

ALTCRAFT PLATFORM

Administrator Documentation

Contents

Sections and articles in this PDF.

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 Altcraft Cloud SMTP	admin-panel/senders/altcraft-cloud-smtp.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 to Percona Server	platform-configuration/mongodb-to-percona.mdx
Adding sender IP addresses	platform-configuration/3114051.mdx
Deduplication request settings	platform-configuration/41484334.mdx
PostgreSQL database for Market 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
Configuring MongoDB log 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
Custom Channels	custom-channels/index.mdx
Creating a Channel	custom-channels/creating-a-channel.mdx
External Objects (Entities)	custom-channels/external-objects.mdx
Templating Language	custom-channels/templating.mdx
Sending Files	custom-channels/sending-files.mdx
Presets (Field Sets)	custom-channels/presets.mdx
Debugging	custom-channels/debugging.mdx
Technical Limitations	custom-channels/limitations.mdx
Pipelines	custom-channels/pipelines/index.mdx
Message	custom-channels/pipelines/message.mdx
Schedule	custom-channels/pipelines/schedule.mdx
Listener	custom-channels/pipelines/listener.mdx
Moderate	custom-channels/pipelines/moderate.mdx
Stop	custom-channels/pipelines/stop.mdx
HTTP Request	custom-channels/pipelines/pipes/http-request.mdx
Pack	custom-channels/pipelines/pipes/pack.mdx
Unpack	custom-channels/pipelines/pipes/unpack.mdx
Eventer	custom-channels/pipelines/pipes/eventer.mdx
Scheduler	custom-channels/pipelines/pipes/scheduler.mdx
Selector	custom-channels/pipelines/pipes/selector.mdx
SQL	custom-channels/pipelines/pipes/sql.mdx
Store Set	custom-channels/pipelines/pipes/store-set.mdx
Store Get	custom-channels/pipelines/pipes/store-get.mdx
Log	custom-channels/pipelines/pipes/log.mdx

Result	custom-channels/pipelines/pipes/result.mdx
Error	custom-channels/pipelines/pipes/error.mdx
RMQ Publisher	custom-channels/pipelines/pipes/rmq-publish.mdx
Personnel requirements	platform-maintenance/33063284.mdx
Platform maintenance processes	platform-maintenance/33063294.mdx
Platform updating	platform-maintenance/2196347.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
Using the aktool utility	platform-maintenance/aktool.mdx
Users and directories engaged by the platform	platform-maintenance/users-directories.mdx
Platform service monitoring	platform-maintenance/9406592.mdx
Process and mailing monitoring via Prometheus	platform-maintenance/prometheus-monitoring.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 user	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

Administrator documentation: Contents

Contents of this section.

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 Altcraft Cloud SMTP	admin-panel/senders/altcraft-cloud-smtp.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 to Percona Server	platform-configuration/mongodb-to-percona.mdx
Adding sender IP addresses	platform-configuration/3114051.mdx
Deduplication request settings	platform-configuration/41484334.mdx
PostgreSQL database for Market 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
Configuring MongoDB log 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
Custom Channels	custom-channels/index.mdx
Creating a Channel	custom-channels/creating-a-channel.mdx
External Objects (Entities)	custom-channels/external-objects.mdx
Templating Language	custom-channels/templating.mdx
Sending Files	custom-channels/sending-files.mdx
Presets (Field Sets)	custom-channels/presets.mdx
Debugging	custom-channels/debugging.mdx
Technical Limitations	custom-channels/limitations.mdx
Pipelines	custom-channels/pipelines/index.mdx
Message	custom-channels/pipelines/message.mdx
Schedule	custom-channels/pipelines/schedule.mdx
Listener	custom-channels/pipelines/listener.mdx
Moderate	custom-channels/pipelines/moderate.mdx
Stop	custom-channels/pipelines/stop.mdx
HTTP Request	custom-channels/pipelines/pipes/http-request.mdx
Pack	custom-channels/pipelines/pipes/pack.mdx
Unpack	custom-channels/pipelines/pipes/unpack.mdx
Eventer	custom-channels/pipelines/pipes/eventer.mdx
Scheduler	custom-channels/pipelines/pipes/scheduler.mdx
Selector	custom-channels/pipelines/pipes/selector.mdx
SQL	custom-channels/pipelines/pipes/sql.mdx
Store Set	custom-channels/pipelines/pipes/store-set.mdx
Store Get	custom-channels/pipelines/pipes/store-get.mdx
Log	custom-channels/pipelines/pipes/log.mdx
Result	custom-channels/pipelines/pipes/result.mdx

Error	custom-channels/pipelines/pipes/error.mdx
RMQ Publisher	custom-channels/pipelines/pipes/rmq-publish.mdx
Personnel requirements	platform-maintenance/33063284.mdx
Platform maintenance processes	platform-maintenance/33063294.mdx
Platform updating	platform-maintenance/2196347.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
Using the aktool utility	platform-maintenance/aktool.mdx
Users and directories engaged by the platform	platform-maintenance/users-directories.mdx
Platform service monitoring	platform-maintenance/9406592.mdx
Process and mailing monitoring via Prometheus	platform-maintenance/prometheus-monitoring.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 user	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

This article will help you start using the Altcraft Platform. It provides a standard workflow for installing and configuring the platform in the Admin Panel and logging into the User Interface. Extended information about each step is available via links to related articles.

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 in mind that you should select hardware based on your specific needs. To clarify requirements in your case, you can send a request to support@altcraft.com.

Hardware Requirements

Server / Resource	Minimum Requirements	Recommended Requirements
Customer database size	up to 5 mln	up to 10 mln
Daily delivery volume	up to 1 mln emails	up to 5 mln emails
Management server — CPU	6 dedicated cores *	12 dedicated cores *
Management server — RAM	32 GB **	64 GB **
Management server — Disk	1 TB SSD ***	1 TB NVME ***
Sending server — CPU	3 dedicated cores *	6 dedicated cores *
Sending server — RAM	8 GB	16 GB
Sending server — Disk	0.5 TB ****	1.5 TB ****

* Assumes core allocation without considering hyper-threading level of Intel Xeon Gold 6240R or Intel Xeon Platinum 8260 or Intel Xeon Silver 4316 or AMD EPYC 7413 or AMD Ryzen Threadripper 2970WX or equivalent in performance.

** Additional memory allocation may be required when building suboptimal segments and automation mechanics.

*** Assumes write speed availability of 80 MB/s for HDD, 450 MB/s for SSD SATA, 1500 MB/s for NVME disks. Server-class disks are preferred. The calculation is made without considering the storage of generated content on the platform side. The number of operations and inodes on disks should not be limited.

**** The disk size is calculated for an average Email message size of 70 KB and a message lifetime of up to 72 hours. The number of operations and inodes on disks should not be limited.

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	Version	Source
Percona Server for MongoDB	8.0	https://www.percona.com
PostgreSQL *	16.1	https://www.postgresql.org
RabbitMQ	3.12.9 (4.0+ also supported)	https://www.rabbitmq.com
Kvrocks	Latest	https://github.com/apache/kvrocks
ClickHouse	24.8+ (LTS)	https://clickhouse.com
SSDB	1.8.0	https://github.com/ideawu/ssdb
xvfb	Any current version	Ubuntu repository **
cutycapt	Any current version	Ubuntu repository **

* 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 should have the following skills and knowledge:

- 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 (Percona Server for 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 methods in [this section](#).

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
```

Login

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. This section briefly describes what these entities are and how to configure them.

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 same server as the Altcraft management components;
- [Tariff](#) — a set of restrictions for a platform account. The Altcraft Platform license agreement does not impose any restrictions on data usage, but you can define your own restrictions for certain accounts;
- [Account](#) — the workspace for marketers and related specialists working with the Altcraft User Interface;

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 your web browser. Go to the list of account users and click the rocket button next to the desired user:

Users @ #1

<

1

>

Username ↕	Name ↕	Email ↕	Active ↕	Master ↕	Last login date ↕	Last login IP ↕	
astro	Astero Fernandez	asterofernandez@gmail.com	true	true	Fri, 28 Jun 2019 07:41:38 GMT	192.168.97.2	  
stainless	James DiGriz	asterofernandez@yandex.ru	true	false	Wed, 27 Feb 2019 16:23:11 GMT	192.168.97.2	  

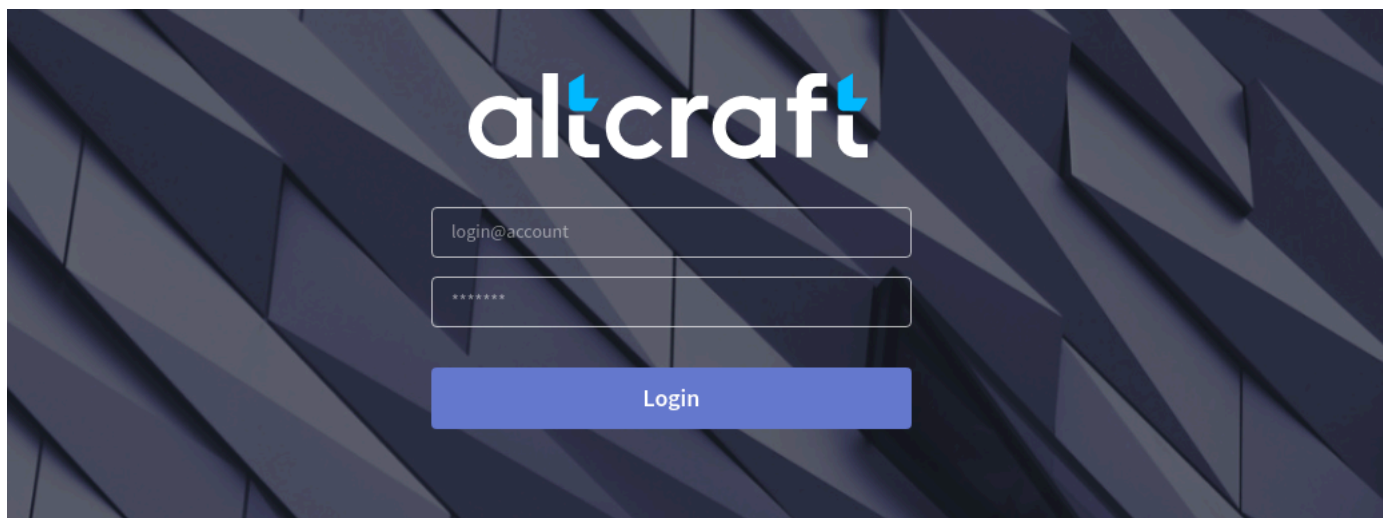
Access via Link

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 data, as well as to automate communications with customers through many channels on the Internet. The software can work as a cloud service or be hosted on the customer's server.

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 Services
- Integration with Amazon SNS
- Receiving applications from LPGenerator, Tilda
- Sending SMS via SMSCenter, BulkSMS, Svyaznoy Loading, Devino Telecom, SMSTraffic, Infobip, MTS, Sky Media, Play Mobile, Rapporto, etc
- 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. It 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

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 platform interface is accessible via HTTP and is supported by most modern browsers. Responsive layout allows you to use different devices for work. Details of working with the platform interface can be found in [“User documentation”](#).

Interface language

The platform is designed to support multiple user languages. Now two languages are available on the platform:

- 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 extremist and banned in Russia.*

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 101 and up
- Mozilla Firefox, version 102 and up
- Microsoft Edge, version 101 and up
- Opera, version 91 and up
- Chromium, version 101 and up
- Safari, version 15.0 and up
- Yandex Browser, version 23.2 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 in your case, please send a request to support@altcraft.com.

Deployment architecture

It is recommended to place key components on separate servers:

- **Database server** — PostgreSQL and Percona Server for MongoDB
- **Message queue server** — RabbitMQ
- **Control module server** — AKD
- **Sending module server** — AKMTA

Each database has its own memory requirements that must be taken into account when planning resources.

WARNING

NUMA architecture is not recommended for database servers. NUMA can lead to uneven memory distribution and performance degradation during intensive database operations.

Processor

- **Benchmark:** AMD EPYC 9354 or equivalent in performance
- **Minimum:** 8 cores / 16 threads
- **Recommended:** 16 cores / 32 threads

Memory

- **Recommended capacity:** 96 GB
- **Type:** DDR4 (high-performance modules)

Each database (PostgreSQL, MongoDB) has its own memory requirements. When hosting multiple databases on a single server, sum their requirements.

Disk subsystem

For servers with databases and message queues:

- **IOPS (4K random):** > 15,000 operations/sec for both read and write
- **Recommended type:** NVMe SSD — 4K random $\geq 500K$ IOPS read / $\geq 300K$ IOPS write, 512K sequential ≥ 3500 MB/s read / ≥ 3000 MB/s write
- **Minimum level:** Enterprise SATA SSD — 4K random $\geq 75K$ IOPS read / $\geq 20K$ IOPS write, 512K sequential ≥ 500 MB/s read / ≥ 450 MB/s write

For the sending module server (AKMTA spool):

- Standard SATA/SAS disks or network storage are acceptable

Network requirements

- **Between services:** 10 Gbit/s
- **To the internet:** minimum 250 Mbit/s, recommended 1 Gbit/s

Server requirements table

Server / Resource	Minimum requirements	Recommended requirements
Customer database size	Up to 5 million	Up to 10 million
Daily message sending volume	Up to 1 million	Up to 5 million

Server / Resource	Minimum requirements	Recommended requirements
Control server - CPU	8 dedicated cores *	16 dedicated cores *
Control server - RAM	64 GB **	96 GB **
Control server - Disk	1 TB NVMe ***	1 TB NVMe ***
Sending server - CPU	4 dedicated cores *	8 dedicated cores *
Sending server - RAM	16 GB	32 GB
Sending server - Disk	0.5 TB ****	1.5 TB ****

* Assumes core allocation without considering hyper-threading level of AMD EPYC 9354 or equivalent in performance.

** Additional memory allocation may be required when building suboptimal segments and automation mechanics.

*** Assumes IOPS (4K random) > 15,000 for both read and write. Recommended type — NVMe SSD: 4K random ≥ 500K IOPS read / ≥ 300K IOPS write, 512K sequential ≥ 3500 MB/s read / ≥ 3000 MB/s write. Minimum level — Enterprise SATA SSD: 4K random ≥ 75K IOPS read / ≥ 20K IOPS write, 512K sequential ≥ 500 MB/s read / ≥ 450 MB/s write. Server-class disks are preferred. The calculation is made without considering the storage of generated content on the platform side. The number of operations and inodes on disks should not be limited.

**** The disk size is calculated for an average Email message size of 70 KB and a message lifetime of up to 72 hours. Standard SATA/SAS disks or network storage are acceptable. The number of operations and inodes on disks should not be limited.

For cluster deployment

For cluster deployment, at least a threefold increase in resources will be required, taking into account the desirable separation of data storage locations (MongoDB, PostgreSQL).

Infrastructure requirements

A frequent question concerns public IPv4 addresses, so the requirement is explained as a separate paragraph.

In its standard configuration, the AKD server uses **three public (white) IP addresses**. One is intended for the control panel, the second is needed for the user action tracking system in email and SMS messages (tracking), and the third is required for the user action tracking system in push notifications and web, as well as for implementing web layer functions, displaying forms, and popups. An SSL certificate will be required for the user action tracking system on websites.

The AKMTA email sender can use the same IP addresses as the control host for operation if it is hosted directly on it, but this configuration is not recommended. The number of required addresses depends on the individual configuration of the system and the approach to message distribution. For precise determination, consultation with a marketer specializing in email mailings is required. For a volume of up to 1 million per day, 2-4 IP addresses are recommended. For a volume of 5-10 million per day, it is recommended to have 4 to 16 IP addresses.

Here and in the following, we are talking about IPv4 addresses, as IPv6 is currently not fully supported by network service providers for sending email.

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 license renewal (payments). In the absence of access, the platform will stop launching new mailings, but the mechanics of already sent messages will continue to work.

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 container](#)), depending on your preference.

The following open-source software is used during the system installation:

Software	Version	Source
Percona Server for MongoDB	8.0	https://www.percona.com
PostgreSQL	16.1	https://www.postgresql.org
RabbitMQ	3.12.9 (4.0+ versions are also supported)	https://www.rabbitmq.com
ClickHouse	24.8+ (LTS)	https://clickhouse.com
Kvrocks	Latest	https://github.com/apache/kvrocks
SSDB	1.8.0	https://github.com/ideawu/ssdb
xvfb	Latest	Ubuntu repository **
cutycapt	Latest	Ubuntu repository **

** 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	TCP Ports *	Access
RabbitMQ Message Broker - Message Exchange	5672 (AMQP), 25672 (Cluster)	For the control server, for the sending server
RabbitMQ Message Broker - control	15672 (HTTP)	For administrators
Tracking user actions in messages (dedicated IP)	80 (HTTP), 443 (HTTPS)	Public, for everyone
Control panel (dedicated IP)	80 (HTTP), 443 (HTTPS)	Public or internal network
Tracking actions in web and web layer (dedicated IP)	443 (HTTPS)	Public, for everyone
Percona Server for MongoDB (MongoDB-compatible) for storing all system configurations	27017	For the control server
PostgreSQL	5432	For the control server
SSDB for storing email addresses that cannot be delivered	4420	For the control server
SSDB for storing notifications	4430	For the control server
ClickHouse for storing profile history	9000	For the control server
Notification of reports and security system	25 (SMTP) or 465 (SMTPS)	Used to connect to SMTP for sending notifications
Kvrocks for checking duplicates (contacts) in mailings	6666	For the control server

AKMTA ports (Sending module)

Description	Ports	Access
For send messages	25 (SMTP)	Ensure that all IP addresses from which you send
For receive and process complaints and FBL	110 (POP3)	Ensure that all IP addresses from which you send
For auto remote interact with the sender	22 (SSH)	Ensure that the control server communicates with the sending server

* All ports can be reassigned.

** The choice of HTTPS or SSH senders management method is optional.

Processing HTTP/HTTPS traffic

NOTE

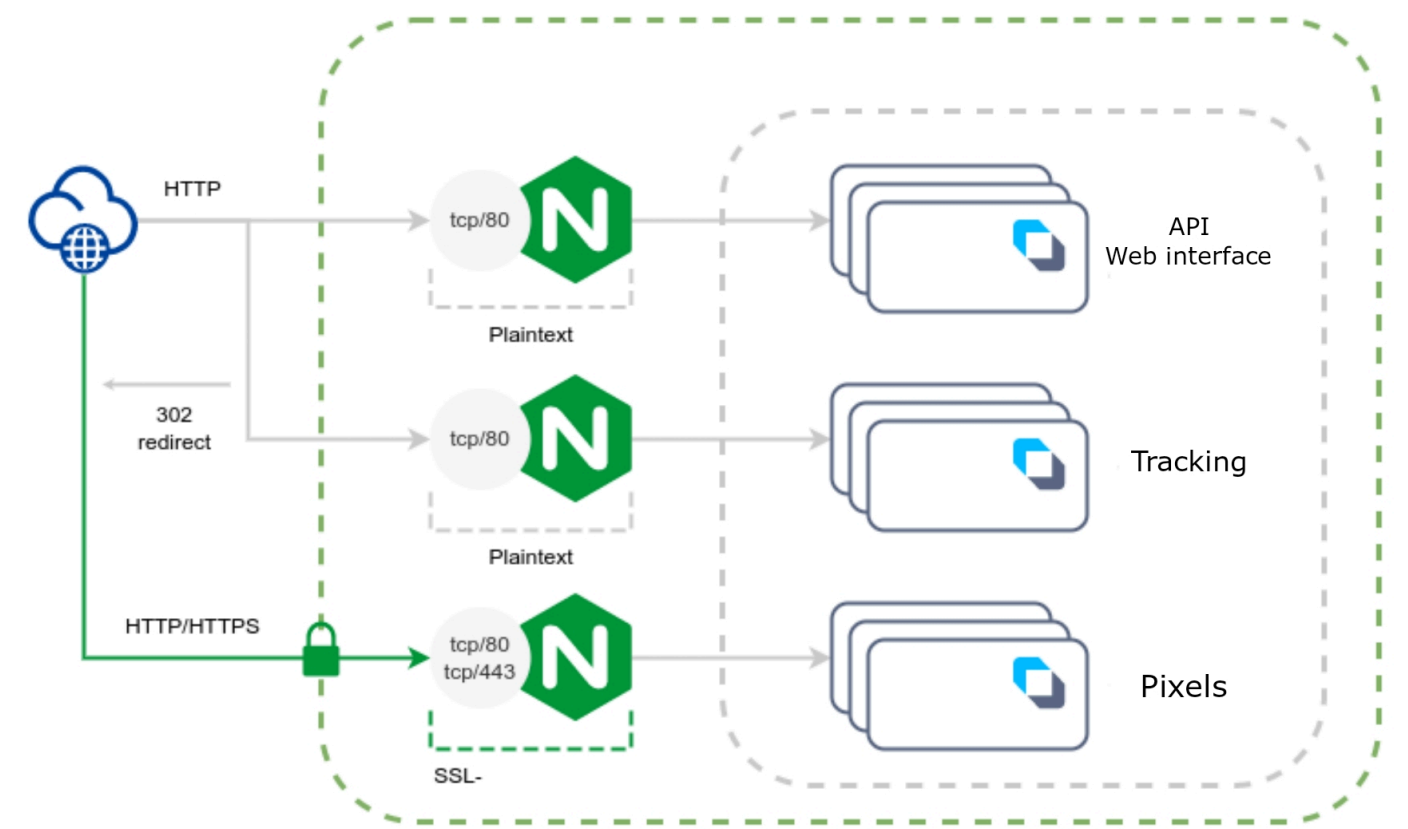
For a complete list of all configurable parameters, see the section - Configuration file. To apply changes to the platform configuration, restart AKD (./akd restart).

TIP

To get free Let's Encrypt certificates, you can use the certbot utility.

Default traffic processing scheme on the Altcraft side

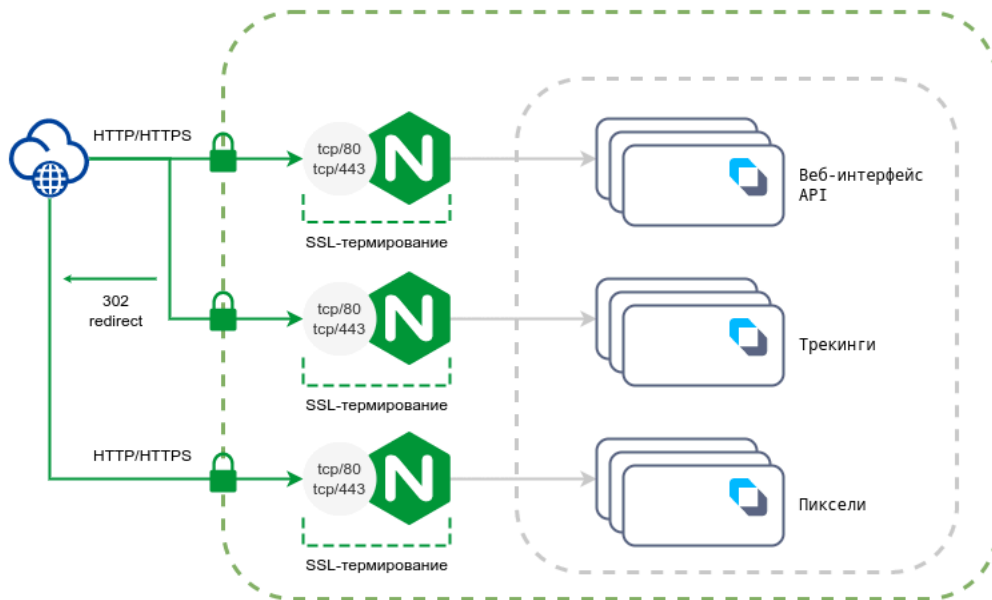
By default, HTTPS is required for pixels to work and custom cookies to be processed. Setting the COOKIESAVER_SSL_ON parameter to false will not affect the presence of the https:// protocol in links generated by mailings.



Required parameters

Parameter	Type	Value	Description
SSL_ON	bool	false	Disabling SSL termination for the web interface
TRACKING_SSL_ON	bool	false	Disabling SSL termination for tracking
COOKIESAVER_SSL_ON	bool	true	Enable SSL termination for pixels
COOKIESAVER_SSL_CERT_PATH	string	/etc/letsencrypt/live/pxl.example.com/fullchain.pem	Path to certificate for pixels
COOKIESAVER_SSL_KEY_PATH	string	/etc/letsencrypt/live/pxl.example.com/privkey.pem	Path to the private key to the certificate for pixels

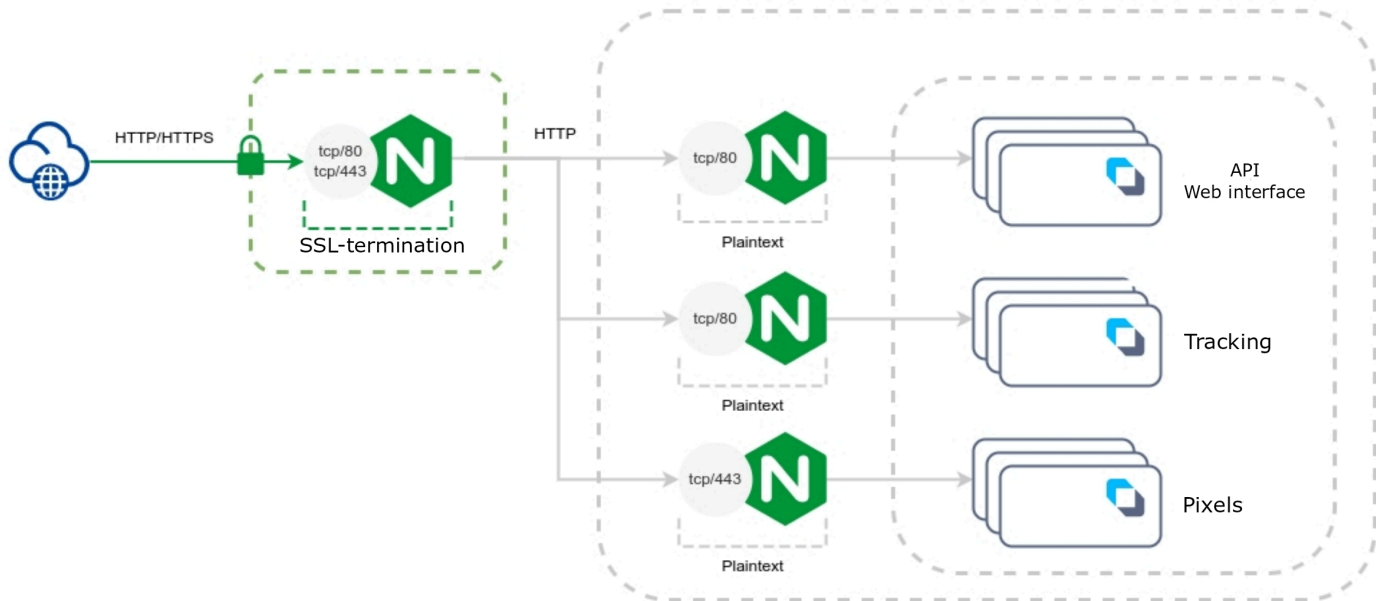
Scheme for processing traffic with full SSL termination on the Altcraft side



Required parameters

Parameter	Type	Value	Description
SSL_ON	bool	<code>true</code>	Enabling SSL termination for the web interface
SSL_CERT_PATH	string	<code>/etc/letsencrypt/live/altcraft.example.com/fullchain.pem</code>	Path to the certificate for the web interface
SSL_KEY_PATH	string	<code>/etc/letsencrypt/live/altcraft.example.com/privkey.pem</code>	Path to the private key to the certificate for the web interface
TRACKING_SSL_ON	bool	<code>true</code>	Enabling SSL termination for tracking
TRACKING_SSL_CERT_PATH	string	<code>/etc/letsencrypt/live/click.example.com/fullchain.pem</code>	Path to a trekking certificate
TRACKING_SSL_KEY_PATH	string	<code>/etc/letsencrypt/live/click.example.com/privkey.pem</code>	Path to the private key to the tracking certificate
COOKIESAVER_SSL_ON	bool	<code>true</code>	Enable SSL termination for pixels
COOKIESAVER_SSL_CERT_PATH	string	<code>/etc/letsencrypt/live/pxl.example.com/fullchain.pem</code>	Path to certificate for pixels
COOKIESAVER_SSL_KEY_PATH	string	<code>/etc/letsencrypt/live/pxl.example.com/privkey.pem</code>	Path to the private key to the certificate for pixels

Traffic processing scheme with full SSL termination on the external side



Required parameters

Parameter	Type	Value	Description
SSL_ON	bool	<code>false</code>	Disabling SSL termination for the web interface
TRACKING_SSL_ON	bool	<code>false</code>	Disabling SSL termination for tracking
TRACKING_SSL_EXTERNAL_ON	bool	<code>true</code>	Setting the sign of using SSL termination at the top for tracking
COOKIESAVER_SSL_ON	bool	<code>false</code>	Disable SSL termination for pixels
NGINX_TRUSTED_PROXIES	array	<code>["Внешний IP-адрес прокси"]</code>	List of trusted proxy IP addresses
NGINX_REALIP_HEADER	string	<code>"X-Forwarded-For"</code>	Header from which a real IP address of a client will be obtained

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 updated to reflect 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 the API reference combined into a single PDF file.

- **Version 76** — [PDF](#)
- **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 archived versions; they remain in the state in which they were archived.

Architecture Overview

Description

Altcraft is a scalable marketing automation platform built on a microservice architecture. The microservice approach ensures fault tolerance (a failure of one service does not affect the entire system) and horizontal scalability. All services are written in Go and designed to run in a Linux environment. Some parts of the system can be distributed across network zones (e.g., Internet, DMZ, Data Storage) for increased security. A common practice is isolating databases and MTA (`akmtad`) in separate segments.

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 termination are handled by the Nginx web server, supplied as a pre-built package with Altcraft. Three Nginx instances run simultaneously, each serving its own traffic type.

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 session storage. Authorization depends on the control panel:

- `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 external sources or APIs. An example of such a service is `procworkflow` .

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 configured rules and channels. Email delivery is handled by the `akmtad` service, which can operate as a separate node. Multiple `akmtad` instances provide scalability and help bypass blocks. SSH is used to speed up deployment.

Customer Action Tracking

Implemented using pixels, HTTP redirects, and cookies. Access control and scalability are managed via the `procrpc` RPC service. For handling website and message traffic, several Nginx instances are used: `nginx-cookiesaver` and `nginx-trk` .

Event Processing and Analytics

Data collection, aggregation, and transmission to reports and external services. Nearly all Altcraft services participate in these processes.

Periodic Tasks

Customer data processing, import, and export are performed on schedule by dedicated Altcraft services.

Utility Applications

Examples:

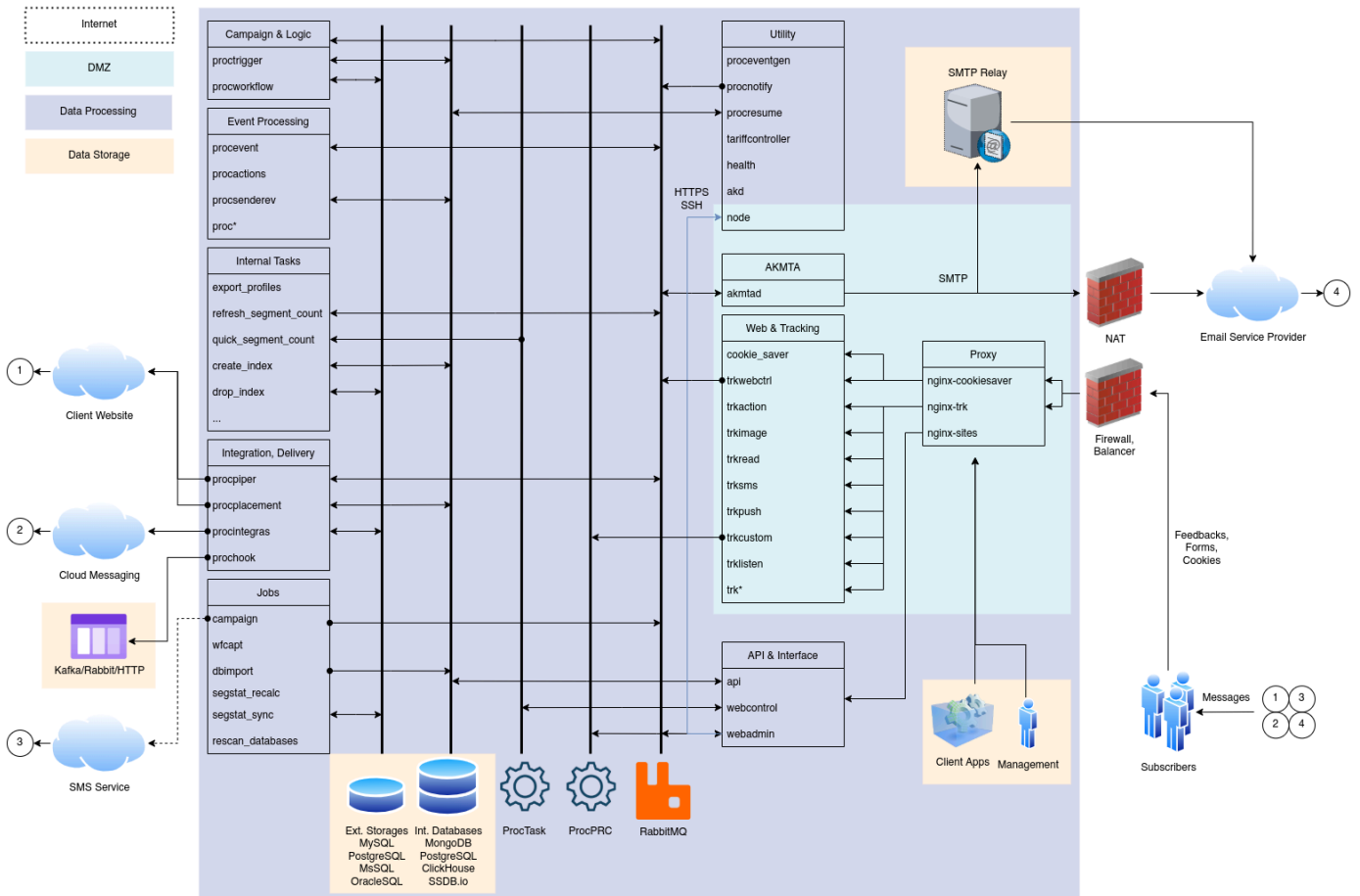
- `akd` — starts and stops services;
- `altool` — performs migrations and manages certain entities created in Altcraft;
- `health` — evaluates service health.

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 messages from external sources;
- **DMZ** — demilitarized zone for receiving HTTP and SMTP traffic. Services located in the DMZ are isolated and have no direct access to databases;
- **Data Processing** — data processing zone where the core business logic of the platform resides. This is where calculations, aggregation, and data transformation occur;
- **Data Storage** — secure segment for storing system and customer data (including external sources).

Altcraft Platform v2022.1+



In the diagram, platform components are grouped by functional purpose. Some services run continuously (daemons), others run periodically or on demand (**Jobs** and **Internal Tasks**).

- **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 clicks).
- **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's external proxy.

All components support horizontal scaling. However, some services (e.g., `procworkflow` and `proctask`) operate in master-worker mode and must be configured properly.

Component Description

Services

API and Web Interfaces

Services providing access to the Altcraft platform for users and administrators, as well as interaction with internal components.

- `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 segments
- `webloyalty` — the loyalty program module interface

Tracking and Analytics

Services that collect, route, and process tracking events (clicks, opens, subscriptions, forms, etc.), as well as serve static content.

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 pixel)
 - 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 pixel)
 - 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 events for `proctrigger` and `prochhook`
- `procpixel` — processes web-layer pixel events, imports and updates profiles, generates events for `proctrigger` and `prochhook`
- `procpush` — processes Push channel events, imports and updates profiles, creates push events
- `procsenderev` — processes events from senders: deliveries, bounces, returns, responses (deprecated)
- `procsmscv` — processes outgoing SMS gateway events, polls gateways, generates events for `proctrigger` and `prochhook`
- `procsmslisten` — processes incoming SMS gateway events, receives provider hooks, generates events for `proctrigger` and `prochhook`
- `proctrkweb` — processes web form and popup events, writes data to the database, generates events for `proctrigger` and `prochhook`
- `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
 - tasks of high-priority scenarios are processed by a dedicated worker pool on the cluster hosts that have `HIGH_PRIORITY_WORKERS` enabled; when there are no priority tasks, the workers pick up regular ones
 - if priority is disabled on all cluster hosts, the service does not start
- `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 queue (broker) for further processing by senders
- `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.g., Facebook, VK)

Data Processing

Services for internal data processing, integrations, and subscription management.

- `procresume` — manages subscriptions: automatically unfreezes (resumes) subscriptions for profiles after suspension expiration
- `proctrigger` — processes and manages trigger (event-based) campaigns
- `prochhook` — manages outgoing webhooks: generates and sends notifications to external systems based on Altcraft events
- `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 `prochhook`
- `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 other common data required for Altcraft operation.

<https://www.mongodb.com/>

ClickHouse

ClickHouse — an analytical DBMS used to store subscriber activity history and Altcraft events (and in some cases for caching external SQL query results). These data are used for building segments and generating reports.

<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 hashes of contact data).

<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.

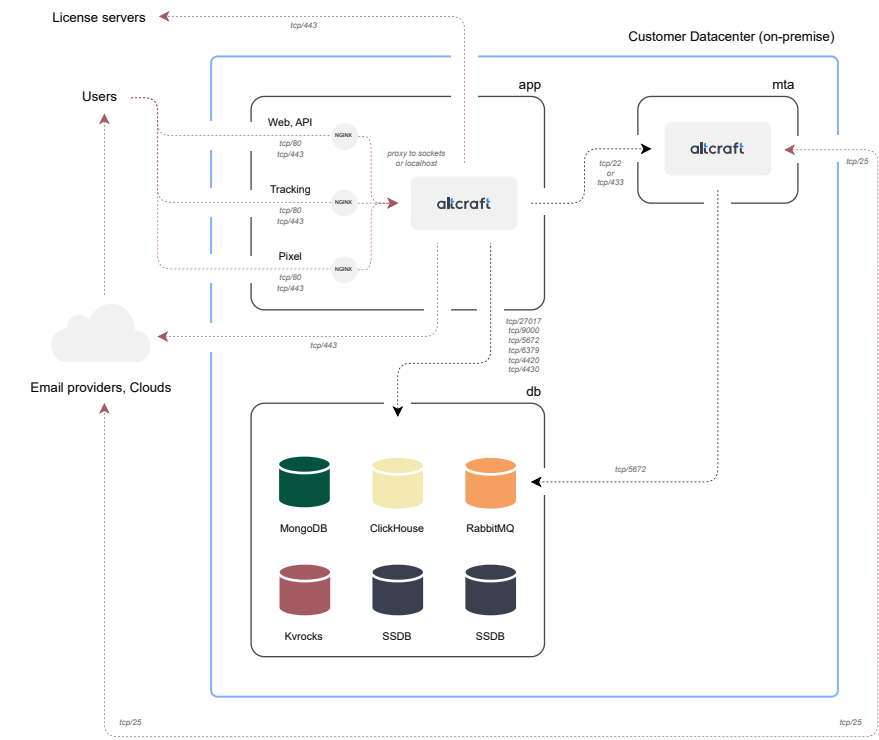
Message Brokers

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 page presents several simplified and most common deployment schemes in a customer's infrastructure. These diagrams are intended for general reference and help to understand the basic principles of system design, on the basis of which a specific architecture is then formed according to project requirements. Key components in the diagrams:

- **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 delivery.
- **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 *profiles*.
- **Email services (ISP)** — email providers (e.g., Gmail, Yandex, etc.) that receive email messages.

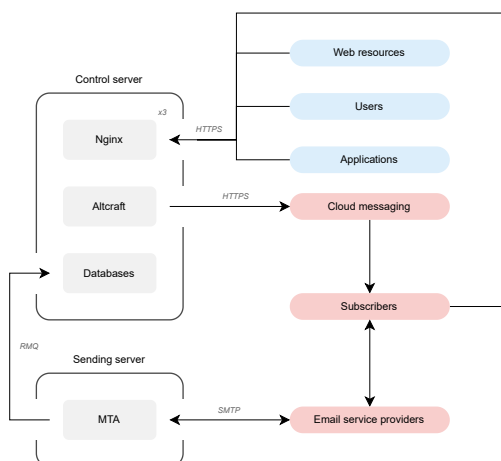
Regardless of the deployment method, the following questions must be addressed in advance:

- 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 feature of this scheme is SSL termination handled by Altcraft itself — the certificate is placed inside the service.

INFO

At the moment, Altcraft runs three separate Nginx instances, which requires three IP addresses on the server. Otherwise, launching on the same ports will cause conflicts.

The MTA interacts with ISPs directly, which allows the use of bypass mechanisms (various retry rules for automatic resending).

INFO

At least two IP addresses are required for sending. Depending on mailing volume and content type, more addresses may be needed to increase delivery speed and improve deliverability.

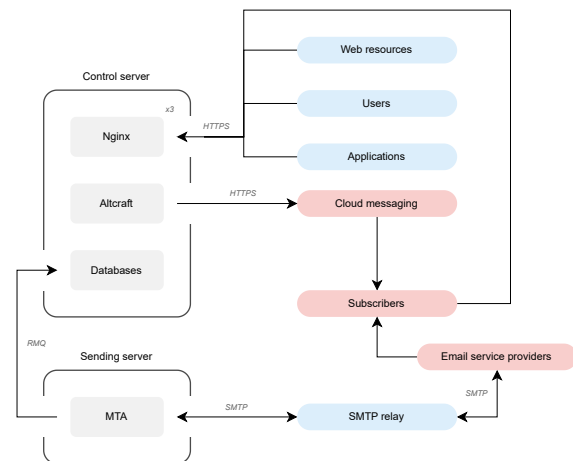
Using an SMTP Relay

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.

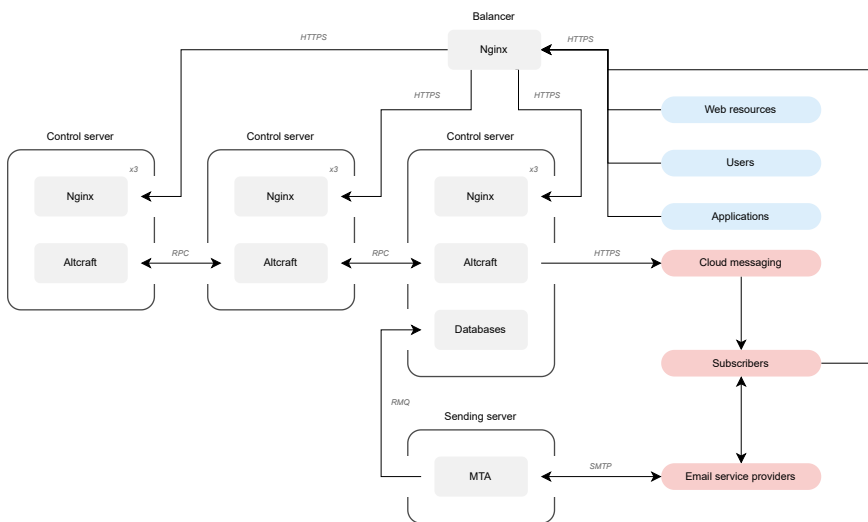
When using an SMTP relay, the Altcraft MTA must receive delivery reports (DSN, Delivery Status Notification). Such a report may be generated either by the relay itself or by the destination email service. The key requirement is that the domain in MAIL FROM (Return-Path header) must point to one of the IP addresses configured in the Altcraft sender (or the domain must resolve to the relay only for external servers — this can be achieved with Split DNS; in that case, the relay must forward received messages to the MTA). Upon receiving a report, Altcraft automatically processes it and records a delivery 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 handled by the customer's infrastructure.

INFO

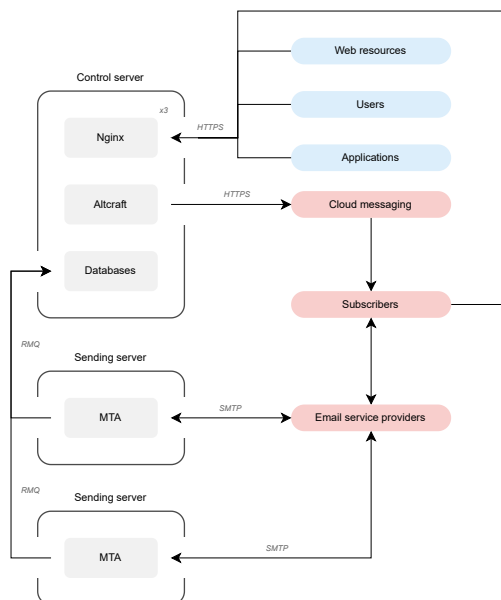
- All databases can be replicated, thereby improving the overall fault tolerance of the solution.
- The load balancer must pass the client's real address in the `X-Real-IP` or `X-Forwarded-For` header, and its IP address must be listed as trusted in the Altcraft configuration.
- When deploying the service in multiple instances, configure **sticky sessions** on the load balancer. This ensures that all requests from a single user are routed to the same instance, which prevents cache desynchronization and incorrect system behavior.

Other Options

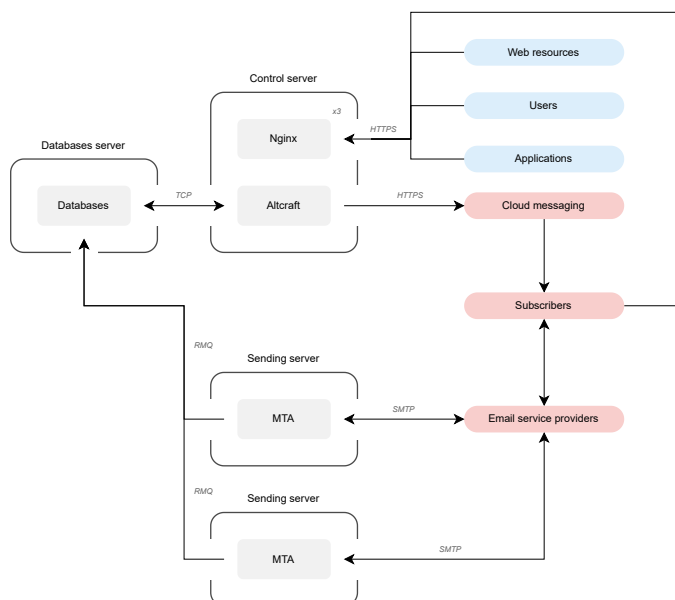
For example, to ensure high-performance sending (millions of emails per hour), multiple MTA servers can be deployed.

INFO

If a server has multiple IP addresses, one of them can be used for sending via an SMTP relay by creating a dedicated sender.



And to simplify maintenance, all databases can be moved to a dedicated server and resources redistributed.



The specific scheme depends on the tasks at hand — for example, a prioritized delivery channel. Details of the optimal configuration can be clarified with the Altcraft team: support@altcraft.com.

Nodes

Description

A **Node** is a network interface of the sending server used by AKMTA senders. A node can be located on the same server as Altcraft management components. However, the sending module is usually deployed on a separate server and connected to the management component via SSH or HTTPS protocols.

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**.

altcraft

Admin

Account Area

Settings

Nodes

Senders

Reports

Tools

John Doe
v2025.1.71.983

That's what we have today

Accounts

1

Email Senders

1

SMS Senders

1

Nodes

1

External data configurations

3

Databases

5

Tariffs

2

API Tokens

0

Admins

1

The list of existing nodes will appear. On the right side of the table, there are buttons for **viewing error logs**, **editing**, and **deleting**. You can also open editing by clicking on the name or ID of a node (on the left side).

NOTE

In the node list, you can start or stop the **MTA** (mailer) on a specific node or on all available ones. After making changes, it is often necessary to restart the node so that the settings are synchronized with the Altcraft management server. The synchronization status is shown in the **Synced** column.

To create a new node, click **+ Create**:

altcraftAdmin

Account AreaSettingsNodesSendersReportsTools

John Doe
v2025.1.71.617

Nodes

Start allStop allCreate

Search

Matches 2125 / page

ID	Name	Host	Mode	Enable	Synced	MTA	Resources	Actions
9	Node1	localhost	SSH	☒	✓ Yes	Alive	10.200.1.60	

For the new node, specify a unique **Name**, the network **Host** address, and the **Node installation directory** on the sending server:

Nodes > #9

* Name

NewNode

* Host ?

111.1.1.11

* Node installation directory ?

/opt/ak

Node with SSH Access

By default, SSH protocol is used to access the server with the node. For this type of connection, you must specify the **User** and the **SSH key**:

☒ SSH mode☐ HTTPS mode

* User

root

* SSH key ?

1 - ssh1

Before saving the node:

1. Add public SSH key «1 - SSH key» to your remote host 111.1.1.11

2. Add host 111.1.1.11 to all your marketing platform hosts:ssh-keyscan -t rsa 111.1.1.11 >> ~/.ssh/known_hosts

See [documentation](#) to learn more about it. To create an RSA encoded SSH key, Setup -> [keys](#).

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 will be available for download after switching the node to **HTTPS mode**. You must also specify the **API key** for node access:

☐ SSH mode☒ HTTPS mode

* API key

xxxxxxxxxxxxxxxxxxxxxxxx

HTTPS node requires manual installation and update

Download [node package here](#). Please install and start it **before** saving the node. Documentation managing nodes - [HTTPS access node](#).

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": "046FtVEU1UdrMDwdCAORyDJelirt4r0W",
  "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 other parameters:

MTA Settings

* Priority channel size

* Other channel size

* Spool channel size

* Number of spool writers

* Retry channel size

* Event channel size

* Spool cleaner timeout (days)

* DNS lookup timeout (sec)

* DNS lookup cache expire (sec)

* Number of RMQ consumers per sender

* Number of spool reader

* Number of eventer reader

* ESMTP connection timeout

* Fetcher prior spool timeout

* Rebuilder workers count

* Override FBL header name

SMTP administrator email

POP3 listen only interfaces

SMTP listen only interfaces

☒ Write SMTP responses to system log

☒ Enable MTA observer

☐ Enable profiling

☐ Clear spool messages after delivery

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 messages.
- **Override FBL header name** — override the service header name for [receiving complaint reports](#) (FeedBack Loop).
- **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 of lost connection.
- **Enable profiling** — enable profiling (pprof).
- **Clear spool messages after delivery** — with this option enabled, delivered messages will be automatically deleted from the spool.

Debugging and Troubleshooting

Synchronizing Settings

After making changes to the [sender](#) configuration, you must synchronize the management and sending modules of Altcraft. When necessary, after saving the sender configuration, you will see this message:

Saved but not synced!

The changes are saved, but will not be applied until you restart the AKMTA servers on the nodes. Do you want to go to the nodes list to sync them manually?

Stay in senders

Go to nodes

The list of nodes will open. You must stop and start the sender on the desired node by clicking **Stop** and **Start** in the MTA column:

ID	Name	Host	Mode	Enable	Synced	MTA
9	test	localhost	SSH	☒	No	Stop MTA ☒ Alive

Tariffs

Description

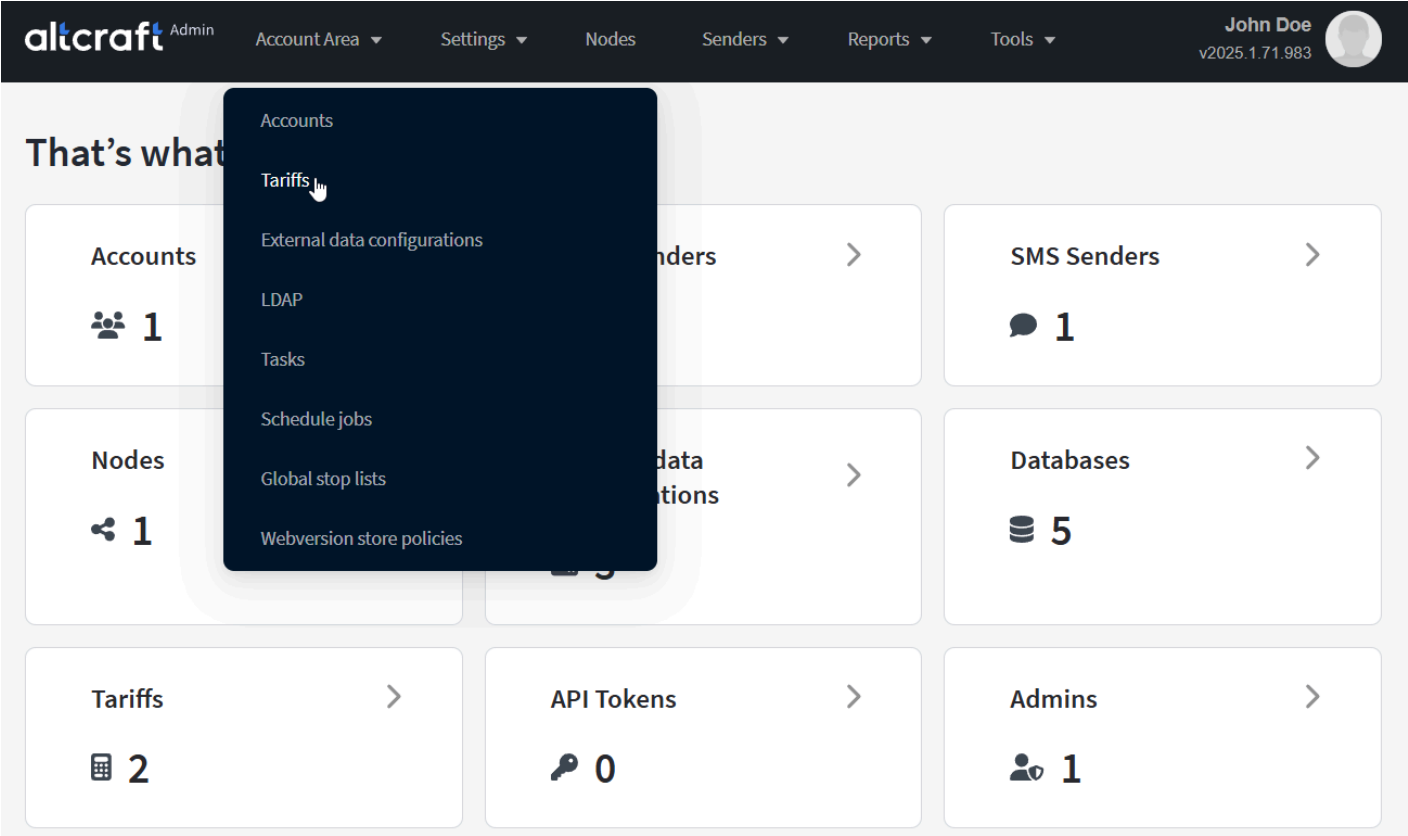
A **tariff** is a set of limits for a platform account. The license agreement for the Altcraft marketing platform does not impose any restrictions on data usage, but you can set your own limits for specific [platform accounts](#).

CAUTION

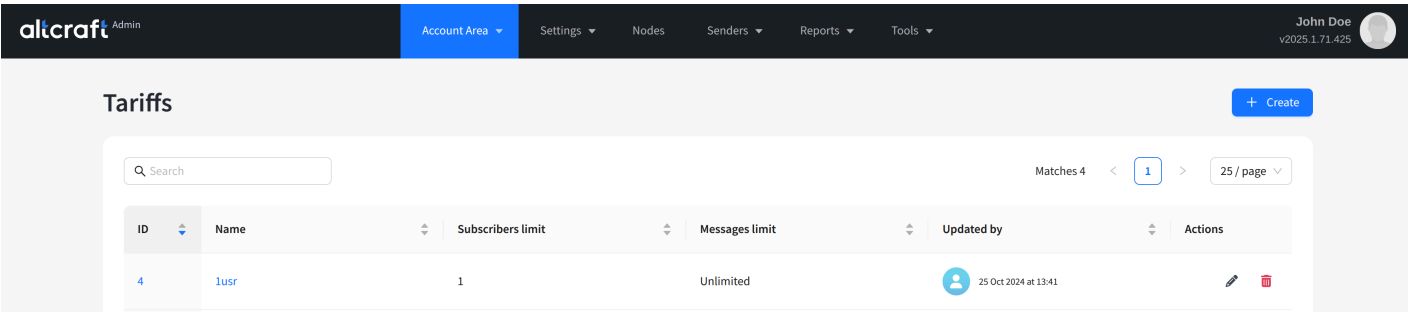
A tariff is a mandatory attribute of an account. Create at least one basic tariff even if you do not plan to limit data or communications.

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 also click the tariff identifier on the left to edit. To create a new tariff, click **+ Create**:



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 are counted).
- **Messages limit** — limit on the number of messages sent (all communication channels are counted).

Tariffs > Edit

* Name

Basic Tariff

Limits

* Subscribers limit

0

* Messages limit

0

Save

Close

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 databases.

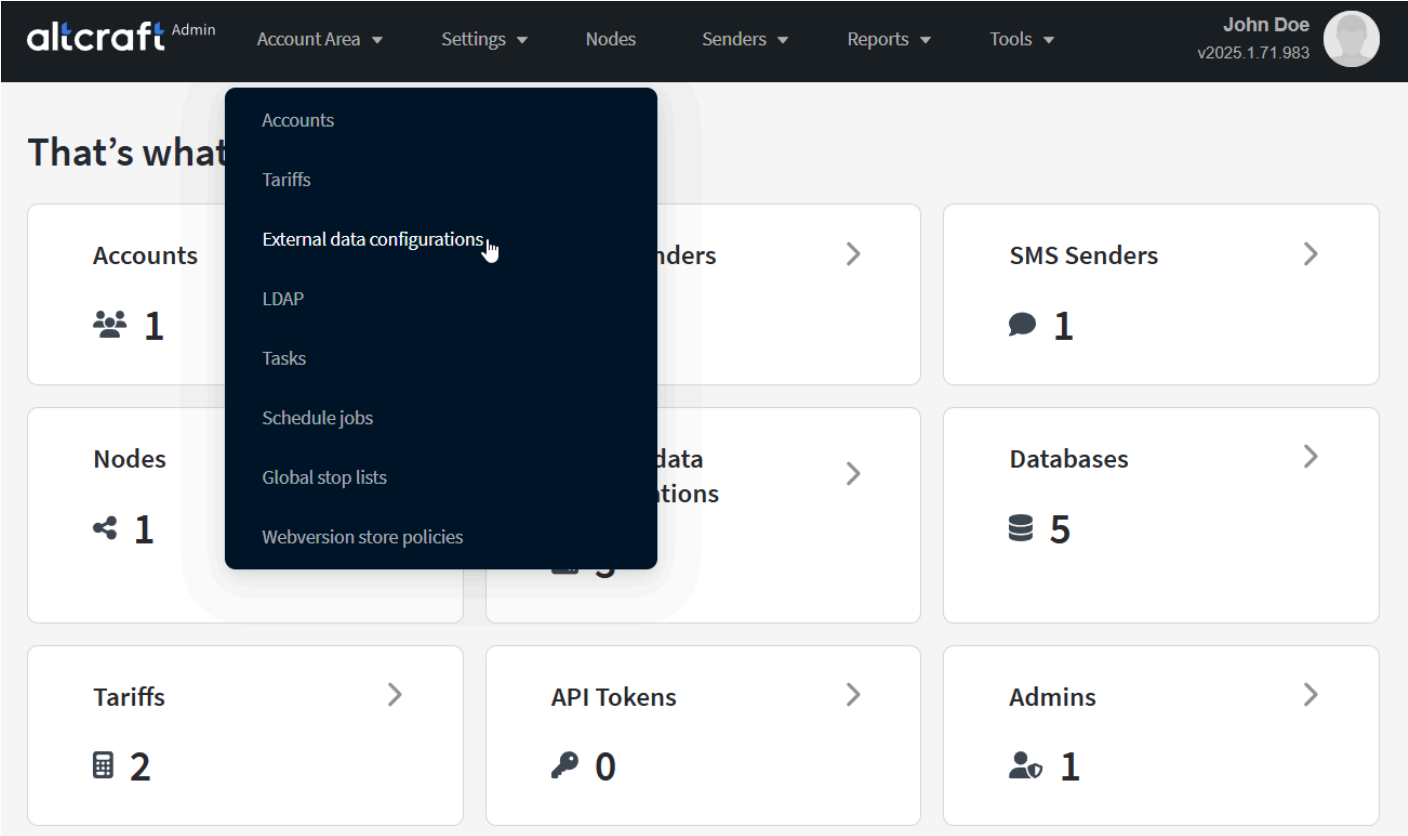
Capabilities

- [Scheduled import of customer data](#)
- [Segmentation of subscribers by external databases](#)
- [Dynamic content in message templates](#)

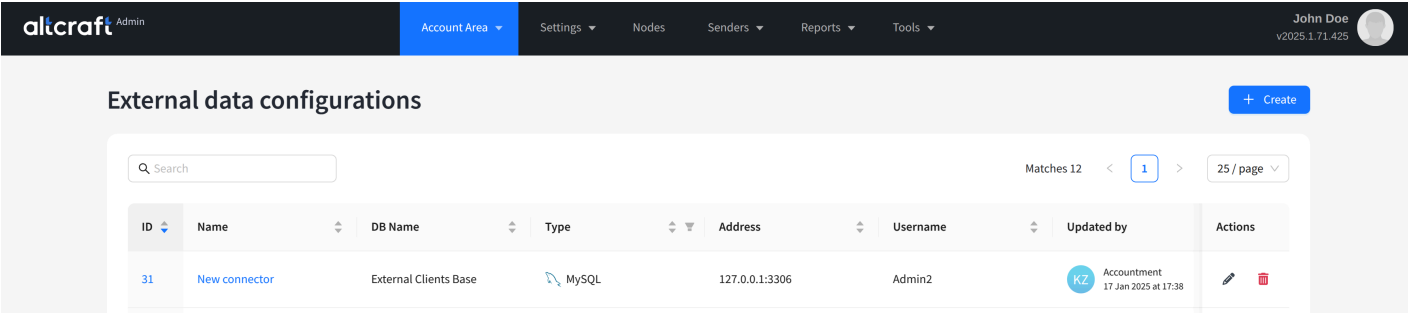
Creating, Editing, and Deleting

Integration with external SQL databases is configured in the **Altcraft Admin Panel**, while queries to databases and scheduled imports are created in the account's user interface.

To manage external data source connections, in the main menu select **Account Area — External data configurations**.



The list displays existing connectors to external databases. On the right side of the table are the **edit** and **delete** buttons. You can also click the connector identifier on the left to edit. To create a new connector, click **+ Create**:



For each connector, specify the following details:

- **Name** — connector name.
- **Type** — the DBMS type of the external database. Supported types are ****MySQL, PostgreSQL, MSSQL, Oracle, Hive****, and ****ClickHouse****.
- **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, PostgreSQL, and ClickHouse.

Hive DBMS supports different authentication types: PLAIN, GSSAPI (Kerberos), LDAP (also PLAIN), and custom types. Authentication itself is configured in Hive's config; in the platform's dsn option, you need to specify the required `auth` parameter (NOSASL, PLAIN, GSSAPI).

TIP

You can specify the **Collation** used for connecting to the database in the Additional dsn options field using the `collation_connection=` key.

External data configurations > Edit

* Name

New connector

Type

MySQL

* Database name

default

* Connection timeout

1

* Query timeout

1

* Protocol

TCP

* Host

localhost

* Port

1234

Test connection

Username

Password

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 format:

☒ With SSL

* SSL certificate

-----BEGIN CERTIFICATE-----

* SSL key

-----BEGIN ~~RSA~~ PRIVATE KEY-----

* SSL CA certificate

-----BEGIN CA CERTIFICATE-----

Connecting a connector to an account

After the connector is created, assign it to a platform account. In the admin panel, open **Account Area — Accounts** and select the account to edit. In the **Account databases** section, choose the connector created earlier. One account can have multiple connectors to different external databases:

External data ?

2 - SQL database connector ×



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 company employees to use their credentials from the corporate directory service, such as Active Directory, to log into the platform with the appropriate set of [groups and access roles](#).

NOTE

This article describes managing directory server connections for access to the Altcraft User Interface. LDAP access configuration for the Admin Panel is described in [this guide](#).

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. You can also click a connection on the left to edit. To create a new directory server connector, click **+ Create**:

altcraftAdmin

Account Area

Settings

Nodes

Senders

Reports

Tools

John Doe
v2025.1.71.425

LDAP connections

+ Create

Search

Matches 34

12

25 / page

ID	Name	Host	Port	Username	Updated by	Actions
35	Main LDAP	localhost	1111	admin	acJohn Doe 22 Jan 2025 at 13:33	

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 fails.
- **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:

LDAP connections > Edit

* Name

default

* Host

localhost

* Port

389

Test connection

Secondary host

Secondary port

Test connection

* Username

admin

* Password

* Base DN

DC=example,DC=com

☐ Use SSL

Save

Close

CAUTION

The account must have sufficient permissions to access the groups that will be used for platform authentication.

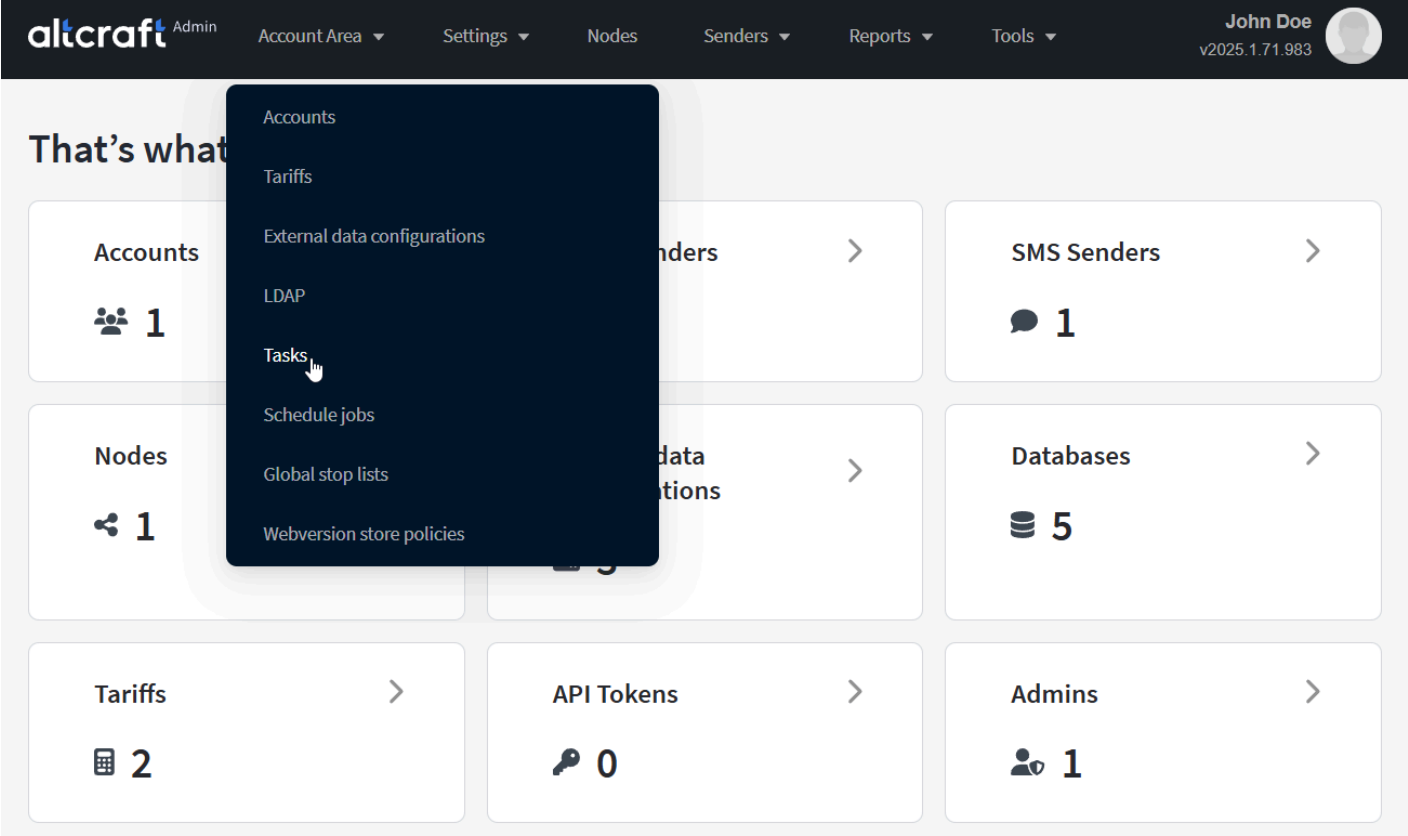
Tasks

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.

ID	Task	AccID / User	Machine ID	State	Created	Updated by	Actions
15f5829e-1822-42d3-a251-5624f6a7a585	Notify templates update	1 - com	Default master	completed 100%	03 Feb 2025 at 18:01	Dm Default master 03 Feb 2025 at 18:01	 
c8e77c61-9cb8-4994-beea-e57236c86523	WhatsApp templates update	1 - com	Default master	completed 100%	03 Feb 2025 at 18:01	Dm Default master 03 Feb 2025 at 18:01	 
b195978f-e2f1-4104-85b2-af2a146b354d	WhatsApp templates update	1 - com	Default master	completed 100%	03 Feb 2025 at 18:16	Dm Default master 03 Feb 2025 at 18:16	 
674fa089-8b4a-400b-9977-ed221dc89d89	Notify templates update	1 - com	Default master	completed 100%	03 Feb 2025 at 18:16	Dm Default master 03 Feb 2025 at 18:16	 
f2c5ea13-7264-49bd-a3d9-5f3674d63ca8	WhatsApp templates update	1 - com	Default master	completed 100%	03 Feb 2025 at 18:32	Dm Default master 03 Feb 2025 at 18:32	 

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.

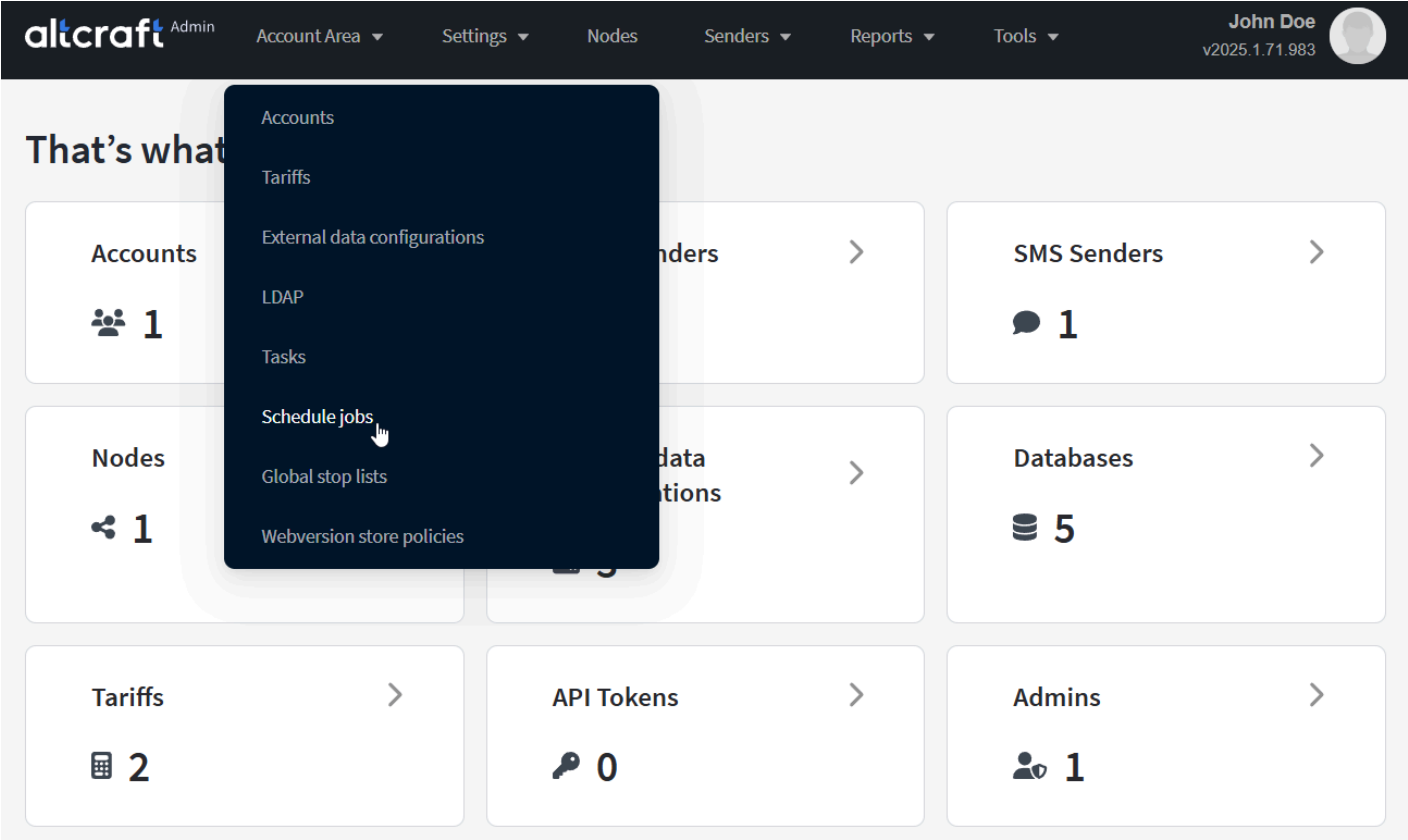
Schedule Jobs

Description

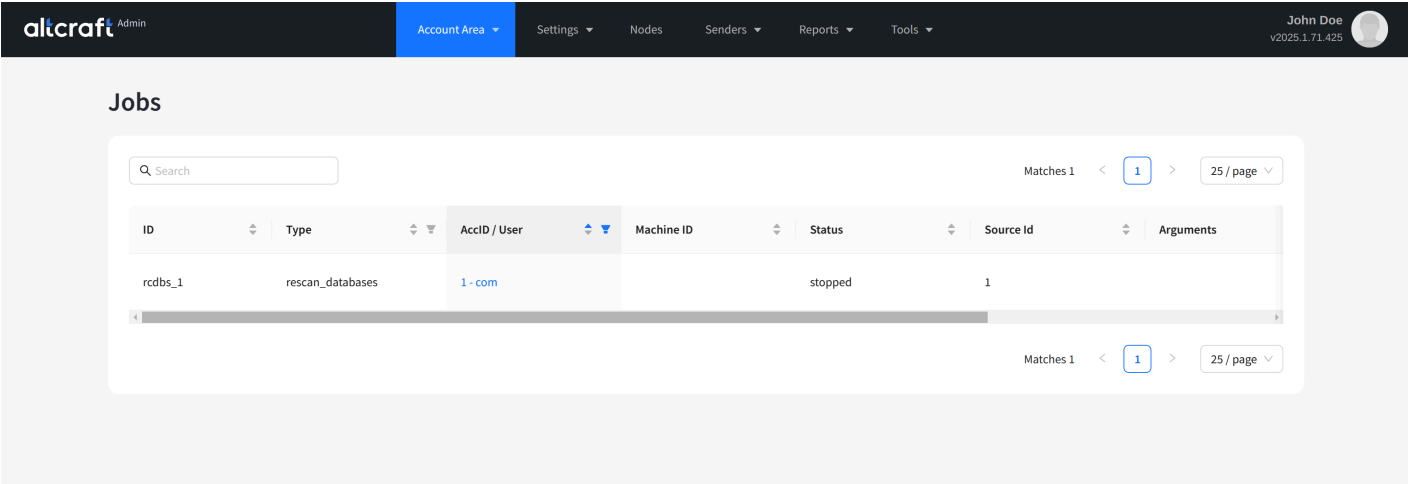
Scheduled jobs are processes launched periodically to support platform functionality. Schedule Jobs differ from Tasks in that they run as standalone applications, while Tasks are executed within the `proctask` service.

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 of four parts:

- **Suppress emails** — the **blacklist** of email addresses. These email addresses may be **valid**, but for various reasons sending to them is **blocked** for all platform accounts. This may include recipients who unsubscribed from all mailings, and resending messages to them increases complaint rates.
- **Hard bounce emails** — a list of **invalid emails** that returned 5xx class errors from mail servers — in most cases these mailboxes do not exist.
- **Suppress phones** — the **blacklist** of phone numbers. These phone numbers may be valid, but for various reasons sending SMS to them is blocked for all platform accounts. This may include recipients who unsubscribed from all mailings, and resending messages to them increases complaint rates.
- **Hard bounce phones** — a list of invalid phone numbers that returned errors when sending SMS — in most cases these numbers do not exist.

To view the stop lists, in the main menu select **Account Area — Global stop lists**:

altcraftAdmin

Account Area

Settings

Nodes

Senders

Reports

Tools

John Doe
v2025.1.71.425

Global stop lists

Suppress emailsHard bounce emailsSuppress phonesHard bounce phones

* Enter the email address you want to add or remove from the suppress list ?

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 are interested in into the input field, separating them by spaces or new lines. Then click **Check**:

* Enter the email address you want to add or remove from the suppress list ?

dog@example.com
cat@example.com
cockroach@example.com

Check

Result

Found:1 Not found:2

Add all not found

Remove all found

Emails	Status	
cat@example.com	Not found	+
cockroach@example.com	Found	-
dog@example.com	Not found	+

Matches 3

<

1

>

25 / page

You can remove or add entries to the list using the + and - buttons next to each item. The **Add all not found** button adds all values not found in the list, and the **Remove all found** button removes all values found in the list, allowing you to manage large numbers of addresses or phone numbers at once.

Webversion Store Policies

Description

A **web version storage policy** is a set of rules that defines the directory and retention period for web versions, as well as the channels where they will be stored.

Pre-configuration

S3 storage is used to store web version objects. You can connect any S3-compatible storage such as [Minio](#), [Yandex Cloud](#), or [Selectel](#). To connect, add the parameters to the [configuration file](#) `main.json` and restart the platform:

