    <users>
        <altcraft>
            <password>abcdefghiasdwqedwqjklmnopqrstuvwxyzABCDEF</password>
            <networks>
                <ip>::/0</ip>
            </networks>
            <profile>default</profile>
            <quota>default</quota>
            <allow_databases>
                <database>altcraft_system</database>
            </allow_databases>
        </altcraft>
    </users>
</yandex>
```

> **CAUTION**
>
> **The character set “abcdefghiasdwqedwqjklmnopqrstuvwxyzABCDEF” is used as the password.**
> **Be sure to generate a more complex password consisting of at least 32 characters.**

Step 3. Enable the `input_format_import_nested_json` option to insert JSON data with nested objects (see the do

```
<?xml version="1.0"?>
<yandex>
    <!-- Profiles of settings. -->
    <profiles>
        <!-- Default settings. -->
        <default>
            <!-- Maximum memory usage for processing single query, in bytes. -->
            <max_memory_usage>8589934592</max_memory_usage>
            <max_memory_usage_for_user>8589934592</max_memory_usage_for_user>
            <max_bytes_before_external_group_by>4294967296</max_bytes_before_external_group_by>
            <max_bytes_before_external_sort>4294967296</max_bytes_before_external_sort>
            <receive_timeout>3600</receive_timeout>
            <send_timeout>3600</send_timeout>
            <max_threads>2</max_threads>
            <max_partitions_per_insert_block>10000</max_partitions_per_insert_block>
            <!-- Use cache of uncompressed blocks of data. Meaningful only for processing many of very short queries. -->
            <use_uncompressed_cache>0</use_uncompressed_cache>

            <!-- How to choose between replicas during distributed query processing.
                 random - choose random replica from set of replicas with minimum number of errors
                 nearest_hostname - from set of replicas with minimum number of errors, choose replica
                  with minimum number of different symbols between replica's hostname and local hostname
                  (Hamming distance).
                 in_order - first live replica is chosen in specified order.
                 first_or_random - if first replica one has higher number of errors, pick a random one from replicas with minimum number of err
            -->
            <load_balancing>random</load_balancing>
            <input_format_import_nested_json>1</input_format_import_nested_json>
        </default>
```

Step 4. In the main configuration file `main.json`, configure the settings for connecting to and working with

```
"CLICKHOUSE_SYSTEM": {
    "HOST": "localhost",
    "PORT": 9000,
    "USER": "altcraft",
    "MAX_CONNECTION": 100,
    "CONN_TIMEOUT": 15,
    "PASSWORD": "abcdefghiasdwqedwqjklmnopqrstuvwxyzABCDEF",
    "IS_DEBUG": false
},
```

Step 5. Run migration. For these purposes, a special utility has been prepared — `actions_migrate`. Full list o

Parameter	Default value
--accounts string	**All active accounts**
--appname bool	**false**
--batch_timeout int	**5**
--ch_batch int	**100000**
--ch_debug bool	**false**
--ch_host string	**The main configuration `main.json` is used**
--ch_password string	**The main configuration `main.json` is used**
--ch_port int	**The main configuration `main.json` is used**
--ch_user string	**The main configuration `main.json` is used**
--config string	**/opt/MarketingPlatform/config/main.json**
--exclude string	**“”**
--level_log string	**INFO**
--log_path string	**./actions_migrate.log**
--range string	**“”**
~~**--rmq bool**~~	~~**false**~~
--version bool	**false**
--worker_size int	**-1**

TIP

Migration can take a long time, up to 3 or more days. To continue the migration after disconnecting from

Example of transferring all historical data up to September 2021:

```
/opt/akd/ak/bin/actions_migrate --accounts "1" --config "/opt/akd/config/main.json" --batch_timeout 60 --ch_batch 500000 --range "-09.21"
```

Example of transferring all historical data for November 2021:

```
/opt/akd/ak/bin/actions_migrate --accounts "1" --config "/opt/akd/config/main.json" --batch_timeout 60 --ch_batch 500000 --range "11.21-11.21"
```

Example of transferring all historical data from January 2021 to the current moment:

```
/opt/akd/ak/bin/actions_migrate --accounts "1" --config "/opt/akd/config/main.json" --batch_timeout 60 --ch_batch 500000 --range "01.21-"
```

Overriding API addresses of push services for testing

The configuration is done in the main.json configuration file.
The platform can send push messages to the following services:

- **APNs (Safari)**
- **APNs (iOS)**
- **Firebase**
- **Mozilla Firefox**
- **Appmetrica**
- **HMS**

You can override the API address for each of them using the `ACCOUNT_OVERRIDES` field. For the Firebase servic

Utility for importing push subscriptions to Firebase project

Description

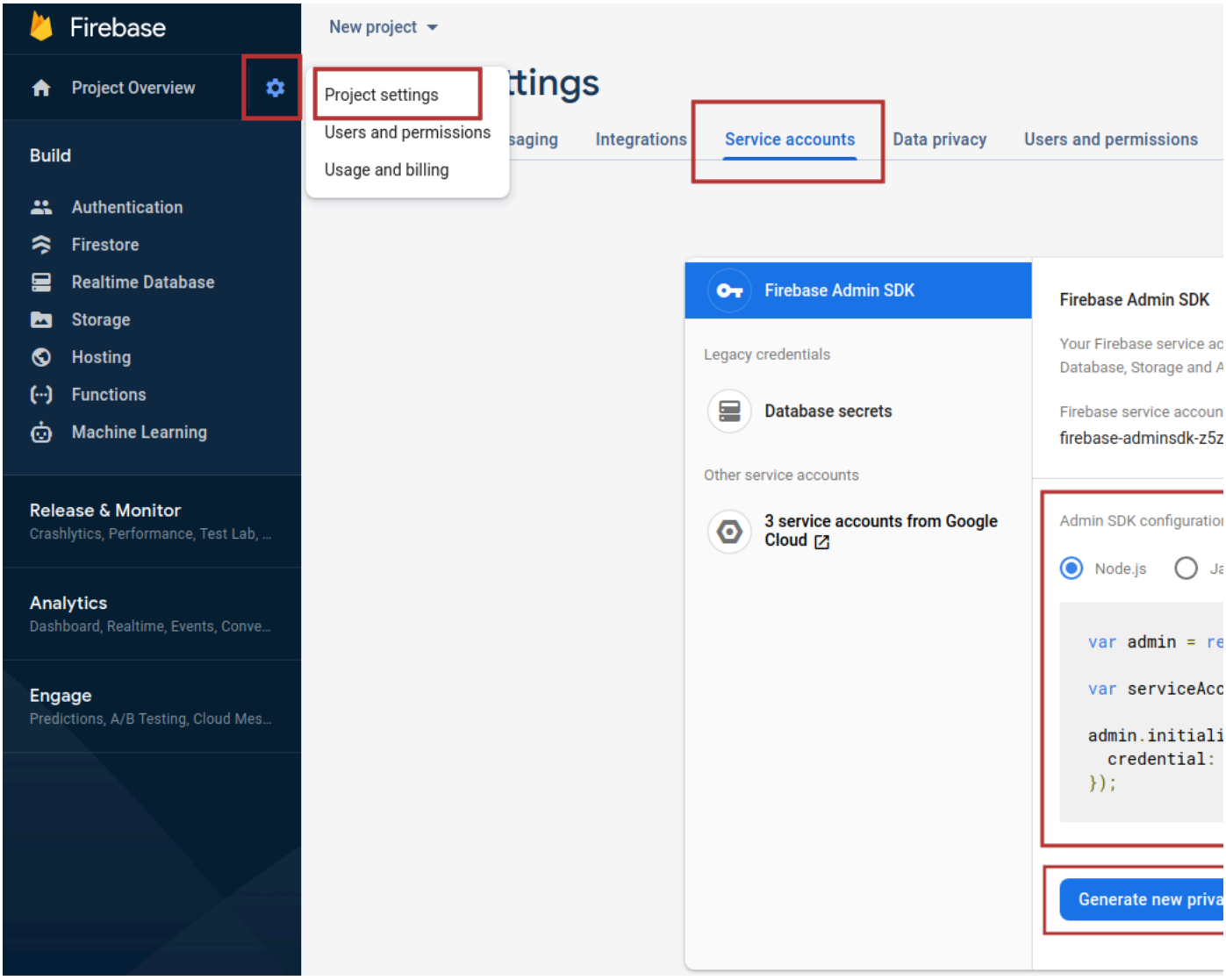
Utility `firebase_import_subs` is a tool for importing Web Push subscriptions of a third-party push service into

How does it work?

1. **A CSV file is sent as the input data. It contains fields for subsequent matching during the import proc**
2. **A new CSV file is sent as the output data. All data from the source file is saved in the file, but also the**

How to do a request?

- **The Altcraft platform user transfers the CSV file with subscriptions to the platform administrator.**
- **You need a service account JSON file. You can get it in the Firebase console: https://firebase.google.c new private key" to download the JSON file.**



- **Go to the directory where the Altcraft platform is installed. By default, it is /opt/MarketingPlatform/**

```
cd /opt/MarketingPlatform/
```

- **Use the utility:**

```
ak/bin/firebase_import_subs --in ~/test_firebase_import/in.csv --out ~/test_firebase_import/out.csv --cred_json ~/test_firebase_import/cred.jsc
```

Parameters

`in`
`out`
`cred_json`
`auth_field_name`
`p256_field_name`
`endpoint_field_name`
`publ_vapid_key`

Additional parameters
—
proxy
config

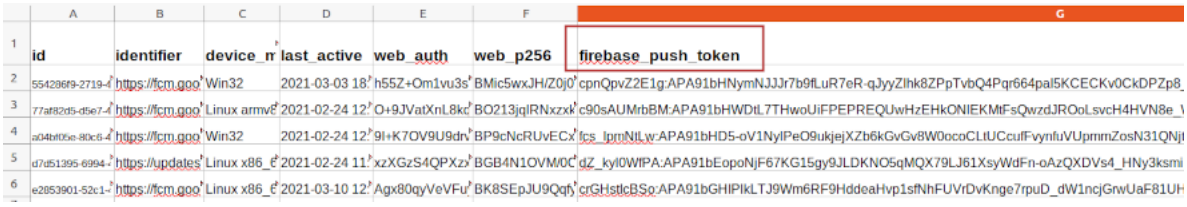
Utility example:

```
root@dev-polina-zakaryan:/opt/MarketingPlatform$ ak/bin/firebase_import_subs --in ~/test_firebase
_import/in.csv --out ~/test_firebase_import/out.csv --cred_json ~/test_firebase_import/cred.json
--auth_field_name web_auth --p256_field_name web_p256 --endpoint_field_name identifier --publ_vap
id_key BDkO8iue5By4UcLUk5SCL0LLtDl9RWvSyqr0WOIjiXQTJmwH0yHxcrqidZ-PyJ9QwksTBHkh9wx88YK4xnn17_Y
 6 / 6 [============================================================] 100.00% 2s
Number of mistakes:  1
```

NOTE

All errors that happened when you start the utility are recorded in the log file.

- **If you've done everything right, the new out.csv contains:**
  1. **the same data that the initial file contains**
  2. **column firebase_push_token.**



	A	B	C	D	E	F	G
1	id	identifier	device_n	last_active	web_auth	web_p256	firebase_push_token
2	554286f9-2719-4	https://fcm.goo	Win32	2021-03-03 18:	h55Z+Om1vu3s	BMic5wxJH/Z0j0	cpnQpvZ2E1g:APA91bHNymNJJJr7b9fLuR7eR-qJyyZIhk8ZPpTvbQ4Pqr664pal5KCECKv0CkDPZp8_
3	77af82d5-d5e7-4	https://fcm.goo	Linux armv8	2021-02-24 12:	O+9JVatXnL8kc	BO213jqlRNxzxk	c90sAUMrbBM:APA91bHWDtL7THwoUiFPEPREQUwHzEHkONIEKMtFsQwzdJROoLsvcH4HVN8e_
4	a04bf05e-80c6-4	https://fcm.goo	Win32	2021-02-24 12:	9l+K7OV9U9dn	BP9cNcRUvECx	fcs_IpmNtLw:APA91bHD5-oV1NylPeO9ukjejXZb6kGvGv8W0ocoCLtUCcufFvynfuVUpmmZosN31QNjt
5	d7d51395-6994-4	https://updates	Linux x86_6	2021-02-24 11:	xzXGzS4QPXzx	BGB4N1OVM/0C	dZ_kyl0WfPA:APA91bEopoNjF67KG15gy9JLDKNO5qMQX79LJ61XsyWdFn-oAzQXDVs4_HNy3ksmi
6	e2853901-52c1-4	https://fcm.goo	Linux x86_6	2021-03-10 12:	Agx80qyVeVFu	BK8SEpJU9Qqf	crGHstlcBSo:APA91bGHIPlkLTJ9Wm6RF9HddeaHvp1sfNhFUVrDvKnge7rpuD_dW1ncjGrwUaF81UH

- **Send the resulting csv file to the platform user.**

Administrator API

This part of documentation describes Altcraft Marketing Platform administrator API interface.

NOTE

All administrator API requests use POST method and need a token, created in Altcraft MP administrative
Some requests are similar to user API requests — the difference comes with multiple accounts in one Alt

Get accounts list

Description

Retrieves Altcraft service instance accounts list.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/accounts/list/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**100**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "from_id": 300,
    "limit": 10
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <from_id>300</from_id>
    <limit>10</limit>
</xml>
```

Response example

JSON

```
{
    "data": [],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 1
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>1</total_count>
</xml>
```

Response parameters

Parameter
error
error_text
id
name
timezone
tariff_id
is_active
pixels_enabled
pushes_enabled

Parameter
is_smart
is_subscribers_suspend_all
is_tariff_exceed

Add a new account

Description

Create a new platform account

Request URL

Method: POST

`https://example.com/api/v1.1/admin/accounts/save`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**
name	**string**	**“Account name”**
title	**string**	**“example title”**
tz	**string**	**“UTC”**
tariff	**int/string**	**“example tariff”**
web_version_policy	**int/string**	**“0”**
account_postgresql	**int/string**	**“2”**
account_cache_clickhouse	**int/string**	**“3”**
external_data_connectors	**int**	**[1,2,3,4]**
reports_day	**int**	**4**
reports_hour	**int**	**18**
force_2fa_auth	**bool**	**false**
enable_push	**bool**	**true**
custom_push_domain	**string**	**“example.com”**
enable_smart_settings	**bool**	**true**
enable_pixel	**bool**	**true**
enable_track_email_replies	**bool**	**false**