```
"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 available. To enable MongoDB storage, add the following parameter to the configuration file:

```
"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, click the policy identifier or name. To create a new policy, click **+ Create**:

alcraftAdmin

Account Area ▾

Settings ▾

Nodes

Senders ▾

Reports ▾

Tools ▾

John Doe
v2025.1.71.425

Web version policies

+ Create

Q Search

Matches 1 < 1 > 25 / page ▾

ID ▾	Name ▾	Bucket ▾	Channels ▾ ▾	TTL ▾ 🔍	Status ▾	Updated by	Actions
1	Default	adm1	All channels		OK	KZ Admin 15 Jan 2025 at 12:41	

Matches 1 < 1 > 25 / page ▾

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 name can include only lowercase Latin letters, numbers, dots (.) and hyphens (-). Length must be from 3 to 63 characters.
- **Prefix** (optional) — a string at the beginning of the object key name, used to organize data inside the bucket.
- The **Disable compression** option disables data compression within the policy.

Web version policies > Edit

* Name

* Bucket Name

Prefix

☐ Disable compression

In the **Channel filter** block, select the channels where web versions of messages should be saved: all channels (**Any channel**) or only specific ones (**Some channels**). Web version storage is available for all active channels in the account.

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.

Channel filter

☐ Any channel ☒ Some channels

* Channels

Email × SMS ×

▼

TTL Policy

☐ Not set / Use bucket defaults ☒ Set custom TTL rule

* TTL (days)

Save

Close

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 Interface](#) of the Altcraft marketing platform.

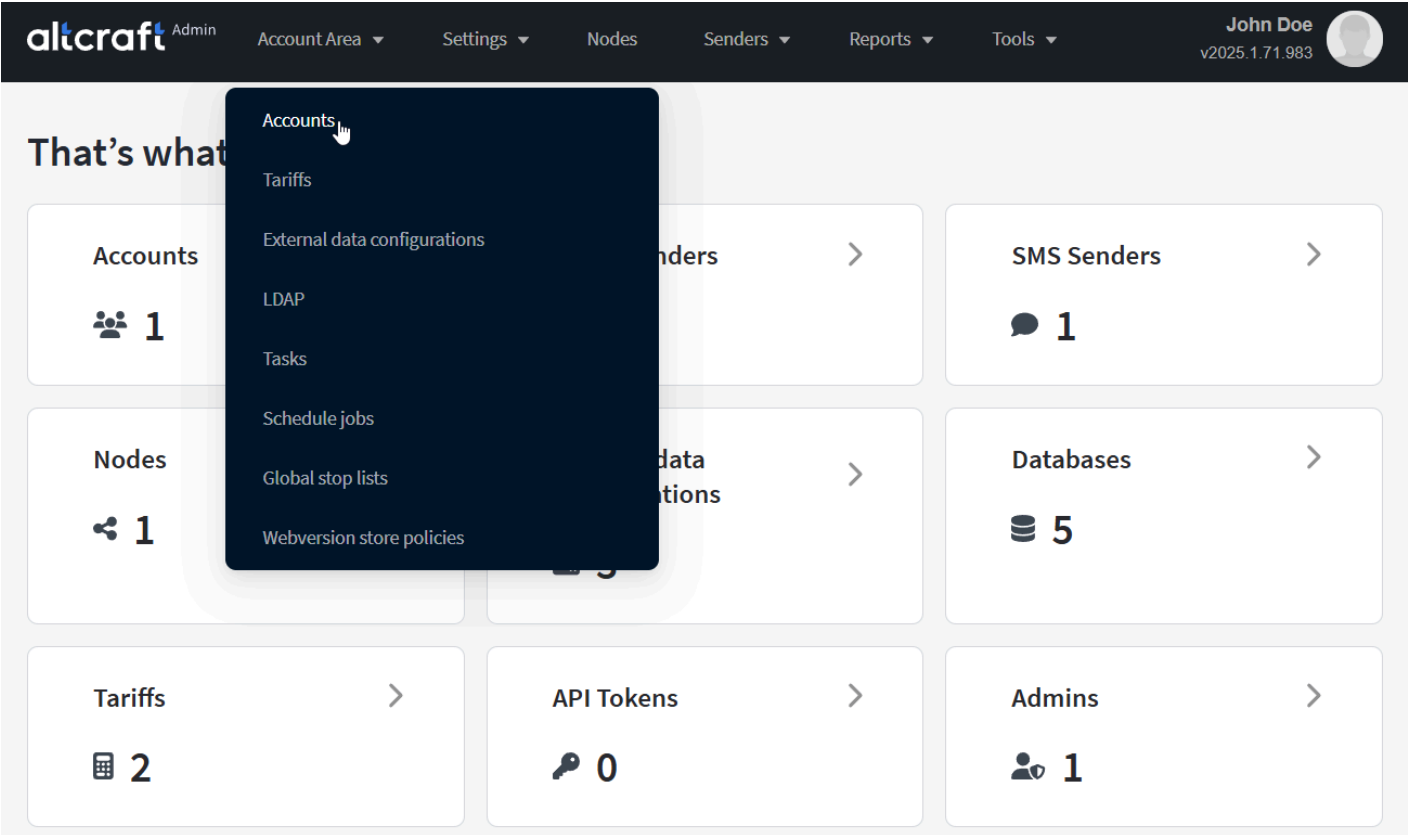
[Master users](#) of an account have access to all objects within the account's access groups, including consolidated statistics on sends and actions, as well as an audit of other users' activities.

An account uses shared internal and external platform resources:

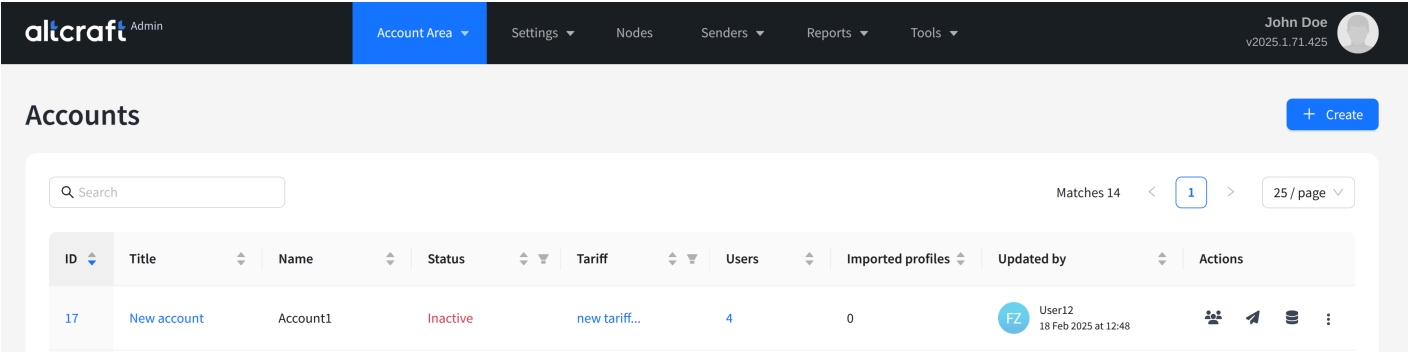
- [Tariffs](#)
- [Databases and Job Processing Server](#)
- [Senders](#)
- [Connectors to External Databases](#)
- [Connectors to External Directory Services \(LDAP\)](#)

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** toggle at the top of the screen.

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.

Accounts > Edit

☒ Active

* Title:

New account

* Login name 

* Tariff 

Not chosen

* Time zone 

Not chosen

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 basis for segments and reports.
- **External data** — SQL connectors for integration with external infrastructure.

Account databases

Postgres DB 

Not chosen

ClickHouse DB (cache) 

Not chosen

External data 

NOTE

When deploying platform components in a cluster to implement a [fault-tolerant operation scheme](#), configuration records must be created for each MongoDB shard and each SSDB replica, and added to the account.

Web Version Settings

In this section, specify the **web version storage policy** for the account. Channels and retention periods are configured when creating a [separate policy](#). Here you can also enable **saving web versions of all email messages**:

Web version

Web version policy [?](#)

No policy: web versions disabled



Always save email web versions [?](#)

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 sending.
- **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](#).

Additional options



Enable AK Pixel [?](#)



Enable push channel [?](#)



Enable smart settings [?](#)



Capture email replies [?](#)

Custom pixel and pushes domain

pxl.your-pixel-domain.com

Notifications

Additionally, you can set the day of the week for sending weekly email reports to account users (**Send weekly reports on**) and the hour for daily reports (**Send daily reports at**):

Notification

Send weekly reports on [?](#)

Monday



Send daily reports at [?](#)

7



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 into the platform with the appropriate set of [groups and access roles](#).

TIP

A step-by-step guide for configuring LDAP access to Altcraft is available [here](#).

Authorization



Force 2fa for users [?](#)



Use LDAP [?](#)

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 be sent to account users.
- **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:

Accounts > #17

☒ Active

☐ Frozen ?



* Title:

New account

Or from the account list using the quick action buttons on the right side of the table:

Updated by		Actions			
	User12 18 Feb 2025 at 12:48				

Account Users

Managing account user accounts is available both in the Altcraft Admin Panel and in the [User Interface](#). This article covers the Admin Panel functionality.

To manage users, in the list of accounts click the quick action button:

Accounts

Search

Matches 15 < 1 > 25 / page

ID	Title	Name	Status	Tariff	Users	Imported profiles	Updated by	Actions
18	Main Account	main	Active	1usr	1	0	JD John Doe 06 Mar 2025 at 15:19	Users

Or open the account editing menu and click the **Users** button:

Accounts > #18

Active

Frozen

Title:

Main Account

Login name

main

The list of users will open displaying existing accounts. On the right side of the table, there are buttons for **logging in as the user**, **editing**, and **deleting accounts**. You can also click the user account identifier on the left to edit. To create a new account, click **+ Create**:

Users

Search

Matches 2 < 1 > 25 / page

Username	Name	Email	Active	Master	Last login date	Last login IP	Actions
smitha	Angelo Smith	asmith@example.com	Yes	Yes	01 Jan 0001 at 02:30		
brianadams	Brian Adams	brianadams@example.com	Yes	Yes	01 Jan 0001 at 02:30		

Matches 2 < 1 > 25 / page

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@accountname).
- **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).

Account #18 > Users > Edit

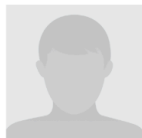
☒ Active

Main settings

Alerts

Safety

Access



Upload avatar

* Login

Login

* First Name

New

* Last name

User

* Contact email address

example@example.com

Contact phone number

+70000000000

Language

Russian Pro (RU)

Delete user

Save

Close

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 select which information the user will receive:

- **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.

Account #18 > Users > Edit

☒ Active

Main settings

Alerts

Safety

Access

Email for alerts

example@example.com

☒ Exhaustion of limits

☒ Subscriber complaints

☒ Hard bounces

☐ Soft bounces

☐ Daily reports

☒ Weekly reports

Save

Close

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 saving.
- **User must change password on next login** — when enabled, the user must change the password at the next login.

At the bottom of the screen, you can view the user’s login history in the platform showing login date, browser, IP, and location:

Account #18 > Users > Edit

☒ Active

Main settingsAlertsSafetyAccess

* Set Password ?

Generate

☐ Send an email with access to the user

☐ User must change password on next login

Login date	Device	Browser	Location	IP
No data				

SaveClose

Access Settings

For each user, you can configure access to specific account objects and Altcraft features. More about access rights is described [in this article](#).

- **Assigned roles** — assign access Roles to the user for account objects. Roles are created and edited in the master settings of the User Interface.
- **Assigned groups** — grant the user access to Groups of account objects. Master users can create object groups when editing any account object.
- **Master user** — grant the user access to the Main Group of the account, management of other users, and master settings.
- **Is LDAP user** — use the [LDAP protocol](/en/admin-guide/admin-panel/account-area/ldap) for user authentication. To enable this, the corresponding setting must be active in the account.

Account #18 > Users > Edit

☒ Active

Main settings

Alerts

Safety

Access

* Assigned groups

All groups



Assigned roles



☒ Master user

☐ Is LDAP user

Save

Close

Account Virtual Senders

Description

Virtual senders group the platform's [real senders](#) and set rules and proportions for sending through them. Managing account user accounts is available both in the Altcraft Admin Panel and in the [User Interface](#). This article covers the Admin Panel functionality.

NOTE

Users can select a virtual sender in mailing [Smart settings](#). To enable this, the **Smart settings** option must be activated for the account. If Smart settings are not activated, the first matching virtual sender will be used for mailings.

Creating, Editing, Deleting

To manage virtual senders, in the list of accounts click the quick action button:

Accounts

+ Create

Q Search

Matches 15 < 1 > 25 / page v

ID	Title	Name	Status	Tariff	Users	Imported profiles	Actions
18	Main Account	main	Active	1usr	2	0	

Or open the account editing screen and click the **Virtual senders** button:

Accounts > #18

Active

Frozen ?

Virtual senders

* Title:

Main Account

Login name ?

main

The list of existing virtual senders will open. On the right side of the table are the **edit** and **delete** buttons. You can also click the virtual sender identifier on the left to edit. To create a new virtual sender, click + **Create**:

Virtual Senders

+ Create

Q Search

Matches 1 < 1 > 25 / page v

ID	Name	Channels	Updated by	Actions
1	MainVirtualSender		JD John Doe 18 Mar 2025 at 11:52	

General Settings

In the general settings, specify a unique **Name** for each virtual sender and assign the [account object Groups](#) where the virtual sender will be available (**Groups**):

Account #18 > Virtual Senders > Edit

*

Name:

Description

Enter a description

Groups

?

▼

Email Sender Rules

Email sender rules determine which real senders and in what proportions will be used under certain conditions. To add a rule, click **+ Add rule**.

Rules for email senders ?

Email rules are not configured for the virtual sender

+ Add rule

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

akmta ▼

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 always be last. You can adjust the priority of other rules using the **Move up** and **Move down** buttons. To delete a rule, click **Remove rule**.

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 field for which the index was created.

Creating, Editing, Deleting

To manage indexes, in the list of accounts click the quick action button:

Accounts

Q Search

Matches 15 < 1 > 25 / page

ID	Title	Name	Status	Tariff	Users	Imported profiles	Actions
18	Main Account	main	Active	1usr	2	0	Account Databases

Or open the Account for editing and click the **Databases** button:

Accounts > #18

Active

Frozen

Account databases

Title

Main Account

Login name

main

A list of all account databases containing customer profiles will open. For each database in the list, information about the number of customer profiles, creation time, and last modification time is displayed. To manage indexes, click the lightning-shaped button on the right side of the table:

Account Databases

Q Search

Matches 1 < 1 > 25 / page

ID	Name	Total number	Updated by	Actions
1	New galactic database	1	smitha 18 Mar 2025 at 12:23	Edit

Matches 1 < 1 > 25 / page

A list of all database field indexes will open. Most of them are system indexes and cannot be managed. They are linked to fields such as email, ID, registration date, and others. To add a new index, click **Create**:

Database indexes - #1 New galactic database

+ Create

Some indexes are created, or re-created automatically on system start. There indexes can't be managed

Matches 42 < 1 2 > 25 / page

Name	System	Unique	Actions
subscriptions.status_1_subscriptions.is_del...	Yes	No	
subscriptions.resource_id_1_subscriptions....	Yes	No	
subscriptions.resource_id_1_subscriptions....	Yes	No	
subscriptions.resource_id_1_subscriptions....	Yes	No	

In the modal window that opens, select the required database fields and add indexes:

Creating an index



* Select fields

☐ Reverse index

☐ Unique index

☐ Add primary xxh key

Close

Save

The following options are available for an index:

- Reverse index** — indexes for records are set in reverse order. Used when reverse search occurs more frequently than direct search.
- Unique index** — guarantees that each record will have its own unique index. It will also be impossible to create multiple profiles with the same field value.
- Add primary xxh key** — adds a history identifier to the index. Used when searches by this field are often performed along with profile activity history.

Databases

Description

Databases is a section in the Altcraft platform designed for managing connections. In this section, you configure connectors to various databases used for storing data about orders, products, user statistics, and reports.

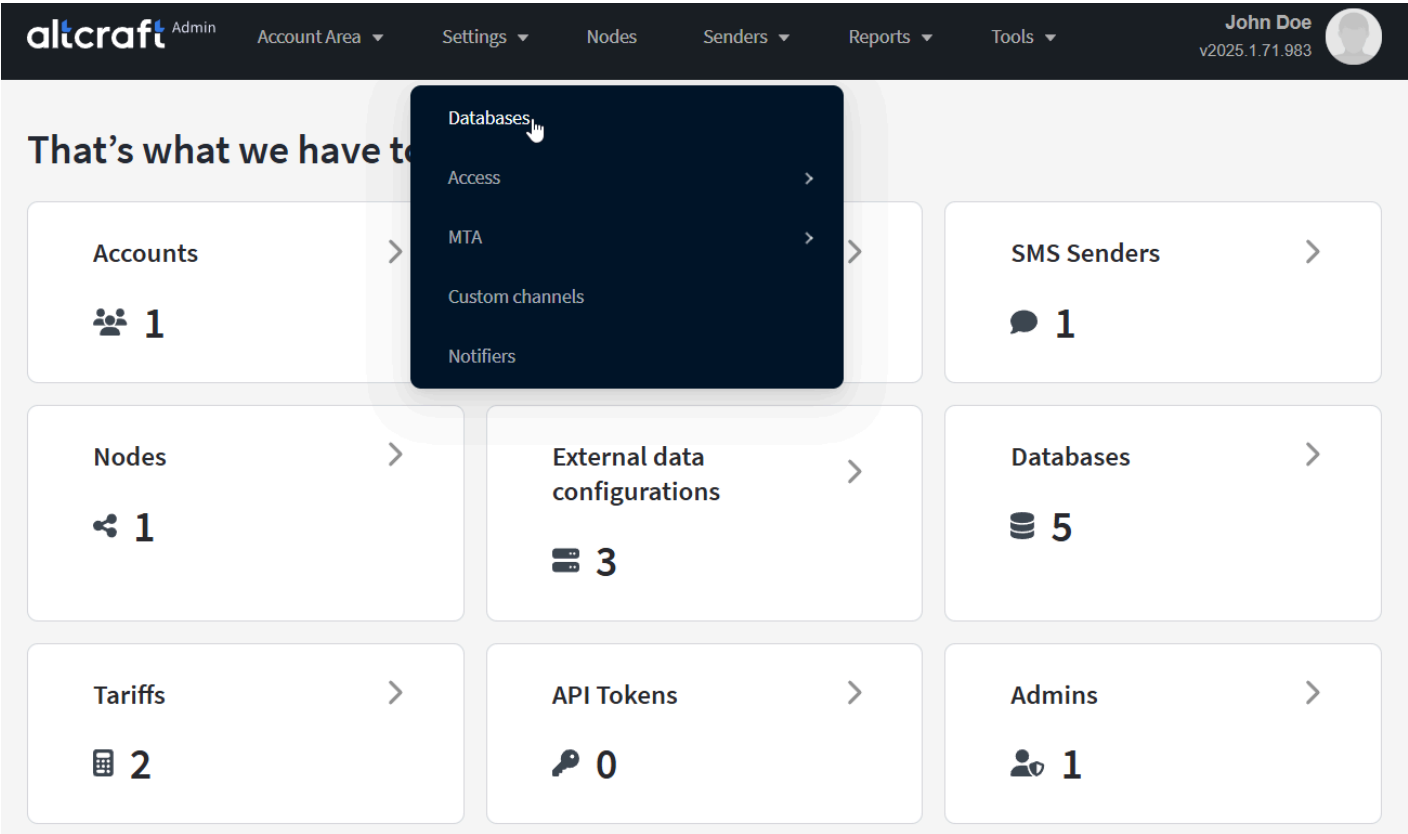
Database Types

For each account in the Altcraft marketing platform, you can configure connectors to the following databases:

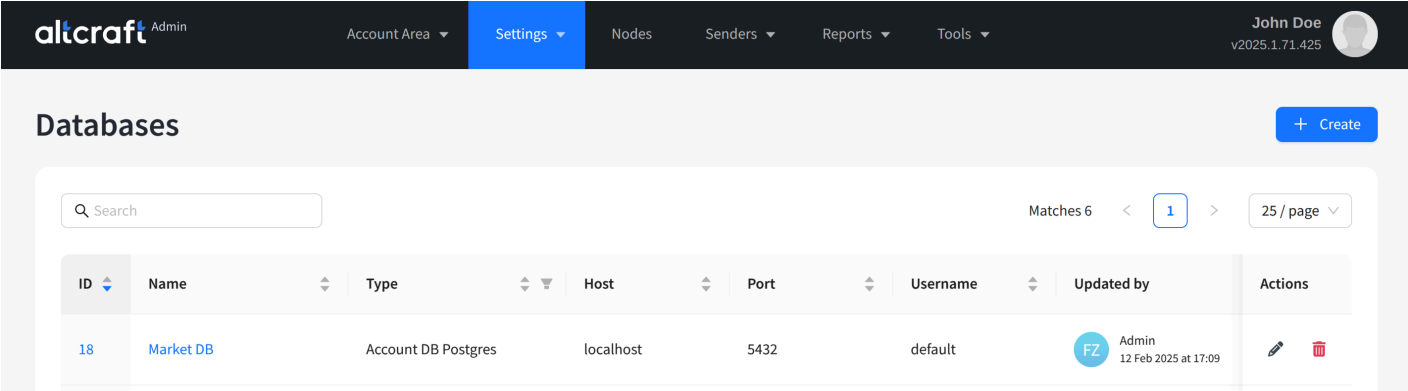
- **Account DB Postgres** — used by the **Market** module for storing data about products and orders.
- **Account DB ClickHouse** — used to store data about [profile relations](#) and cache queries to external databases.

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. You can also edit a connector by clicking its identifier or name on the left. To create a new connector, click **+ Create**.



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 password is not used for local database deployments.

Databases > Edit

* Name

New DB

* Type

Account DB ClickHouse

* Host

localhost

* Port

9000

Test connection

* Username/Server token

default

Password/Secret

☐ With SSL

Save

Close

For ClickHouse databases, you can enable **SSL** support. To do this, activate the **Via SSL connection** parameter and fill in the necessary fields:

☒ With SSL

* SSL certificate

-----BEGIN CERTIFICATE-----

* SSL key

-----BEGIN RSA PRIVATE KEY-----

* SSL CA certificate

-----BEGIN CA CERTIFICATE-----

Save

Close

NOTE

When deploying platform components in a cluster to implement a [high-availability deployment scheme](#), you must create configuration records for each MongoDB cluster shard and add records to the account.

Notifiers

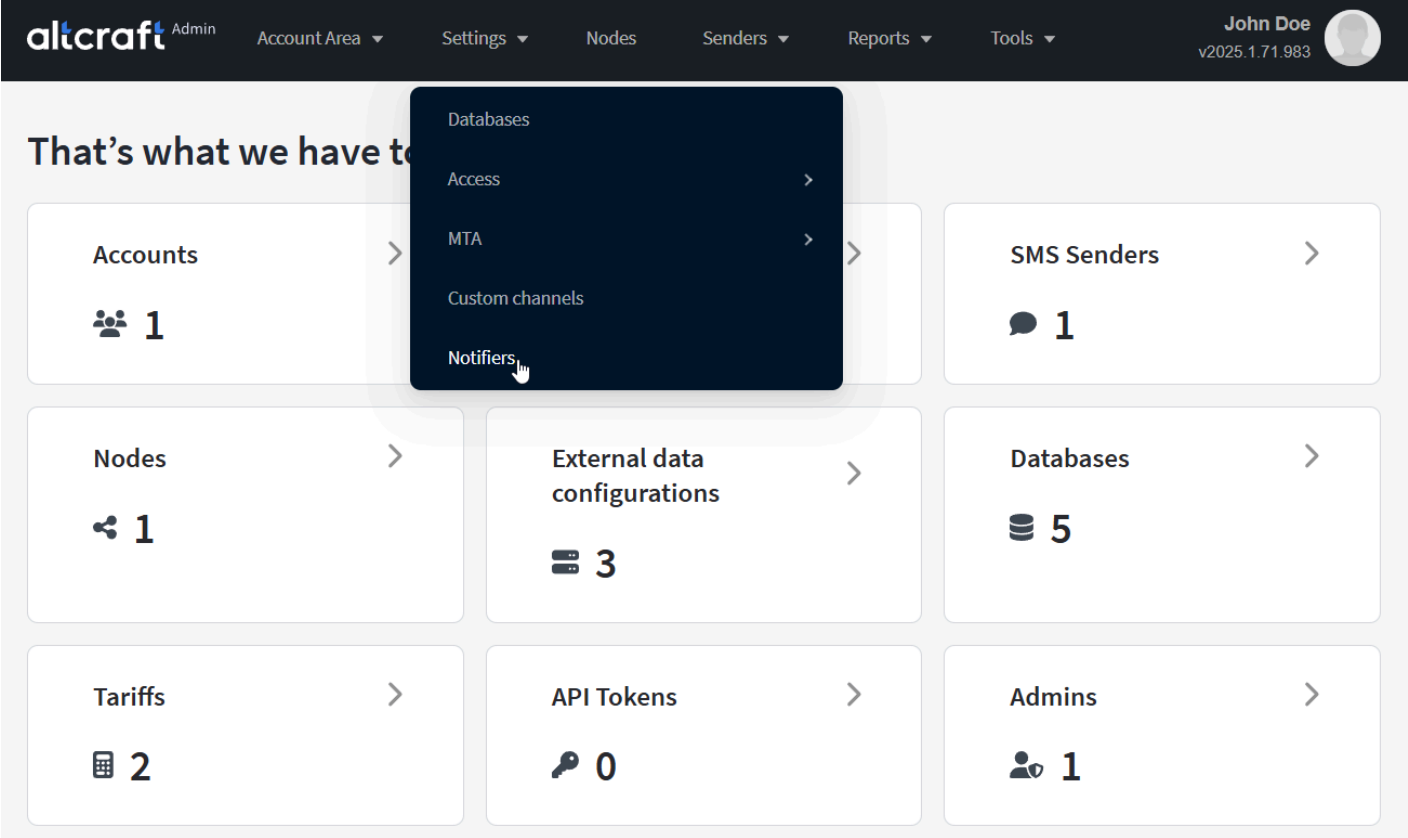
Description

Notification Service is a mechanism in the Altcraft platform designed to automatically inform administrators about issues with message delivery.

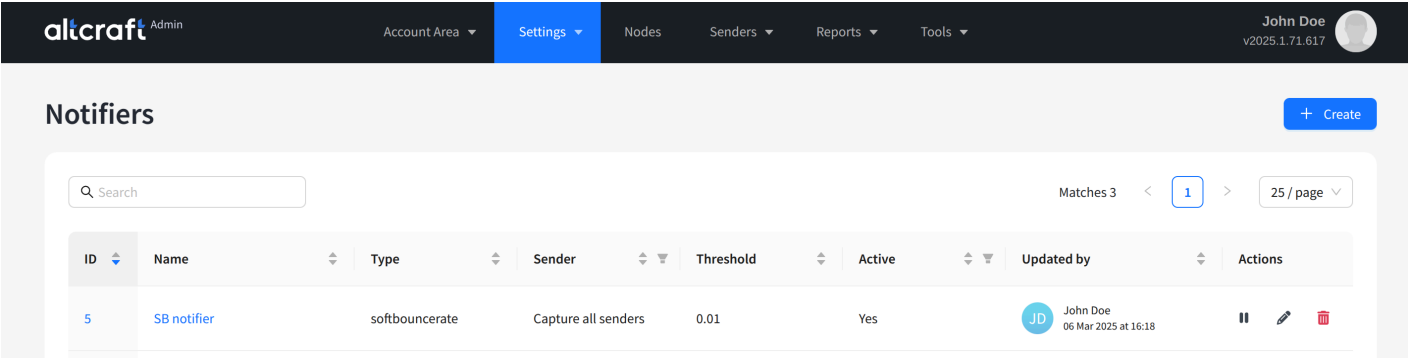
It allows you to promptly respond to failures and ensure the stable operation of the platform.

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 delete buttons, as well as the activation/deactivation button. You can also edit an object by clicking its identifier or name on the left. To create a new object, click **+ Create**:



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 rate exceeds the specified threshold;
 - **When Soft bounce rate is higher than threshold** — notification will be sent if the Soft bounce rate exceeds the specified threshold;
 - **When Complaints rate is higher than threshold** — notification will be sent if the complaints rate exceeds the specified threshold;
 - **When Undelivery rate is higher than threshold** — notification will be sent if the undelivery rate exceeds the specified threshold;
 - **When Performance rate is lower than threshold** — notification will be sent if the performance rate drops below the specified threshold;
- **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

* **Minimum actions**

10

Actions

* **Send notification emails**



* **Freeze interval (minutes)**

120

Save

Close

Admins

Description

Administrator in Altcraft is an account in the Admin Panel. Administrator records are created separately from user accounts of the platform. However, each administrator can access any account within the platform.

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**.

altcraftAdminAccount Area▼Settings▼NodesSenders▼Reports▼Tools▼John Doe v2025.1.71.983

That's what we have to

Accounts1

Nodes1

Tariffs2

Databases5

API Tokens0

Admins1

Databases

Access

MTA

Custom channels

Notifiers

Admins

API tokens

Senders1

Databases

Admins

The opened list displays existing administrator accounts. On the right side of the table, there are **edit** and **delete** buttons. You can also edit by clicking the identifier or name of the administrator account (on the left). To create a new account, click **+ Create**:

altcraftAdminAccount Area▼Settings▼NodesSenders▼...John Doe v2025.1.71.425

Admins

+ Create

Search

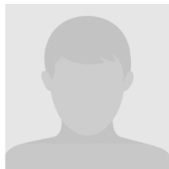
Matches 18125 / page

ID	Username	Master	Last login date	Updated by	Actions
26	Mikhail_admin	No	28 Feb 2025 at 16:02	t2 Mikhail_admin 28 Feb 2025 at 15:02	editdelete

The following parameters are configured for an administrator account:

- **Username** and **Password** — username and password. These credentials are used to log in to the Admin Panel.
- **First name** and **Last name** — administrator's first and last name.
- **Contact email address** — administrator's contact email.
- **Language** — Admin Panel interface language.

Admins > Edit



Upload avatar

* Username

* Password

↺

* First Name

Ivan

* Last name

Ivanov

* Contact email address

example@example.com

* Language

English

▼

☐ Master administrator ?

☐ Is LDAP user

Save Close

Select the **Master administrator** option if you want to allow the administrator to manage other accounts in the Panel.

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, open the **Accounts** list and go to the list of users:

Accounts

+ Create

Q Search

Matches 15 < 1 > 25 / page ▼

ID	Title	Name	Status	Tariff	Users	Imported profiles	Updated by	Actions
18	Main Account	main	Active	1usr	0	0	JD John Doe06 Mar 2025 at 15:19	Users 🚀📄⋮

Click the rocket button to access the selected Altcraft user account:

Users

+ Create

Q Search

Matches 1 < 1 > 25 / page ▼

Username	Name	Email	Active	Master	Last login date	Last login IP	Updated by	Actions
smitha	Angelo Smith	asmith@example.com	Yes	Yes	01 Jan 0001 at 02:30		👤 06 Mar 2025 at 15:22	Log in 🚀📄🗑️

Matches 1 < 1 > 25 / page ▼

NOTE

Your web browser must allow pop-ups.

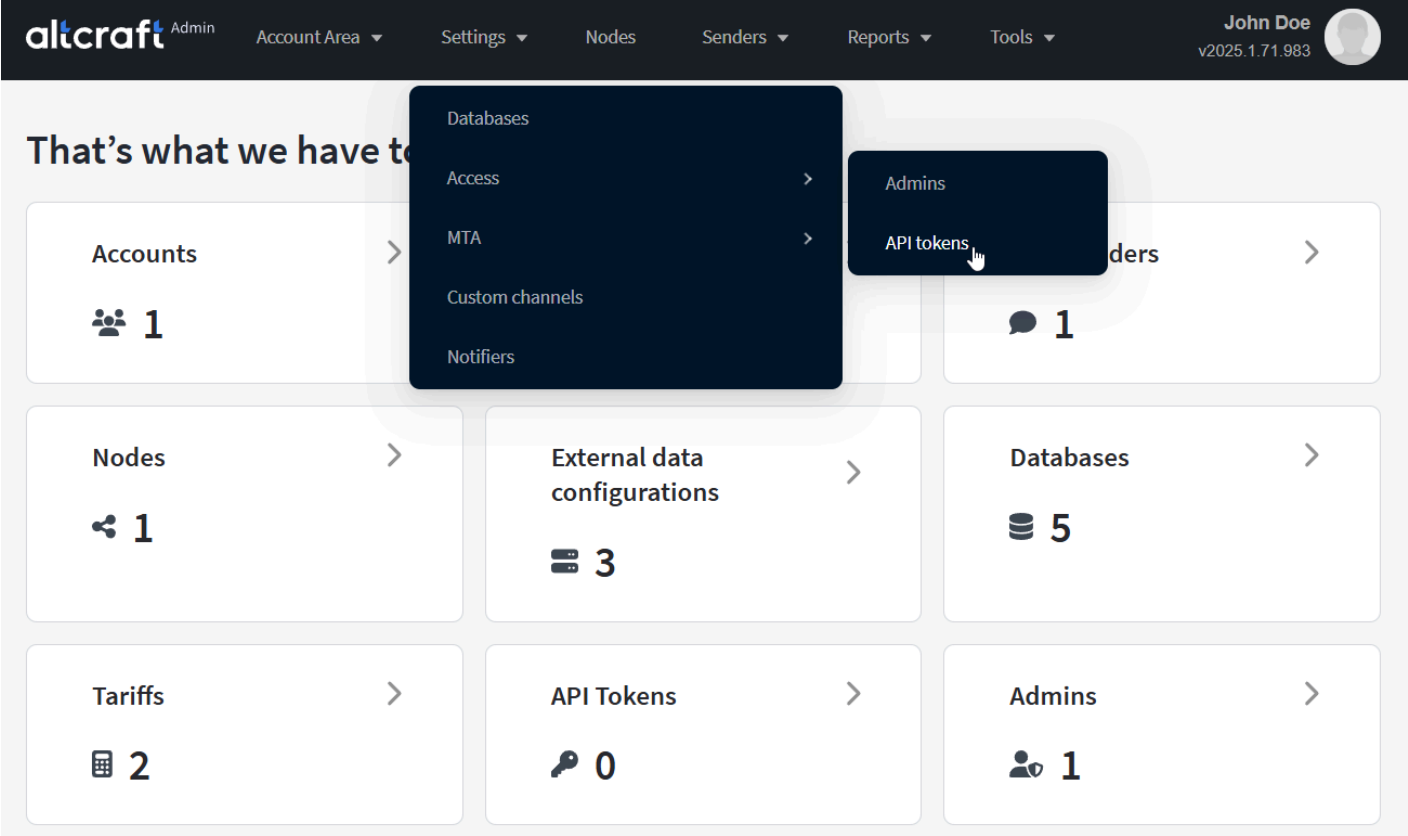
API tokens

Description

API tokens are tools used for API requests, mainly related to managing senders (message delivery agents). The library of administrator API calls is available [at this link](#).

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. You can also edit by clicking the identifier or name of the token (on the left). To create a new token, click **+ Create**.

NOTE

The secret part of the token used in API requests is displayed in the "Token" column of the table.

altcraftAdmin

Account Area

Settings

Nodes

Senders

...

John Doe
v2025.1.71.617

API Tokens

+ Create

Q Search

Matches 5

<1>

25 / page

Token	Name	Active	Updated by	Actions
c12376991331458c8c20439c6f0ff6b3	Token	Yes	JD John Doe 06 Mar 2025 at 15:49	
39203e675dfc497c9f3ab9a852e40343	API token main	Yes	JD John Doe 06 Mar 2025 at 15:49	

You can set a unique name for the token and define whether the token is active or not. Using multiple administrative tokens is justified if you need to disable some while keeping others active.

API Tokens > Edit

☒ Active

* Name

Save

Close

MTA

Description

AKMTA (Altcraft Mail Transfer Agent) is an email sender developed by the Altcraft team and included in the Altcraft Platform. Altcraft MTA allows you to flexibly configure the sending speed of messages and the behavior when receiving negative responses from email providers in order to 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 simultaneously track sending statistics and perform SPF alignment.

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 bottlenecks. One solution for optimizing message delivery is to implement a caching DNS server.

When the system initiates a DNS query to convert a domain name into an IP address, it usually relies on the default DNS resolver provided by the operating system or network. Although these resolvers are generally reliable, they may not be optimized for high-volume message delivery. High-volume sending often requires frequent DNS lookups to resolve recipient domain names. Default resolvers can inefficiently cache these query results or not cache them at all, leading to redundant requests for the same information. This inefficiency can result in increased latency, network overload, and a higher risk of service disruptions.

By storing previously resolved domain-to-IP mappings, DNS caching servers reduce the need for redundant queries. This not only speeds up the resolution process but also saves bandwidth and minimizes load on upstream DNS servers. Additionally, DNS caching servers can enhance security by verifying 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/` directory, for example `/etc/unbound/unbound.conf.d/default.conf`. You can delete the configuration file created during installation in that directory and create a new one, or modify it.

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 maintained to monitor the mail agent's operation, analyze its performance, and troubleshoot delivery issues. Logs help administrators track how the system processes each sending step, identify negative 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 understand the current state and the processing history of each message.

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@gmail.com','116.69.16.162','gmail-smtp-in.l.google.com.','500 dial tcp 231.127.65.9:0->116.69.16.162:25: i/o timeout','undefined','DIAL','3','68','7','0','abcdefghijklmnoP-123QrstuvwXYZ','asikjbyrinnvjfikdig123-dfgretyhvvbvasdaa''<nil>'
```

- 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.

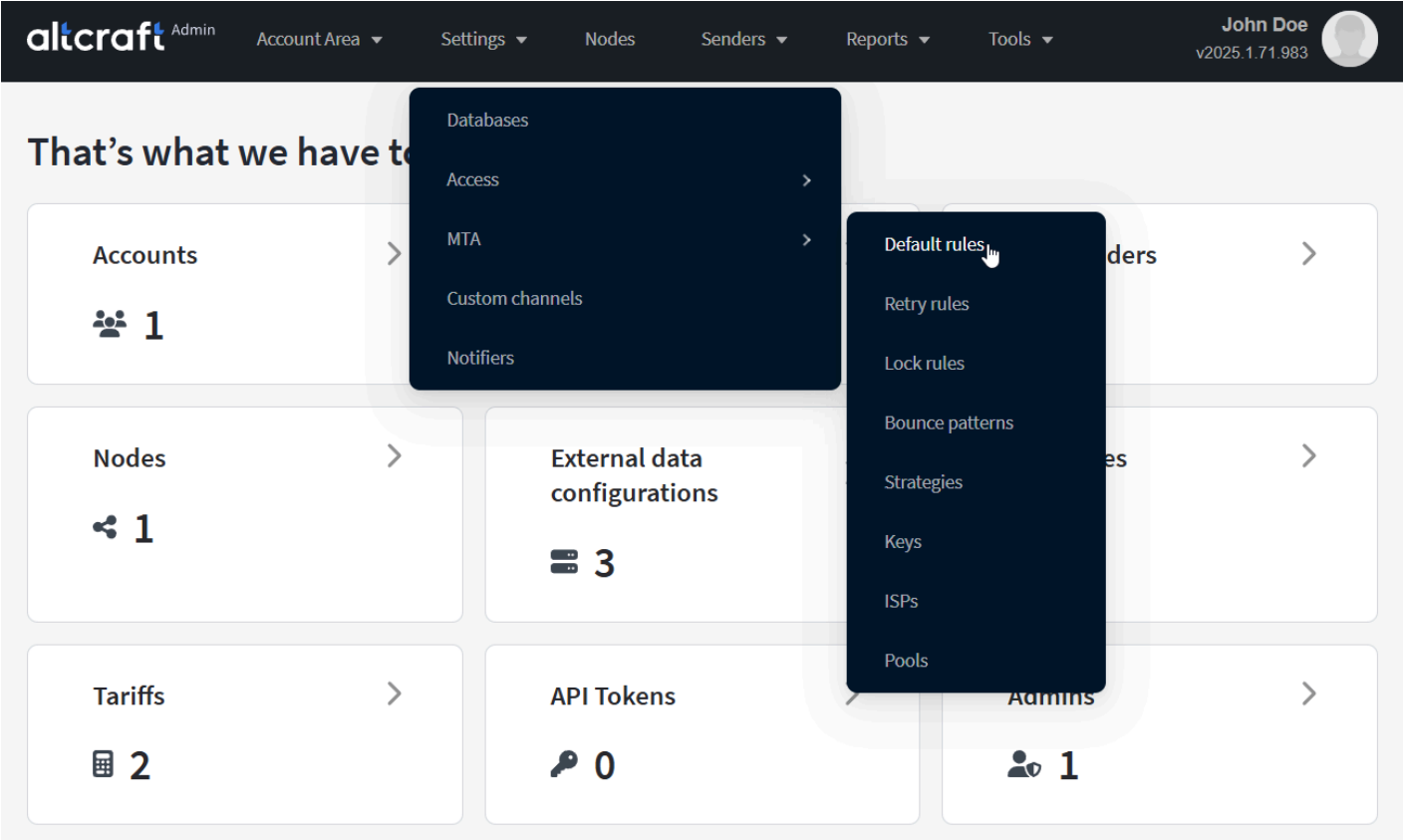
Default rules

Description

Default rules are configuration records that define AKMTA sending scenarios when receiving negative responses from all or specific ISPs, as well as from certain groups of ISP domains.

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.

AKMTA default rules

Retry rules

Lock rules

Default retry rules

Bounce patterns

47 - socket\stimeout × 48 - socket\terror ×
49 - operation\sreturned\s*\stimeout.*expired × 50 - greylist ×
420 - .*ry\sagain\slater.* × 421 - ^([42|45])[0-9]\s ×



Rule

1 - Geometric Retry 15 30 60 120 240 480 min

× Remove

- **Lock rules** — scenarios that use sending lock rules.

AKMTA default rules

Retry rules

Lock rules

Default lock rules

Bounce patterns

558 - 540.*recipient\saddress\srejected.*Inactive ×



Rule

1 - Lock ISP on IP for 30 min

× Remove

Add bounce patterns

Most MTA behavior scenarios for negative ISP server responses are already configured by Altcraft specialists. You can modify existing patterns and add new ones. If any data is outdated, please let us know: support@altcraft.com.

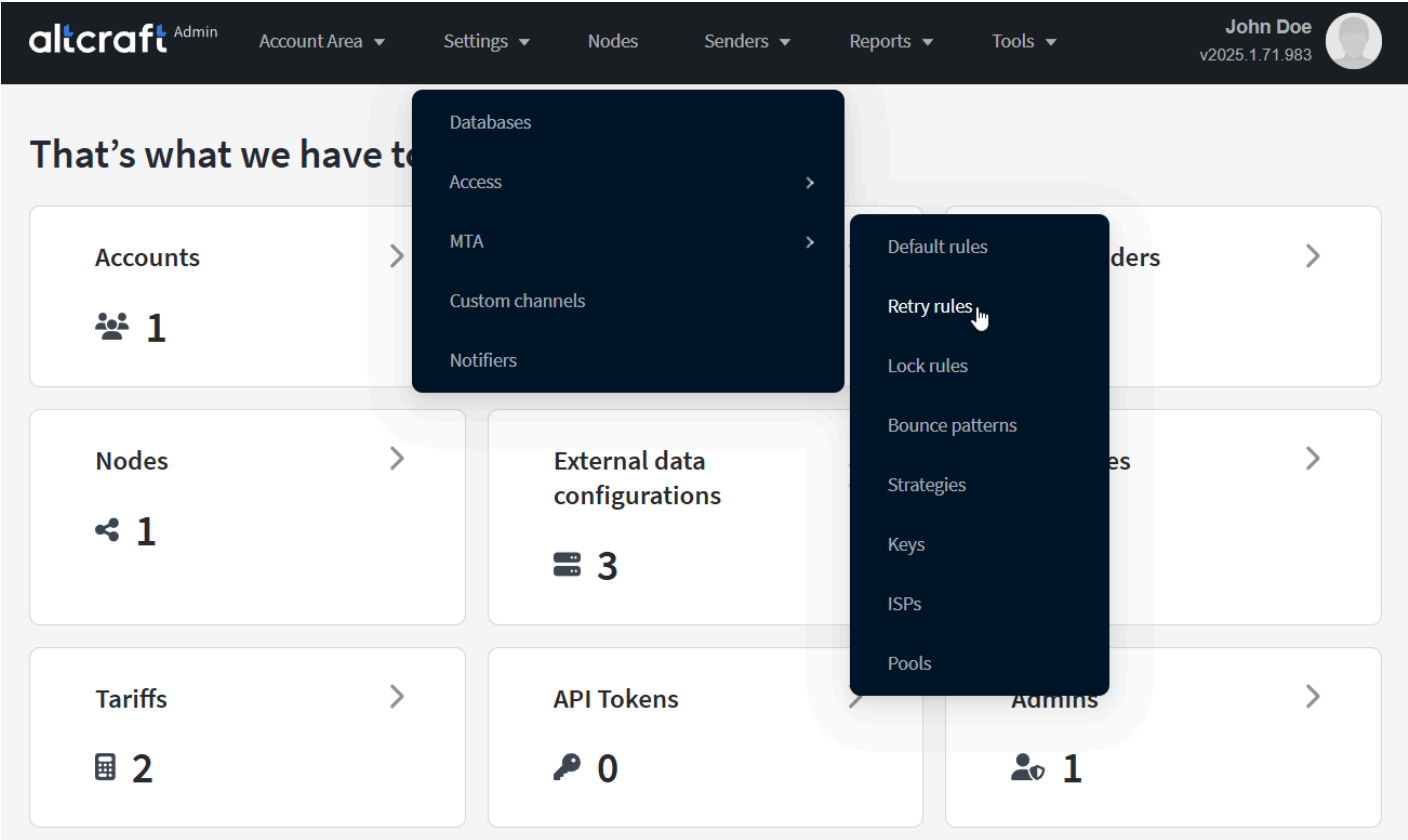
Retry rules

Description

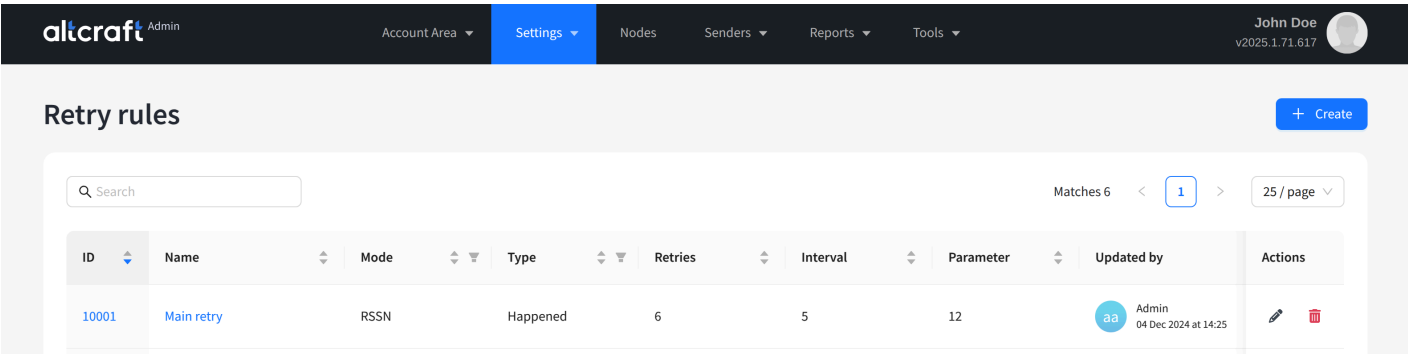
Retry rules are a tool used in cases when the ISP server responds that the recipient's mailbox is temporarily unavailable.

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** buttons. You can also edit a rule by clicking the identifier or name on the left. To create a new rule, click **+ Create**:



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 scenarios.
- **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 multiplied by (1 + Random modifier).

Retry rules > Edit

* Name

Mode

Retry from the same IP of the same node

▼

Type

Geometric increase delay

▼

* UserNumber of retries

6

* Time interval (min)

5

* Exponent

2

Random modifier

12

Save

Close

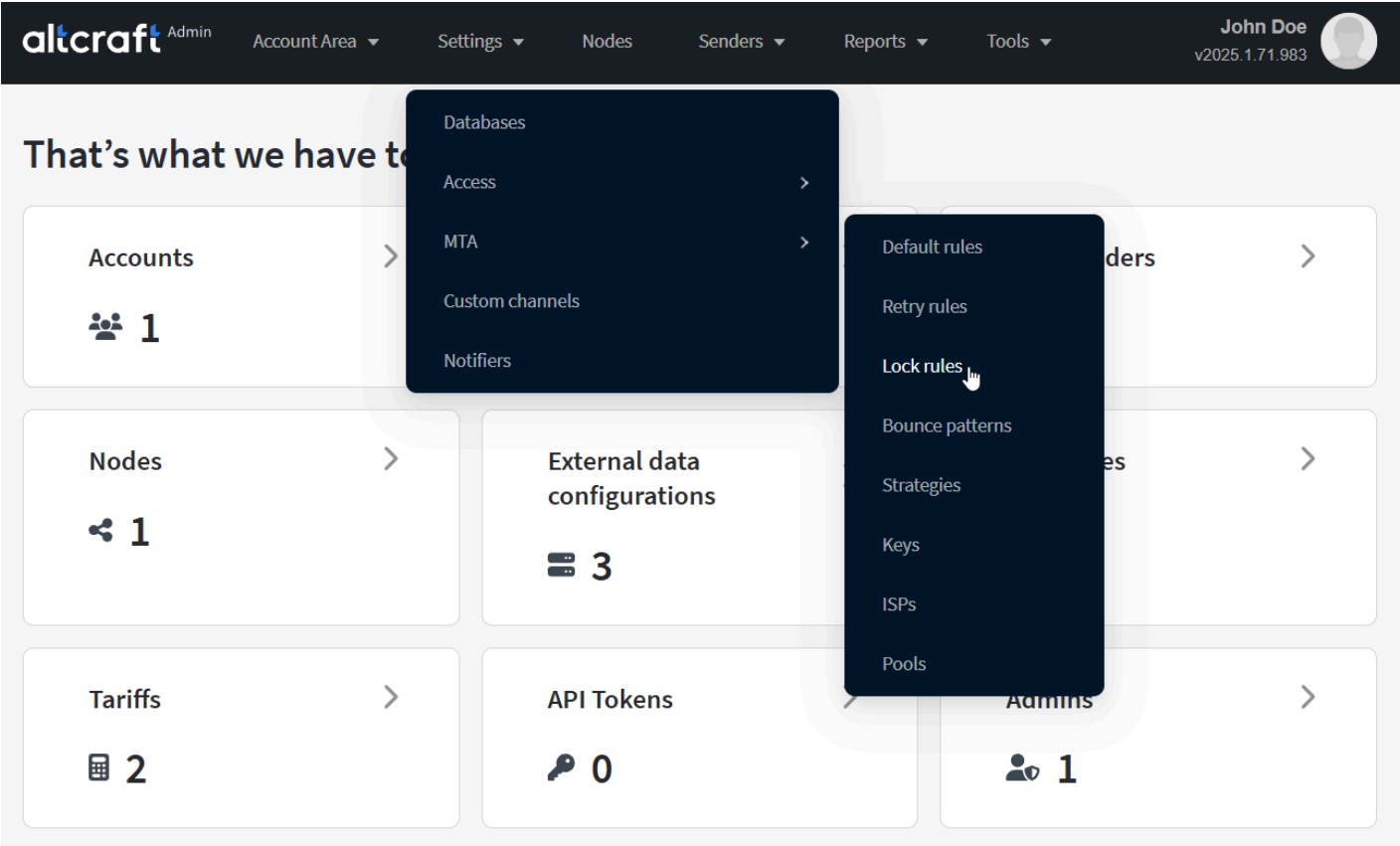
Lock rules

Description

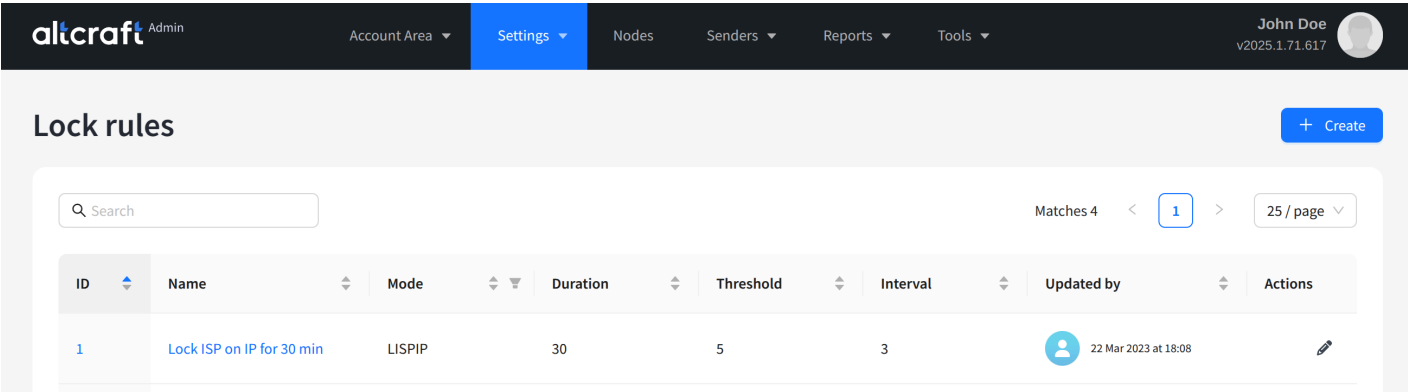
Lock rules are a tool that temporarily prohibits sending messages to a domain group of specific ISPs.