Request example

JSON

```
{
    "token": "aabcdefghijklmnqrstuvwxyz",
    "name": "Steve",
    "title": "ASdf",
    "tz": "UTC",
    "tariff": "example tariff",
    "reports_day": 5,
    "reports_hour": 22
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <name>Steve</name>
    <title>ASdf</title>
    <tz>UTC</tz>
    <tariff>example tariff</tariff>
    <reports_day>5</reports_day>
    <reports_hour>22</reports_hour>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 6
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
    <data>
        <saved_id>6</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter
error
error_text
saved_id

Delete the account

Description

Delete the platform account

Request URL

Method: POST

`https://example.com/api/v1.1/admin/accounts/delete`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**1**

Request example

JSON

```
{
    "token": "aabcdefghijklmnqrstuvwxyz",
    "format": "json",
    "id": 9
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <format>json</format>
    <id>9</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Update an Existing Account

Description

Modify an existing platform account

Request URL

Method: POST

`https://example.com/api/v1.1/admin/accounts/users/update`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
account_id	**int**	**1**
password	**string**	**“abcdefghijk1232313!”**
name	**string**	**“name”**
fname	**string**	**“Fname”**
lname	**string**	**“Lname”**
lang	**string**	**“RU”**
email	**string**	**“abcdefghij@example.com”**
master	**bool**	**true**

INFO

- **Only letters, dashes, underscores, and numbers are allowed in the username**
- **The password must contain 8 to 32 characters, including letters, numbers, and special characters**

Request example

JSON

```
{
    "token": "abcdefghijklmnopqrst",
    "format": "json",
    "account_id": 1,
    "password": "asdfGHJKOP12!_",
    "name": "example",
    "lname": "Example",
    "fname": "Example",
    "lang": "RU",
    "email": "example@gmail.com",
    "master": false
}
```

XML

```
<xml>
    <token>abcdefghijklmnopqrst</token>
    <format>json</format>
    <login>example@example</login>
    <password>asdfGHJKOP12!_</password>
    <name>example</name>
    <lname>Example</lname>
    <fname>Example</fname>
    <lang>RU</lang>
    <email>example@gmail.com</email>
    <master>false</master>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Add a new account

Description

Create a new platform account. To update an existing user profile, use this method.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/accounts/users/add`

Request parameters

Parameter	Type	Example	Required
token	**string**	**“abcdefghijklmnqrstuvwxyz”**	**Yes**
format	**string**	**“json”**	**No**
account_id	**int**	**1**	**Yes**
password	**string**	**“abcdefghijk1232313”**	**Yes**
name	**string**	**“name”**	**Yes**
fname	**string**	**“Fname”**	**Yes**
lname	**string**	**“Lname”**	**Yes**
lang	**string**	**“RU”**	**Yes**
email	**string**	**“abcdefghij@example.com”**	**Yes**
master	**bool**	**true**	**Yes**
send_welcome:	**bool**	**true**	**No**

INFO

- **Only letters, dashes, underscores, and numbers are allowed in the username**
- **The password must contain 8 to 32 characters, including letters, numbers, and special characters**

Request example

JSON

```
{
    "token": "abcdefghijklmnopqrst",
    "format": "json",
    "account_id": 1,
    "password": "asdfGHJKOP12!_",
    "name": "example",
    "lname": "Example",
    "fname": "Example",
    "lang": "RU",
    "email": "example@gmail.com",
    "master": false
}
```

XML

```
<xml>
    <token>abcdefghijklmnopqrst</token>
    <format>json</format>
    <account_id>example@example</account_id>
    <password>asdfGHJKOP12!_</password>
    <name>example</name>
    <lname>Example</lname>
    <fname>Example</fname>
    <lang>RU</lang>
    <email>example@gmail.com</email>
    <master>false</master>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Delete a user

Description

Delete an account user

Request URL

Method: POST

`https://example.com/api/v1.1/admin/accounts/add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
account_id	**int**	**1**
name	**string**	**“example”**

Request example

JSON

```
{
    "token": "aabcdefghijklmnqrstuvwxyz",
    "format": "json",
    "account_id": 1,
    "name": "example"
}
```

XML

```
<xml>
    <token>aabcdefghijklmnqrstuvwxyz</token>
    <format>json</format>
    <account_id>1</account_id>
    <name>Example</name>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get a list of users

Description

Get a list of account users

Request URL

Method: POST

`https://example.com/api/v1.1/admin/accounts/users/list`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
account_id	**int**	**9**
active	**bool**	**true**
master	**bool**	**false**
lang	**string**	**RU**

Request example

JSON

```
{
    "token": "aabcdefghijklmnqrstuvwxyz",
    "format": "json",
    "account_id": 9,
    "active": true,
    "master": false,
    "lang": "RU"
}
```

XML