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** buttons. You can also edit a rule by clicking the identifier or name on the left. To create a new rule, click **+ Create**.



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 domains.
- **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

* Time interval (min)

5

Save

Close

Bounce patterns

Description

Bounce patterns are templates of responses from email providers (ISPs) about delivery problems. AKMTA analyzes these responses and classifies them to determine the reasons for delivery failures and correctly handle unsuccessful sends.

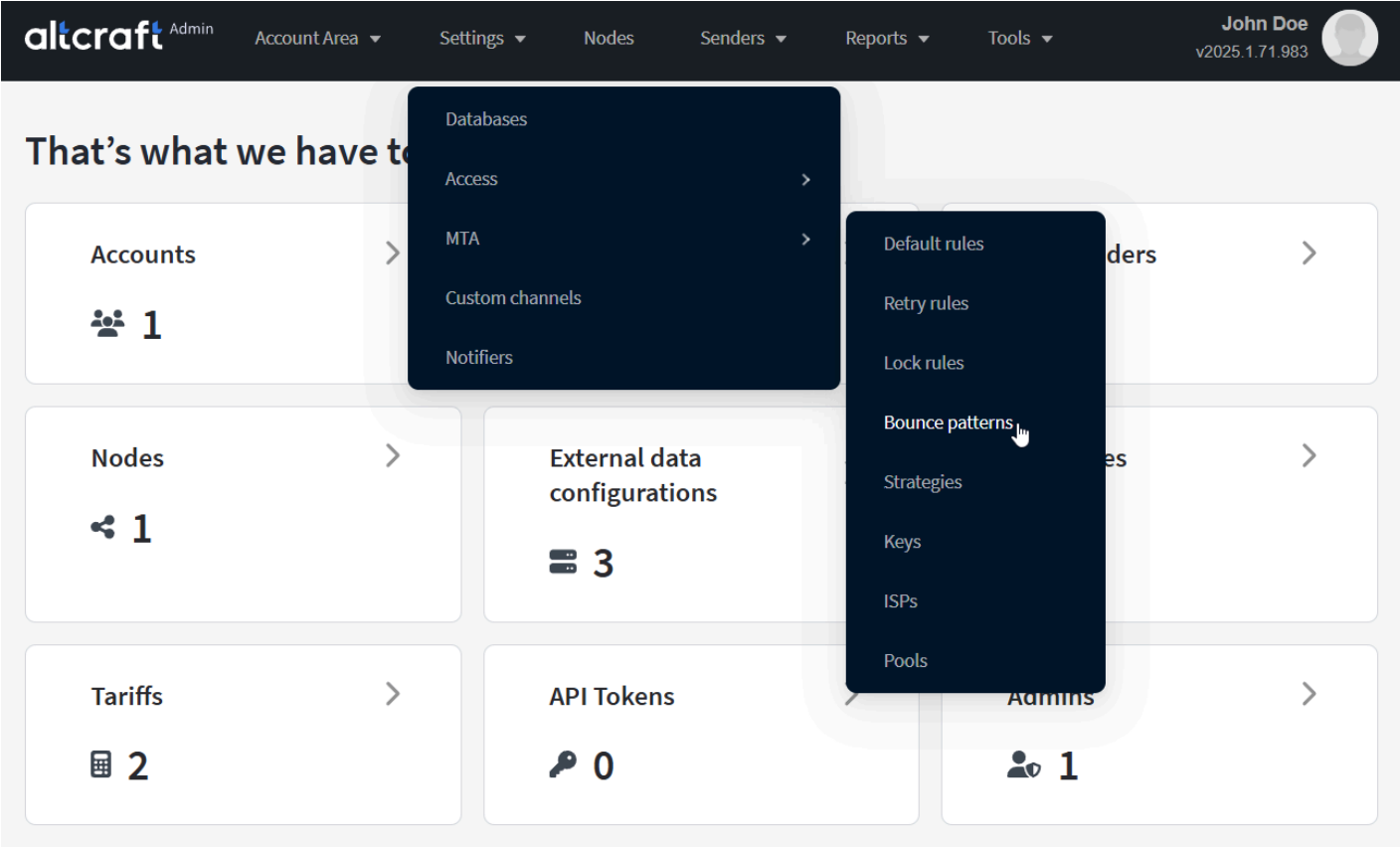
Types of negative ISP responses

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 exist. Sending to such addresses is blocked to avoid damaging the domain and sender IP reputation. The customer's email subscription is marked as **Hard bounced**, preventing further delivery attempts.
- **Soft bounce (SB)** contains client-class errors (**4xx**), indicating **temporary unavailability** of the mailbox. In such cases, retry or temporary blocking scenarios can be applied.
- **Bounce (B)** — this type includes **other delivery errors** that are not Hard or Soft bounce. Typically, retrying delivery is not advisable, and the message is marked as **Undelivered**.

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 already exist in the default AKMTA configuration. To **edit** an existing entry, click the quick action button on the right side of the list or the entry identifier on the left. To create a new entry, click **+ Create**.

CAUTION

Only **custom** ISP response templates can be deleted.

Bounce patterns

+ Create

Search

Matches 375 < 1 2 3 ... 15 > 25 / page

ID	Type	ISP	Match String	Message	Updated by	Actions
1	HB		DNS\sname\sdoes\snot\sexist	DNS name does not exist	17 Apr 2024 at 16:57	

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 any provider.
- **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

Save

Close

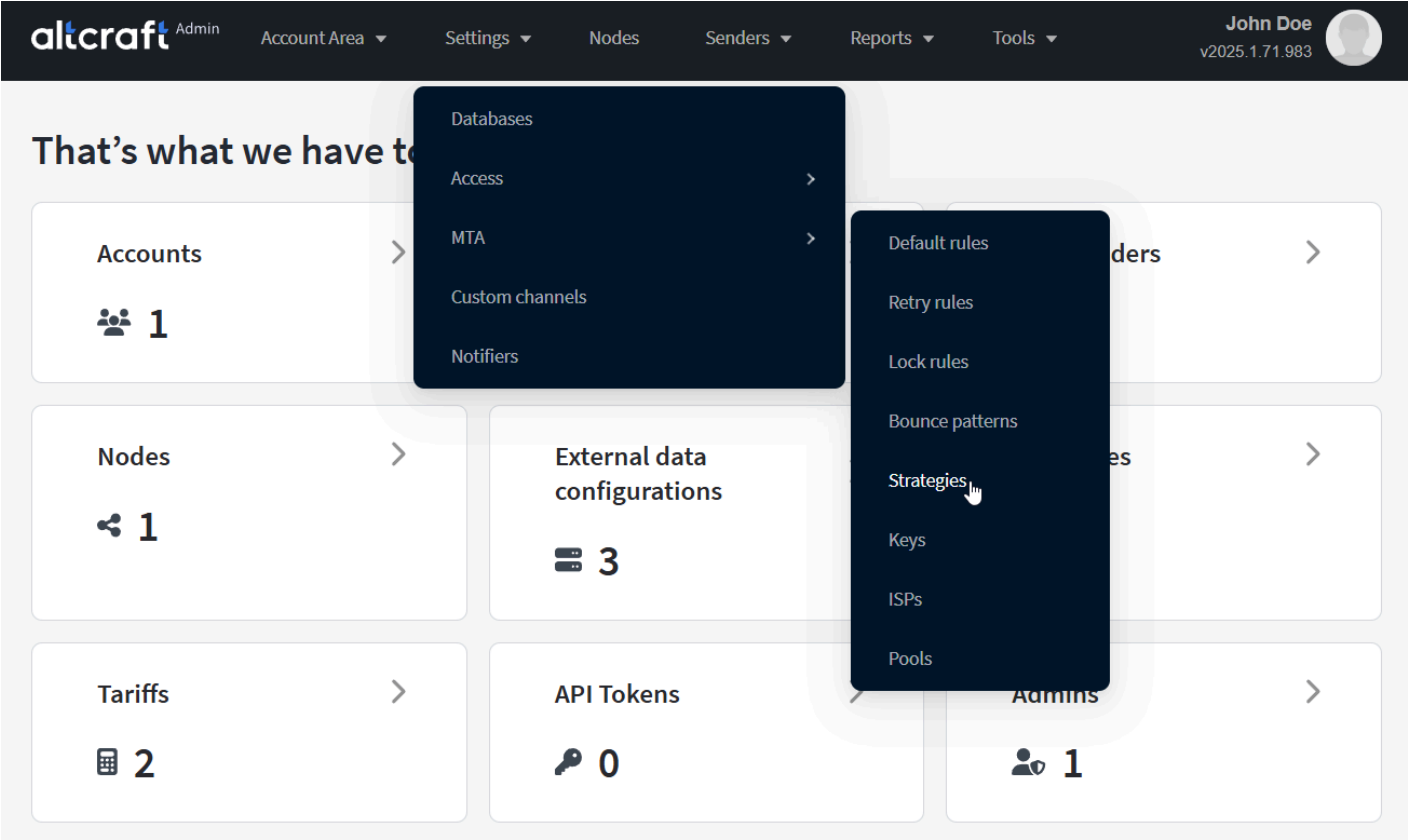
Strategies

Description

Strategies are a set of rules and settings that define message delivery parameters via **AKMTA strategy**. They control the sending speed, depending on the number of IP addresses of the [sender](#) and the selected sending strategies (for different IP addresses or different ISPs).

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 — 60 messages every 60 seconds. On the right side of the table are the **edit** and **delete** buttons. You can also edit a strategy by clicking its identifier or name on the left. To create a new strategy, click **+ Create**.

CAUTION

The default strategy cannot be deleted from the system. However, if you only need one configuration, you can modify the system record.

altcraft Admin

Account Area

Settings

Nodes

Senders

Reports

Tools

John Doe

v2025.1.71.617

Strategies

+ Create

Matches 8 < 1 > 25 / page

ID	Name	Concurrent	Messages / Connections	Connections / Interval	Interval	Drop	Actions
1	TLS, 60 per 60 sec	2	15	2	60	No	

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

* Messages per connection

* Connection attempts

* Time interval (seconds)

Average strategy speed per IP address

In 1 day

In 1 hour

In 1 minute

1.4K / 1 day

60 / 1 hour

1 / 1 minute

HELO type

Greet with IP PTR



☒ Try TLS first

☐ Drop messages

Save

Close

The number of messages sent per interval equals the product of the first three parameters. The **Average strategy speed per IP address** table shows the sending speed per day, hour, and minute for one sender IP address.

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 may be necessary if you want to create a sender stub for testing purposes.

Keys

Description

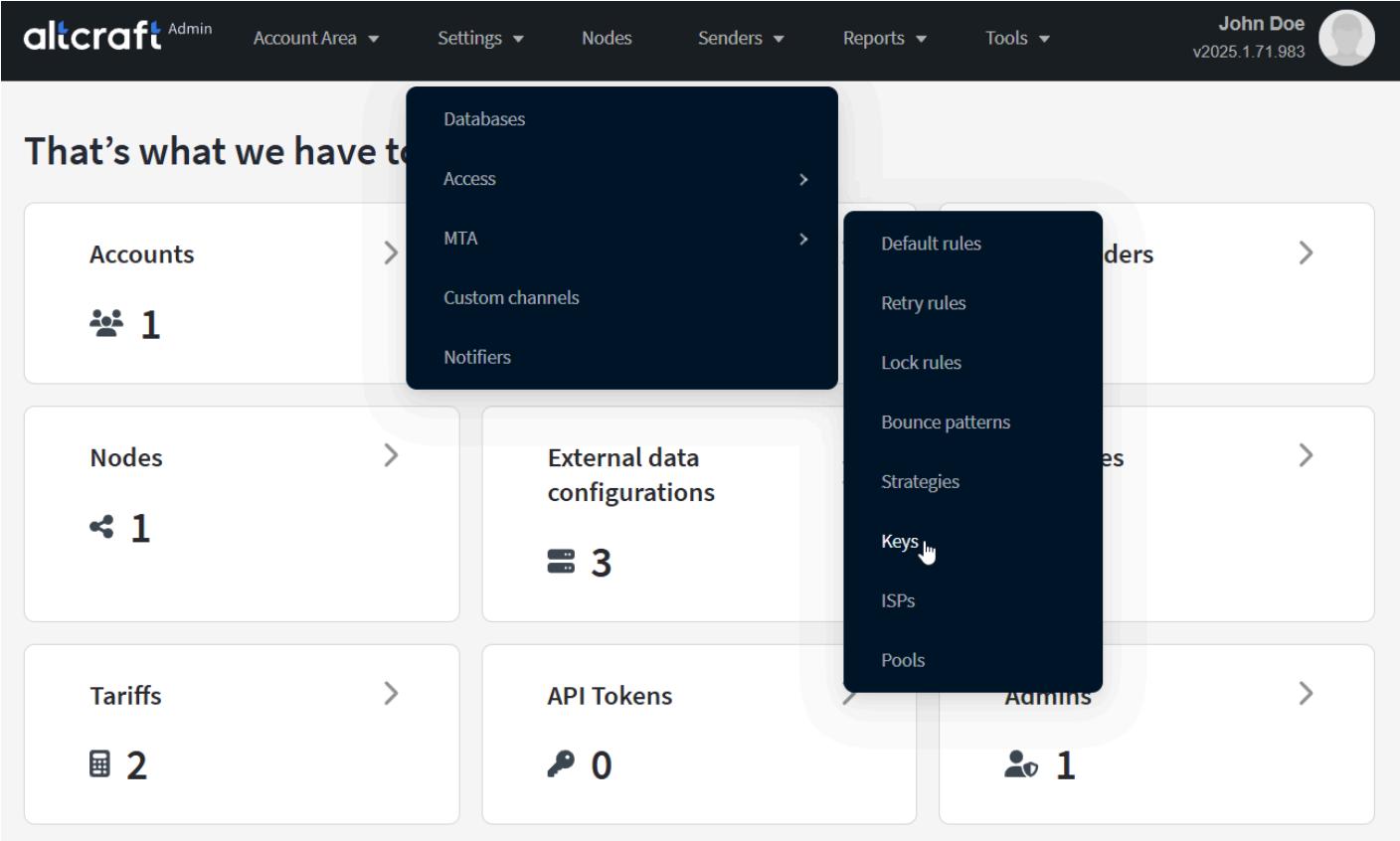
Keys are cryptographic tools used to ensure the secure transfer of data.

The platform supports two types of keys:

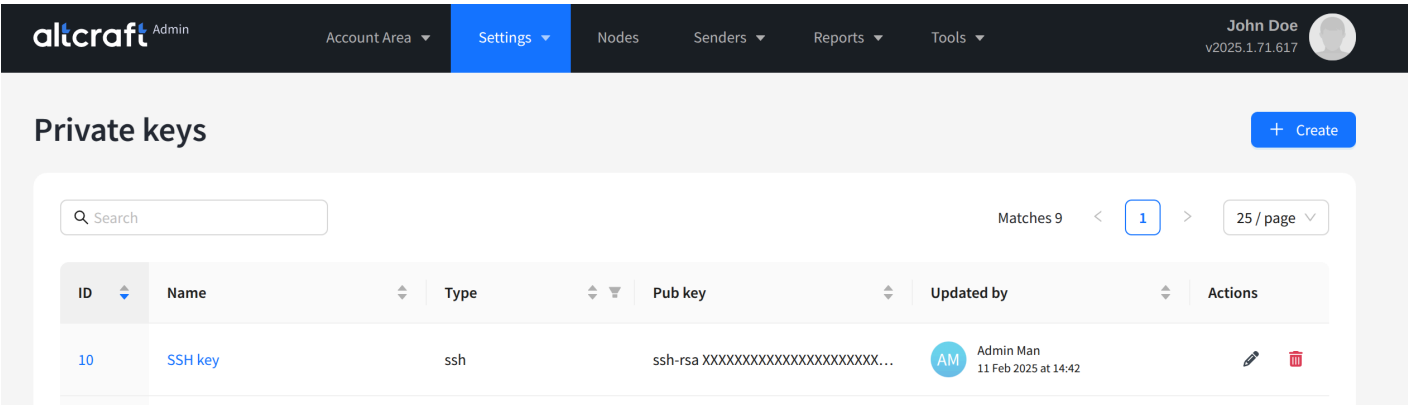
- **DKIM (DomainKeys Identified Mail)** — used to authenticate the sender of email messages and protect against spoofing.
- **SSH (Secure Shell)** — used for securely connecting to the server for administration.

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 **delete** buttons. You can also open editing by clicking the key identifier or name on the left. To create a new key pair, click **+ Create**:



Select the key type (**DKIM** or **SSH**). After that, you can paste existing keys into the **Public key** and **Private key** fields or generate a new pair by clicking **Generate key pair**:

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

Close

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 rules for different domain groups belonging to specific ISPs.

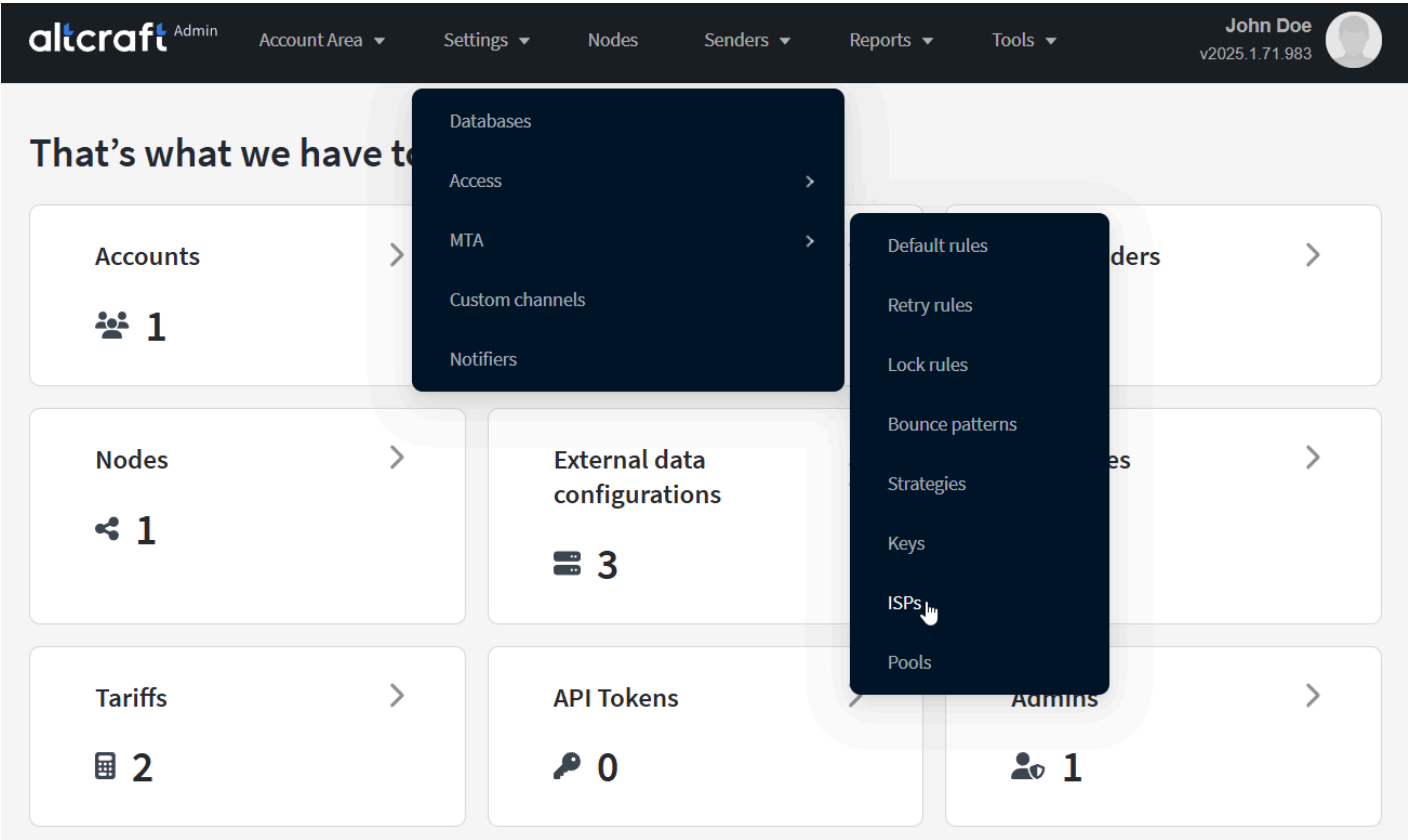
When is customization needed?

Configuring ISPs can be useful in the following cases:

- **Sending limits** — some providers limit the number of incoming messages as part of bulk mailing policies.
- **Improving domain and IP reputation** — setting individual rules for ISPs helps improve email deliverability.
- **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

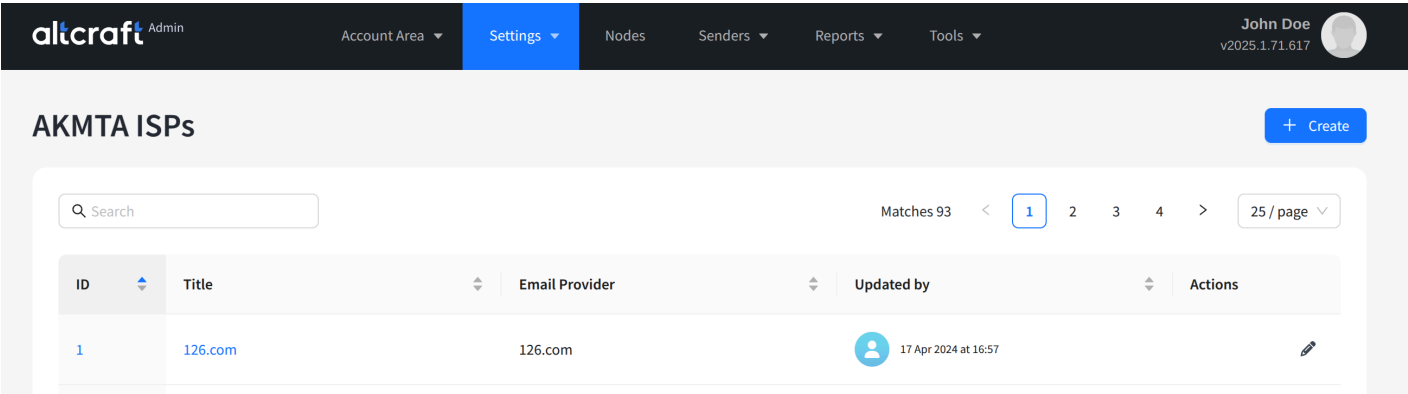
To manage email providers, in the main menu select **Settings — MTA — ISPs**.



The opened list displays existing ISP records in the AKMTA module. For most email providers, entries already exist in the default AKMTA configuration. To **edit** an existing ISP record, click the quick action button on the right side of the list or the record identifier on the left. To create a new record, click **+ 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, Base64, 8-bit.
- **Domains** — the domains of the email provider's mailboxes.

AKMTA ISPs > Edit

* Name

Global encoding

HTML encoding

Text encoding

Sender encoding



Sender encoding



Sender encoding



* Domains

One domain per line

Headers overriding

Added headers

+ Add header

Deleted headers

— Delete header

Save

Close

If you need to separately configure delivery to part of the email provider’s domains, create a separate ISP record and move the required domains into it.

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.

Headers overriding

Added headers

Key	Value	</>v	
Key	Value	</>v	

+ Add header

Deleted headers

Key	! This header will be removed	
Key	! This header will be removed	

— Delete header

Save

Close

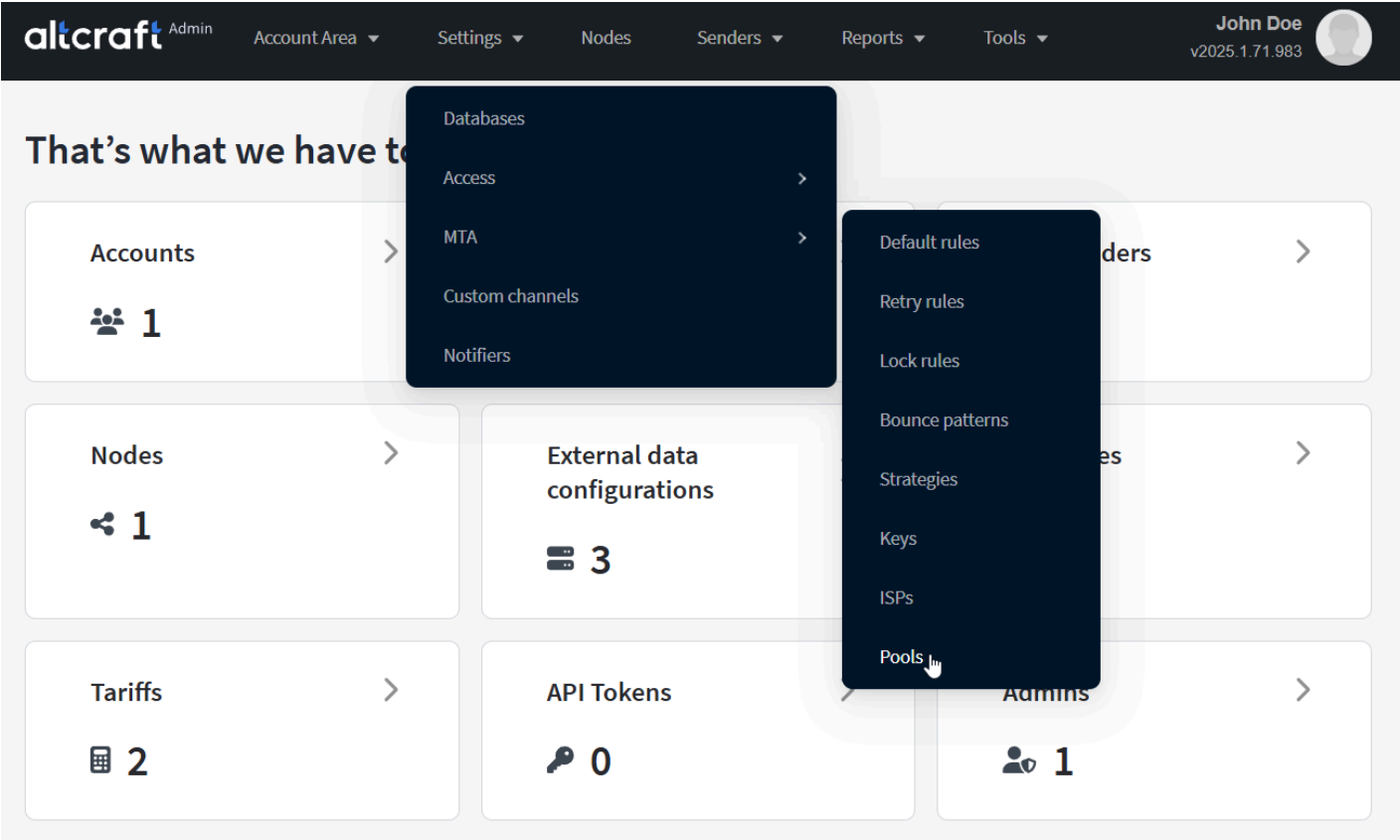
Pools

Description

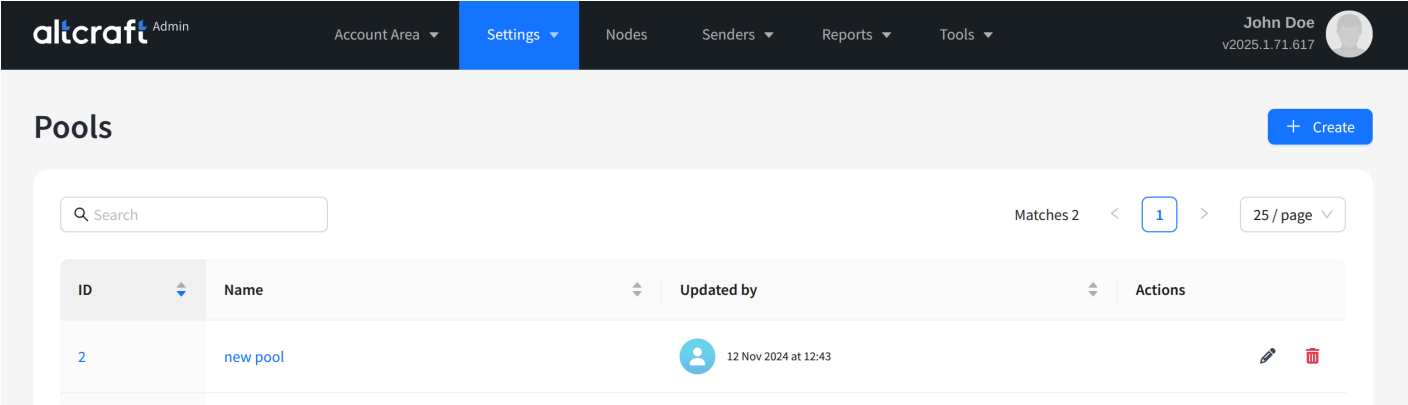
A **Pool** is a set of resources required to send emails from a specific domain. The pool allows you to simultaneously track sending statistics and perform SPF alignment.

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 can also open editing by clicking the pool identifier or its name on the left. To create a new pool, click **+ Create**:



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

Pools > Edit

Name

main pool

Email channel

* Sender domain

mta.example.com

* Sender subdomain

mta

* SPF domain

spf.mta.example.com

* DKIM key

Select DKIM



DKIM selector

acmp

Save

Close

After saving, the created pool will appear in the list. You can use it when configuring other entities in the Admin Panel.

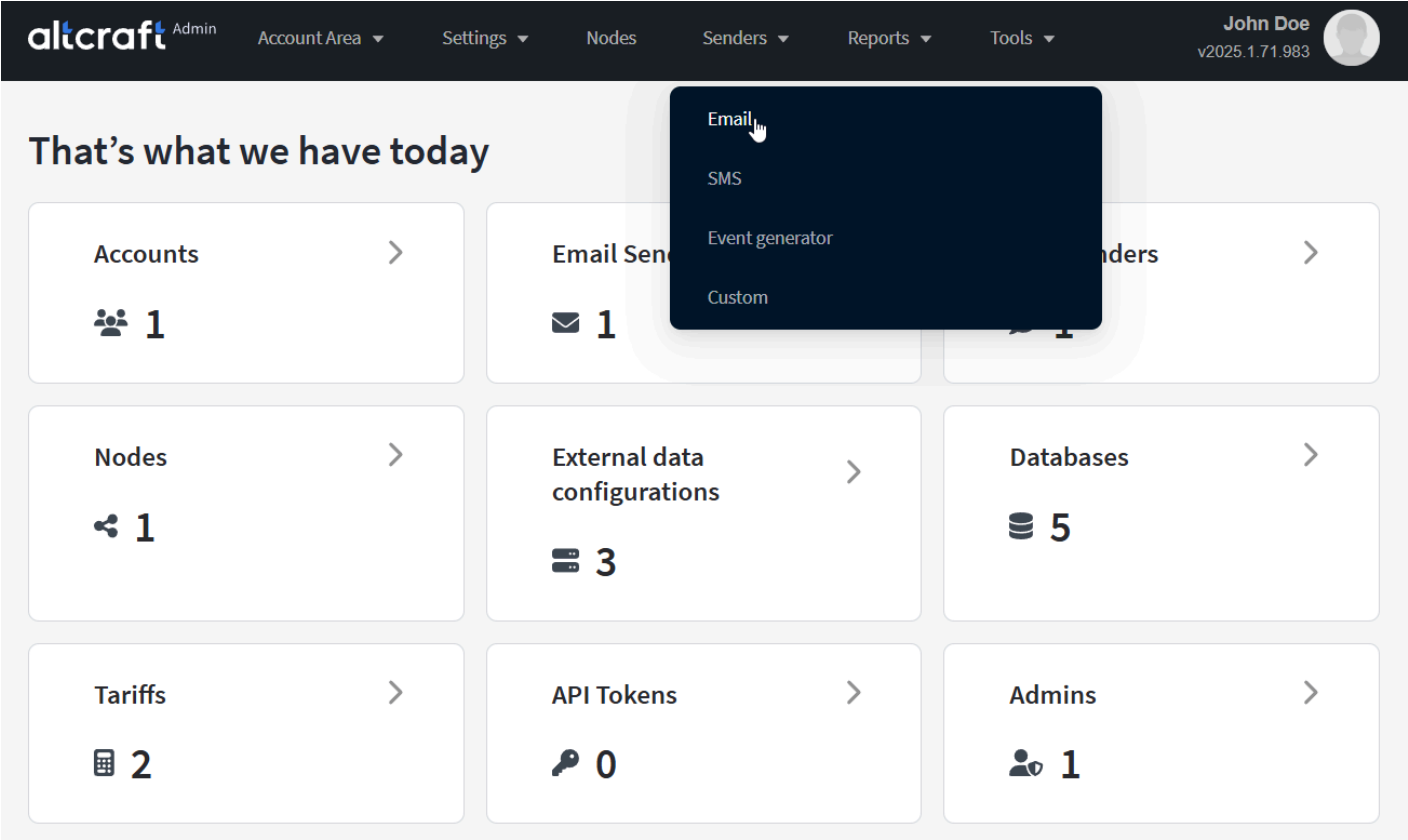
Email

Description

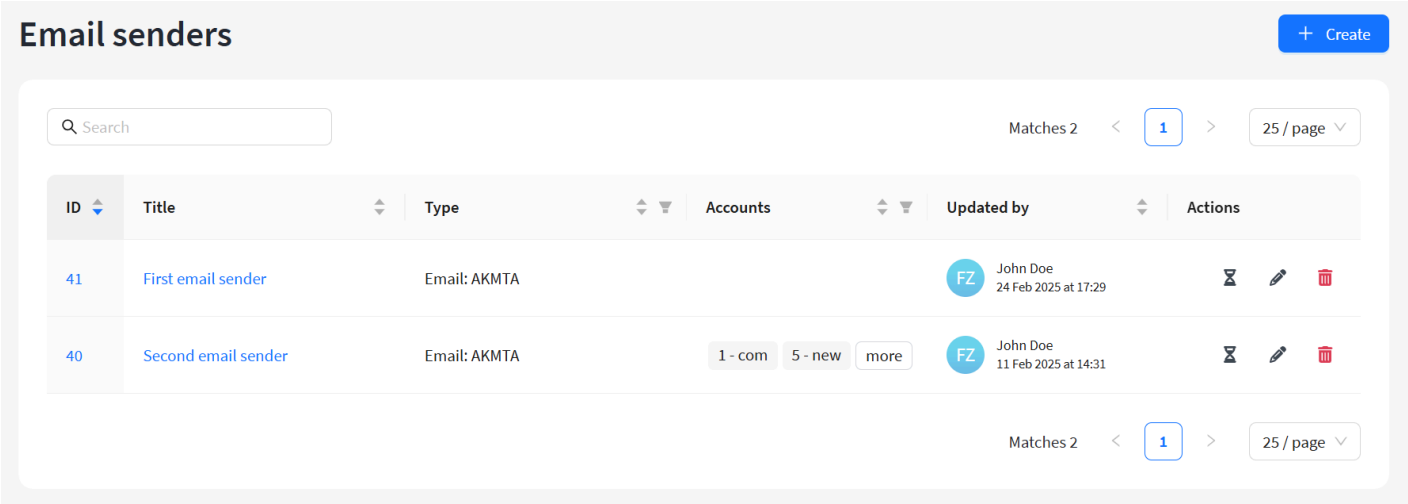
Sender in Altcraft is a configuration of the message delivery service. Senders are configured for email and SMS communication channels. As an email sender, Altcraft Mail Transfer Agent (**AKMTA**) can be used — a built-in email delivery module with extensive configuration capabilities.

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** buttons. You can also edit by clicking the sender ID (on the left). To create a new sender, click **Create**.



For a new sender, in the **Name** field specify a unique name, and in the **Assigned accounts** menu select [platform accounts](#) that will be able to use the sender for delivering messages.

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 the next section.
- **Amazon SES** (Simple Email Service) — integration with Amazon's transport module. Documentation link: <https://docs.aws.amazon.com/ses/index.html>
- **Sendsay** — integration with Sendsay SMTP relay. [Webhooks](#) are used to obtain delivery statuses. Documentation link: <https://docs.sendsay.ru/integrations/webhooks/>

NOTE

The platform also provides integration with the **Get-n-post** sender. Contact our support team to learn more: support@altcraft.com.

AKMTA Sender Configuration

* Name

First email sender

Type

Email: AKMTA

Assigned accounts

5 - new

Per IP strategy

Average strategy speed per IP address

	In 1 day	In 1 hour	In 1 minute
Minimum	691.2K / 1 day	28.8K / 1 hour	480 / 1 minute
Maximum	3.1M / 1 day	129.6K / 1 hour	2.2K / 1 minute

When selecting the built-in delivery module (**Type - Email: AKMTA**), the MTA configuration settings for the current sender will open.

General Settings

General

Rules

Relay

Headers

Incoming

* IP addresses

8 - altcraft - 00.000.0.00

Use MAIL FROM domain part from

Domain part from From header

From domain DKIM key

Disabled or not required

From domain DKIM selector

acmp

Secondary DKIM key

Disabled or not required

Secondary DKIM selector

acmp

In the **General** block, you configure sender domains and IP addresses. The following parameters are available:

- **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 and From header**** domain modification.
- **Pool** — resource pool needed for sending emails from a specific domain. Used when choosing the ****Aligned prefix and From header**** domain modification. The pool must be created separately. Learn how to create it [\[here\]](#)(/en/admin-guide/admin-panel/settings/mta/pools).

Rules

TIP

AKMTA settings are covered in a [separate section](#) of the documentation. It describes ISP management, sending strategies, Lock rules, Retry rules, Bounce patterns.

Each AKMTA sender can be finely tuned to deliver messages depending on the number of available IP addresses, network load, IP and domain reputations, and other factors.

- **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 sender if you need to add new rules or modify the built-in AKMTA rules. Be sure to transfer any required rules from the AKMTA default rules menu.

General
Rules
Relay
Headers
Incoming

Speed
Retry
Lock

IPs

00.000.0.00 ×

▼

ISP's

96 - o2.pl ×
94 - iG Mail ×

▼

Strategy

7 - TLS, 240 per 60 sec

▼

×

Remove rule

+

Add ISP's strategy

Speed — managing delivery speed

You can separately control the sending speed for different IPs, different email providers, and combinations of IPs and ISPs. To control the sending speed from IP addresses, click **Add rule** under the speed calculation table.

- **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 specified separately. The **Any other ISP** option is used to configure delivery speed to any other servers.

- **Strategy** — sending strategy parameters. Created in [AKMTA settings](/en/admin-guide/admin-panel/settings/mta).

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 forwarding node. On the relay side, messages can be further processed before leaving the company perimeter. To enable this option, check **Send via SMTP relay**.

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.

General

Rules

Relay

Headers

Incoming

Enable sending via SMTP relay

SMTP relay host

relay.example.com:25

Auth type

CRAM-MD5

Username

username

Password

password

Use Start TLS

Use webhook

- **Use webhook** — enable receiving email statuses via webhook instead of DSN. After enabling this, you'll need to select a gateway. The platform automatically generates a webhook URL that will receive delivery statuses. Copy this URL and specify it in your gateway webhook settings.

Use webhook

Webhook type

Sendsay

Use webhook

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 right field, the header value.

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.

GeneralRulesRelay**Headers**Incoming

Headers overriding

Added headers

Key

Value

</>v

+ Add header

Deleted headers

— Delete header

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 process emails.

Forward unparsed emails to — email address for messages the server failed to process correctly.

GeneralRulesRelayHeaders**Incoming**

Additional mailbox domains

One domain per line

Forward unparsed emails to

example@abucat.lan

Sending to allowed addresses

The **Allowed only on specified FROM domains** and **Allowed only on specified TO emails** options restrict sending from the sender to a list of specified domains or addresses.

- Allowed only on specified FROM domains** — sending will only occur to addresses with the specified From-domain. Each domain must be specified on a separate line.
- Allowed only on specified TO emails** — sending will only occur to the specified email addresses. You can use the * symbol, which will match any set of characters. Each address must be specified on a separate line.

When sending, the domain is checked first; then, if the domain matches, the specific recipient address is checked.

Only verified FromDomains

example.com

Only verified ToEmails (wildcards)

*-testing@example.com
test-mailbox@example.com

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 platform, go to **Settings — Virtual Senders**, and configure the rules for sending through the created sender.

You can learn more about virtual senders in [this article](#).

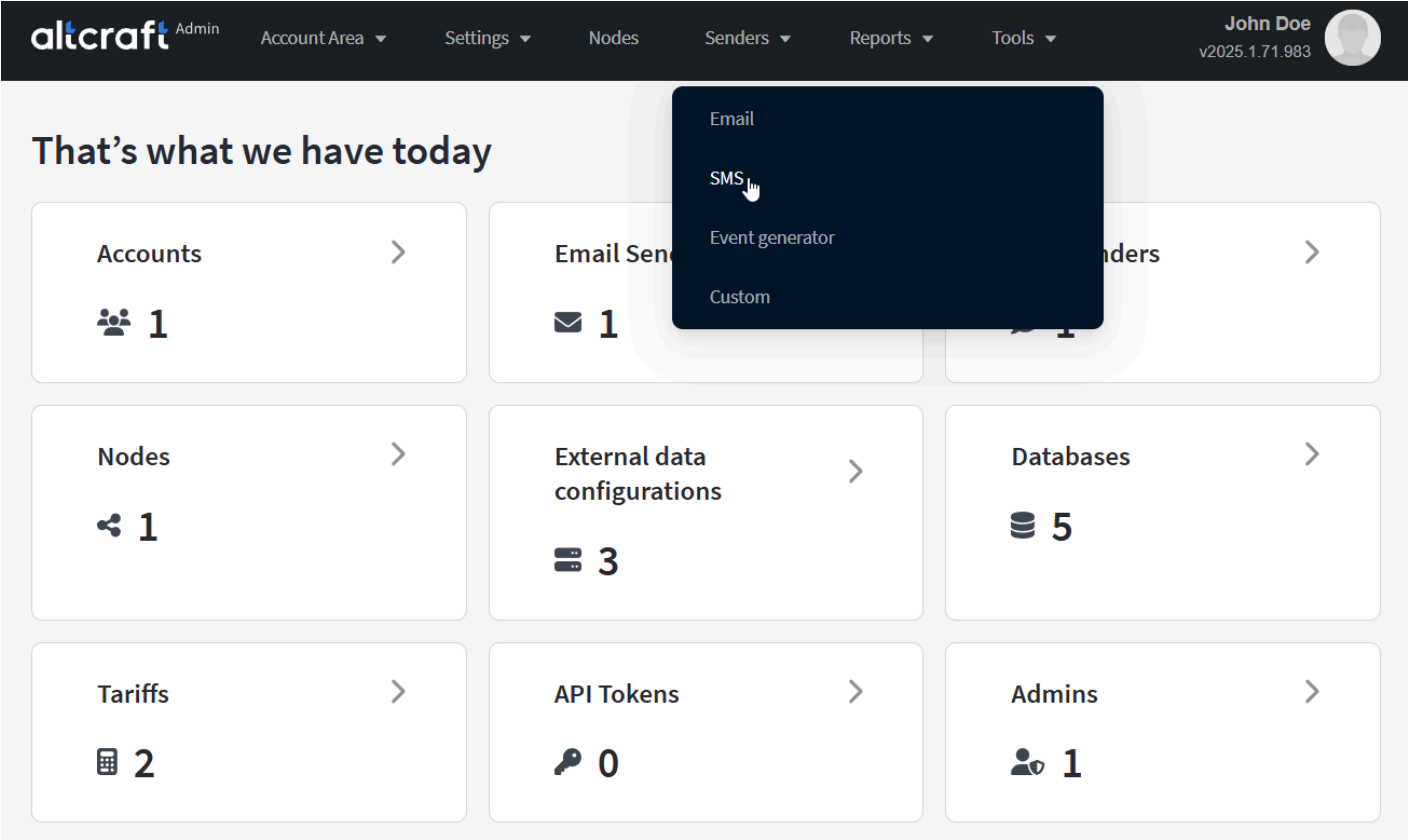
SMS

Description

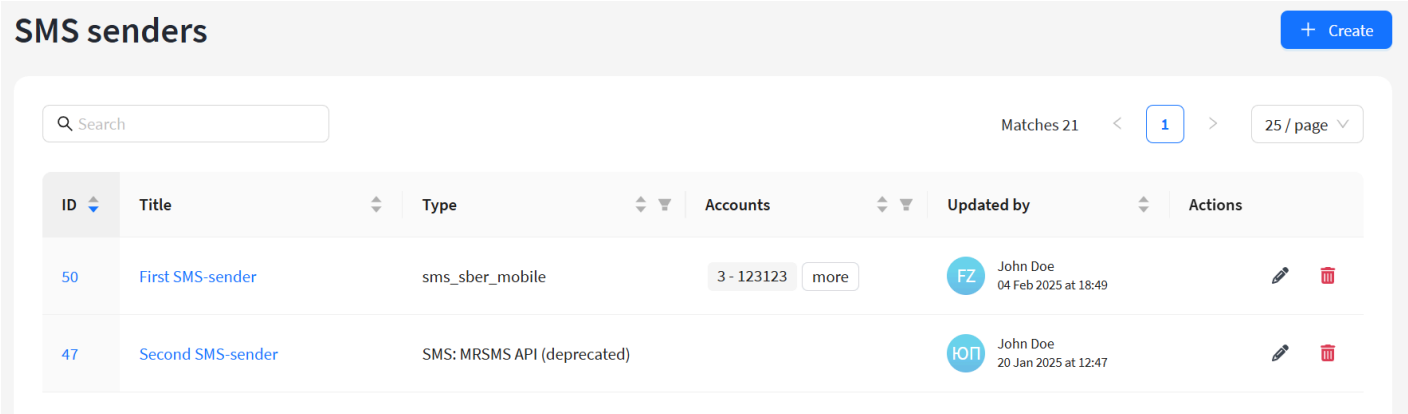
Sender in Altcraft is a configuration of the message delivery service. Senders are configured for email and SMS communication channels. As an email sender, Altcraft Mail Transfer Agent (**AKMTA**) can be used — a built-in email delivery module with extensive configuration capabilities.

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** buttons. You can also edit by clicking the sender ID (on the left). To create a new sender, click **Create**:



For a new sender, in the **Name** field specify a unique name, and in the **Assigned accounts** menu select [platform accounts](#) that will be able to use the sender for delivering messages:

SMS and telecom gateways

After that, in the **Keywords** field, you can add stop words. When received, the platform will add the phone number to the suppression lists of **all accounts linked to this sender**. The word **STOP** is included by default.

The **Add to global suppress list** option also adds the unsubscribed phone number to the platform's global blacklist.

 **SMS Receiving URL**
https://click.

☐ Suppress phone number if keyword is received

☐ Add to global suppress list

Keywords:

STOP ×

i-Digital 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 **password**. To ensure correct operation of such configurations, you must specify в параметрах **Login field name** и **Password field name** the names of the authentication fields. If values for these fields are not specified, the platform will pass the standard names (**login** и **password**).

SMS service url:

https://gate10.mfms.ru:9876/CLIENT/connectorX

Login:

Input login...

Login field name:

login

Password:

Input password...

Password field name:

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 limits on the quantity and volume of sent requests imposed by the provider. The limit for sending requests is configured in the [configuration file](#) via the following parameter:

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 receiving message delivery statuses.

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 accessed.
- **Login** — client credentials for authorization in the SMS gateway.
- **Password** — client credentials for authorization in the SMS gateway.

SMS.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, the platform registers a non-delivery with the following message:
`Message is expired`.

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 shortened.

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 to restrict names).
- **Only verified Phones** — list of numbers to which sending is allowed (used for test gateways).

Phone numbers support wildcards: `?` matches any single digit, and `*` matches any sequence of digits at the end of the number:

* SMS service sending script

https://api.smstraffic.ru/multi.php

* Login

Input login...

* Password

Input password..

Only verified CallerIds ?

domain.tld

Only verified Phones (wildcards) ?

*@example.com

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 platform, go to **Settings — Virtual Senders**, and configure the rules for sending through the created sender.

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 test the operation of various platform mechanisms without actual user interaction.

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**

altcraftAdmin

Account Area

Settings

Nodes

Senders

Reports

Tools

John Doe
v2025.1.71.983

That's what we have today

Accounts
1

Nodes
1

Tariffs
2

Email Senders
1

External data configurations
3

API Tokens
0

Senders
1

Databases
5

Admins
1

Email

SMS

Event generator

Custom

The opened sender list displays existing senders. On the right side of the table, there are **edit** and **delete** buttons. You can also edit by clicking the sender ID (on the left). To create a new sender, click **Create**.

Event Generators

+ Create

Search

Matches 2 < 1 > 25 / page

ID	Title	Type	Accounts	Updated by	Actions
49	First event generator	Any: Event Generator		JD John Doe Date is unknown	
7	Second event generator	Any: Event Generator		Date is unknown	

Matches 5 < 1 > 25 / page

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 platform, go to **Settings — Virtual Senders**, and configure the rules for sending through the created sender.

You can learn more about virtual senders in [this article](#).

Integration with Altcraft Cloud SMTP

Altcraft Cloud SMTP is an external SMTP relay through which the Altcraft platform can send email messages and receive delivery events via HTTP Webhook.

The integration is used when direct email sending from the platform server is not possible or not desirable. For example, when Altcraft is hosted in a private cloud, outgoing connections on SMTP port 25/TCP may be blocked by the cloud provider. In such a configuration, emails are passed to an external SMTP relay, which performs delivery on behalf of the system.

Altcraft Cloud SMTP also sends back delivery events such as delivery and bounce. This allows the platform to correctly record message statuses, generate mailing statistics, and handle non-deliveries.

Sender Configuration

- Go to the Admin Panel, section **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

Specify the following details:

- **SMTP relay host:** the address provided by Altcraft support
- **Start TLS:** Disable
- **Use webhook:** Enable

The authentication type and credentials should be filled in based on the data provided by Altcraft technical support.

General Rules **Relay** Headers Incoming

☒ Enable sending via SMTP relay

* SMTP relay host

relay.example.com:25

Auth type

Without authorization

☐ Use Start TLS

☒ Use webhook

Webhook type

Altcraft Cloud SMTP

X-Tenant, X-Campaign and X-Feedback-URL headers will be added automatically

Use webhook

https://pxl-sg3npsg3npsg3npy.dev.altcraft.com/integrations/altcraft_cloud_smtp?sh=BI

Connecting to Virtual Sender

After creating and configuring the sender, you need to add it to a **virtual sender** through the platform's user interface. To do this, open the platform, go to **Settings — Virtual Senders**, and configure the rules for sending through the created sender.

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

General

Rules

Relay

Headers

Incoming

☒ Enable sending via SMTP relay

SMTP relay host

relay.example.com:25

Auth type

CRAM-MD5

Username

username

Password

password

☐ Use Start TLS

☐ Use webhook

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](#)

☒ Use webhook

Webhook type

Sendsay



Use webhook



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](/en/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](/en/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 clicks. Unsubscribes via the List-Unsubscribe header may also not be recorded in Sendsay.

:::

:::note

Altcraft reports are recommended as the primary source of analytics, as the platform provides more detailed statistics and correctly identifies special events (unsubscribes).

:::

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 platform, go to **Settings – Virtual Senders**, and configure the rules for sending through the created sender.

You can learn more about virtual senders in [\[this article\]\(/en/user-guide/settings/9405938\)](/en/user-guide/settings/9405938).

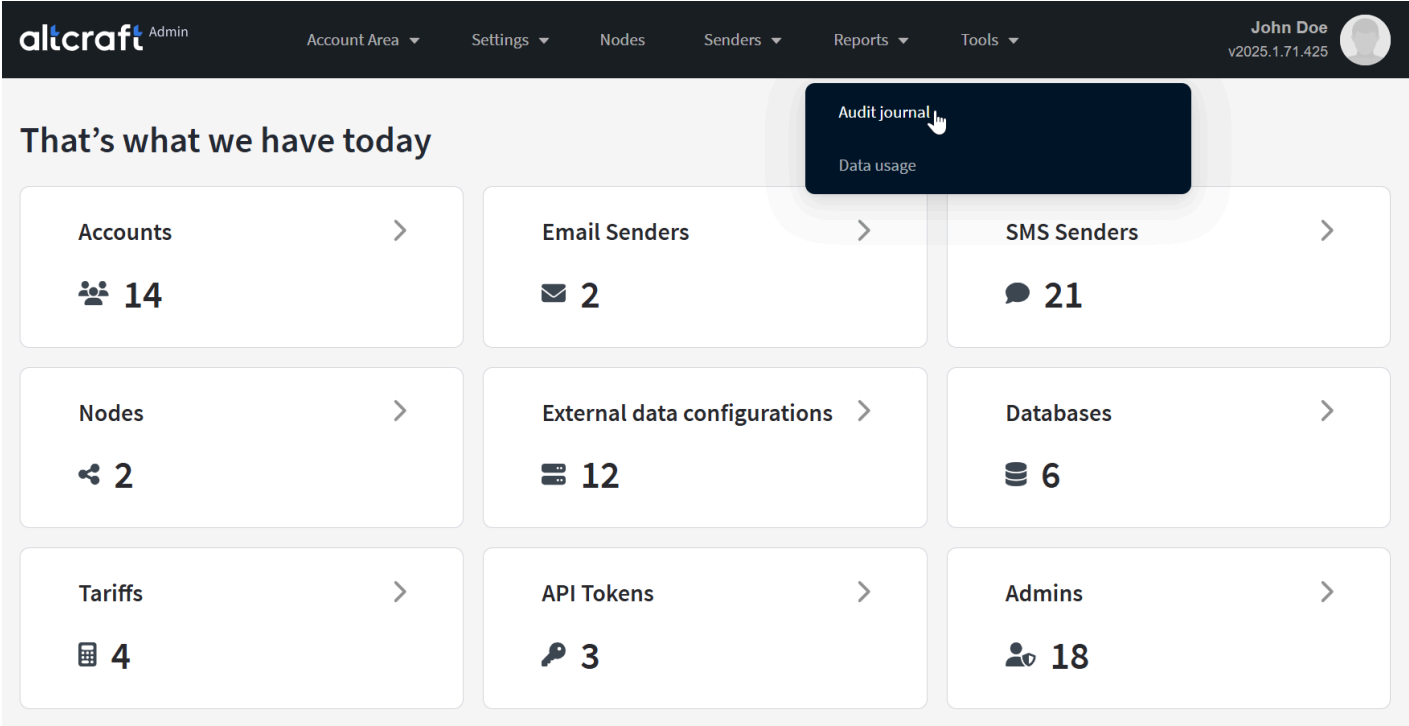
Audit Journal

Description

The **Audit Journal** is an important security component of the Altcraft platform. It collects information about the actions performed by [platform administrators](#) on various objects in the Administrative Panel.

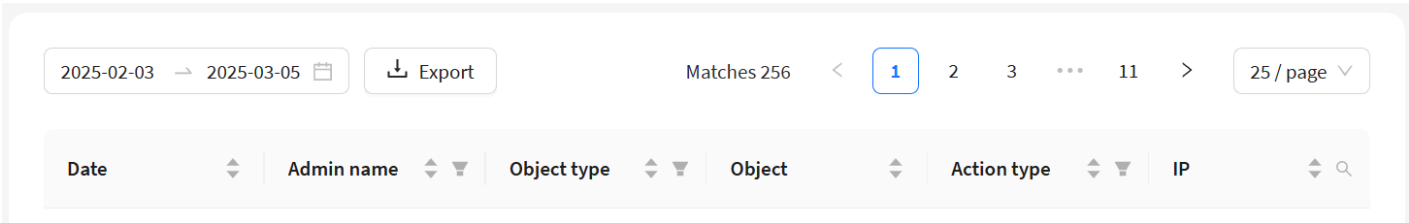
Working with the Audit Journal

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 about all actions in the Administrative Panel, select **All** in the filters. You can download the report in CSV or XLS format by clicking **Export**.

The report displays the date of the action, administrator account, IP address, action type, and the ID and name of the object.

Date	Admin name	Object type	Object	Action type	IP
2025-02-03 14:17:09	admin	System	-	Login	192.168.40.101
2025-02-03 16:04:47	admin	System	-	Login	::ffff:192.168.40.101
2025-02-03 18:41:53	admin	Sender	50 New sender	Create	::ffff:192.168.40.101
2025-02-03 18:45:14	admin	Sender	50 New sender	Save	::ffff:192.168.40.101
2025-02-03 18:45:19	admin	Sender	50 New sender	Save	::ffff:192.168.40.101
2025-02-04 12:39:28	admin	System	-	Login	::ffff:192.168.40.101

Data Usage

Description

Data Usage is a section designed to view the volume of data used in the platform. It contains information about the current data volume, its distribution across different system components, and the dynamics over the past 30 days.

Current Data Usage Statistics

Current data usage

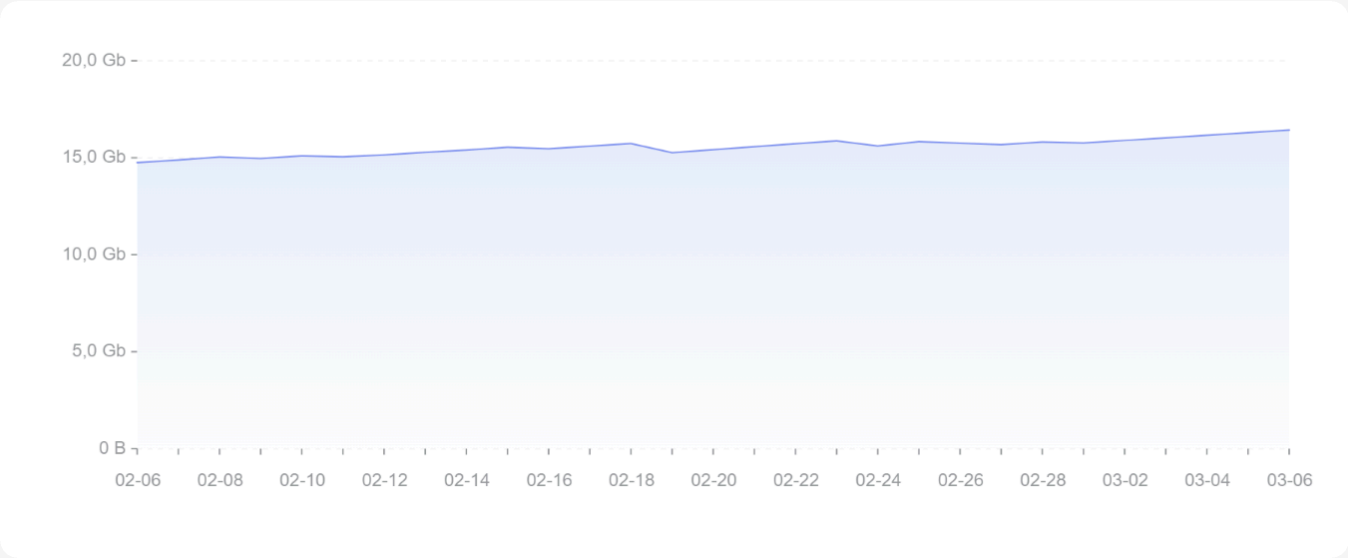
Total 16,4 Gb	Node files 318,6 Mb	Account databases 38,4 Mb	Control database 22,9 Mb
Clickhouse database 1,3 Mb	Suppress database 0 B	Notify database 1,0 Mb	AKD files 13,4 Gb

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

Current total last 30 days



The graph shows the dynamics of changes in data volume over the past month. It displays the total volume of data per day, which helps analyze and forecast storage requirements.

Data Usage Details

Data usage details

2025-02-27 → 2025-03-06

Total Account Node

Total details

Export

Matches 4125 / page

Date	Account database	Control database	SSDB	AKD Files	Node Files
2025-02-27	36,8 Mb	23,1 Mb	1,1 Mb	15,3 Gb	318,7 Mb
2025-02-28	36,8 Mb	23,0 Mb	1,1 Mb	15,4 Gb	318,7 Mb
2025-03-01	37,1 Mb	23,0 Mb	1,1 Mb	15,4 Gb	318,7 Mb
2025-03-02	37,1 Mb	23,0 Mb	1,1 Mb	15,5 Gb	318,7 Mb

Matches 8125 / page

The **Data Usage Details** section provides detailed information about the distribution of data across different categories for the selected period. It contains a table with daily statistics, including:

- **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 about the original message based on the received complaint.

Connection Setup

In the main menu, select **Tools — ARF decoder**:

altcraftAdmin

Account Area ▾

Settings ▾

Nodes

Senders ▾

Reports ▾

Tools ▾

John Doe
v2025.1.71.425

That's what we have today

Accounts
14

Email Senders
2

Nodes
2

External data configurations
12

Tariffs
4

API Tokens
3

Databases
6

Admins
18

ARF decoder

URL decoder

SMID decoder

License

Paste the message with service headers into the corresponding field and click **Decode**. The result will contain:

- **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: fH8cmmbi13SIFFbalHnkOMXAAQMQUQEIQEAABARAQGqOTVbnkOMXZ...

SenderID: 1

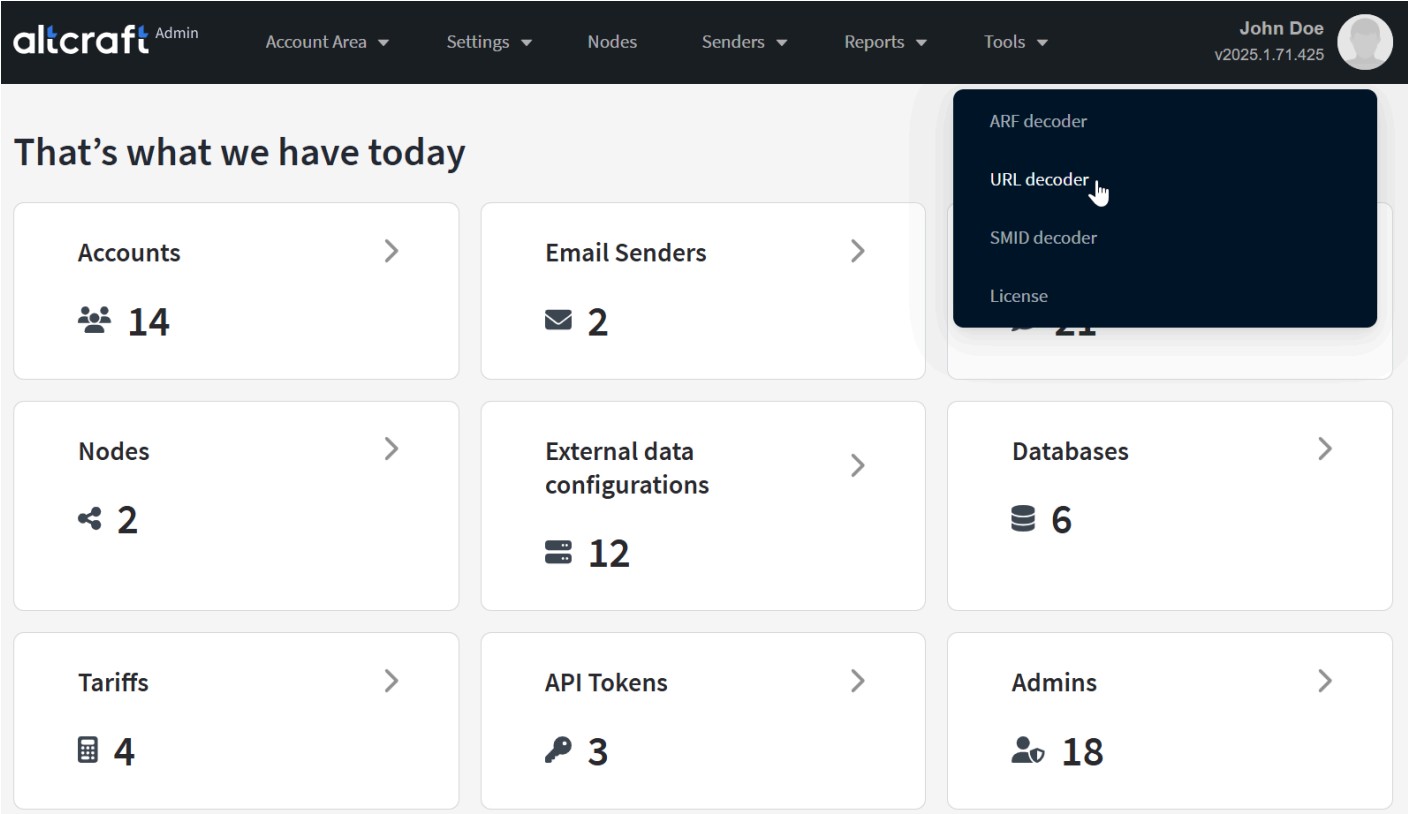
URL decoder

Description

URL Decoder is a tool used to decode links containing information about the recipient and click tracking in email messages.

Connection Setup

The links you add to message templates redirect recipients to the tracking server to collect click tracking information. From the text of such a link, you can identify the message recipient. In the main menu, select **Tools — URL decoder**:



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-K1DiypwL7WhOp9yIbK1mfXFN32GwJ19i1GWIF-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 :	fAAAAAAAAAAAAAAAAAAAAAAQMQUQEIAAAEBEBAao5NVueQ4xdmmidVXJEZufQnTfxlZmM0Y...
Resource ID :	1
Channel :	email channel
Launch ID :	4_6_4ZKDyqhwiMc
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 specific recipients.

Connection Setup

By using the identifier of a message sent to a particular recipient, you can retrieve detailed information about it. In the main menu, select **Tools — Message ID decoder**:

altcraftAdmin

Account Area ▾

Settings ▾

Nodes

Senders ▾

Reports ▾

Tools ▾

John Doe
v2025.1.71.425

That's what we have today

Accounts
14

Email Senders
2

Nodes
2

External data configurations
12

Tariffs
4

API Tokens
3

Databases
6

Admins
18

ARF decoder

URL decoder

SMID decoder

License

Paste the message identifier (SMID, FSMID, ESMID, or USMID) into the corresponding field and click **Decode**. Depending on the [identifier type](#), the result may include information such as:

- 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 permanent link to the web version.

Message ID decoder

* Paste SMID, FSMID, ESMID or USMID here ?

fNZiyGcQc8uGNR4AMY57ykYAAAEQMQUEQEAEABChAQcXSOMY57ykYZ6HOj1Tvqh_-6t5DZDJmMWM0ODmA1bL9A

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 : 67a1ce8f54efaa1ffeade43

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, including validity period and update capabilities. The license can be linked to a specific platform version and includes notifications about its status and expiration, as well as mechanisms for automatically updating license information via the akttool utility.

Connection Setup

altcraftAdmin

Account Area

Settings

Nodes

Senders

Reports

Tools

John Doe
v2025.1.71.425

That's what we have today

Accounts
14

Email Senders
2

Nodes
2

External data configurations
12

Tariffs
4

API Tokens
3

Databases
6

Admins
18

ARF decoder

URL decoder

SMID decoder

License

In the **License** section, you can view detailed information about the platform version, license, and its expiration date:

License details

Status:

Valid

Product ID:

License ID:

Issued for:

Version:

v2025.1.71.569-gceb6eda43e4

Updated:

2025-03-05T08:46:09Z

Till:

2025-06-01T00:00:00Z

Notify about license change to:

license@altcraft.com

☐ Reset license cache

In the **Notify about license changes to** field, you can specify the email address where notifications will be sent about license status changes or expiration. Messages will start being sent 2 weeks before the license expiration date and also if its status changes.

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, technical requirements, and operation schemes.

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 - abcdefghijklmnopqrstuvwxyzABCDEF. Be sure to generate a more complex password consisting of at least 32 characters.

Archive Loading

All clients are provided with a compressed archive with the extension `.tar.gz`. Unpack the archive and move its contents to the `/opt/akd` directory:

```
{
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 statistics. Configure an NTP server and verify synchronization:

```
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

Install the following dependencies:

```
{
apt update
apt install --no-install-recommends xvfb cutycapt unzip gcc g++ libsnappy-dev make autoconf libpcre3-dev libgomp1
}
```

Installation of Percona Server for MongoDB or MongoDB Community Edition

MongoDB is used to store system settings and user data.

INFO

Additional materials:

- [Install Percona Server for MongoDB on Ubuntu](#)
- [Install Percona Server for MongoDB on RHEL](#)
- [Tune parameters](#)

The platform supports two options:

- **Percona Server for MongoDB** — the recommended option from Percona with additional features: enhanced data-at-rest encryption, extended auditing, improved diagnostics, and performance optimization.
- **MongoDB Community Edition** — the standard version from MongoDB Inc.

TIP: Recommendation

For production environments, it is recommended to use **Percona Server for MongoDB** as a more secure and performant solution. The instructions for creating users and configuring connections remain the same for both options.

Choose one of the installation options:

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` , `/var/log/mongodb`), the `mongodb` user and group, and install additional utilities, including `mongosh` .

```
{
  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 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 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` parameter and specify how the master key will be stored. This guide covers two master key storage options: a local key file and HashiCorp Vault.

Encryption can only be enabled on an empty database. If the database already contains data, use a separate migration procedure.

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 Secrets Engine v2 with versioning enabled. On first startup, the server generates a new random master key and stores it in Vault at the specified path.

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 own strong password with a minimum length of 32 characters.

MongoDB Community Edition

Go to the [download page](#), select version 8.0, your OS version, and the `.tgz` archive. Copy the link and download the archive:

```
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](#).

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 Percona Server for MongoDB. To improve security, you can use the SCRAM mechanism supported by the platform (see the [documentation](#)).
- **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 installation guide using DEB packages.

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 addresses.

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 instruction for installing from DEB packages hosted on Cloudsmith.

Add the required repositories to the system:

```
{
curl -sLf https://dl.cloudsmith.io/public/rabbitmq/rabbitmq-server/setup.deb.sh | bash
curl -sLf 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 suitable for handling large volumes of data. In Altcraft, starting from version [v2025.2.72](#), Kvrocks is used for deduplication in mailings. Install the dependencies for the build:

```
{
 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 abcdefghijklmnopqrstuvwxyzABCDEFGF
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 description of all parameters is available on the [Configuration File](#) page.

Example:

```
{
 "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.

This completes the installation. The platform is ready to launch and further configuration. To continue setup, follow the guide on the [\[Getting Started\]\(/en/admin-guide/how-to-begin\)](#) page.

# 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 Docker containers.

## Installation

### Minimum requirements

The table contains the minimum software versions that must be installed on the target host to run the platform.

Software	Version	Description
Docker Engine	>= 25.x	See the <a href="#">Docker Engine</a> documentation for installation instructions
Docker Compose	>= 1.25.0	See the <a href="#">Docker Compose</a> documentation for installation instructions

### Volume locations

The Altcraft container uses volumes to store persistent data. You can customize their location according to your requirements.

Location in Docker container	Description
/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 <b>Note:</b> the output of logs to the <code>STDOUT</code> of the container is configured by the <code>LOG_STDOUT</code> environment variable. Set it to <code>true</code> if you need it.

### Minimal main.json configuration

Before starting the platform, you need to configure the minimum configuration `$ALTCRAFT_HOME/config/main.json`. Configuration example:

**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_HOSTNAME": "click.example.com",
 "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
 },
 "SYSTEM_REDIS": {
 "MODE": "standalone",
 "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
 "NODES": [
 {
 "HOST": "192.168.20.1",
 "PORT": 6666
 }
]
 },
 "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 to `true`.

```
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 options.

### 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: "abcdefghijklmnopqrstuvwxyzABCDEF"  
 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 parameter to -1

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
 "RPC_HOST": "", // RPC replica address
 "RPC_PORT": "", // 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
 "RPC_HOST": "", // RPC replica address
 "RPC_HOST_CLI": "", // address used by clients to connect to the replica
 "RPC_PORT": 7962, // RPC replica port
 "FORCE_MASTER_ROLE": false, // use as master server
 "READERS_GROUP": "default", // reader group served by the replica
 "READERS_GROUPS": {}, // custom reader groups (set on the master host)
 "HIGH_PRIORITY_WORKERS": false // dedicated worker pool for high-priority scenario tasks
}]
```

Object fields:

- `HOST_ID` — unique host identifier.
- `RPC_HOST` — the address on which the service accepts connections.
- `RPC_HOST_CLI` — the address clients use to connect.
- `RPC_PORT` — the RPC port.
- `FORCE_MASTER_ROLE` — forces the master role.
- `READERS_GROUP` — the reader group served by the host. Default: `default`.
- `READERS_GROUPS` — custom reader groups: `{"group_id": {"INCLUDE": "1-5", "EXCLUDE": "2"}}`. Set on the master host only. See [Configuring Custom Reader Groups](#).
- `HIGH_PRIORITY_WORKERS` — designates the host as a handler of high-priority scenario tasks: the dedicated worker pool processes high-priority tasks first and picks up regular ones when there are none. If the parameter is not set, the flag is enabled on the master host ( `FORCE_MASTER_ROLE: true` ) and disabled on all other hosts (an unset parameter on a slave host is equivalent to `false` ). If priority is effectively disabled on all hosts, the `procworkflow` service does not start.

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 asynchronously

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 queue

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 processing starts

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 the database

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 scenarios

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 remote servers, including as a cluster for [fault-tolerant platform operation](#).

**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\_CERT\_PATH** `string`

Description: Path to TLS encryption certificate

Default: None

Required: No

**CONTROLDB\_TLS\_CA\_CERT\_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 specified, the values will be inherited from this object.

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\_CERT\_PATH** `string`

Description: Path to TLS certificate  
Default: None  
Required: **Yes**, if TLS enabled

**KVR\_TLS\_CA\_CERT\_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\_CERT\_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 attached to the account

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 nodes is lost, the platform will automatically attempt to reconnect to that node or switch to other available nodes. With these parameters, you can override the location of the RabbitMQ message broker 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\_CERT\_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 processing of duplicate messages

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 unprocessed events exceeds this value, some of them may be lost.

With a value of 1024 the probability of loss is low, but pay attention to the log warning "Some events will dropped"

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 previous RabbitMQ configuration settings. Otherwise, all previous settings will be ignored and priority will be given to the settings specified in the `RABBITMQ_CLUSTER` section.

#### **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 processed messages

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 before being processed

Default: 50000

Required: No

---

## PebbleDB

Processing a large number of Action Hooks can lead to high RabbitMQ load. To offload the broker, you can use a local buffer database to read messages from the RMQ queue. In this case PebbleDB is used as the storage.

**PROCHOOK\_BUFFER\_PEBBLEDDB\_ENABLE** `bool`

Description: Enables buffer database mode

Default: false

Required: No

**PROCHOOK\_BUFFER\_PEBBLEDDB\_PATH** `string`

Description: Path to the buffer database

Default: None

Required: No

**PROCHOOK\_BUFFER\_PEBBLEDDB\_INSERT\_BATCH\_SIZE** `int`

Description: Insert batch size

Default: 200

Required: No

**PROCHOOK\_BUFFER\_PEBBLEDDB\_BATCH\_ASSEMBLING\_TIME\_MS** `int`

Description: Insert batch assembly time (ms)

Default: false

Required: No

**PROCHOOK\_BUFFER\_PEBBLEDDB\_SCAN\_TIME\_MS** `int`

Description: Interval between scans (ms)

Default: 1500

Required: No

**PROCHOOK\_BUFFER\_PEBBLEDDB\_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 attempts

Default: 60

Required: No

---

## Tracking platform module

The following parameters are used to configure the external connection of the platform tracking module, as well as internal connections of the [tracking module processes](#) with the platform control server.

### 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 links in template previews

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 processes](#):

#### 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

---

#### procpush Settings

With these parameters you can optimize the `procpush` process during profile event history merging:

**NUMBER\_OF\_MONTHS\_TO\_MERGE** `int`

Description: Number of months for which profile history is merged. A value of `-1` means merge for all time. For example, `36` — merge history for the last 3 years. This parameter avoids heavy ClickHouse queries when dealing with large data volumes.

Default: `-1`

Required: No

**NUMBER\_OF\_WORKER\_FOR\_HANDLE** `int`

Description: Number of workers for asynchronous profile processing during merge. Increasing this value speeds up processing through parallelism.

Default: `1`

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 requests, the request will not be recorded as duplicate. Default – 1 day

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 [procsmstlisten 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 events in Altcraft Platform [automation Scenarios](#):

### 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 must run on different hosts  
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 complete within the set time (300 seconds), the worker aborts it

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\_TRIGGERS\_CREATE\_WORKER\_SIZE** `int`

Description: Number of workers for creating triggers. If not specified, half of the number of CPU cores is used

Default: half of the number of CPU cores

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 scenario execution

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 always greater than 1.

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\_BATCHES\_LIMIT `int`

Description: Limit on the number of batches collected by the condition node worker.

If `COND_NODE_BATCHES_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 is greater than 0 — reporting is enabled

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 process for all records in the batch does not finish within the set time, the Scenario will return the error “Exceeded maximum processing time”

Default: 15000

Required: No

---

### Asynchronous batch profile status update

`ProfileStatusUpdater` is an asynchronous profile status updater for automation scenarios with buffering and batching via `BulkWrite`. Updates are grouped by `(accID, listID, profileID, workflowID)`, and only the latest update is kept for a given key. Flush occurs every 500 ms or when 500 records are accumulated. Updates are grouped by collection (`accID+listID`) and written via `BulkWrite(SetOrdered(false))`.

#### WORKFLOW\_PROFILE\_STATUS\_FLUSH\_MS `int`

Description: Interval for forced flush of the profile status update buffer (in milliseconds)

Default: 500

Required: No

#### WORKFLOW\_PROFILE\_STATUS\_FLUSH\_SIZE `int`

Description: Threshold of records in the buffer that triggers a forced flush

Default: 500

Required: No

#### WORKFLOW\_PROFILE\_STATUS\_RETRY\_MS `int`

Description: Initial backoff for retry attempts on partial BulkWrite failure (in milliseconds)

Default: 1000

Required: No

#### WORKFLOW\_PROFILE\_STATUS\_RETRY\_MAX `int`

Description: Maximum number of retry attempts for failed records from BulkWrite

Default: 3

Required: No

#### **WORKFLOW\_PROFILE\_STATUS\_QUEUE\_SIZE** `int`

Description: Size of the incoming queue (buffer for batch accumulation). When full, the worker waits; updates are not dropped

Default: 10000

Required: No

#### **WORKFLOW\_PROFILE\_STATUS\_FLUSH\_WORKERS** `int`

Description: Maximum number of profile update workers. Used for parallelizing updates across different collections (profile databases). If the batch contains profiles from 3 databases, 3 workers will be used instead of the specified maximum

Default: 8

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\_MEM\_LIMIT** `string`

Description: Disk memory limit allocated for sorting, e.g., “2G”, “500M”

Default: “10M”

Required: No

#### **SEGMENT\_SORT\_PARALLEL** `int`

Description: Number of CPU cores. If <= 0, all cores are used

Default: 1

Required: No

#### **SEGMENT\_SORT\_DEDUPLICATE** `bool`

Description: True if duplicates should be removed

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 written to disk in a single operation when this value is reached. Required: No

#### SEGMENT\_SQL\_DATA\_GROUP\_OPTIMIZATION bool

Description: Enables optimization for combining segmentation queries to external SQL databases. When enabled, the platform combines queries from the same connector and the same profile field into a single SQL query using INTERSECT, UNION, or EXCEPT operations, instead of executing them sequentially and intersecting results within the platform. Default: false Required: No

#### SEGMENT\_SQL\_SIS\_EXPANSION bool

Description: Enables segment-inside-segment expansion (InSegment). When enabled, the platform expands conditions of a nested dynamic segment and includes its external database queries in the general grouping for combination with the parent segment queries. Default: false Required: No

#### SEGMENT\_SQL\_SIS\_MAX\_DEPTH int

Description: Maximum depth of segment-inside-segment expansion. Value 0 means unlimited. Default: 3 Required: No

#### SEGMENT\_GROUP\_EXPANSION bool

Description: Enables expansion of redundant condition groups. When enabled, the platform expands nested groups with matching operators (AND in AND, OR in OR), single-child groups, and empty AND groups, so that queries from different nesting levels can be combined. Default: false Required: No

#### SEGMENT\_SQL\_SIS\_NOT\_EXPANSION bool

Description: Enables expansion of “Not in segment” conditions via De Morgan transformation. When enabled, the platform expands the nested segment, inverts all operators inside, and includes queries in the general grouping for combination. Default: false Required: No

#### SEGMENT\_SQL\_SIS\_DATATABLE\_INVERSION bool

Description: Disables inversion of conditions in data table (“Not in data table”). When disabled, it may affect segment calculation correctness. Default: true Required: No

#### SEGMENT\_ACTION\_GROUP\_OPTIMIZATION bool

Description: Flag to enable/disable the entire action\_group optimization (combining profile history conditions in a segment into a single query). Default: false Required: No

#### SEGMENT\_ACTION\_GROUP\_DATE\_TOLERANCE int

Description: Date range expansion threshold in days when combining profile history conditions. Value 0 means exact date range intersection only. Value greater than 0 sets a tolerance in days for range expansion. Value -1 combines all entries into one group regardless of dates. Default: 0 Required: No

#### SEGMENT\_ACTION\_GROUP\_OPTIMIZATION bool

Description: Flag to enable/disable the entire action\_group optimization (combining profile history conditions in a segment into a single query). Default: false Required: No

#### SEGMENT\_ACTION\_GROUP\_DATE\_TOLERANCE int

Description: Date range expansion threshold in days when combining profile history conditions. Value 0 means exact date range intersection only. Value greater than 0 sets a tolerance in days for range expansion. Value -1 combines all entries into one group regardless of dates. Default: 0 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 “Update segment daily” option is enabled. The start will occur in the account’s time zone

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 — delete all profiles matching the segment conditions

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\_CLEANNER\_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 parameters of size and connection frequency:

### **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\_PIPER\_METRIC\_HOST** `string`

Description: Host for the `procpiper` metrics HTTP server

Default: localhost

Required: No

**PROC\_PIPER\_METRIC\_PORT** `int`

Description: Port for the `procpiper` metrics HTTP server

Default: 8917

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 [prochhook 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 removed from the queue and a delivery failure event ( `expired` ) is recorded. If the value exceeds `MTA_MAX_TTL` , it will be capped to that limit.  
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 inherited from `MTA_DEFAULT_TTL`  
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 avoid data loss

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



#### SCHEDULE\_RUN\_INTERVAL\_MIN int

Description: Minimum allowed interval for scheduled tasks. Currently used for the start node of scheduled segments; planned to be extended to mailings and imports in the future

Default: 15

Required: No

#### WFCAPT\_SCHEDULE\_RUN\_INTERVAL\_MIN int

Description: Minimum allowed interval for launching the start node of a scheduled segment. If not set directly, inherits from SCHEDULE\_RUN\_INTERVAL\_MIN

Default: 0

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 customer data and registering target actions:

**APPMETRICS\_EVENTS\_REFRESH\_WORKERS\_PERIOD\_SEC** `int`

Description: Event handler cache cleanup frequency

Default: 60

Required: No

**APPMETRICS\_EVENTS\_REQUEST\_PERIOD\_SEC** `int`

Description: Frequency of event retrieval requests

Default: 60

Required: No

**APPMETRICS\_EVENTS\_DELAY\_COMPENSATIONS\_REQUEST\_PERIOD\_SEC** `int`

Description: If lag occurs, overrides `APPMETRICS_EVENTS_REQUEST_PERIOD_SEC`

Default: 600

Required: No

**APPMETRICS\_EVENTS\_REQUEST\_DATE\_RANGE\_SEC** `int`

Description: Event request time range

Default: 60

Required: No

**APPMETRICS\_PROC\_GOALS\_WORKER\_SIZE** `int`

Description: Number of goal registration handlers

Default: 2

Required: No

**APPMETRICS\_PROC\_GOALS\_PREFETCH\_SIZE** `int`

Description: "prefetch\_size" setting of the RMQ queue for goal registration

Default: 2000

Required: No

**APPMETRICS\_PROC\_IMPORTS\_WORKER\_SIZE** `int`

Description: Number of profile import handlers

Default: 1

Required: No

**APPMETRICS\_PROC\_IMPORTS\_PREFETCH\_SIZE** `int`

Description: "prefetch\_size" setting of the RMQ queue for profile import

Default: 2000

Required: No

**APPMETRICS\_EVENTS\_RESPONSE\_LIMIT** `int`

Description: Limit on the number of events received from AppMetrica

Default: 10000

Required: No

---

## Message templates settings

These parameters allow you to manage limits for content and attachment sizes in message templates.

### 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

### TEMPLATE\_HTML\_FIELD\_LIMIT\_BYTES int

Description: Maximum size of the HTML field in an email or custom channel template (in bytes)

Default: 512000 (500 KB)

Required: No

### TEMPLATE\_AMP\_FIELD\_LIMIT\_BYTES int

Description: Maximum size of the AMP field in an email template (in bytes)

Default: 512000 (500 KB)

Required: No

### TEMPLATE\_TEXT\_FIELD\_LIMIT\_BYTES int

Description: Maximum size of the text field (email text, SMS, push, custom channel) (in bytes)

Default: 102400 (100 KB)

Required: No

### TEMPLATE\_CONTENT\_LIMIT\_BYTES int

Description: Maximum total content size across all channels in a message (in bytes)

Default: 2097152 (2 MB)

Required: No

### TEMPLATE\_ATTACH\_LIMIT\_BYTES int

Description: Maximum attachment size in a message (email + MAX channel) (in bytes)

Default: 12582912 (12 MB)

Required: No

### TEMPLATE\_TOTAL\_LIMIT\_BYTES int

Description: Maximum total template size (content + attachments) (in bytes)

Default: 14680064 (14 MB)

Required: No

---

## Profile import and update

These parameters configure the processes responsible for creating and updating customer profiles, including automatic import of customer data from the company's SQL storage:

### 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 can find the instructions [at this link](#):

**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 Telegram chats:

### “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 [ClickHouse documentation](#):

**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\_timeout", "none")

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\_CERT\_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 set logging time and limits:

**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 the queue before being processed  
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 requests:

- `/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 send a message**:

**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 for the last 6 months to determine the optimal time to send messages

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 default, the model is already loaded during the build process and specifying this variable is not required. Models are provided upon request

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 default value -1 means the limit is disabled

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 SQL tables

Default: None

Required: No

**MAX\_PARALLEL\_ML\_ENRICH\_JOBS** `int`

Description: Maximum number of parallel runs of the machine-learning-based profile data recalculation process

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 interface

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. Default -1 means the limit is disabled

Default: -1

Required: No

---

### **Global control groups configuration**

With these parameters you can override [GCG](#) settings:

#### **GCG\_MIN\_SIZE** int

Description: Minimum size of a global control group

Default: 1000

Required: No

#### **GCG\_MAX\_SOURCE\_SIZE** int

Description: Limit on the source data size for a global control group. If the source exceeds this value, the platform uses only the first records up to the limit and will not display an error (only if the formation method is random distribution from a segment or profile database) Default: 50000000

Required: No

#### **GCG\_MAX\_PERCENT** int

Description: Maximum size of a global control group as a percentage of the data source (only if the formation method is random distribution from a segment or profile database)

Default: 25

Required: No

#### **GCG\_RUN\_HOUR** []int

Description: Overrides the start time for calculating global control groups (default 00:00 in the account time zone)

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 connection

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](mailto: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 mode)  
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 seconds)  
Default: 60

**AUTOCLEAN\_DOWNLOAD\_TASKREPORT\_MINUTES** `int`

Description: Interval in minutes for checking the directory and cleaning up stale files, default once per day  
Default: 1440  
Required: No

**MAX\_DOWNLOAD\_TASKREPORT\_FILE\_AGE\_MINUTES** `int`

Description: File lifetime in minutes, default 30 days  
Default: 43200  
Required: No

---

*The Meta organization, which owns Instagram, Facebook, and WhatsApp, is recognized as extremist and banned in the Russian Federation.*

# Domain settings

This guide contains examples of records that should be added for your domains.

The following IP addresses are used as examples. They only exist within this documentation and cannot be used as working addresses.

- 192.0.2.1 — IPv4 address for the admin panel;
- 192.0.2.2 — IPv4 address for message tracking;
- 192.0.2.3 — IPv4 address for website tracking;
- 192.0.2.4 — IPv4 address for the sender (AKMTA).

## Admin panel domain settings

Choose a subdomain and add an A record.

Domain	Record type	Contents
<a href="#">altcraft.example.com</a>	A	192.0.2.1

### NOTE: Where else this is configured

The domain and IP are set in `main.json` : `WEBCONTROL_HOSTNAME` and `WEBCONTROL_PUBLIC_IP` .

## Message tracking domain settings

Choose a subdomain for your from-domain and add an A record.

Domain	Record type	Contents
<a href="#">click.example.com</a>	A	192.0.2.2

Additionally, you can use as many tracking domains as needed. The settings will be the same, or you can use CNAME records.

### NOTE: Where else this is configured

The domain and IP are set in `main.json` : `TRACKING_HOSTNAME` (or `TRACKING_PREFIX` + `WEBCONTROL_HOSTNAME` ) and `TRACKING_PUBLIC_IP` . In the UI when creating a resource, the **Tracking Domain** field allows you to set a domain for a specific resource.

## Web site tracking domain settings

Choose a subdomain and add an A record.

Domain	Record type	Contents
<a href="#">pxl.example.com</a>	A	192.0.2.3

Additionally, you can use as many tracking domains as needed. The settings will be the same, or you can use CNAME records.

### NOTE: Where else this is configured

The domain and IP are set in `main.json` : `COOKIESAVER_PUBLIC_HOSTNAME` and `COOKIESAVER_PUBLIC_IP` .

## Sender domain settings (AKMTA)

Choose a domain or subdomain and configure the following records in your domain's DNS zone:

Domain	Record type	Contents
<a href="#">mta-1.example.com</a>	A	192.0.2.4
<a href="#">example.com</a>	MX	<a href="#">mta-1.example.com</a>

The PTR record is not configured in the domain's DNS zone, but in the reverse zone with your IP address provider:

Reverse zone	Record type	Contents
4.0.2.192.in-addr.arpa.	PTR	<a href="#">mta-1.example.com</a>

### NOTE: Where else this is configured

- **A-record** — the domain and IP are set in the Admin Panel when creating a **Node** (sender IP). The A-record must point to the same IP.
- **PTR-record** — configured with your hosting provider or IP management panel (reverse zone). The PTR value must match the A-record (e.g., `mta-1.example.com` ).
- **MX-record** — points to the A-record domain of the sender.

## SPF settings

Sender Policy Framework (SPF) is a technology that allows verifying whether the sender domain has been spoofed. To make it work, you need to add a special TXT record to your sending domain (sender domain, from domain), which defines the policy for allowing this domain to send from various hosts.

We recommend adding records for different versions:

Domain	Record type	Contents
example.com.	TXT	v=spf1 ip4:192.0.2.4 -all
example.com.	TXT	spf2.0/pra ip4:192.0.2.4 ~all

**CAUTION: A domain can have only one SPF record**

If an SPF record already exists on your domain, **do not create a new one** — it will overwrite the existing one. Instead, add `ip4:<your_IP>` to the existing record.

Check the current record with:

```
dig example.com TXT
```

Example output if a record already exists:

```
; <<>> DiG 9.18.28 <<>> example.com TXT
;; ANSWER SECTION:
example.com. 300 IN TXT "v=spf1 include:sendgrid.net ~all"
```

**Good result:** ANSWER SECTION contains one record starting with `v=spf1`. **Bad result:** `NXDOMAIN` (record not found) or multiple `v=spf1` records (this violates the standard).

If a record is found, combine mechanisms into a single line:

Was	Became
v=spf1 include:sendgrid.net ~all	v=spf1 include:sendgrid.net ip4:192.0.2.4 -all

The directive `-all` (or `~all`) must remain **single** and be placed at the **end** of the record.

**NOTE: Where else this is configured**

The domain used for MAIL FROM is determined in the Admin Panel:

- **Senders** — `mail_from_type` ( `from_email`, `ptr`, `custom`, `aligned` ), `sender_domain`, `mail_from_subdomain`, `pool_id`
- **Sender pools** — `Sender domain`, `SPF domain`, `Sender subdomain`

Make sure the SPF record for your domain allows sending from the domain that will be used in MAIL FROM.

...

**DKIM settings**

DomainKeys Identified Mail (DKIM) technology adds a digital signature to the email, linked to the from-domain. The signature is automatically verified on the recipient side, then used to refine reputation and marked for the user. To sign the email, a private key is used, which is set up on the sender side and is unknown to anyone else. The public key is placed as a special TXT record on a subdomain of the from-domain.

In the platform Admin Panel, you can generate or add an existing key. Choose a selector — for example, `altcraft`. The selector is the prefix in the DKIM subdomain name. Selectors allow different senders to use the same domain: for example, `altcraft._domainkey.example.com` for Altcraft and `mail._domainkey.example.com` for another service.

Domain	Record type	Example contents
altcraft._domainkey.example.com.	TXT	v=DKIM1; k=rsa; p=AAAAAB3NzaC1yc2EAAAADAQABAAQDLWvAVr5k+4nneMRKhUeb/yGjRB1pGu9CLx4LJ3b1dUZeNmBCbjpJN8P5h1XD3tyml8aB3dHwtudeBvauUmxoxIfnrkQW6XNwiplAXB/2bsECd1k51PRnWzGAt8WOVI/z
_domainkey.example.com.	TXT	0=-;

**NOTE: Important!**

For one sender in the Admin Panel, you can select only one key.

**NOTE: Where else this is configured**

DKIM key and selector are set in the Admin Panel:

- **Keys** — creation (generate key pair) or paste an existing key. Key type: `dkim`.
- **Senders** — `From domain DKIM key` (select from list) and `From domain DKIM selector` (e.g., `altcraft`). If `mail_from_type` is not `aligned`, additionally set `Secondary DKIM key` and `Secondary DKIM selector`.
- **Sender pools** — `DKIM key` and `DKIM selector` (used when `mail_from_type = aligned`)



The selector in DNS must match the selector in the sender settings. For example, if the sender specifies selector `altcraft`, the DNS record should be `altcraft._domainkey.example.com`.

## DMARC settings

DMARC technology allows the mail server to decide what to do with email if something is wrong with the DKIM and SPF records.

We recommend setting rules that reject messages received from other IP addresses or signed incorrectly.

Domain	Record type	Example contents
_dmarc.example.com.	TXT	v=DMARC1; p=reject; sp=reject; rua=mailto:report@example.com

You will be able to receive reports at `report@example.com`.

### NOTE: Where else this is configured

DMARC is configured only in DNS. The report email (`rua=mailto:...`) is specified by the domain administrator. DMARC verifies that the domain in `From` aligns with SPF and DKIM results — make sure the from-domain (determined by the **Sender Email** field in the resource or template) matches the domain for which SPF and DKIM are configured.

## Additional settings

### BIMI (optional)

BIMI technology ([Brand Indicators for Message Identification](#)) is used together with SPF, DKIM, and DMARC to display a branded image next to the email in the recipient's inbox. This setting is optional — it is needed if you want your company logo to appear next to emails in supporting mail clients.

To use this technology, you need to specify a path to a file in **svg** format. The image should be square and have no additional layers.

Add the following record in your domain settings:

Domain	Record type	Example contents
default._bimi.example.com	TXT	v=BIMI1; l=https://example.com/bimi/bimi.svg

You can also use the record generator at <https://bimigroup.org/bimi-generator/>.

## Troubleshooting

If DKIM, SPF, DMARC, or antis spam headers show poor values after a test send, check your domain settings, sender configuration, and email content.

### DKIM, SPF, or DMARC fail verification

If the email headers show `dkim=fail`, `spf=fail`, or `dmarc=fail`, the issue is most likely related to domain configuration.

Check:

- DKIM selector in the admin panel;
- DKIM key in the domain DNS record;
- SPF record for the domain;
- DMARC record for the domain;
- domain in the `From` field;
- reverse address domain;
- domain alignment for DMARC to pass.

If the domain uses a single DKIM selector, the admin panel must specify the same selector. For example, if the DNS publishes the record:

`mail._domainkey.example.com`

then the admin panel must specify the selector for DKIM as:

`mail`

If the admin panel specifies a different selector, the mail server will not find the correct DKIM key or will verify the email against a different DNS record.

Also check the DKIM key itself. The public key value in DNS must match the key used to sign the email. If the key was replaced in DNS but not updated in the admin panel, DKIM may fail verification.

### SPF fails verification

If the headers show `spf=fail`, check the SPF record for the domain used in the reverse address.

The reverse address is usually displayed in the header:

```
Return-Path: <bounce@example.com>
```

or in the SPF verification result:

```
spf=fail smtp.mailfrom=bounce@example.com
```

SPF is verified not against the visible sender address in `From`, but against the domain from `smtp.mailfrom` / `Return-Path`.

Check that the domain's SPF record allows sending from the required server or service. For example:

```
example.com. TXT "v=spf1 ip4:192.0.2.4 -all"
```

If sending through an external service, it must be added to the SPF record via `include` or another recommended mechanism.

Also check that the domain does not have multiple SPF records. For a single domain, there should be one TXT record starting with:

```
v=spf1
```

### DMARC fails verification

If DKIM and SPF show `pass`, but DMARC shows `fail`, the issue is most often related to mismatched domains.

Example of a problematic situation:

```
From: News <news@example.com>
Return-Path: <bounce@mailer-service.com>
```

```
Authentication-Results: mx.example.com;
dkim=pass header.i=@mailer-service.com;
spf=pass smtp.mailfrom=mailer-service.com;
dmarc=fail header.from=example.com
```

In this case, DKIM and SPF pass verification, but not for the `example.com` domain specified in `From`.

For DMARC to pass, at least one of the following conditions must be met:

- DKIM passes verification, and the DKIM domain aligns with the domain in `From`;
- SPF passes verification, and the `smtp.mailfrom` domain aligns with the domain in `From`.

If a different domain is used in the `From` field, it must also be configured: it needs DKIM, SPF, and DMARC records.

### A different From-domain is used

Issues may arise if the email uses one sender domain, but the settings are configured for a different domain.

For example, the admin panel has the domain configured:

```
example.com
```

but the email is sent from the address:

```
From: News <news@promo.example.net>
```

In this case, DKIM, SPF, and DMARC must also be configured for the `promo.example.net` domain. If this domain is not configured, verification may fail.

Before a test send, make sure the domain in `From` matches the domain configured in the admin panel, or is a properly configured subdomain of it.

### Admin panel values override other settings

Settings specified in the admin panel may override values from other locations. For example, if the panel specifies a separate DKIM selector or reverse address, these values will be used when sending.

Check the settings for:

- sender domain;
- sender;
- virtual sender;
- reverse address;
- DKIM selector;
- template or campaign, if they allow specifying the sender manually.

If the settings differ, the email may go out with a different `From`, different `Return-Path`, or different DKIM signature.

### Domain or IP address DNS records not configured

For proper sending, DNS records for the domain and sending IP address are required.

Check:

- `A` record for the domain;
- `MX` record for the domain;
- `PTR` record for the sending IP address;
- correctness of the reverse address;
- domain availability in DNS.

The `PTR` record is especially important for the sending IP address. It links the IP address to a domain name and is used by mail systems as one of the reputation signals. The PTR record is configured with your hosting provider or IP management panel, not in the domain's DNS zone.

### Verifying records with `dig`

Check basic records:

```
dig example.com A
dig example.com MX
dig example.com TXT
```

Example successful output for an A-record:

```
; <<>> DiG 9.18.28 <<>> example.com A
;; ANSWER SECTION:
example.com. 300 IN A 192.0.2.4
```

**Good result:** ANSWER SECTION contains a record with an IP address. **Bad result:** `NXDOMAIN` (domain not found) or empty ANSWER SECTION.

Check the PTR record (reverse DNS) for the sender IP:

```
dig -x 192.0.2.4
```

Example successful output:

```
; <<>> DiG 9.18.28 <<>> -x 192.0.2.4
;; ANSWER SECTION:
4.0.2.192.in-addr.arpa. 300 IN PTR mta-1.example.com.
```

**Good result:** PTR points to a domain matching the A-record (`mta-1.example.com`). **Bad result:** `NXDOMAIN` (PTR not configured) or PTR points to a different name.

Check the DKIM record (replace `altcraft` with your selector):

```
dig altcraft._domainkey.example.com TXT
```

Example successful output:

```
; <<>> DiG 9.18.28 <<>> altcraft._domainkey.example.com TXT
;; ANSWER SECTION:
altcraft._domainkey.example.com. 300 IN TXT "v=DKIM1; k=rsa; p=MIGfMA0GCS..."
```

**Good result:** record found, starts with `v=DKIM1; k=rsa; p=`. **Bad result:** `NXDOMAIN` (selector not found) or `p=` key is empty.

Check the DMARC record:

```
dig _dmarc.example.com TXT
```

Example successful output:

```
; <<>> DiG 9.18.28 <<>> _dmarc.example.com TXT
;; ANSWER SECTION:
_dmarc.example.com. 300 IN TXT "v=DMARC1; p=reject; sp=reject; rua=mailto:report@example.com"
```

**Good result:** record found, contains `v=DMARC1` and policy `p=`. **Bad result:** `NXDOMAIN` (DMARC not configured) or policy `p=none` (not recommended for production).

### Domain or IP address not yet warmed

Even with correct DKIM, SPF, and DMARC, an email may receive a poor antispam score if sending started at a high rate to an unwarmed domain or IP address.

A new domain or new IP address requires gradual warming. Start with a small volume of sends and increase the rate gradually.

If you launch mass sending at a high rate immediately, mail systems may start rejecting emails, sending them to spam, or lowering the sender's reputation.

During warming, monitor:

- deliverability;
- opens and clicks;
- spam complaints;
- bounces;
- spam placement;
- domain and IP address reputation.

### Email fails antispam filters

If DKIM, SPF, and DMARC pass verification, but the email receives a high `SCL`, `BCL`, or ends up in spam, the issue is most likely not related to domain authentication, but to email content or sender reputation.

Different mail providers use different antispam filters with different headers and scores. For example, SpamAssassin uses `X-Spam-Status` and `X-Spam-Score`, Microsoft 365 uses `SCL` and `BCL`, and Gmail does not publish numeric scores. For universal checking, use online testers (see below).

Check:

- email subject;
- email body text;
- HTML markup;
- text-to-image ratio;
- links;
- domains in links;
- attachments;
- presence of URL shorteners;
- domain reputation;
- IP address reputation;
- sending frequency;
- complaint and bounce history.

The email should look acceptable to antispam filters: no suspicious links, aggressive wording, unnecessary attachments, broken markup, or phishing indicators.

### Useful tools for verification

For diagnostics, you can use external services:

- [Mail-Tester.com](#) — send a test email to a unique address, receive a 0–10 score with detailed analysis of DNS, headers, SPF, DKIM, DMARC, and content.
- MXToolbox — DNS record, SPF, DKIM, DMARC, PTR, and blacklist checking.
- Google Postmaster Tools — monitoring domain reputation, IP addresses, delivery errors, and complaints for Gmail.
- Postmaster [Mail.ru](#) — monitoring reputation and send quality for [Mail.ru](#).
- Microsoft SNDS — monitoring IP address reputation in Microsoft infrastructure.

After making corrections, perform another test send and check the email headers again. A normal authentication result:

```
dkim=pass
spf=pass
dmarc=pass
```

If you use Microsoft 365 or Outlook, additionally check:

```
compauth=pass
SCL:0
BCL:0
```

# LDAP access configuration

## Overview

You can configure LDAP access to your Altcraft Platform account and Administrative panel. This allows your employees to use company’s directory service like Active Directory to log into the platform with a predefined set of [access groups and roles](#).

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` configuration file:

```
"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_CERT_PATH": "/etc/ssl/certs/ca.crt",
 "KEY_PATH": "/etc/ssl/private/ldap_client.key"
},
"LDAP_CHECK_USER_TIME_PERIOD_SEC": 300
```

Parameter	Type	Description
Main parameters		
<code>LDAP_ONLY</code>	bool	Prohibits access to the admin panel using login/password created within the platform. All authentication requests are directed to the LDAP server.
<code>PRIMARY_HOST</code>	string	Network address of the primary LDAP server.
<code>PRIMARY_PORT</code>	int	Port of the primary LDAP server (usually 389 for LDAP, 636 for LDAPS).
<code>SECONDARY_HOST</code>	string	Network address of the backup LDAP server for fault tolerance.
<code>SECONDARY_PORT</code>	int	Port of the backup LDAP server.
<code>BINDLOGIN</code>	string	Login of the account for connecting to the LDAP server (e.g., <code>cn=admin,dc=example,dc=com</code> ).
<code>BINDPASS</code>	string	Password of the account for connecting to the LDAP server.
<code>BASE_DN</code>	string	The base DN (Distinguished Name) from which the directory search begins.
<code>AUTH_GROUP</code>	string	The distinguished name of the LDAP access group that determines employee access to the admin panel.
Secure connection parameters		
<code>IS_SSL</code>	bool	Enables LDAPS mode (LDAP over SSL). The connection is encrypted from the start on a separate port (usually 636).
<code>START_TLS</code>	bool	Enables the STARTTLS extension. Encryption is established on top of the regular connection on the standard LDAP port (389) after executing a special command.
<code>CRT_PATH</code>	string	Path to the client certificate file (if mutual authentication is required).
<code>CA_CERT_PATH</code>	string	Path to the Certificate Authority (CA) certificate file.
<code>KEY_PATH</code>	string	Path to the client's private key file.
Additional parameters		
<code>LDAP_CHECK_USER_TIME_PERIOD_SEC</code>	int	Time interval (in seconds) for checking the compliance of user rights with the granted access. Set outside the LDAP object. Default: 300.

INFO

- The account for connecting to the server ( `BINDLOGIN` / `BINDPASS` ) must have sufficient rights to read the attributes of the group specified in `AUTH_GROUP` .
- The `IS_SSL` and `START_TLS` parameters are mutually exclusive — only one encryption mode can be enabled at a time. More information about the STARTTLS mechanism can be found in [this article](#).

After making changes to the `main.json` configuration file, restart Altcraft Platform ( `./akd restart` ).

## LDAP User interface access

LDAP access to account User interface is set up for every account separately in Altcraft **Administrative panel**.

## LDAP binding creation

Enter Altcraft Platform Administrative panel and open **Create → LDAP binding** or **Setup → LDAP binding** and press **Create**.

## LDAP Binding #new

Name:

Host:  Port:

Username:  Password:

☒ Use SSL [?](#)

Specify LDAP connector name, directory server network address and port. Enter **Username** and **Password** for directory server database access. If you are planning to use a secure connection activate **Use SSL** option and set up directory service SSL certificates.

### CAUTION

Directory service connection account must have sufficient access rights to the groups that will be used for platform authorisation.

### Setting up account LDAP access

Open an existing account settings or create a new account. Activate **Use LDAP** option to get access to the settings:

### NOTE

**LDAP only** option restricts Altcraft login and password authorisations. All requests will be processed via LDAP to directory service.

Time interval between checking user DistinguishedName and access rights matching is configured in `main.json` configuration file in `LDAP_CHECK_USER_TIME_PERIOD_SEC` property with an integer value. By default — 300 seconds.

**LDAP options**

LDAP only

Assigned LDAP connector:

new LDAP binding #1

Authorized group:

CN=Altcraft-Users,OU=AltcraftMP,DC=ad,DC=ldap-server,DC=devel

Master group:

CN=Altcraft-Master\_user,OU=AltcraftMP,DC=ad,DC=ldap-server,DC=devel

Default groups:

Corporate clients project ✕ default ✕

Default roles:

default ✕

Groups matching:

CN=East\_coast,OU=AltcraftMP,DC=ad,DC=ldap-server,DC=devel

main ✕

✕

CN=East\_cowst,OU=AltcraftMP,DC=ad,DC=ldap-server,DC=devel

main ✕

HR recruiting project ✕

✕

+ Add Group

Roles matching:

CN=East\_coast,OU=AltcraftMP,DC=ad,DC=ldap-server,DC=devel

email designer ✕

✕

+ Add Role

In a dropdown select an **Assigned LDAP connector**: LDAP binding, you are going to use for this account access.

### Basic access settings

For configuring basic account access you will need directory service DistinguishedNames for the following groups:

- **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 and to objects outside groups.

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 used: **CN** — common name, **OU** — organisation unit and **DC** — domain component.

Select or create **Default Groups** and **Default Roles** to be accessible by default for all users with **Authorized group** DistinguishedName.

#### NOTE

**WARNING! Users with no assigned groups or roles cannot log into account.**

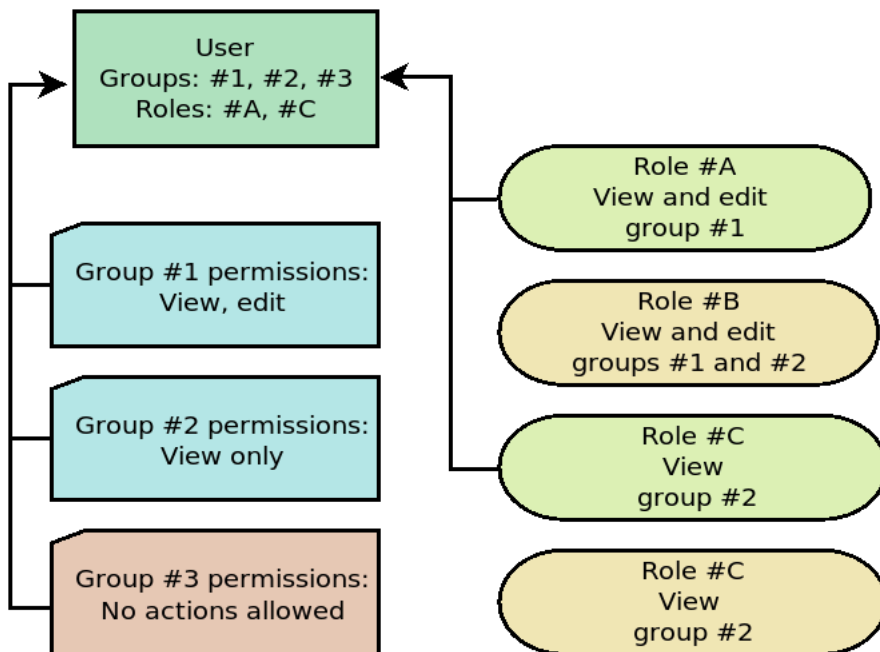
In case a user **role** allows to interact with a certain **group** make sure the corresponding group is assigned to the user as well. Otherwise no access to the group will be permitted.

### Access differentiation

You can create directory service DistinguishedNames matching custom Altcraft Platform [groups and roles](#).

Enter the DistinguishedNames into these fields:

- **Groups matching** — to set access groups matching.
- **Roles matching** — to set access roles matching.



#### Platform Roles Mode

By default, every time an LDAP user logs in, the platform **overwrites** their roles and groups based on LDAP group mapping and **Default Roles/Default Groups** settings. This means roles assigned manually through the platform UI are lost on the user's next login.

If all employees in your organization belong to a single LDAP group but need different roles in Altcraft, enable the **Platform roles** mode:

## Authorization

☐ Force 2fa for users ?

☐ Use LDAP ?

☒ Use Keycloak ?

☐ Keycloak only ?

☒ Platform roles ?

\* Assigned Keycloak connector ?

When **Platform roles** is enabled:

- LDAP is used **only for authentication** — password check and **Auth Group** membership verification.
- User roles and groups are **not overwritten** on login or session refresh.
- Roles and groups are assigned manually through the platform UI ([Users](#)) just like for regular users.
- The following fields continue to sync from LDAP: email, first name, last name, active status (**IsActive** — based on **Auth Group** membership) and master status (**IsMaster** — based on **Master Group** membership).

#### TIP

Use this mode when:

- All users belong to one Active Directory group, but different roles are needed in Altcraft.
- You want to manage roles through the platform UI instead of LDAP group mapping.

Do not enable this mode if you need automatic LDAP group-to-role mapping (default behavior).



A similar **Platform roles** mode is available for [Keycloak integration](#).

## 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 password.

If **LDAP only** option is disabled, both Altcraft Platform 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 about security and data loss prevention, that will add a relay (proxy or DLP) to your sending process.

### 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 Platform Administrative panel [create a sender](#) and enable “Send via SMTP relay”.

## Sender #1

Name:

Default sender (via SMTP relay: relay.altkraft.com)

Type:

Email: AKMTA

Sender Domain Type:

Use custom domain

Sender Domain:

sending-server.devel

Additional mailbox domains:

One domain per line

### Mail boxes access



Host	Port	Username	Password
10.10.0.15	110	all@sending-server.devel	K4qrReRL3jHxfQIE

New password

Sender Domain Key:

Disable DKIM

Sender Domain DKIM Selector:

From Domain Key:

Disable DKIM

From Domain DKIM Selector:

IP Addresses:

1 - Default node - 10.10.0.15 x

☒ Send via SMTP relay

SMTP Relay Host:

relay.altkraft.com

### 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 support DSN. Otherwise it will resend the report to the receiving server. The solution here is to disable DSN for all servers.

### 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/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 and delivery via HTTPS.

This standalone domain is required in any infrastructure design where you want to use behavior tracking and push [notifications.it](#) uses cookies for user identification — so it is highly advisable not to change this domain in the future.

There are two ways of pixel handling in Altcraft Platform, see [Infrastructure design variations](#) for details.

The following terms are used in the article below:

- **Control server** — the main server or server group of Altcraft Platform, processing control panel and databases
- **cookiesaver** — Altcraft Platform service processing pixel and push domain
- **main.json** — the main Altcraft Platform configuration file, placed in **config** directory of Altcraft Platform installation directory.
- **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 interfaces. NAT usage is also possible.

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 px1.example.com;
 ssl on;
 ssl_certificate /etc/letsencrypt/live/px1.example.com/fullchain.pem;
 ssl_certificate_key /etc/letsencrypt/live/px1.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 Platform installation directory:

```
./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.

☒ Enable Pixel

☒ Enable Pushes

To make sure the pixel domain is functional, check if hyperlinks in sent messages are processed within the new domain:

```
wget http://trk.example.com/0f1d00e4d93978fc/TdguHgCa<cut>w 2>&1 | grep Location:
Location: https://px1.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 subscribers.

The following pixel and push configuration is set up in Altcraft Platform user interface: [pixel settings](#), [web push: settings & implementation](#)

# Cluster and Replication Setup

## Deployment Scheme



### DANGER

For passwords and secret combinations, the character set abcdefghijklmnopqrstuvwxyzABCDEF is used. Be sure to generate a more complex password consisting of at least 32 characters.

## Percona Server for 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/docs/manual/replication/>

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 Percona Server for MongoDB, connect to the database via `mongosh` 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
 }
]
}
```

#### INFO: Number of Percona Server for MongoDB replicas

With the right approach, the number of Percona Server for MongoDB replicas with data should be **odd**.

If an even number of replicas with data is used, the configuration usually includes an **arbiter**. An arbiter can participate in voting when electing the primary replica, but it does not store data or acknowledge writes. This option is used when there are not enough resources for a full-fledged replica with data.

In this case, we recommend configuring the `defaultWriteConcern` parameter. It defines how many Percona Server for MongoDB replicas must acknowledge a data write for the write to be considered successful.

For example:

```
db.adminCommand({ setDefaultRWConcern: 1, defaultWriteConcern: { w: 3 } })
```

With this configuration, at least 3 replicas must acknowledge the write. The `w` value is selected individually based on the number of replicas with data and fault tolerance requirements.

## ClickHouse Cluster Setup

According to the [deployment diagram](#), the replica (a single replicated shard) and Zookeeper are hosted on **db1**, **db2**, and **db3**.

#### 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 current host's name.

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 the configuration **[db1, db2, db3]**:

```

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-docs/replication.html>

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"
 }
]
}

```

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 "ABCDEFGHJKLMN" > /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-mode":"automatic"}'; done
```

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 status [**db1**, **db2**, **db3**]:

```
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.json configuration](#) [app1, app2]:

```
{
 "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":"2023-10-17T07:00:00Z","message":"Altcraft Platform [proctask] v2023.3.65.2060-g8377e3e65b ((HEAD) @Now app1 started on app1:8962, is master true"}
{"level":"info","proc":"procworkflow","pid":306827,"hid":"app1","ver":"v2023.3.65.2060","file":"workflow/controlworkflow.go:108","time":"2023-10-17T07:00:00Z","message":"Altcraft Platform [procworkflow] v2023.3.65.2060-g8377e3e65b ((HEAD) @Now started, workers: 6, process: 5896e0fa252eefa7_306827, host_id: app1, is_primary: true, rpc addr: app1:7962"}
```

## Configuring Custom Reader Groups

You can configure **custom reader groups** that will process tasks for specific accounts only. By default, there is a single reader group called `default`. If an account has no assigned reader group, its tasks are automatically placed into the `default` group.

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 exclude account 2 from the acc1-5 group in the example, the service will fail to start. After you make the changes, each process will handle tasks only for the accounts specified in its respective group. If no group parameters are set for a slave host, it will process tasks from the default group.

#### High-priority scenarios

Tasks of [high-priority scenarios](#) are processed by a dedicated worker pool on the hosts that have `HIGH_PRIORITY_WORKERS: true` set in `PROC_WORKFLOW`. If the parameter is not set, the dedicated pool is enabled on the master host (`FORCE_MASTER_ROLE: true`) and disabled on all other hosts.

To move high-priority tasks to a separate host **app3**:

```

{
 "PROC_WORKFLOW": [
 {
 "HOST_ID": "app1",
 "RPC_HOST": "0.0.0.0",
 "RPC_HOST_CLI": "app1",
 "RPC_PORT": 7962,
 "FORCE_MASTER_ROLE": true,
 "HIGH_PRIORITY_WORKERS": false
 },
 {
 "HOST_ID": "app2",
 "RPC_HOST": "0.0.0.0",
 "RPC_HOST_CLI": "app2",
 "RPC_PORT": 7962,
 "FORCE_MASTER_ROLE": false
 },
 {
 "HOST_ID": "app3",
 "RPC_HOST": "0.0.0.0",
 "RPC_HOST_CLI": "app3",
 "RPC_PORT": 7962,
 "FORCE_MASTER_ROLE": false,
 "HIGH_PRIORITY_WORKERS": true
 }
]
}

```

In this example, the dedicated pool runs only on **app3**: its workers process high-priority tasks first and pick up regular ones when there are none. The other hosts process only regular tasks.

A cluster without priority hosts does not start: if the parameter is explicitly set to `false` on the master host and is unset or set to `false` on all other hosts, the `procworkflow` service does not start and writes the following error:

```

invalid cluster config: HIGH_PRIORITY_WORKERS is effectively
disabled on all PROC_WORKFLOW nodes, no workers would process
priority tasks

```

# System notifications configuration

## Mailbox configuration

To allow Altcraft Platform to send weekly and daily reports, credentials changes and other email notifications, `NOREPLY_MAILER` object is configured in `main.json` file.

This configuration consists of SMTP access credentials for notifications mailbox. If you do not have a dedicated mailbox, ask your system administrator to create one like [altcraft@example.com](mailto:altcraft@example.com), where `example.com` is your corporate email domain.

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	Description
server	string	Yes	Email server domain
port	int	No	Port. By default - 25
email	string	Yes	Email address
login	string	*	Email server username, often the same as email address
pass	string	No	Email server password
helo	string	No	Email server greeting mode. By default - PTR record
insecure	bool	No	Disable SSL certificate verification. By default - false
force_no_ssl	bool	No	Disable SSL including standard ports. By default - false

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,
 "PERFORMANCE_WARNING_PERCENT": 90,
 "SELECT_INTERVAL_HOURS": 1,
 "NOTIFY_INTERVAL_HOURS": 12,
 "MINIMAL_DIVIDER": 1000
}
```

## Notification parameters

Parameter	Description
HARDBOUNCES_WARNING_PERCENT SOFTBOUNCES_WARNING_PERCENT COMPLAINS_WARNING_PERCENT	Define <b>Hard bounces</b> , <b>Soft bounces</b> and <b>Complaints</b> warning thresholds.  By default – 1, 30 and 0.07.
PERFORMANCE_WARNING_PERCENT	Defines the threshold for triggering a notification when the message <b>delivery rate</b> drops (in percentage terms).  By default – 90.
SELECT_INTERVAL_HOURS	Assessment time interval for data aggregation.  By default – 1 hour. Maximum – 24 hours.
NOTIFY_INTERVAL_HOURS	Duplicated notifications time interval.  By default – 12 hours (If the same problem occurs within 12 hours - no new notification will be sent).
MINIMAL_DIVIDER	Minimum deliveries number to aggregate statistics.  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-cokera:

Parameter	Example
"COOKIESAVER_UNIXSOCK_PATH"	"/opt/akd/socks/akd-cookiesaver.socket"
"PUSH_HTTP_SERVER_UNIXSOCK_PATH"	"/opt/akd/socks/akd-pushserver.socket"
"SMS_LISTENER_UNIXSOCK_PATH"	"/opt/akd/socks/akd-smslistener.socket"
"TRACKING_IMAGE_UNIXSOCK_PATH"	"/opt/akd/socks/akd-imgtrackingnew.socket"
"TRACKING_PUSH_UNIXSOCK_PATH"	"/opt/akd/socks/akd-trackingpush.socket"
"TRACKING_READ_UNIXSOCK_PATH"	"/opt/akd/socks/akd-trackingnewread.socket"
"TRACKING_SMS_UNIXSOCK_PATH"	"/opt/akd/socks/akd-trackingnewsms.socket"
"TRACKING_UNIXSOCK_PATH"	"/opt/akd/socks/akd-trackingnew.socket"
"TRACKING_WEB_VERSION_UNIXSOCK_PATH"	"/opt/akd/socks/akd-trackingwebversionnew.socket"
"WEBAPI_UNIXSOCK_PATH"	"/opt/akd/socks/akd-api.socket"

## 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 any changes.

## 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 Community to Percona Server

## NOTE

For new installations, Percona Server for MongoDB is installed by default. See [Manual installation](#).

This article is intended for customers who already use MongoDB Community Edition and want to migrate to Percona Server for MongoDB.

This article describes how to replace **MongoDB Community Edition** with **Percona Server for MongoDB** with disk encryption enabled.

The standard in-place upgrade method (replacing binary files on the existing server) **is not suitable** when encryption needs to be enabled, because existing data files cannot be encrypted in place. To solve this, a temporary **MongoDB replication** setup is used.

The procedure involves minimal platform downtime. Most of the migration is performed without stopping Altcraft Platform, with a brief downtime required only during the final switchover to the new server.

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 ( `repSet` ) is enabled, and the server becomes PRIMARY.
2. A second clean server is provisioned with Percona Server for MongoDB installed and encryption enabled.
3. The second server is added to the replication set and performs an initial sync. During synchronization, it receives data from the PRIMARY and writes it to disk in encrypted form.
4. After the initial sync completes and replication status is verified, the PRIMARY role is switched to the new server.
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 for the new server to complete the initial sync and catch up with changes written to the database during synchronization.

## Check MongoDB Version

On the current server, ensure that the MongoDB Community version matches the Percona Server for MongoDB version to be installed:

```
/usr/local/mongodb/bin/mongod --version | head -1
```

The Percona Server for MongoDB version must be from the same major branch as MongoDB Community. For example: MongoDB 8.0.10 → Percona Server for MongoDB 8.0.x.

## 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 packages:

```
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 an encrypted partition or container, with permissions set to `600`.

Disk encryption reduces the risk of data exposure in case of media compromise, such as disk theft or improper hardware disposal.

## Using HashiCorp Vault Instead of a Local Key File

Instead of a local `encryptionKeyFile`, Percona Server for MongoDB can retrieve the master key from HashiCorp Vault.

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 container.

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 significant lag behind the PRIMARY.

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 changing the primary server.

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 decommissioned.