```
<xml>
    <token>aabcdefghijklmnqrstuvwxyz</token>
    <format>json</format>
    <account_id>9</account_id>
    <active>true</active>
    <master>false</master>
    <lang>RU</lang>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "username": "example",
            "first_name": "Example",
            "last_name": "Example",
            "email": "example@gmail.com",
            "notify_email": "example@gmail.com",
            "phone": "",
            "lang": "RU",
            "active": true,
            "master": false
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 1
}
```

XML

```
<xml>
     <data>
        <user>
            <username>example</username>
            <first_name>Example</first_name>
            <last_name>Example</last_name>
            <email>example@gmail.com</email>
            <notify_email>example@gmail.com</notify_email>
            <phone></phone>
            <lang>RU</lang>
            <active>true</active>
            <master>false</master>
        </user>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>1</total_count>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
username	**string**
first_name	**string**
last_name	**string**
email	**string**
notify_email	**string**
phone	**string**
lang	**string**
active	**bool**
master	**bool**

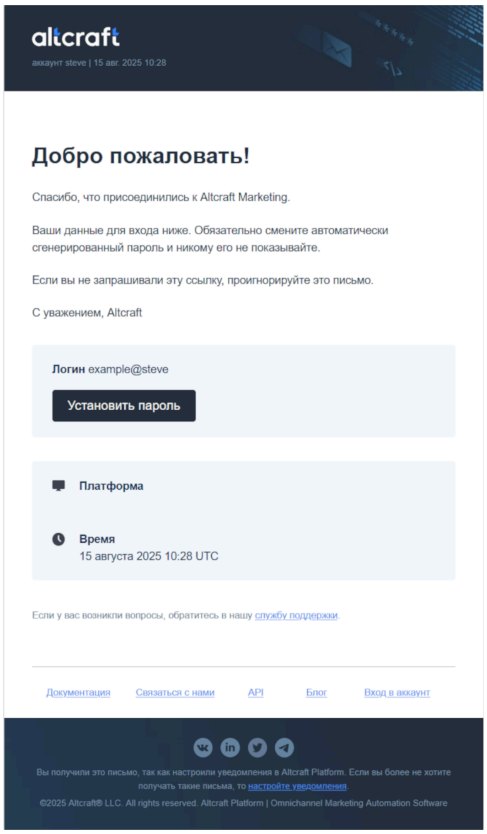
Sending a Welcome Email

Description

Sends a welcome email with a link to set a password. The email provides personal data about the initiato

- **Platform (UserAgent of the request)**
- **Device location (Geo-decoding of IP, if data is available in GeoDB)**
- **Operation time (approximate time of request execution)**

Example of a welcome email



URL Address

Method: POST

`https://example.com/api/v1.1/admin/accounts/users/send_welcome`

Request Parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
account_id	**int**	**1**
name	**string**	**“example”**

Request Example

JSON

```
    {
    "token": "abcdefghijklmnqrstuvwxyz",
    "account_id": 7,
    "name": "example"
    }
    ```

XML

```xml
    <xml>
      <token>abcdefghijklmnqrstuvwxyz</token>
      <account_id>Successful operation</account_id>
      <name>example<name>
    </xml>
    ```

Response Example
JSON

```json
    {
      "error": 0,
      "error_text": "Successful operation"
    }
    ```

XML

```xml
    <xml>
      <error>0</error>
      <error_text>Successful operation</error_text>
    </xml>
    ```

Response Parameters
Parameter	Type	Description
error	int	Error code
error_text	string	Error message text

Synchronize node MTA configuration

Description

Synchronizes node message transfer agent configuration.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/nodes/mta/sync/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**10**
format	**string**	**“json”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get nodes list

Description

Retrieves Altcraft service instance nodes list.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/nodes/list/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**100**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 1,
            "name": "new",
            "host": "localhost",
            "mode": "SSL"
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 1
}
```

XML

```
<xml>
    <data>
        <host>localhost</host>
        <id>1</id>
        <mode>SSL</mode>
        <name>new</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>1</total_count>
</xml>
```

Response parameters

Parameter	Ty
error	**int**

Parameter	Typ
error_text	**string**
id	**int**
name	**string**
host	**string**
mode	**string**

Get node MTA status

Description

Retrieves node message transfer agent status.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/nodes/mta/status/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**10**
format	**string**	**“json”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 10
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>10</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "synced": false,
        "running": true
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <running>true</running>
        <synced>false</synced>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	Type
error	**int**
error_text	**string**
synced	**bool**
running	**bool**

Activate node MTA

Description

Activates message transfer agent on a node.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/nodes/mta/start/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**10**
format	**string**	**“json”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Deactivate node MTA

Description

Deactivates message transfer agent on a node.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/nodes/mta/stop/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**10**
format	**string**	**“json”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Create or update AKMTA sender

Description

Creates or updates Altcraft message transfer agent instances.

Request URLs

Method: POST

Create AKMTA sender:

`https://example.com/api/v1.1/admin/senders/add`

Update AKMTA sender:

`https://example.com/api/v1.1/admin/senders/update`

Use with “id”:“%senderID%” property.

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
name	**string**	**“AKMTA mail sender”**
type	**string**	**“akmta”**
assigned_accounts	**JSON array**	**[123, 456]**
verified_domains	**JSON array**	**[“mail.roo”, “rambler.roo”]**
akmta_obj	**JSON object**	
id	**int**	**16**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 3,
    "name": "AK MTA mail sender",
    "type": "akmta",
    "channel_type": "email",
    "verified_domains": [],
    "akmta_obj": {
        "dkim_id": 0,
        "dkim_selector": "hello",
        "ips": [
            {
                "ip": "192.168.111.111",
                "node": 10
            }
        ],
        "lock_rules_obj": {
            "common_rules": [],
            "isp_rules": []
        },
        "redirect_email": "blackhole@altcraft.com",
        "retry_rules_obj": {
            "common_rules": [],
            "isp_rules": []
        },
        "sender_dkim_id": 0,
        "sender_dkim_selector": "altcraft",
        "sending_rules_obj": {
            "common_rules": [
                {
                    "isps": [
                        0
                    ],
                    "strategy": 2
                }
            ],
            "ip_rules": []
        },
        "users": [
            {
                "domain": "example.org",
                "pop3_pass": "jkhdsbasdb",
                "redirect_email": "redirect@example.org"
            }
        ]
    },
    "assigned_accounts": [
        306
    ]
}
```


XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>3</id>
    <name>AK MTA mail sender</name>
    <type>akmta</type>
    <channel_type>email</channel_type>
    <akmta_obj>
        <dkim_id>0</dkim_id>
        <dkim_selector>hello</dkim_selector>
        <ips>
            <ip>192.168.111.111</ip>
            <node>10</node>
        </ips>
        <lock_rules_obj>
        </lock_rules_obj>
        <redirect_email>blackhole@altcraft.com</redirect_email>
        <retry_rules_obj>
        </retry_rules_obj>
        <sender_dkim_id>0</sender_dkim_id>
        <sender_dkim_selector>altcraft</sender_dkim_selector>
        <sending_rules_obj>
            <common_rules>
                <isps>0</isps>
                <strategy>2</strategy>
            </common_rules>
        </sending_rules_obj>
        <users>
            <domain>example.org</domain>
            <pop3_pass>jkhdsbasdb</pop3_pass>
            <redirect_email>redirect@example.org</redirect_email>
        </users>
    </akmta_obj>
    <assigned_accounts>306</assigned_accounts>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 7
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
  <data>
    <saved_id>7</saved_id>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get AKMTA sender information

Description

Retrieves AKMTA sender detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/get/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**10**
format	**string**	**“json”**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>1</id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "akmta_obj": {
            "dkim_id": 1,
            "dkim_selector": "abcdefghijklmnqrstuvwxyz",
            "ips": [
                {
                    "ip": "10.200.1.93",
                    "node": 1
                },
                {
                    "ip": "10.200.1.94",
                    "node": 1
                }
            ],
            "lock_rules_obj": {
                "common_rules": [],
                "isp_rules": []
            },
            "redirect_email": "",
            "retry_rules_obj": {
                "common_rules": [],
                "isp_rules": []
            },
            "sender_dkim_id": 1,
            "sender_dkim_selector": "abcdefghijklmnqrstuvwxyz",
            "sending_rules_obj": {
                "common_rules": [
                    {
                        "isps": [
                            0
                        ],
                        "strategy": 7
                    }
                ],
                "ip_rules": []
            },
            "users": []
        },
        "assigned_accounts": [
            1
        ],
        "channel_type": "email",
        "id": 3,
        "name": "Email sender",
        "type": "akmta",
        "verified_domains": [],
        "verified_recipients": []
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML

```
<xml>
    <data>
        <akmta_obj>
            <dkim_id>1</dkim_id>
            <dkim_selector>abcdefghijklmnqrstuvwxyz</dkim_selector>
            <ips>
                <ip>10.200.1.93</ip>
                <node>1</node>
            </ips>
            <ips>
                <ip>10.200.1.94</ip>
                <node>1</node>
            </ips>
            <lock_rules_obj>
            </lock_rules_obj>
            <retry_rules_obj>
            </retry_rules_obj>
            <sender_dkim_id>1</sender_dkim_id>
            <sender_dkim_selector>abcdefghijklmnqrstuvwxyz</sender_dkim_selector>
            <sending_rules_obj>
                <common_rules>
                    <isps>0</isps>
                    <strategy>7</strategy>
                </common_rules>
            </sending_rules_obj>
        </akmta_obj>
        <assigned_accounts>1</assigned_accounts>
        <channel_type>email</channel_type>
        <id>3</id>
        <name>Email sender</name>
        <type>akmta</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
id	**int**
name	**string**
type	**string**
channel_type	**string**
assigned_accounts	**JSON array**
verified_domains	**JSON array**
akmta_obj	**JSON object**

Assign account to sender

Description

Assigns an account to a sender.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/add_to_account/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
id	**int**	**10**
format	**string**	**“json”**
account_id	**int**	**1**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 1,
  "account_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <account_id>1</account_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get senders list

Description

Retrieves Altcraft service instance senders list.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/list/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
from_id	**int**	**1**
limit	**int**	**100**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 3,
            "name": "Email sender",
            "type": "akmta",
            "channel_type": "email"
        },
        {
            "id": 6,
            "name": "new sender",
            "type": "sms_notify",
            "channel_type": "custom"
        }
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 4
}
```