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 new server:

```
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 Platform administrative panel open node settings and save it to enable new IPs.
- 3. Reopen settings to make sure the new IP addresses are available:

MTA Settings

Priority channel size: \*

1000

Other channel size: \*

1000

Spool channel size: \*

10000

Spool writers num: \*

4

Retry channel size: \*

1000

Eventor channel size: \*

10000

Cleaner spool timeout days: \*

3

Cleaner events timeout days: \*

7

Cleaner attach timeout days: \*

3

FBL header name: \*

X-AKMTA-FBL

SMTP Listen:

127.0.0.1:25, default listening on all sending IP

SMTP Administrator email:

sample@example.com, default using from sender

RMQ Consumers count / Sender: \*

1

Write SMTP Responses to syslog

☒

Enable MTA Observer

☒

Node IP's:

10.0.2.46

192.168.2.1

192.168.2.2

192.168.2.3

192.168.2.4

192.168.2.5

192.168.2.6

192.168.2.7

192.168.2.8

192.168.2.9

192.168.2.10

- 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 request. A repeated request can be sent if the connection failed at the moment of receiving the data.

The platform will not accept a repeated request if it changes the data in order to avoid duplicate events. This scheme works by default and does not require additional configuration. By default, the platform deduplicates within 1 day from the first request.

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	Description
API_DEDUPLICATION_ENABLE	bool	true	No	Enable deduplication
API_DEDUPLICATION_LEVELDB_PATH	string	data/api/deduplication	No	LevelDB storage location directory
API_DEDUPLICATION_LIVE_TIME_SEC	int	86400	No	The maximum number of seconds between identical requests. If more time passes between requests, the request will not be treated as a repeated one.  by default – 1 day
API_DEDUPLICATION_PERIOD_SEC	int	1800	No	Deduplication database cleanup period (in seconds)  by default – 30 minutes

# PostgreSQL database for Market data

## PostgreSQL installation

PostgreSQL must be connected to enable the [Market](#) module. For correct operation, **PostgreSQL version 16.1 or higher** is required.

### INFO

You can find PostgreSQL installation instruction for Ubuntu [here](#).

## User and database configuration

### CAUTION

A standard variant with local database installation is described. In this example we use `altcraft` as the username, but you can specify any arbitrary name.

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 **Settings — Databases** section. It is necessary to create a new configuration for connecting to the database; to do this, click the **Create** button.

Select the connection type **Account DB Postgres**. Specify the host, port and credentials of the user created in PostgreSQL. To test the connection, click **Test connection**:



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

Test connection

\* Username

altcraft

Password

altcraft

•

Additional dsn options

key1=value1;key2=value2;

☐ With SSL

Save

Close

Add the created configuration to your account. To do this, edit your account and select the configuration in the **Account databases** item:

Account databases

Postgres DB ?

5 - market

▼

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 no proxy server address is specified for any of the connections, the platform will use the value from the `HTTP_PROXY` parameter. You can also set "noproxy" for any of the parameters, in which case the proxy will not be used for this connection.

Parameter	Description
HTTP_PROXY	Proxy server for http connections
PREVIEWS_PROXY	Proxy for previews and thumbnails
LICENSE_PROXY	Proxy server for http connections to the License Server
YM_PROMOCODES_PROXY	Proxy server for Yandex.Metrics
NET_GOOGLE_PROXY	Proxy server for Google Ads audiences
NET_FACEBOOK_PROXY	Proxy server for Facebook Ads Manager™*
NET_VK_ADS_PROXY	Proxy for VK Ads

## Exceptions for proxy server

The platform may access the user's internal resources for some information (for example, data for templates or for API requests). If the user has a proxy server configured, it will not be possible to get data from the local address. For such situations, you can specify **exception list** in the configuration file. It 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 validation sequence for the proxy server in such a case would look like this:

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 extremist and banned in Russia.*

# Keycloak Integration with Altcraft

Access to Altcraft can be provided via the **Keycloak centralized authentication system**. This allows using a single user and access management service to sign in to the platform, with groups and roles assigned according to the configuration in Keycloak.

Keycloak supports modern authentication and authorization protocols (OpenID Connect, OAuth 2.0, SAML) and provides centralized management of user accounts, groups, and access permissions.

## 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 belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

---

Resource file	Drag a file here or browse to upload    1      Upload a JSON file	Browse...   Clear
Realm name *	oauth_app	
Enabled	On	
	Create   Cancel	

For isolated testing, enable user registration. In **Realm Settings** → **Login**, enable the **User Registration** parameter to allow user self-registration.

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#)

- General
- Login
- Email
- Themes
- Keys
- Events
- Localization
- Security defenses
- Sessions
- Tokens
- Client policies
- User profile
- User registration

Login screen customization

- User registration  ☒ On
- Forgot password  ☐ Off
- Remember me  ☐ Off

Create a user via the realm account page: `http://localhost:8080/realms/oauth_app/account`

# OAUTH\_APP

## Register

\* Required fields

**Username \***

**Password \***



**Confirm password \***



**Email \***

**First name \***

**Last name \***

---

[« Back to Login](#)

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`

## Create client

Clients are applications and services that can request authentication of a user.

1 General settings

2 Capability config

3 Login settings

Client type ?

OpenID Connect

Client ID \* ?

altcraft\_admin

Name ?

Description ?

Always display in UI ?

Off

- Enable the `Client authentication` parameter:

## Create client

Clients are applications and services that can request authentication of a user.

1 General settings

2 Capability config

3 Login settings

Client authentication ?

On

Authorization ?

Off

Authentication flow

☒ Standard flow ?

☒ Direct access grants ?

☐ Implicit flow ?

☐ Service accounts roles ?

☐ OAuth 2.0 Device Authorization Grant ?

☐ OIDC CIBA Grant ?

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 required role to the user.

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 whether a user has access to the platform or the administrator panel.

Client scopes

Client scopes are a common set of protocol mappers and roles that are shared between multiple clients. [Learn more](#)

▼

Name

Q

Search for client scope

→

Create client scope

Change type to ▼

⋮

Refresh

1 - 10 ▼

◀ ▶

<input type="checkbox"/> Name	Assigned type	Protocol	Display order	Description
<input type="checkbox"/> <a href="#">acr</a>	Default ▼	OpenID Connect	–	OpenID Connect scope for add acr (authentication context class reference) to the token ⋮
<input type="checkbox"/> <a href="#">address</a>	Optional ▼	OpenID Connect	–	OpenID Connect built-in scope: address ⋮
<input type="checkbox"/> <a href="#">altcraft_platform-dedicated</a>	None ▼	OpenID Connect	–	Dedicated scope and mappers for this client ⋮
<input type="checkbox"/> <a href="#">basic</a>	Default ▼	OpenID Connect	–	OpenID Connect scope for add all basic claims to the token ⋮
<input type="checkbox"/> <a href="#">email</a>	Default ▼	OpenID Connect	–	OpenID Connect built-in scope: email ⋮
<input type="checkbox"/> <a href="#">microprofile-jwt</a>	Optional ▼	OpenID Connect	–	Microprofile - JWT built-in scope ⋮
<input type="checkbox"/> <a href="#">offline_access</a>	Optional ▼	OpenID Connect	–	OpenID Connect built-in scope: offline_access ⋮
<input type="checkbox"/> <a href="#">organization</a>	Optional ▼	OpenID Connect	–	Additional claims about the organization a subject belongs to ⋮
<input type="checkbox"/> <a href="#">phone</a>	Optional ▼	OpenID Connect	–	OpenID Connect built-in scope: phone ⋮
<input type="checkbox"/> <a href="#">profile</a>	Default ▼	OpenID Connect	–	OpenID Connect built-in scope: profile ⋮

To add this parameter to the token, it is recommended to use the predefined **dedicated client scope** (`<client-name>-dedicated`) by adding a **mapper** that passes the user's groups.

# Add mapper

If you want more fine-grain control, you can create protocol mapper on this client

Mapper type

Group Membership

Name \* ?

groups

Token Claim Name ?

groups

Full group path ?

Off

Add to ID token ?

On

Add to access token ?

Off

Add to lightweight access token ?

Off

Add to userinfo ?

Off

Add to token introspection ?

Off

Save

Cancel

## User Attributes

oauth\_app

Enabled

Action

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#)

General

Login

Email

Themes

Keys

Events

Localization

Security defenses

Sessions

Tokens

Client policies

User profile

User registration

Attributes

Attributes Group

JSON editor

All groups

Create attribute

	Attribute [Name]	Display name	Attribute group
	username	\${username}	
	email	\${email}	
	firstName	\${firstName}	
	lastName	\${lastName}	

An attribute is created (or imported) at the realm level: **Realm Settings** → **User profile** → **Attributes**



# Username

## General settings

Attribute [Name] \* ⓘ

Display name ⓘ

Multivalued ⓘ

☐
Off

Attribute group ⓘ

None
▼

## Jump to section

- General settings
- Permission
- Validations
- Annotations

## Permission

Who can edit? ⓘ

☒ User
☒ Admin

Who can view? ⓘ

☒ User
☒ Admin

## Validations

+ Add validator

Validator name	Config	
length	{"min":3,"max":255}	Delete
username-prohibited-characters	{}	Delete
up-username-not-idn-homograph	{}	Delete

## Annotations

### Annotations

No Annotations have been defined yet. Click the below

button to add Annotations, key and value are required for a key pair.

[+ Add Annotations](#)

- **Attribute (Name)** — the technical name of the attribute. Used in code, tokens, and APIs. Must be unique within the realm;
- **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 Info);
- **Who can edit / Who can view** — defines which user categories are allowed to view and modify the attribute value;
- **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 do this, in the **Client Scopes** section, select an existing **scope** used by the client and enable the **Include in token scope** parameter.

[Client scopes](#) > Create client scope

## Create client scope

Name * ?	altcraft_platform_scope
Description ?	Набор необязательных полей, предоставляемых клиенту
Type ?	Optional
Protocol ?	OpenID Connect
Display on consent screen ?	<input checked="" type="checkbox"/> On
Consent screen text ?	Набор необязательных полей, предоставляемых клиенту
Include in token scope ?	<input checked="" type="checkbox"/> On
Display Order ?	

Save

Cancel

To pass the attribute value into the token, add a **mapper** with the following parameters:

# Add mapper

If you want more fine-grain control, you can create protocol mapper on this client

Mapper type

User Attribute

Name \* ?

Phone mapper

User Attribute ?

▼

phone

Token Claim Name ?

phone

Claim JSON Type ?

String ▼

Add to ID token ?

Off

Add to access token ?

Off

Add to lightweight access token ?

Off

Add to userinfo ?

On

Add to token introspection ?

Off

Multivalued ?

Off

Aggregate attribute values ?

Off

- **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 done from the administrator panel.

To configure access, log in to the **Altcraft Administrator Panel** and navigate to **Account Area** → **Auth Connectors**. In this section, create a new authentication connector with the **Keycloak** type.

When creating the connector, specify the following parameters: **connector name**, **Keycloak host** and **port**, **realm name**, **client identifier (Client ID)**, and **Client Secret**. These values must match the settings of the client previously created in Keycloak.

## Auth connectors > Edit

<b>* Name</b>		<b>* Type</b>
Keycloak connector		Keycloak
<b>* Host</b>	<b>* Port</b>	
localhost	389	Test connection
<b>Secondary host</b>	<b>Secondary port</b>	
		Test connection
<b>* Realm</b>	<b>* Client ID</b>	<b>* Client secret</b>
oauth_app	account	*****
<input type="checkbox"/> With SSL		
		Save   Close

After saving the connector, it becomes available for selection in the account settings and can be used for user authentication via Keycloak.

### 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 determines which authentication type is associated with the user's account (local, LDAP, or Keycloak). If a Keycloak connector is configured and active for the account, the user is automatically redirected to the Keycloak login page.

On the Keycloak side, the user enters their credentials and completes authentication. After successful verification, Keycloak redirects the user back to Altcraft with authentication tokens.

Altcraft validates the received token, extracts the user's groups and attributes, and determines access rights to the platform or the administrator panel based on this data. If validation is successful, the user is granted access to the system with the appropriate roles and groups.

### Platform Roles Mode

By default, every time a Keycloak user logs in, the platform **overwrites** their roles and groups based on Keycloak group mapping and **Default Roles / Default Groups** settings. This means roles assigned manually through the platform UI are lost on the user's next login.

If all employees in Keycloak belong to a single group but need different roles in Altcraft Platform, enable the **Platform roles** mode in the account settings (**Authorization — Keycloak**):

# Authorization

☐ Force 2fa for users ?

☐ Use LDAP ?

☒ Use Keycloak ?

☐ Keycloak only ?

☒ Platform roles ?

\* Assigned Keycloak connector ?

When **Platform roles** is enabled:

- Keycloak is used **only for authentication** — credential check and **Auth Groups** membership verification.
- User roles and groups are **not overwritten** on login or session refresh.
- Roles and groups are assigned manually through the platform UI ([Users](#)) just like for regular users.
- The following fields continue to sync from Keycloak: email, first name, last name, phone, language, active status (**IsActive** — based on **Auth Groups** membership) and master status (**IsMaster** — based on **Master Groups** membership).

## TIP

Use this mode when:

- All users belong to one Keycloak group, but different roles are needed in Altcraft Platform.
- You want to manage roles through the platform UI instead of Keycloak group mapping.

Do not enable this mode if you need automatic Keycloak group-to-role mapping (default behavior).

A similar **Platform roles** mode is available for [LDAP integration](#).

# Getting HTTP service statuses

Correct balancing of web services requires the user to constantly and accurately **transmit the status of processes**. IN order to achieve this, special API requests to process status handlers are implemented.

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` , `trkwebctrl` or `cookiesaver` .

## List of requests

Request	Name of a service	Description
GET <code>/webcontrol/health</code>	<code>webcontrol</code>	Checks if the <code>webcontrol</code> service is running
GET <code>/webadmin/health</code>	<code>webadmin</code>	Checks if the <code>webadmin</code>
GET <code>/webapi/health</code>	<code>webapiapi</code>	Checks if the <code>webapiapi</code>
GET <code>/trkaction/health</code>	<code>trkaction</code>	checks if the <code>trkaction</code>
GET <code>/trkwebctrl/health</code>	<code>trkwebctrl</code>	Checks if the <code>trkwebctrl</code>
GET <code>/cookiesaver/health</code>	<code>cookiesaver</code>	Checks if the <code>cookiesaver</code>
GET <code>/webcontrol/readiness</code>	<code>webcontrol</code>	Checks if the <code>webcontrol</code> service is ready to accept traffic
GET <code>/webadmin/readiness</code>	<code>webadmin</code>	Checks if the <code>webadmin</code> service is ready to accept traffic
GET <code>/webapi/readiness</code>	<code>webapi</code>	Checks if the <code>webapi</code> service is ready to accept traffic
GET <code>/trkaction/readiness</code>	<code>trkaction</code>	Checks the <code>trkaction</code> service to see if it is ready to accept traffic
GET <code>/trkwebctrl/readiness</code>	<code>trkwebctrl</code>	Checks the <code>trkwebctrl</code> service to see if it is ready to accept traffic
GET <code>/cookiesaver/readiness</code>	<code>cookiesaver</code>	Checks if the <code>cookiesaver</code> service is ready to accept traffic

## 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 Mongo

### INFO

More information about the occurred 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, outdated logs can take up a significant amount of memory on the server. In order for the system to clean them regularly, it is necessary to configure **log rotation**. In this article we will describe log rotation for **MongoDB**.

To configure rotation using the `logrotate` utility, create the `/etc/logrotate.d/mongod` file with the following contents:

```
/var/log/mongodb/*.log {
 size 100M
 rotate 7
 compress
 missingok
 notifempty
 sharedscripts
 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 customize the rotation to your liking.

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 parameters are listed on [this page](#).

# Configuration of system constants and directories

The system settings of the platform are regulated **in configuration files**. Such JSON files contain various parameters of the platform components, such as templates, dictionaries and lists of constants. If necessary, these files can be modified by users to customize the system to suit their needs.

### CAUTION

If you are not sure that you will be able to create and customize a custom configuration yourself, we strongly recommend that you do not make changes to the config files. If you do decide to change the system configuration, do not edit the original configuration files. In case they are damaged or lost, 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 `"custom.<original_name>"`. Next, you can make the necessary changes to the copy of the file and save it. The next time you run the platform, the system uses the new custom config file.

For example, you want to add new cities to the `"cities.json"` config. To do this, **create a copy** of the `"cities.json"` file, **rename it** to `"custom.cities.json"` and move it back to the `"json_consts"` folder. Any changes to the platform are made **only after it has been rebooted**, so you will need to restart the platform after saving the new config.

Settings from custom files like `custom.` will be loaded in priority. If critical errors are detected in this config file, the system will notify the user that the configuration is not working correctly. In such a case it is necessary to delete the created **custom-file** from the folder, then at the next startup the platform will load the default configuration.

To configure [custom communication channels](#) from the **"channels "** folder, it is not necessary to add the `custom.` string to the name of the custom config. Files with the **"mta\_"** prefix can be reconfigured via [Admin Panel](#).

## List of JSON configurations

File	Description
<a href="#">bots_ip.json</a>	Contains a list of IP addresses to be identified as bots, and the actions associated with those addresses
<a href="#">bots_ua.json</a>	Contains a list of UserAgents that will be identified as bots and the actions associated with these agents
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
<a href="#">genders.json</a>	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
tlids.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	Description
mta_globals.json	General MTA-sender settings
mta_isps.json	MTA-sender ISP strategies
mta_lock_rules.json	MTA-sender locking strategies
mta_retry_rules.json	Strategies for retries on the MTA sender
mta_strategies.json	Strategies for sending on the MTA sender



# Filtering bot actions

The **Altcraft** platform can detect bot activity and modified proxy settings and is able to filter out these events. Thanks to this, clients do not receive distorted data about user activity on their sites. This mechanism allows you to get accurate information from analytics and reports on the platform.

The platform checks each connection and compares its **pattern** or **subnet** with those in the directory. Settings for filtering bot actions are contained in two config files: **"bots\_ua.json "** and **"bots\_ip.json "**. Currently, **1139 network rules and 575 UserAgent rules** have been added to these directories.

Validation can be performed on the following events:

Event	Description
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	Description
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 protection system operation) should be processed.
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., using a VPN), the user data should be ignored.
ip_ignore	Ignores the IP address of the event and all associated data, including <code>geo_ip</code> . Applies when a user performs an action, but because there has been data corruption (e.g., VPN usage), the event data should be ignored.
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.json "** or **"custom.bots\_ip.json "**, depending on the config. Do not make any changes to the original files!

## 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 specify an empty set, the default actions **ua\_ignore** and **ip\_ignore** will be executed.

# Directory of gender markers

Gender markers for the platform are contained in the “**genders.json**” directory file. By default, it contains male gender, female gender, and the most common designations for them.

If you need to change the configuration of the directory, for example, to add other designations, create a **copy** of the config file, name it “**custom.genders.json**”, then make changes to existing objects or add a new one.

## CAUTION

Do not make any changes to the original files! They should remain unchanged in case there is an error in the user configs. If the original file is present, the system will be able to start correctly by taking data from it.

## 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"
]
 }
]
```

# Custom Channels

A **channel** is an entity that defines interaction with a subscriber (profile) through an external integration. Such interaction can be sending an SMS, a message to a CRM, a post in a social media community, and much more.

The platform has two types of channels:

- **Built-in** (Email, SMS, Push, messenger channels) — channels described in the system code. Email, SMS, and Push operation cannot be configured, while messenger channel configuration is stored in the `json_consts/channels` directory.
- **Custom** — fully configurable channels that are created and configured by the platform administrator.

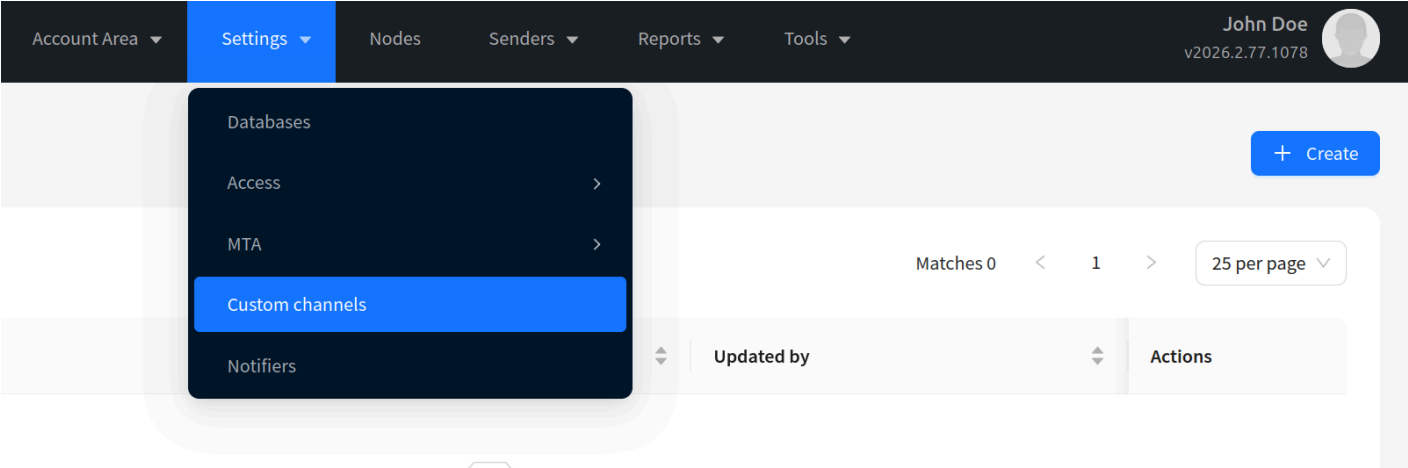
## Why custom channels are needed

Built-in channels cover standard communication scenarios. But if you need to connect an external service that is not available out of the box — for example, a messenger, an advertising platform, a CRM system, or a custom delivery gateway — a custom channel is created for this purpose.

A custom channel allows you to:

- Define what data is sent to the external service and in what format
- Configure the fields that the user sees when creating templates, mailings, and senders
- Define the events that are registered after sending (delivery, click, error, etc.)
- Connect external data sources — for example, select profiles from a CRM directly in the template editor
- Fully control the interaction logic with the external service API through **pipelines**

Custom channels are created and edited in the admin panel under **Settings — Custom Channels**.



## Preparation before creating a channel

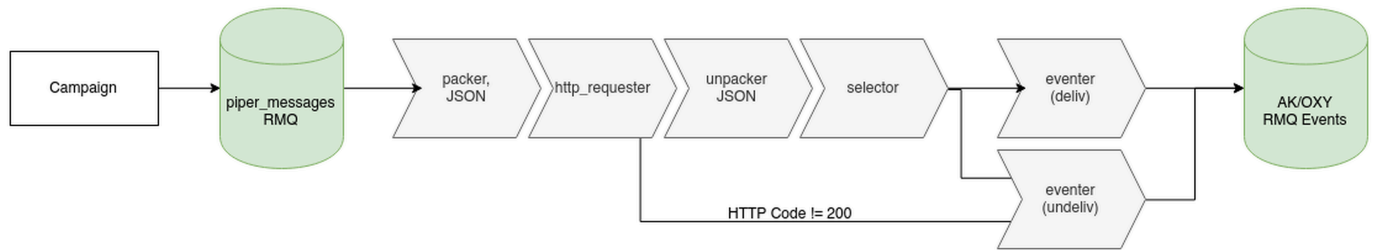
Before creating a channel, it is recommended to prepare the following information:

- Documentation for the external service API: message sending requests, delivery status check requests, callbacks, etc.
- Which fields are used when sending a message
- Which fields are used to identify a subscription
- Which fields will be configured by the user and which by the administrator
- Which events will occur on the channel and which platform events they correspond to

## Channel parameters and settings

The following can be configured for a custom channel:

Parameter	Description
Name and identifiers	Human-readable channel name, unique string ID for API, reports, and exports, unique numeric ID
Icon	Visual channel icon in the platform interface
Communication type	Direct (personalized sending) or Indirect (one content for everyone)
Mailing types	List of campaign types where the channel is available: broadcast, regular, splittest, trigger, placement
Moderation	Option to enable moderation mode for indirect channels
Events	System events linked to platform event types (send, deliv, undeliv, etc.), and custom events
System forms	Field sets for Template, Resource, Campaign, Sender, Subscription, Account — define the data the user or administrator sees (account and sender fields)
Presets	Field sets for different content types in Template. More details — in the <a href="#">Presets</a> article
External objects (Entities)	Entities for interacting with external APIs: List, Read, Create, Update, Delete operations with <b>pipelines</b> and forms
Pipelines	JSON chains of <b>pipes</b> for Message, Schedule, Listener, Stop, Moderate — describe the logic of sending, processing responses, registering events, stopping placements, and moderation
Localization	Translations of field names, events, and presets for different language versions of the platform



### String ID

Each channel has a unique numeric and a unique string identifier. They are used to store the channel flag in events, data structures, exports, and API requests. After the channel is created, the identifiers cannot be changed.

### Communication types

When creating a channel, you must select a communication type:

- **Direct** — personalized sending. A mailing creates a separate message for each profile, using profile data (subscription fields, etc.). For a direct channel to work, at least one field in the Subscription form is required. For example, a subscription field will be available in the **pipeline** as `$subscription.client_id`.
- **Indirect** — non-personalized sending. One content is sent for the entire mailing, without binding to a specific profile. Suitable for cases where interaction occurs with an undefined set of profiles (for example, a bot in a group or messenger channel) or when profiles are absent from the platform. For an indirect channel to work, at least one field in the Resource form is required.

### Moderation

For **indirect** channels, the **Enable moderation** (With moderate) option is available. It enables the mailing moderation mode and opens the **Moderate pipeline** for editing.

How this works from the user's perspective:

1. The user activates a placement campaign on a channel with moderation — the campaign receives the **ModerationPending** status. The edit form is locked.
2. The platform launches the mailing in moderation mode: the message is routed to the **Moderate** pipeline instead of **Message**.
3. The Moderate pipeline sends content to an external approval service (for example, via HTTP Request). The result is processed by the **Eventer** pipe:
  - **moderate\_pass** — status changes to **ModerationPassed** (Moderation passed). The mailing will launch on schedule if configured, or via the Run Now button.
  - **moderate\_error** — status changes to **ModerationError** (Moderation error). A red warning icon and the error text from the Eventer **message** field are displayed in the interface.
4. If moderation fails, the user fixes the content and saves the campaign — it is automatically sent for re-moderation.
5. After successful moderation, the user launches the campaign, and messages are delivered through the regular **Message** pipeline.

#### NOTE

- A campaign in **ModerationPending** or **ModerationPassed** status cannot be re-activated — it can only be deactivated.
- The "Start now" checkbox is hidden when creating a campaign on a channel with moderation.

### Channel events

Events can be defined in a channel, which are reflected in reports, profile history, segments, and other platform modules. Each event is assigned:

- Unique numeric and string identifiers
- Name (Title)
- Platform event type (for example, send — sending, undeliv — non-delivery)

Events are divided into **system** (System Events) and **custom** (Custom Events).

**System events** are registered automatically by the platform (for example, a click on a tracking link) or through a **pipeline** using the **Eventer** pipe.

**Custom events** are created in the Custom Events section and registered only through a **pipeline** with type **custom**. These are events that do not correspond to any system event type, for example, a like or adding to favorites. Like system events, they will be displayed in analytics and segmentation.

### Platform event types (Event type)

When creating an event in the Events section, you need to select a platform event type. The type determines how the event is processed by the system:

Event type	Purpose	Automatic registration
send	Message sending	Yes — generated automatically when a campaign is sent
deliv	Message delivery	No — via Eventer
undeliv	Message non-delivery	No — via Eventer
hbounce	Hard bounce	No — via Eventer
sbounce	Soft bounce	No — via Eventer
open	Message open	Yes — when HTML content with open pixel is opened
uopen	Unique open	Yes — when HTML content with open pixel is opened
click	Link click	Yes — when following a tracking link
uclick	Unique click	Yes — when following a tracking link

Event type	Purpose	Automatic registration
uclick_link	Unique click with link data	Yes — when following a tracking link
click_link	Click with link data	Yes — when clicking a tracking link in HTML content
subscribe	Subscription	Yes — registered automatically when subscription status changes
unsubscribe	Unsubscription (via API or manually)	Yes
unsub	Unsubscription	Yes — registered automatically when a profile unsubscribes
uunsub	Unique unsubscription	Yes — automatically when unsubscribing by link
uunsub_link	Unique unsubscription by link	Yes — automatically when unsubscribing by link
unsub_link	Unsubscription by link	Yes — automatically when unsubscribing by link
complain	Complaint (spam)	No — via Eventer
read	Message read	No — via Eventer (sends to trkread)
reply	Reply to message	No — via Eventer
confirm	Confirmation	No — via Eventer
glanced	Glanced	Yes — when tracked via tracking
skimmed	Skimmed	Yes — when tracked via tracking
custom	Custom event	No — via Eventer

CAUTION

The types `unsubscribe` and `unsub` are different:

- `unsubscribe` — unsubscription via API or manually in the interface. The event is registered when an administrator or an external service changes the subscription status.
- `unsub` — unsubscription by link. The event is registered automatically when a user follows the unsubscribe link in a message.

If you do not add an event with type `unsub` in the channel settings, unsubscription by link will not work: the user will see the unsubscribe page, but the subscription status will not change.

To register events in a **pipeline**, the [Eventer pipe](#) is used.

# Custom Channels > Edit

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

System eventsCustom events

In this section user defines which system events will be available for the channel. Some of these events will be registered by the platform in automatic mode, the rest should be configured using the Pipeline. Send and undelivery events are mandatory for channel configuration.

Event ID ?	Title	String ID ?	Event type ?	Operation
not defined	send	string_id_send	Send	<a href="#">Edit</a> <a href="#">Delete</a>
not defined	undeliv	string_id_undeliv	Undeliv	<a href="#">Edit</a> <a href="#">Delete</a>

-

Enter a title

string\_

Enter a

deliv

Ok

Cancel

+ Add event

Read

Glanced

Skimmed

Reply

Confirm

Subscribe

Unsubscribe

Offence

Save

Close

## Pipelines

Interaction with external services when sending messages and receiving events occurs through **pipelines**. Each **pipeline** consists of a JSON configuration that describes sending requests, transforming data, saving and loading values, and creating events in the platform.

Types of **pipelines**:

- [Message](#) — main processing of mailing messages
- [Schedule](#) — deferred processing on a schedule
- [Listener](#) — incoming events from external services
- [Moderate](#) — mailing moderation (for indirect channels)
- [Stop](#) — completion of a placement mailing

## Custom Channels > Edit

General

Events

External objects

Object fields

Pipelines

Languages

Message

Scheduled

Listener

Stop

Moderate

```
1 [
2 {
3 "id": 1,
4 "type": "log",
5 "params": {
6 "message": "[Message] Start | campaign: $in.c_id | send_message_id: $in.send_message_id",
7 "level": "info"
8 },
9 "outs": {
10 "success": 2
11 }
12 },
13 {
14 "id": 2,
15 "type": "pack",
16 "params": {
17 "type": "JSON",
18 "template": {
19 "chat_id": "$subscription.chat_id",
20 "text": "$template.text",
21 "parse_mode": "HTML",
22 "webhook_url": "$getWebhookURL($in.send_message_id)"
23 }
24 },
25 "outs": {
```

Save

Close

### System forms

Channel content and settings are defined through fields that can be configured for the following platform entities:

- **Template** — content delivered to the user. This is where all dynamics are assembled: profile data, events, and other variables are substituted. Template fields: text message body, image, attachments, etc.
- **Resource** — a means for organizing client subscriptions to specific content over specific communication channels.
- **Campaign** — placement-level settings: display periods, placement locations, and other parameters. A campaign has **no** personalization — profile data is not substituted.
- **Sender** — sender settings: API key (not accessible to the user), URL for content delivery. Often used for migration scenarios — for example, an old sender with an old URL and a new one with a new URL, while the mailing percentage between them is overridden and everything works without downtime.
- **Subscription** — identifiers for a contact and their properties: email, phone, user ID in CRM, etc. Links a resource to a profile in a channel.
- **Account** — global channel settings and permissions that need to be hidden from the user (for example, a kill-switch).

#### CAUTION: Important

Dynamic content (substitution of profile and event data) is assembled **only in Template**. Personalization does not work in other entities.

For the channel to work correctly, the **Template** and **Subscription** forms (for direct channels) or **Template** and **Resource** forms (for indirect channels) are mandatory. At minimum, you need to define Template fields and the fields of entities that depend on the selected communication type.

#### TIP

To create a custom channel from scratch, go to the step-by-step guide: [Creating a Channel](#).



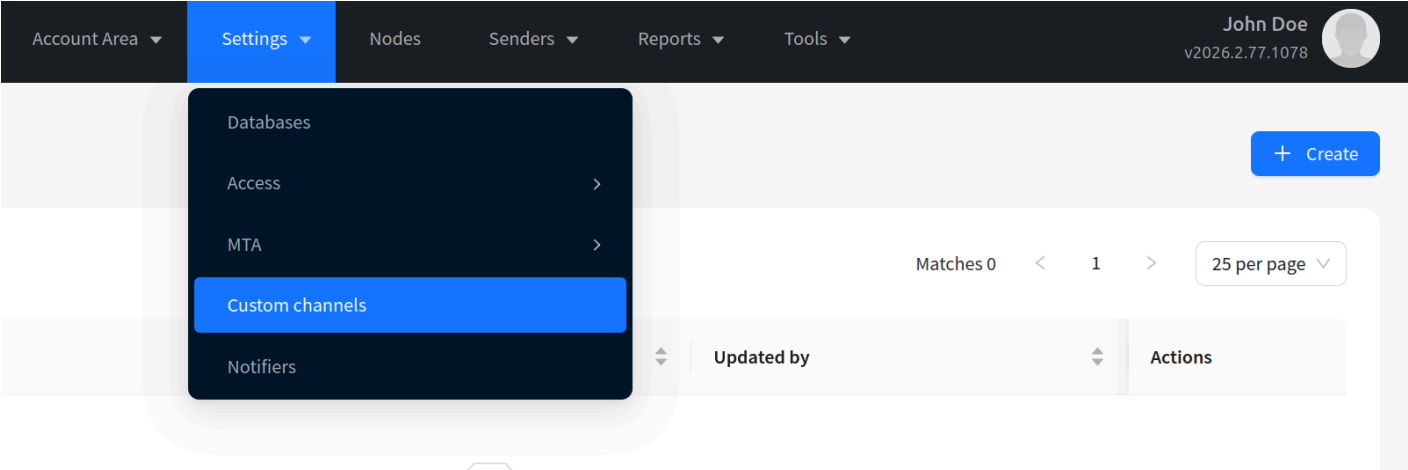
# Creating a Channel

This is a step-by-step guide to creating a custom channel from scratch. We will go through all interface tabs in the order they open.

Before you begin, make sure you have:

- Documentation for the external service API (sending requests, status checks, callbacks)
- An understanding of which fields are needed for sending a message and identifying a subscription
- An understanding of which fields are configured by the user and which by the administrator
- A list of events that will occur on the channel

Go to **Settings — Custom Channels** and click the button to create a new channel:



## General

On this tab, the main channel parameters are configured:

Field	Description
Name	Channel name displayed in the interface
String ID	String identifier. Used in reports, analytics, exports, and API requests. Cannot be changed after creation
Custom channel icon	Channel icon in the platform interface
Types of campaigns	Mailing types where the channel will be available: broadcast, regular, splittest, trigger, placement
Communication type	Communication type: <b>Direct</b> (personalized sending for each profile) or <b>Indirect</b> (one content for the entire mailing)
Enable moderation	Moderation toggle. Available for indirect channels. Enables the <a href="#">Moderate</a> pipeline

## Custom Channels > Edit

General

Events

External objects

Object fields

Pipelines

Languages

\* Name

New Custom Channel

\* String ID ?

custom\_channel

Custom channel icon ?



bullhorn



Types of campaigns ?

Broadcast ×



Communication type ?

Direct



☐ Enable moderation

Save

Close

After filling in the fields, click **Save**. The channel can only be saved after configuring at least one field in Template and Subscription (for direct channels) or Resource (for indirect channels). The remaining tabs will become available for editing.

### Events

On this tab, channel events are configured, which are displayed in reports, profile history, and segments.

#### System Events

System events of the channel. Some are registered automatically by the platform (for example, `click` when following a tracking link), the rest — through a **pipeline** using the [Eventer](#) pipe. Events **send** and **undeliv** are mandatory for channel configuration.

When adding an event ( `Add event` ), the following are configured:

Field	Description
Event ID	Numeric identifier of the event
Title	Event name
String ID	String identifier. Used in reports, analytics, exports, and API
Event type	Platform event type (see the table below)

#### Platform event types

Event type	Purpose
<code>send</code>	Message sending
<code>deliv</code>	Message delivery
<code>undeliv</code>	Message non-delivery
<code>click</code>	Link click (automatically when following a tracking link)
<code>uclick</code>	Unique click
<code>subscribe</code>	Subscription
<code>unsubscribe</code>	Unsubscription via API or manually
<code>unsub</code>	Unsubscription by link (automatically when following the unsubscribe link)
<code>hbounce</code>	Hard bounce
<code>sbounce</code>	Soft bounce
<code>open</code>	Message open
<code>read</code>	Message read (automatically via <code>{read}</code> tracking pixel, view time >= 8s)
<code>glanced</code>	Brief view (automatically tracked, view time 0-4s)

Event type	Purpose
skipped	Skipped (automatically tracked, view time 4-8s)
complain	Spam complaint
reply	Reply to message
confirm	Subscription confirmation

NOTE

The events `glanced`, `skipped`, and `read` are registered automatically by the tracking service based on the time spent viewing the message page:

- `glanced` — 0 to 4 seconds
- `skipped` — 4 to 8 seconds
- `read` — 8 seconds or more

For these to work, the message template must include the `{pixel}` or `{read}` tracking pixel.

CAUTION

The types `unsubscribe` and `unsub` are different:

- `unsubscribe` — unsubscription via API or manually in the interface
- `unsub` — unsubscription by link. If you do not add an event with type `unsub`, unsubscription by link will not work: the user will see the unsubscribe page, but the subscription status will not change

Each event has **Edit** and **Delete** buttons.

Custom Channels > Edit

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

System eventsCustom events

In this section user defines which system events will be available for the channel. Some of these events will be registered by the platform in automatic mode, the rest should be configured using the Pipeline. Send and undelivery events are mandatory for channel configuration.

Event ID ?	Title	String ID ?	Event type ?	Operation
not defined	send	custom_channel_send	Send	<a href="#">Edit</a> <a href="#">Delete</a>
not defined	undeliv	custom_channel_undeliv	Undeliv	<a href="#">Edit</a> <a href="#">Delete</a>

+ Add event

SaveClose

Custom Events

Custom events are events that do not correspond to any system event type, for example, a like or adding to favorites. They are created in the Custom Events section and registered only through a **pipeline** with type `custom`. Like system events, they are displayed in analytics and segmentation.

When adding, the same fields are configured: Event ID, Title, String ID.

## Custom Channels > Edit

General

Events

External objects

Object fields

Pipelines

Languages

System events

Custom events

In this section the user defines which events (in addition to the platform system events) will be available for the channel. These events can be, for example, likes or dislikes. Custom events are always registered through the pipelines.

Event ID ?

Title

String ID ?

Operation

-

registration

custom\_cl

reg

Ok Cancel

+ Add event

Save

Close

### External Objects

On this tab, external entities (Entities) are created — objects with which the channel interacts through an API. An Entity allows you to load, create, and edit external data directly from the interface when configuring channel content.

To create, click **Add** and fill in:

Field	Description
Name	Entity name
Shortname	Unique string identifier
Description	Brief description

Then configure **Object Structure** (fields returned by the API), **Operations** (List, Read, Create, Update, Delete with Input Fields and Pipeline for each), and **Output Operations Form** (result display forms).

For a detailed description, see [External Objects \(Entities\)](#).

# Custom Channels > Edit

General Events **External objects** Object fields Pipelines Languages

External\_data × +

\* Name

External\_data

\* Shortname

ext\_data

▼ Object Structure

This section defines the structure of the external object. The structure defines interaction with the external API, data from which can be requested, created and edited directly in the user interface when customizing the channel content. This data will be further available in the Pipeline. The object structure is a list of fields of different types.

Title		Shortname	Type	Operation
Data	 	data	int	 

+ Add fields

Primary field ?

Data

Preview field ?

Data

## Object Fields

On this tab, system forms for channel entities are configured. Each subsection corresponds to its own entity.

Subsection	Entity	Required
Template	Message content	Yes
Resource	Resource	Yes for indirect
Campaign	Mailing	No
Sender	Sender (API keys, URL)	No
Subscription	Subscription (profile identifiers)	Yes for direct
Account	Account (global settings)	No

The following buttons are available in the subsections:

- **Add field row** — adds a row with fields to the form. Fields are added inside rows
- **Add object** — adds a field with a bound External Entity (available on Template and Campaign)

### Field configuration

One-, two-, three-, and four-column layouts are available for field rows. Each row can have its own set of columns. When adding a field, the following are configured:

Field	Description
Title	Display name of the field
Shortname	Name for access in the <b>pipeline</b> through templating
Type	Data type

Available field types: `int` , `string` , `bool` , `json` , `text` , `html` , `file` , `enum` , `color` , `image_upload` .

CAUTION

The set of available field types may vary depending on the entity.

On the **Layout element** tab, decorative elements are added: `header` (heading), `message` (text), `divider` (separator).

Binding an External Entity

The **Add object** button allows you to bind a previously created External Entity to a form field. The field occupies the entire row in the Layout.

When binding, the following are configured:

Field	Description
Shortname	Name for access in the <b>pipeline</b> (for example, <code>\$template.profile</code> )
Entity button description	Button text in the interface (for example, "Select profile")
Entity	Entity selection from the list
Entity operation	Specifies which operation result (List or Read) will be sent to the sender

For the List operation, **Select entity from list**, **Multiple selection**, and **Save selected entity** are additionally available. Depending on the settings, the List operation works in one of four modes:

- **Automatic loading** — all data from List is automatically pulled when sending
- **Single entity selection** — the user selects one record, data is pulled through Read
- **Multiple selection** — the user selects one or more records from the list, data is pulled through Read
- **Saving at template creation** — if the option is activated, the mailing will send the entity data that was saved when the template was saved. If the option is not activated, the mailing will fetch fresh data using the Read pipeline before sending.

For details, see [External Objects \(Entities\)](#).

Account Area ▾Settings ▾NodesSenders ▾Reports ▾Tools ▾

Custom Channels > Edit

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

Template \*ResourceCampaignSenderSubscription \*Account

System forms

Next, you need to add system forms for the channel. These forms determine the data the platform will collect and creating various objects related to the channel. Forms can be created for Templates, Resources, Campaigns, Send Subscriptions and Accounts. For the channel setup, only templates and subscriptions forms are mandatory.

+ Add field row

+ Add object

External object field config

\* Shortname

Campaign\_External

Entity button description

Edit

\* External object

External\_data ▾

Entity operation

List ▾

☒ Select an entity from list

☐ Multiple selections

☐ Save selected entity

Presets (Template only)

The **Presets** block is available in the Template subsection — field sets for different content types. Each preset determines which fields the user sees when creating a template. The **Links shorter enabled** toggle enables link shortening in content.

For details, see the [Presets \(Field Sets\)](#) article.

Pipelines

On this tab, channel **pipelines** are configured — JSON configurations that describe interaction with external services.

Subsection	Pipeline	Description
Message	<a href="#">Message</a>	Main processing of mailing messages
Scheduled	<a href="#">Schedule</a>	Deferred processing on a schedule
Listener	<a href="#">Listener</a>	Incoming events from external services
Stop	<a href="#">Stop</a>	Completion of a placement mailing
Moderate	<a href="#">Moderate</a>	Moderation (only for indirect channels with moderation enabled)

Each subsection contains a JSON editor. By default — an empty array `[]`.

The **Listener** subsection additionally has a **Response body** field — a string with which the tracking service will respond to incoming requests.

Custom Channels > Edit

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

MessageScheduledListenerStopModerate

12345678910111213141516171819202122232425

[  
 {  
 "id": 1,  
 "type": "log",  
 "params": {  
 "message": "[Message] Start | campaign: \$in.c\_id | send\_message\_id: \$in.send\_message\_id",  
 "level": "info"  
 },  
 "outs": {  
 "success": 2  
 }  
 },  
 {  
 "id": 2,  
 "type": "pack",  
 "params": {  
 "type": "JSON",  
 "template": {  
 "chat\_id": "\$subscription.chat\_id",  
 "text": "\$template.text",  
 "parse\_mode": "HTML",  
 "webhook\_url": "\$getWebhookURL(\$in.send\_message\_id)"  
 }  
 },  
 "outs": {  
 "success": 2  
 }  
 }  
]

Save

Close

For a detailed description of **pipelines** and **pipes**, see the [Pipelines](#) section.

Languages

On this tab, localization of field names and channel interface elements is configured.

Text areas are available for each language (English, Russian). Field editing becomes available after saving the channel.

## Custom Channels > #10003

General

Events

External objects

Object fields

Pipelines

Languages

Here you can configure localization of objects of your channel, their descriptions and information messages. Localization is specified in .po format. Example of string localization

Refresh fields

```
#path_to_object
msgid "id_text"
msgstr "Text content"
```

### English

```
#Channel_name
msgid "channel_name"
msgstr "Telegram"

#Template_fields
msgid "template.text.title"
msgstr "Message text"
```

### Russian

```
#Channel_name
msgid "channel_name"
msgstr "Telegram"

#Template_fields
msgid "template.text.title"
msgstr "Текст сообщения"
```

Save

Close

## Saving

After configuring all tabs, click **Save** in the bottom right corner of the page. The channel is ready for use.

## End-to-end example: Telegram channel

Let us consider creating a channel for sending messages to Telegram. This is a direct channel: each message is sent to a specific subscriber.

### Step 1. General

Field	Value
Name	Telegram
String ID	tg
Types of campaigns	broadcast, regular, trigger
Communication type	Direct

### Step 2. Events

Create three system events:

Event ID	Title	String ID	Event type
1	Telegram Send	send	send
2	Telegram Delivered	deliv	deliv
3	Telegram Undelivered	undeliv	undeliv



Step 3. Object Fields

Template — message content:

Title	Shortname	Type
Message text	text	text

Subscription — subscriber identifier:

Title	Shortname	Type
Chat ID	chat_id	string

Sender — API data:

Title	Shortname	Type
Bot Token	token	string

Step 4. Pipelines — Message

Minimal pipeline of 4 pipes:

- 1. Pack ( id: 1 ) — forms the JSON request body
- 2. HTTP Request ( id: 2 ) — sends POST to the Telegram API
- 3. Eventer ( id: 3 ) — registers the deliv event on success
- 4. Eventer ( id: 4 ) — registers the undeliv event on error

```
[
 {
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON",
 "template": {
 "chat_id": "$subscription.chat_id",
 "text": "$template.text",
 "parse_mode": "HTML"
 }
 },
 "outs": {
 "success": 2
 }
 },
 {
 "id": 2,
 "type": "http_request",
 "params": {
 "method": "POST",
 "timeout": 10,
 "content_type": "json",
 "url": "https://api.telegram.org/bot$sender.token/sendMessage",
 "success_codes": [200]
 },
 "outs": {
 "success": 3,
 "error": 4
 }
 },
 {
 "id": 3,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100001,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 },
 {
 "id": 4,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100003,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 }
]
```

Here:

- **Pipe 1** (Pack) takes data from the subscription ( `$subscription.chat_id` ) and the template ( `$template.text` ), packs it into JSON.
- **Pipe 2** (HTTP Request) sends a POST request to the Telegram API. The bot token is taken from the sender ( `$sender.token` ).
- **Pipe 3** (Eventer) registers the delivery event ( `deliv` , action\_type `100001` ) on HTTP 200.
- **Pipe 4** (Eventer) registers the non-delivery event ( `undeliv` , action\_type `100003` ) on error.

## Custom Channels > Edit

General

Events

External objects

Object fields

Pipelines

Languages

Message

Scheduled

Listener

Stop

Moderate