XML

```
<xml>
    <data>
        <channel_type>email</channel_type>
        <id>3</id>
        <name>Email sender</name>
        <type>akmta</type>
    </data>
    <data>
        <channel_type>custom</channel_type>
        <id>6</id>
        <name>new sender</name>
        <type>sms_notify</type>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>4</total_count>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
id	**int**
name	**string**
type	**string**
channel_type	**string**

Delete sender

Description

Does NOT delete a sender but marks it as “DELETED” instead.

> **CAUTION**
>
> **Unlike - when you delete senders from administrative panel - the are actually removed from databases.**

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/delete/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 683
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>683</id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Restore sender

Description

Restores a sender marked as “DELETED” by Delete sender API request.

> **CAUTION**
>
> **Senders removed manually via administrative panel cannot be restored!**

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/undelete/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**

Request example

JSON

```
{
  "token": "abcdefghijklmnqrstuvwxyz",
  "id": 683
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>683</id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get sender queue information

Description

Get detailed information about the number of messages with different status in sender queue.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/queue/stats/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**
account_id	**int**	**12**
campaign_id	**int**	**12**
isp	**string**	**“Gmail”**
group *	**string**	**“total”**

*** “group” parameter can have these values:**

- **account – group messages by account**
- **account_campaign – group messages by mailings within accounts**
- **isp – group messages by ISP**
- **total – do not group messages**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 119
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>119</id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "account_id": 306,
            "campaign_id": 4591,
            "hold": 0,
            "isp": "",
            "new": 100,
            "proc": 100,
            "retry": 0,
            "total": 200
        },
        {
            "account_id": 306,
            "campaign_id": 4914,
            "hold": 80,
            "isp": "",
            "new": 10,
            "proc": 10,
            "retry": 40,
            "total": 140
        }
    ],
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
  <data>
    <account_id>306</account_id>
    <campaign_id>4591</campaign_id>
    <hold>0</hold>
    <isp></isp>
    <new>100</new>
    <proc>100</proc>
    <retry>0</retry>
    <total>200</total>
  </data>
  <data>
    <account_id>306</account_id>
    <campaign_id>4914</campaign_id>
    <hold>80</hold>
    <isp></isp>
    <new>10</new>
    <proc>10</proc>
    <retry>40</retry>
    <total>140</total>
  </data>
  <error>0</error>
  <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter
error
error_text

Hold sender queue

Description

Puts sender queue on hold thus preventing messages from being sent without cancelling.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/queue/hold/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**
account_id	**int**	**12**
campaign_id	**int**	**12**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 3,
    "account_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>3</id>
  <account_id>1</account_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Release sender queue

Description

Releases a sender queue that was previously put on hold.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/queue/unhold/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**
account_id	**int**	**12**
campaign_id	**int**	**12**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 3,
    "account_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>3</id>
  <account_id>1</account_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Clear sender queue

Description

Purges all messages from a sender queue.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/senders/queue/purge/`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
id	**int**	**12**
account_id	**int**	**12**
campaign_id	**int**	**12**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 3,
    "account_id": 1,
    "campaign_id": 3
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <id>3</id>
    <account_id>1</account_id>
    <campaign_id>3</campaign_id>
</xml>
```

Response example

JSON

```
{
  "error": 0,
  "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Get virtual senders list

Description

Retrieves virtual senders detailed information.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/list`

Request examples

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
account_id	**int**	**306**
from_id	**int**	**11768**
limit	**int**	**10**
format	**string**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "limit": 2,
    "account_id": 1
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <limit>2</limit>
    <account_id>1</account_id>
</xml>
```

Response example

JSON

```
{
    "data": [
        {
            "id": 1,
            "name": "Virtual sender12",
            "sms_channel": false,
            "email_channel": true,
            "custom_channels": {}
        },
    ],
    "error": 0,
    "error_text": "Successful operation",
    "total_count": 10
}
```

XML

```
<xml>
    <data>
        <email_channel>true</email_channel>
        <id>1</id>
        <name>Virtual sender12</name>
        <sms_channel>false</sms_channel>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
    <total_count>10</total_count>
</xml>
```

Response parameters

Parameter	T
error	**int**
error_text	**string**
id	**int**
name	**string**
sms_channel	**bool**
email_channel	**bool**

Get virtual sender information

Description

Retrieves a virtual sender information.

Request URL

Method: POST

`https://example.com/api/v1.1/admin/vsenders/get`

Request example

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
account_id	**int**	**306**
id	**int**	**12**
format	**string**	**“json”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 1,
    "account_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>1</id>
  <account_id>1</account_id>
</xml>
```

Response example

JSON

```
{
    "data": {
        "id": 1,
        "name": "Virtual sender12",
        "email_rules": [
            {
                "type": "default",
                "value": "",
                "senders": [
                    {
                        "from_email": "",
                        "replyto_email": "",
                        "from_name": "",
                        "sender_id": 3,
                        "percent": 100
                    }
                ]
            }
        ],
        "sms_rules": [],
        "custom_channels": {}
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <email_rules>
            <senders>
                <percent>100</percent>
                <sender_id>3</sender_id>
            </senders>
            <type>default</type>
        </email_rules>
        <id>1</id>
        <name>Virtual sender12</name>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
id	**int**
name	**string**
email_rules	**JSON-object**
sms_rules	**JSON-object**

Create virtual sender

Description

Creates a new virtual sender

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/add`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxy**
format	**string**	**“json”**
account_id	**int**	**306**
name	**string**	**“New virtual sender”**
email_rules	**JSON-object**	
sms_rules	**JSON-object**	

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "name": "New Email Virtual Sender",
    "sms_rules": [],
    "email_rules": [
        {
            "senders": [
                {
                    "from_email": "",
                    "from_name": "",
                    "percent": 100,
                    "replyto_email": "",
                    "sender_id": 3
                }
            ],
            "type": "default",
            "value": ""
        }
    ]
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <name>New Email Virtual Sender 1</name>
  <sms_rules array='true'></sms_rules>
  <email_rules array='true'>
    <senders array='true'>
      <from_email></from_email>
      <from_name></from_name>
      <percent>100</percent>
      <replyto_email></replyto_email>
      <sender_id>3</sender_id>
    </senders>
    <type>default</type>
    <value></value>
  </email_rules>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 99
    },
    "error": 0,
    "error_text": "Successful operation"
}
```


XML


```
<xml>
    <data>
        <saved_id>14</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```


Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Update virtual sender

Description

Updates an existent virtual sender configuration.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/update`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvw**
format	**string**	**“json”**
account_id	**int**	**306**
id	**int**	**12345**
name	**string**	**“New virtual sender”**
email_rules	**JSON-object**	
sms_rules	**JSON-object**	

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "name": "Virtual sender",
    "account_id": 3,
    "id": 14,
    "sms_rules": [],
    "email_rules": [
        {
            "senders": [
                {
                    "from_email": "",
                    "from_name": "",
                    "percent": 100,
                    "replyto_email": "",
                    "sender_id": 3
                }
            ],
            "type": "default",
            "value": ""
        }
    ]
}
```

XML

```
<xml>
<token>abcdefghijklmnqrstuvwxyz</token>
  <name>Virtual sender</name>
  <account_id>3</account_id>
  <id>14</id>
  <email_rules>
    <senders>
      <from_email></from_email>
      <from_name></from_name>
      <percent>100</percent>
      <replyto_email></replyto_email>
      <sender_id>3</sender_id>
    </senders>
    <type>default</type>
    <value></value>
  </email_rules>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 14
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>14</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text

Clone virtual sender

Description

Creates a copy of an existing virtual sender.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/clone`

Request parameters

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
account_id	**int**	**306**
format	**string**	**“json”**
id	**int**	**12345**
name	**string**	**“Clone of ...”**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "id": 14,
    "name": "Clone of virtual sender #14"
}
```

XML

```
<xml>
  <token>abcdefghijklmnqrstuvwxyz</token>
  <id>15</id>
  <name>Clone of virtual sender #15</name>
</xml>
```

Response example

JSON

```
{
    "data": {
        "saved_id": 16
    },
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <data>
        <saved_id>16</saved_id>
    </data>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter	
error	**int**
error_text	**string**
saved_id	**int**

Delete virtual sender

Description

Removes a virtual sender entry.

Request URL

Method: POST

`https://example.com/api/v1.1/vsenders/delete`

Request example

Parameter	Type	
token	**string**	**“abcdefghijklmnqrstuvwxyz”**
format	**string**	**“json”**
account_id	**int**	**306**
id	**int**	**12345**

Request example

JSON

```
{
    "token": "abcdefghijklmnqrstuvwxyz",
    "account_id": 1,
    "id": 16
}
```

XML

```
<xml>
    <token>abcdefghijklmnqrstuvwxyz</token>
    <account_id>1</account_id>
    <id>13</id>
</xml>
```

Response example

JSON

```
{
    "data": null,
    "error": 0,
    "error_text": "Successful operation"
}
```

XML

```
<xml>
    <error>0</error>
    <error_text>Successful operation</error_text>
</xml>
```

Response parameters

Parameter
error
error_text