```
1 [
2 {
3 "id": 1,
4 "type": "pack",
5 "params": {
6 "type": "JSON",
7 "template": {
8 "chat_id": "$subscription.chat_id",
9 "text": "$template.text",
10 "parse_mode": "HTML"
11 }
12 },
13 "outs": {
14 "success": 2
15 }
16 },
17 {
18 "id": 2,
19 "type": "http_request",
20 "params": {
21 "method": "POST",
22 "timeout": 10,
23 "content_type": "json",
24 "url": "https://api.telegram.org/bot$sender.token/sendMessage",
25 "success_codes": [200]
```

### Step 5. Verification

After saving the channel, perform the following steps:

1. Create a template with message text for the Telegram channel
2. Create a sender with the bot token
3. Create a subscription with chat\_id
4. Create a test mailing and send it
5. Check the logs ( `procpipe.log` ) — entries from Log **pipes** should appear (if added)
6. Check events in profile history — a `send` or `undeliv` event should appear

#### TIP

If the message was not delivered, check:

- Correctness of the bot token in the sender
- Correctness of chat\_id in the subscription
- Error logs in `$error.code` and `$error.message`
- Correspondence of `channel_id` and `action_type` in Eventer to the created events

#### CAUTION

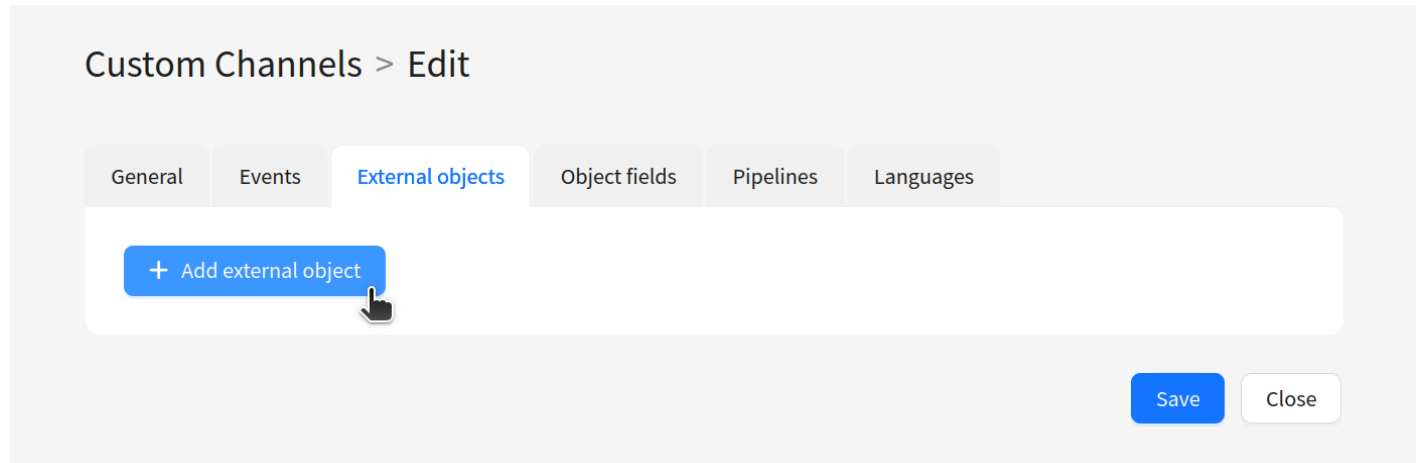
Review the [technical limitations](#) and the [Debugging](#) section for custom channels.

## External Objects (Entities)

An **External Object (Entity)** is an external entity with which a channel interacts through an API. An Entity allows you to load, create, and edit external data directly from the platform interface when configuring channel content. The retrieved data is then available in the main channel **pipelines** (**Message**, **Schedule**, **Listener**, **Stop**, **Moderate**) through the [templating language](#).

Typical scenario: in the template editor, the user selects a profile from an external CRM, and the data of that profile (name, email, ID) is passed to the Message **pipeline** for sending to the external service.

External Objects are created on the **External Objects** tab when configuring a channel:



### Creating an Entity

When creating an external object, you need to specify:

Field	Description
<b>Name</b>	Entity name. Displayed in the interface when binding to a channel entity field
<b>Shortname</b>	Unique string identifier. Entity fields will be accessible in <b>pipelines</b> through templating by this name
<b>Description</b>	Brief description of the entity

# Custom Channels > Edit

General

Events

External objects

Object fields

Pipelines

Languages

New external object ×

+

\* Name

Enter a name

\* Shortname

Enter a shortname

> Object Structure

> Operation

Save

Close

## Object structure

The **Object Structure** section defines which fields the external API returns and which of them will be available in **pipelines**. This is a list of fields of different types.

When adding a field ( `Add Field` ), the following are specified:

Field	Description
Title	Display name of the field
Shortname	Field name for access in a <b>pipeline</b> through templating
Type	Data type

Available field types:

- `Integer` — integer
- `String` — string
- `Boolean` — boolean value
- `Serialized JSON string` — JSON object
- `HTML content` — HTML content
- `Text content` — multiline text
- `Image Field` — image

Account Area ▾Settings ▾NodesSenders ▾Reports ▾Tools ▾

Custom Channels > Edit

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

New external object × +

\* Name

Enter a name

\* Shortname

Enter a shortname

Object Structure

This section defines the structure of the external object. The structure defines interaction with the external API, data from which can be requested, created and edited directly in the user interface when customizing the channel content. This data will be further available in the Pipeline. The object structure is a list of fields of different types.

+ Add fields

Field

\* Title

Enter a title

\* Shortname

Enter a shortname

Type

Integer ▾

Integer

String

Boolean

HTML content

Text content

Serialized JSON string

Image field

Among the fields, you need to designate a **Primary Field** (unique object identifier) and a **Preview Field**. The Preview Field is the field whose value is displayed in the user interface when previewing an entity. If the entity is bound to a screen form in list mode with selection enabled, the Preview Field value will be used to display the selected list items:

Object Structure

This section defines the structure of the external object. The structure defines interaction with the external API, data from which can be requested, created and edited directly in the user interface when customizing the channel content. This data will be further available in the Pipeline. The object structure is a list of fields of different types.

Title		Shortname	Type	Operation
FieldMain	🔑	FieldMain	int	 
FieldPreview	👁	FieldPreview	int	 

+ Add fields

Primary field ?

FieldMain ▾

Preview field ?

FieldPreview ▾

All operations must return data that matches this structure (not necessarily all fields, but the ones that are returned must be described here).

Operations

An **operation** is a method of interacting with an external entity (create, delete, get a list, get a single object, etc.). Each operation consists of input data (Input Fields) and a **pipeline**.

Available operations

Operation	Returns	Effect on the interface
List	Array of objects	"Add" button in a field with a bound Entity. Opens a table with the ability to select
Read	Single object	—
Create	Single object	"+" (create) button in a field with a bound Entity
Update	Single object	Edit button (pencil) in a field with a bound Entity
Delete	—	Delete button in a field with a bound Entity

Input data (Input Fields)

A set of fields that the user fills in the interface before executing the operation. Input data is not required to be configured. For example, an API token and a database ID — values that may change and that you do not want to hardcode in the **pipeline**.

Values from Input Fields are available in the **pipeline** through `$in.<shortname>`.

TIP: Input Field as a filter for SQL.

Often, Input Fields are used to filter data in an SQL query. For example, if a field `id` is added for the Read operation, you can write `SELECT * FROM my_table WHERE id = $in.id` in the SQL query to retrieve a specific record by the entered ID.

Pipeline for operation Read

```
{
 "id": 100,
 "type": "sql_pipe",
 "params": {
 "connector_id": 1,
 "connection_timeout": 15,
 "query_timeout": 15,
 "query": "SELECT * FROM my_table WHERE id = $in.id",
 "map": true
 },
 "outs": {
 "success": 200,
 "error": 150
 }
},
{
 "id": 200,
```

When adding a field ( `Add Row` ), you can select a type:

- **Field** — a data field. Title, Shortname, Type, required flag, default value, value range (Minimum/Maximum), and other parameters are configured.
- **Layout element** — a decorative element: header ( `header` ), text block ( `message` ), divider ( `divider` ). Does not store data, used for visual organization of the form.

Operation

Operations are methods of interacting with the object (create, delete, get list, get one object, etc.). An operation can be configured in the form of input data for the user UI and a mandatory Pipeline. You can read more about pipelines [here](#).

ListReadCreateUpdateDelete

Input fields for operation Read

+ Add row

Pipeline for operation Read

```
{
 "id": 200,
 "type": "result",
 "params": {
 "path": "$#in.sql_query"
 }
},
{
 "id": 150,
 "type": "error",
 "message": "..."
}
```

External object field config

FieldLayout element

\* Title

Enter a title

\* Shortname

Enter a shortname

Type

Integer

This field is required

Add to web version

Default value

Please enter a default value

Value from the list

Minimum

Please enter a minimum value

Maximum

Please enter a maximum value

Operation pipeline

A JSON array of **pipes** that is executed when the operation is launched from the interface. The configuration is similar to other **pipelines** in the channel, with some nuances. The **pipeline** is synchronous — the user who launches the operation in the interface will wait for the result.

Available **pipe** types:

Pipe	Purpose
<a href="#">HTTP Request</a>	HTTP request to an external API
<a href="#">RMQ Publisher</a>	Publish message to RabbitMQ
<a href="#">Pack</a>	Pack data into JSON/XML
<a href="#">Unpack</a>	Unpack JSON/XML
<a href="#">Result</a>	Fix the operation result (required)
<a href="#">Error</a>	Terminate with an error
<a href="#">Selector</a>	Conditional branching
<a href="#">SQL</a>	SQL query to the database
<a href="#">Store_Set</a>	Save data to storage
<a href="#">Store_Get</a>	Read data from storage
<a href="#">Log</a>	Write to log

CAUTION

An event must always pass through the **Result pipe** before exiting the **pipeline** (unless exiting through **Error**).

Output forms (Output Operations Form)

An output form defines how the operation result is displayed in the interface. The **List form** is bound to the List operation, the **Single form** — to the Read operation.

You cannot create new fields in an output form — you can only select and arrange fields already created in the Object Structure section.

Form	Used for	Description
List	List operation	Table of objects. Fields are selected from Object Structure. The field order in the form determines the column order in the table



Form	Used for	Description
Single form	Read operation	Form with fields of a single object. Fields are selected from Object Structure

Fields in the output form are arranged by drag-and-drop.

▼ Output operations form

The output forms are used to customize the display of the data obtained from the operation's pipeline.

FieldMain

FieldMain

FieldPreview

FieldPreview

+ Add row

✕ Remove row

Binding an External Object to a channel entity field

After creating an External Object, it can be bound to a field in the **Object Fields** section of the channel (on the Template or Campaign tabs, where the **Add Object** button is available). A field with a bound External Object occupies the entire row in the Layout.

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

Template \*ResourceCampaignSenderSubscription \*Account

Presets

+

Add

Links shorter enabled ?

System forms

Next, you need to add system forms for the channel. These forms determine the data the platform will collect and display when creating various objects related to the channel. Forms can be created for Templates, Resources, Campaigns, Senders, Subscriptions and Accounts. For the channel setup, only templates and subscriptions forms are mandatory.

+ Add field row

+ Add object

When binding, the following are configured:

Field	Description
Shortname	Name for accessing Entity data in the <b>pipeline</b> (for example, <code>\$template.profile</code> ). See <a href="#">templating language</a>
Entity button description	Button text in the interface that opens Entity editing (for example, "Select profile")
External Object	Selection of an External Object from the list of created ones
Entity operation	Specifies which operation result (List or Read) will be sent to the sender

General

Events

External objects

Object fields

Pipelines

Languages

Template \*

Resource

Campaign

Sender

Subscription \*

Account

Presets

+

Add

Links shorter enabled ?

System forms

Next, you need to add system forms for the channel. These forms determine the data the platform will collect and creating various objects related to the channel. Forms can be created for Templates, Resources, Campaigns, Send Subscriptions and Accounts. For the channel setup, only templates and subscriptions forms are mandatory.

+ Add field row

+ Add object

External object field config

\* Shortname

Enter a shortname

Entity button description

Edit

\* External object

Entity operation

List

\* Preset

☐ Select an entity from list

☐ Multiple selections

☐ Save selected entity

#### Additional settings for the List operation

Field	Description
Select entity from list	If enabled — the user selects objects from the list. Activates the settings below
Multiple selection	Allows selecting multiple objects. If disabled — only one
Save selected entity	Determines whether to save the data when configuring the field, or to load fresh data when launching the mailing

#### List operation work types

Depending on the Entity binding settings to a field, the List operation works in one of four modes:

Mode	Settings	Behavior
Automatic loading	Without additional checkboxes	All data from List is automatically pulled when the campaign is sent. The user does not select records — the List <b>pipeline</b> executes and returns all data
Single entity selection	"Select entity from list" enabled	The user selects one record from the list. When the campaign is sent, data is pulled through the Read operation by the ID of the selected record
Multiple selection	"Select entity from list" + "Multiple selection"	The user selects one or more records from the list. Each is pulled through Read
Saving at template creation	"Save selected entity" enabled	Data is pulled from the external database when the template is saved and stored in MongoDB, not when the campaign is sent

#### List + Read connection

For the "Single entity selection" and "Multiple selection" modes, both operations must be configured:

1. **List** — to display the list of available records (the user sees and selects)
2. **Read** — to load full data of the selected record by ID

When a record is selected from the list, the platform automatically calls the Read operation with the ID of the selected record. Therefore, Read must be configured with an Input Field (for example, `id`) and a **pipeline** that uses this ID for filtering: `SELECT * FROM my_table WHERE id = $in.id`.

#### Pipeline examples for SQL operations

Full **pipeline** examples for all five operations (Read, List, Create, Update, Delete) using the [SQL pipe](#) are in the [SQL](#) article. Key points:

- **Read:** `map: true` + filter `$in.id` → Result with `$.sql_query`
- **List:** without `map` (return array) → Result with `$.sql_query`
- **Create:** INSERT with `$in.<field>` → Result
- **Update:** UPDATE with `$in.<field>`, WHERE `$in.id` → Result
- **Delete:** DELETE WHERE `$in.id` → Result

#### Example

Suppose the List operation returned an array:

```
[
 { "id": 1, "name": "Segment 1" },
 { "id": 2, "name": "Segment 2" },
 { "id": 3, "name": "Segment 3" },
 { "id": 4, "name": "Segment 4" }
]
```

<input checked="" type="checkbox"/>	1	2025-01-01	example
<input type="checkbox"/>	2	2026-01-01	example
<input checked="" type="checkbox"/>	3	2026-02-02	example
<input type="checkbox"/>	4	2026-03-03	example

The user selected Segment 1 and Segment 3. The field will store:

```
[
 { "id": 1, "name": "Segment 1" },
 { "id": 3, "name": "Segment 3" }
]
```

In the **pipeline**, the data can be accessed through templating. For example, if the field is configured in Campaign:

Template	Result
<code>\$campaign.segment.%1.name</code>	Segment 3
<code>\$campaign.segment</code>	<code>[{ "id": 1, "name": "Segment 1" }, { "id": 3, "name": "Segment 3" }]</code>

If **Select entity from list** is disabled, the list of data cannot be edited from the interface — all data will be loaded and written to the field automatically by the operation **pipeline**.

# Templating Language

The templating language is used in **pipelines** and external objects to access channel data. It allows you to insert values of template fields, mailing, sender, subscription, and other entities into **pipe** parameters — for example, when forming a JSON request to an external API.

## Prefixes

Each template starts with one of three prefixes that determine how the value is processed:

Prefix	Behavior	Can be combined with text
\$	Inserts the value as a string	Yes
\$#	Inserts the value preserving the original type (int, bool, array, etc.)	No
\$\$	Unpacks an array into a repeating structure. Used for fields of type "file"	No

### Notation format

For the `$` and `$#` prefixes, two notation variants are available:

`$template.field`

`${template.field}`

The curly brace variant is convenient when you need to clearly separate the template from surrounding text:

`qwerty${in.a_id}qwerty`

For the `$$` prefix, only the notation without curly braces is allowed.

### Concatenation with text

A template with the `$` prefix can be freely combined with constant text:

`https://example.com/handler/v${sender.version}/test`

Templates with the `$#` and `$$` prefixes cannot be concatenated with strings — they return the value strictly of their own type. For example, the notation `qwerty$#in.a_id` will cause an error if `a_id` has type `int`, because a string and a number cannot be concatenated.

## Keywords

**Keywords** are the names of objects available in the **pipeline** context. Access to object fields is done through a dot (`.`), with no limit on nesting depth.

### Pipeline data

Keyword	Type	Description
<code>\$in</code>	Object	Data passed to the current <b>pipe</b> from the previous <b>pipe</b> (if any)
<code>\$in.send_message_id</code>	string	Full message identifier (SMID), generated by the mailing for each profile
<code>\$in.unique_message_id</code>	string	Unique message identifier (SMID with version)
<code>\$in.launch_id</code>	string	Unique sending identifier. One per mailing launch
<code>\$in.a_id</code>	int	Account identifier
<code>\$in.c_id</code>	int	Mailing identifier
<code>\$origin</code>	Object	Original data that came into the <b>pipeline</b>

### CAUTION

In the [Schedule pipeline](#), `$origin` is replaced with the value of `params.template` of the [Scheduler pipe](#). The original **pipeline** data is not available — pass everything needed through the `template` of the Scheduler **pipe**.

### Channel entities

Keyword	Type	Description
<code>\$template</code>	Object	Data filled in the message for the channel. Fields are accessed by shortname: <code>\$template.&lt;field_shortname&gt;</code> . If <a href="#">presets</a> are used, only fields of the selected preset are available. Current preset: <code>\$template.presets_ids</code>
<code>\$sender</code>	Object	Data filled in the sender for the channel
<code>\$resource</code>	Object	Data filled in the resource for the channel
<code>\$campaign</code>	Object	Data filled in the mailing for the channel
<code>\$subscription</code>	Object	Data filled in the subscription for the channel
<code>\$account</code>	Object	Data filled in the account for the channel

Mailing schedule

Keyword	Type	Description
<code>\$camp_schedule</code>	Object	Mailing start and stop settings
<code>\$camp_schedule.start_time</code>	time	Mailing start time in the schedule
<code>\$camp_schedule.stop_time</code>	time	Mailing stop time in the schedule

Errors

Keyword	Type	Description
<code>\$error</code>	Object	Error data from the previous <b>pipe</b>
<code>\$error.code</code>	int	Error code (for example, HTTP code)
<code>\$error.message</code>	string	Text description of the error

Accessing fields and arrays

Object fields

Access to an object field is done through a dot ( `.` ):

```
$template.field_name.inner_field_name
```

Array elements

Access to an array element is done through `.<index>` (zero-based indexing):

```
$template.field_attach.%0.name
```

Functions

The following built-in functions are available in templates:

General functions

Function	Description	Example	Result
<code>\$base64</code>	Encodes a value in base64	<code>\$base64('Hello')</code> or <code>\$base64(\$sender.login:\$sender.password)</code>	<code>SGVsbG8=</code>
<code>\$gen_uniq_int_id</code>	Generates a unique 10-digit number	<code>\$gen_uniq_int_id()</code>	<code>1726573911</code>
<code>\$escapeCharacters</code>	Removes special characters from a string	<code>\$escapeCharacters('Hello \'world\'')</code>	<code>Hello world</code>
<code>\$getWebhookURL</code>	Forms a webhook URL for a callback	<code>\$getWebhookURL(\$origin.send_message_id)</code>	<code>https://altcraft.com/fbp/v1/1/1?ffffffffffff</code>
<code>\$removeCodeFromPhone</code>	Removes the region code from a phone number	<code>\$removeCodeFromPhone('89505062312')</code>	<code>9505062312</code>
<code>\$#toObject</code>	Converts a JSON string to an object	<code>\$#toObject({'a': 'b'})</code>	<code>{"a": "b"}</code>

Getting entity names by message\_id

These functions allow you to get human-readable entity names by message identifier. Useful for logging, report generation, and conditional branching in **pipelines**. If the entity is not found, the function returns an error.

Function	Description	Example	<code>message_id</code>	<code>unique_message_id</code>
<code>\$getCampaignName</code>	Campaign name	<code>\$getCampaignName(\$in.unique_message_id)</code>	Yes	Yes
<code>\$getTemplateName</code>	Template name	<code>\$getTemplateName(\$in.send_message_id)</code>	Yes	—
<code>\$getDatabaseName</code>	Database name	<code>\$getDatabaseName(\$in.unique_message_id)</code>	Yes	Yes
<code>\$getSegmentName</code>	Segment name. If segment is not set — <code>No segment</code>	<code>\$getSegmentName(\$in.send_message_id)</code>	Yes	—
<code>\$getWorkflowName</code>	Scenario name. If scenario is not set — <code>No workflow</code>	<code>\$getWorkflowName(\$in.send_message_id)</code>	Yes	—
<code>\$getResourceName</code>	Resource name. If resource is not set — <code>No resource</code>	<code>\$getResourceName(\$in.unique_message_id)</code>	Yes	Yes

Getting entity IDs by message\_id

Function	Description	Example	<code>message_id</code>	<code>unique_message_id</code>
<code>\$getCampaignID</code>	Campaign identifier (string)	<code>\$getCampaignID(\$in.unique_message_id)</code>	Yes	Yes
<code>\$getTemplateID</code>	Template identifier (string)	<code>\$getTemplateID(\$in.send_message_id)</code>	Yes	—
<code>\$getDatabaseID</code>	Database identifier (string)	<code>\$getDatabaseID(\$in.unique_message_id)</code>	Yes	Yes
<code>\$getSegmentID</code>	Segment identifier (string)	<code>\$getSegmentID(\$in.send_message_id)</code>	Yes	—
<code>\$getWorkflowID</code>	Scenario identifier (string)	<code>\$getWorkflowID(\$in.send_message_id)</code>	Yes	—
<code>\$getResourceID</code>	Resource identifier (string)	<code>\$getResourceID(\$in.unique_message_id)</code>	Yes	Yes

Getting attribute value

The function `$getAttributeValue` returns the value of a custom attribute (custom field) by its identifier. Attributes are set at the mailing, workflow, or campaign level and are inherited hierarchically: the mailing value has the highest priority, then the workflow, then the campaign.

Function	Description	Example
<code>\$getAttributeValue</code>	Returns the attribute value (string). First argument — <code>message_id</code> , second — attribute identifier (string)	<code>\$getAttributeValue(\$in.send_message_id, "attribute_1")</code>

If the attribute is not found at any level of the hierarchy, the function returns an empty string.

Examples

All prefixes

Input data of the **pipe**:

```
{
 "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"
 }
}
```

Result after template processing:

```
{
 "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"
 }
]
}
```

Template in URL

The `params.url` parameter of the [HTTP Request](#) pipe:

```
https://example.com/handler/v${sender.version}/test
```

After processing (when `sender.version = 2`):

```
https://example.com/handler/v2/test
```

Concatenation and types

The `params.template.test_field` parameter of the [Pack](#) pipe:

```
qwerty$in.a_id
```

Result (when `in.a_id = 5`):

```
qwerty5
```

The same notation with the `$$` prefix will cause an error, because a string and a number cannot be "concatenated":

```
qwerty$$in.a_id // error
```



# Sending Files

When a channel template (Template) has a field of type `file` or `image_upload`, the file data is passed to the **pipeline** in base64 format. To send files, the `$$` prefix of the [templating language](#) is used, which unpacks an array of files into a repeating structure.

## File data structure

Each file in the array contains the following fields:

Field	Description
<code>content</code>	File data in base64
<code>name</code>	File name
<code>mime</code>	MIME type of the file (for example, <code>image/png</code> , <code>application/pdf</code> )
<code>size</code>	File size in bytes

## Syntax in Pack

To unpack an array of files in the [Pack pipe](#), the following construct is used:

```
$$template.<shortname>[<array_name>]
```

Where:

- `<shortname>` — shortname of the file field in Template (for example, `file_field`)
- `<array_name>` — key name in the resulting JSON (for example, `array`, `files`, `attachments`)

## Example: single file

If Template has a single file field `file_field`, the Pack **pipe** will look like this:

```
{
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON",
 "template": {
 "message": "$template.message",
 "subid": "$subscription.subid",
 "$$template.file_field[files]": {
 "content": "[${}content",
 "filename": "[${}name",
 "type": "[${}mime",
 "size": "[${}size"
 },
 "campaignId": "$in.c_id"
 }
 },
 "outs": {
 "success": 2,
 "error": 3
 }
}
```

Result in `$in.body` (with one file):

```
{
 "message": "Hello",
 "subid": "sub_123",
 "files": [
 {
 "content": "base64encodeddata...",
 "filename": "document.pdf",
 "type": "application/pdf",
 "size": "45000"
 }
],
 "campaignId": "101"
}
```

## Example: multiple files

If the user uploaded multiple files, each will be added to the `files` array:

```
{
 "message": "Hello",
 "subid": "sub_123",
 "files": [
 {
 "content": "base64data1...",
 "filename": "image1.png",
 "type": "image/png",
 "size": "12000"
 },
 {
 "content": "base64data2...",
 "filename": "image2.png",
 "type": "image/png",
 "size": "8000"
 }
],
 "campaignId": "101"
}
```

## Full pipeline example with a file

```
[
 {
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON",
 "template": {
 "message": "$template.message",
 "subid": "$subscription.subid",
 "$template.file_field[files]": {
 "content": "[$]content",
 "filename": "[$]name"
 },
 "campaignId": "$in.c_id"
 }
 },
 "outs": {
 "success": 2,
 "error": 3
 }
 },
 {
 "id": 2,
 "type": "http_request",
 "params": {
 "method": "POST",
 "timeout": 30,
 "content_type": "json",
 "url": "https://api.example.com/send",
 "success_codes": [200]
 },
 "outs": {
 "success": 4,
 "error": 3
 }
 },
 {
 "id": 4,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100001,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 },
 {
 "id": 3,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100003,
 "event": {
 "send_message_id": "$origin.send_message_id",
 "message": "$error.code $error.message"
 }
 }
 }
]
```

#### TIP

Note the HTTP request timeout: when sending files, it is recommended to increase `timeout` (for example, to 30 seconds), since base64 data increases the request body size.

#### CAUTION

A channel is not designed to send attachments larger than 1 MB per communication. It is recommended not to exceed 500 KB. For more details, see [technical limitations](#).

# Presets (Field Sets)

**Preset** is a set of fields corresponding to a specific message type in a custom channel. Presets allow a single channel to support multiple content types: text message, photo, video, document, etc. The user in the platform interface selects one preset and sees only the fields of that preset.

Presets are configured by the administrator on the **Object Fields** tab — **Template** subsection when setting up a channel.

## Why presets are needed

Without presets, all Template fields are displayed simultaneously, which is inconvenient if the channel supports different content types. For example, a Telegram channel can send text messages, photos, and videos — but each type requires different fields:

- **Text:** "Message text" field
- **Photo:** "Image" field + "Caption" field
- **Video:** "Video file" field + "Description" field

With presets, the user selects a content type and sees only the relevant fields.

## Creating a preset

In the **Template** subsection on the **Object Fields** tab, the **Presets** block is available. To create a preset, click the **Add** button.

A modal window will open with the following fields:

Field	Description
Shortname	Unique identifier for reference in the pipeline (e.g., <code>text</code> , <code>photo</code> , <code>video</code> ). Must contain only letters a-z, digits 0-9, and underscores. Cannot be changed after creation
Title	Preset name displayed in the platform interface (e.g., "Text message")
Icon	Preset icon displayed on the card in the platform interface
Description	Preset description displayed below the icon

Account Area ▾Settings ▾NodesSenders ▾Reports ▾Tools ▾

Custom Channels > #10003

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

Template \*ResourceCampaignSenderSubscription \*Account

Presets

Template\_P...

Add

Links shorter enabled ?

Preset config

\* Title

Text Preset

\* Shortname

text\_preset

Preset icon

comment

Preset description

Text content for a message

After creation, presets are displayed as cards. The order is determined by position — rearrange presets using drag-and-drop.

Each preset has management buttons:

Action	Description
Edit	Edit the preset name, icon, and description. Shortname cannot be changed
Purge preset	Clear bindings — all fields are unbound from this preset
Delete	Delete the preset. Template fields are not deleted

## Binding fields to a preset

After creating a preset, you need to bind Template fields to it. Binding is done when editing each field: open the field via the **Edit** button and in the **Presets** dropdown, select one or more presets.

Account Area ▾Settings ▾NodesSenders ▾Reports ▾Tools ▾

Custom Channels > #10003

GeneralEventsExternal objectsObject fieldsPipelinesLanguages

Template \*ResourceCampaignSenderSubscription \*Account

Presets

Text\_Preset

Sound\_Preset

+

Add

Links shorter enabled ?

System forms

External object field config

Field ?Layout element ?

\* Title

Notification Text

\* Shortname

notification\_text

Type

Text content ▾

This field is required

Add to web version

\* Preset

Text\_Preset × ▾

Text\_Preset ✓

Sound\_Preset

HTML ▾

Markdown

Bold ×Italic ×Strikethrough ×HTML ×

Preformatted ×Underlined ×

Unordered list ×Ordered list ×

A field can be bound to multiple presets simultaneously.

CAUTION

Fields not bound to any preset will not be displayed in the user interface when a preset is selected. If a field should be available, make sure to bind it to a preset.

Presets in the platform interface

When creating a template, the user sees preset cards with icons and descriptions. Selecting a preset determines which fields are available for filling.

## Presets

The diagram illustrates three categories of presets, each represented by a rounded rectangle with a dashed border. The first category, 'Text\_Preset', contains three icons (pencil, eraser, trash) and a blue speech bubble icon. The second category, 'Sound\_Preset', contains the same three icons and a blue bell icon. The third category, 'Add', contains a large plus sign icon.

☒ Links shorter enabled (?)

## System forms

≡

≡≡

≡≡≡

≡≡≡≡

≡≡≡≡≡

≡≡≡≡≡≡

✕ Remove row

✎

🔊

text

Text Content

Please enter a default value

💬

✎

🔊

sound

Element Info

- type: file
- Title: sound

🔔

+ Add field row

+ Add object

AD New custom channel X

+

Text\_Preset ▼

  
Text\_Preset

  
Sound\_Preset

Select one of this channels preset for further editing

TEXT CONTENT

B I       

utm

 ▼

Please enter a value

### Template Preview

You can configure how the data that the user uploads to the template is displayed using **Template Preview** fields. Here you set up the template preview that the user sees when filling in the fields.

For example, if you use a preset with image upload, you can configure the preview as follows:





## Preview for New stellar template

Channels:

AD

New custom channel

▼

Your prize ticket is available for download:



In the preview template, Template fields are accessed via `{ccfield.<shortname>}`. For example, `{ccfield.text}` will substitute the value from the field with shortname `text`.

### Accessing preset in the pipeline

Template field data is available in the pipeline via the [templating language](#) with the syntax:

```
$template.<field_shortname>
```

Where `<field_shortname>` is the field's shortname (e.g., `message`, `image`, `caption`).

The currently selected preset is available via the system field:

```
$template._presets_ids
```

This is an array of preset shortnames (e.g., `["text"]` or `["photo"]`).

#### CAUTION

In the pipeline, only fields corresponding to the selected preset are available. If the user selected the `photo` preset, fields bound only to the `text` preset will be empty. Use `$template._presets_ids` to determine the current preset and implement conditional logic.

### Example: Telegram channel

Consider a Telegram channel with three presets: `text`, `photo`, `video`.

#### Presets

Name	Shortname	Icon	Description
Text message	text		Simple text message
Photo	photo		Message with an image
Video	video		Message with a video file

#### Template Fields

Title	Shortname	Type	Preset binding
Message text	message	text	text
Image	image	image_upload	photo

Title	Shortname	Type	Preset binding
Photo caption	caption	string	photo
Video file	file	file	video
Video description	description	string	video
Webhook URL	webhook	string	text, photo, video

Message Pipeline

Depending on the selected preset, the Pack pipe forms different JSON:

```
{
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON",
 "template": {
 "preset": "$template.presets_ids",
 "chat_id": "$subscription.chat_id",
 "text": "$template.message",
 "photo": "$template.image",
 "photo_caption": "$template.caption",
 "video": "$template.file",
 "video_description": "$template.description",
 "webhook": "$template.webhook"
 }
 },
 "outs": {
 "success": 2,
 "error": 99
 }
}
```

The external service (Telegram bot) determines which API method to call based on the preset field: sendMessage for text , sendPhoto for photo , sendVideo for video . Fields not related to the selected preset will be empty.

Telegram Bot

Select preset

Message

Image content

Audio

Document

Video

Text message (1-4096 characters). Basic formatting is supported. You can use bold, italic, underlined and strikethrough fonts.

# Debugging

## Logs

All custom channel **pipelines** log to `procpiper.log`. For debugging, it is useful to add [Log pipes](#) at key points in the **pipeline** — before and after HTTP requests, after unpacking the response, before registering events.

### Example: logging around an HTTP request

```
[
 {
 "id": 1,
 "type": "log",
 "params": {
 "message": "[Request] URL: $sender.url, method: POST",
 "level": "debug"
 },
 "outs": {
 "success": 2
 }
 },
 {
 "id": 2,
 "type": "http_request",
 "params": {
 "method": "POST",
 "timeout": 10,
 "content_type": "json",
 "url": "$sender.url",
 "success_codes": [200]
 },
 "outs": {
 "success": 3,
 "error": 5
 }
 },
 {
 "id": 3,
 "type": "unpack",
 "outs": {
 "success": 4,
 "error": 5
 }
 },
 {
 "id": 4,
 "type": "log",
 "params": {
 "message": "[Response] status: $in.data.status",
 "level": "debug"
 }
 },
 {
 "id": 5,
 "type": "log",
 "params": {
 "message": "[Error] code: $error.code, message: $error.message",
 "level": "error"
 }
 }
]
```

## Error data

When any **pipe** encounters an error, the following variables are available:

Variable	Description
<code>\$error.code</code>	Error code (number). <b>Not all pipes return a code</b> — see the table below
<code>\$error.message</code>	Text description of the error. Available for all pipes

### Error structure by pipe

Pipe	<code>\$error.code</code>	<code>\$error.message</code>
HTTP Request	Yes (HTTP code: 400, 403, 500, etc.)	Yes (error description: timeout, connection refused, etc.)
Pack	No	Yes (template error description)
Unpack	No	Yes (unmarshalling error description)
SQL	No	Yes (SQL error description)
Eventer	No	Yes (template or format error description)
Scheduler	No	Yes (template error description)
Selector	No	Yes (JsonQL error description)
Store Set / Store Get	No	Yes (database error description)
Log	No	Yes (template error description)
Result	No	Yes (path error or missing data description)
Error	No	Yes (message template error description)

Use them in [Log](#), [Eventer](#), or [Error pipes](#) for diagnostics.

Typical errors

SQL: forgot `map: true` in Read operation

**Symptom:** Result fails with a data type error. **Cause:** Without `map: true`, SQL returns an array `[]map[string]interface{}`, but Result expects an object `map[string]interface{}`. **Solution:** Add `"map": true` to the SQL **pipe** parameters for the Read operation.

Templating: `$` instead of `$$`

**Symptom:** A numeric field arrived as a string, the external API returned a validation error. **Cause:** The `$` prefix always returns a string. The `$$` prefix preserves the original type. **Solution:** Use `$$` for numbers, booleans, arrays, and objects. For example, `$$template.discountAmount` instead of `$template.discountAmount`.

Scheduler: did not pass `send_message_id` in template

**Symptom:** Eventer in the Schedule **pipeline** cannot register an event — `$origin.send_message_id` is missing. **Cause:** In Schedule, `$origin` is replaced with `params.template` of the Scheduler **pipe**. **Solution:** Pass `send_message_id` explicitly through the template:

```
{
 "template": {
 "send_message_id": "$origin.send_message_id",
 "external_id": "$#in.data.externalId"
 }
}
```

And access it in Schedule as `$origin.send_message_id`.

HTTP Request: missing Pack before `content_type: json`

**Symptom:** Error `$in.body not found`.  
**Cause:** `content_type: json` reads the body from `$in.body`, which is formed by the Pack **pipe**.  
**Solution:** Add a Pack **pipe** before HTTP Request in the chain.

HTTP Request: `$in.body` is not a byte array

**Symptom:** Data type error for `$in.body`.  
**Cause:** Pack **pipe** is not configured or `type` is not specified.  
**Solution:** Make sure the Pack **pipe** has `"type": "JSON"` (or `"XML"`) and is placed before HTTP Request.

Eventer: incorrect `channel_id` / `action_type`

**Symptom:** Event does not appear in reports and profile history.  
**Cause:** `channel_id` does not match the numeric ID of the channel, `action_type` does not match the Event ID of the created event.  
**Solution:** Verify the correspondence in the Events section of the channel: `action_type` = Event ID of the event.

Selector: error in JsonQL

**Symptom:** Event goes to `outs.error` of the Selector **pipe**.  
**Cause:** Incorrect JsonQL query syntax or field not found.  
**Solution:** Check the query syntax and make sure the field exists in `$in` or `$template`.

Channel verification after creation

- 1. Create a template with minimal content
- 2. Create a sender and a subscription (or a resource for indirect)
- 3. Send a test mailing
- 4. Check `procpiper.log` — entries from Log **pipes** should appear
- 5. Check events in profile history — a `send` or `undeliv` event should appear
- 6. If there is an error — add Log **pipes** before and after the problematic **pipe** to localize the failure point

# Technical Limitations

## No array iteration in a pipeline

It is not possible to iterate over an array of objects in a **pipeline** and perform an action for each element. This means:

- If an external service returns delivery statuses for multiple messages in a single response, you cannot register events for each message separately.
- If a channel supports multiple attachments in a content field, you cannot send each attachment to the external service separately, receive links, and collect them into an array. The field data will be sent in the original form.

## Media attachment limit

A channel is not designed to send attachments larger than 1 MB per communication. It is recommended not to exceed 500 KB.

This limitation does not apply to fields of type `image url` — such a field passes a link to an image, not the file data itself.

## Synchronous External Objects pipelines

**Pipeline** operations of External Objects (List, Read, Create, Update, Delete) are executed synchronously — the user in the interface waits for the result. Therefore, the **pipeline** must be designed to execute quickly and must always end with a Result or Error **pipe**.

# Pipelines

**Pipeline** is a chain of data processing elements that describes how a channel interacts with external services. A pipeline is defined as a JSON array and configured on the Pipelines tab when setting up a channel.

**Pipe** is an element of a pipeline that performs a single specific action: sends an HTTP request, packs data into JSON, writes a log, registers an event, etc. Each pipe has an input and can have one or more outputs. If a pipe has no outputs, the event exits the pipeline.

A complete list of pipes with parameter descriptions and examples is available in the [Pipes](#) section.

## How a pipeline works

A pipeline works by sequentially passing an event from pipe to pipe:

- Each pipe has a unique `id`, which also serves as its input.
- The pipe's outputs ( `outs` ) specify the `id` of the next pipe that the event will move to on success ( `success` ) or error ( `error` ).
- If an output is not specified, the event exits the pipeline.
- Data passed to a pipe from the previous pipe is accessible via the `$in` keyword.
- The original pipeline data is accessible via `$origin`.

You can access field values inside a pipe using the [templating language](#).

## Flow visualization

```
[Pipe 1: Pack] —success—> [Pipe 2: HTTP Request] —success—> [Pipe 3: Eventer (deliv)]
 |
 error
 v
 [Pipe 4: Eventer (undeliv)]
```

Each arrow is defined through the pipe's `outs`: `success` specifies the next pipe's `id` on success, `error` — on error. If an output is not specified, the event exits the pipeline.

## Custom Channels > Edit

General

Events

External objects

Object fields

Pipelines

Languages

Message

Scheduled

Listener

Stop

Moderate

```
1 [
2 {
3 "id": 1,
4 "type": "log",
5 "params": {
6 "message": "[Message] Start | campaign: $in.c_id | send_message_id: $in.send_message_id",
7 "level": "info"
8 },
9 "outs": {
10 "success": 2
11 }
12 },
13 {
14 "id": 2,
15 "type": "pack",
16 "params": {
17 "type": "JSON",
18 "template": {
19 "chat_id": "$subscription.chat_id",
20 "text": "$template.text",
21 "parse_mode": "HTML",
22 "webhook_url": "$getWebhookURL($in.send_message_id)"
23 }
24 },
25 "outs": {
```

Save

Close

### Working with array data

Some channel entity fields can contain **array values**. This happens in two cases:

- **External Object with List operation** — only available for Template and Campaign. The user can select multiple values (e.g., multiple profiles or segments), and the List operation result enters the pipeline as an array.
- **Fields with multiple selection** — some field types (e.g., `file`, `image_upload`, `enum` with multi-select option) also return arrays.

If you need to extract a specific value from the array, use indexing via `."<index>` (zero-based indexing):

```
$template.profile.%0.id
```

This returns the `id` of the first selected item. To access the second — `%1`, the third — `%2`, and so on.

If you need to access the entire array, use the keyword without indexing:

```
$template.profile
```

To unpack an array into a repeating structure (for example, when building JSON with multiple attachments), use the `$$` prefix. For more details, see the [Templating Language](#) section.

### Pipe structure

Each pipe is described as a JSON object with the following fields:

Field	Example	Required	Description
id	5	Yes	Unique identifier of the pipe in the pipeline. Also serves as its input.
type	"log"	Yes	Pipe type (determines which action is performed).
params	{}	No	Parameters specific to this pipe type.
outs	OutsObject	No	Pipe outputs — determine which pipe the event will move to.

Pipe outputs (OutsObject)

Field	Example	Required	Description
success	3	No	id of the pipe the event will move to on successful execution.
error	4	No	id of the pipe the event will move to on error.

Parallel processing

Each pipe in a pipeline processes events in parallel: multiple independent worker processes are spawned for each pipe. The number of workers is configured via the PIPE\_WORKER\_SIZE parameter in the platform configuration. This allows multiple events for the same pipe to be processed concurrently.

Pipeline example

Below is an example of a pipeline that builds a JSON request, sends it over HTTP, and depending on the result, registers a delivery or non-delivery event:

```
[
 {
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON"
 },
 "outs": {
 "success": 2
 }
 },
 {
 "id": 2,
 "type": "http_request",
 "params": {
 "timeout": 10,
 "content_type": "json",
 "success_codes": [200, 202]
 },
 "outs": {
 "success": 3,
 "error": 4
 }
 },
 {
 "id": 3,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100002,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 },
 {
 "id": 4,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100003,
 "event": {
 "send_message_id": "$origin.send_message_id",
 "message": "$error.code $error.message"
 }
 }
 }
]
```

Here:



1. Pipe **1** (Pack) packs data into JSON and passes the event to pipe **2**.
2. Pipe **2** (HTTP Request) sends an HTTP request. On success, the event goes to pipe **3**; on error, to pipe **4**.
3. Pipe **3** (event) registers a delivery event.
4. Pipe **4** (event) registers a non-delivery event with the error code and message.

# Message

**Message** is the main pipeline of a channel. All messages sent by a mailing go through it.

Here you define the chain of pipes that builds a request to an external service, sends it, processes the response, and registers events (send, delivery, non-delivery, etc.).

The pipeline is configured as a JSON array on the **Pipelines > Message** tab when setting up a channel. By default, the editor contains an empty array `[]`.

## Minimal working example

A typical Message pipeline consists of four pipes: data packing, HTTP request, event registration on success, and event registration on error.

```
[
 {
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON",
 "template": {
 "chat_id": "$subscription.chat_id",
 "text": "$template.text"
 }
 },
 "outs": {
 "success": 2
 }
 },
 {
 "id": 2,
 "type": "http_request",
 "params": {
 "method": "POST",
 "timeout": 10,
 "content_type": "json",
 "url": "https://api.example.com/send",
 "success_codes": [200]
 },
 "outs": {
 "success": 3,
 "error": 4
 }
 },
 {
 "id": 3,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100001,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 },
 {
 "id": 4,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100003,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 }
]
```

Here:

1. **Pack** (id: 1) — builds JSON from subscription and template data
2. **HTTP Request** (id: 2) — sends a request to the external API
3. **Eventer** (id: 3) — registers a delivery event on HTTP 200
4. **Eventer** (id: 4) — registers a non-delivery event on error

For details on pipeline structure and available pipes, see the [Pipelines](#) and [Pipes](#) sections.

# Schedule

**Schedule** is a pipeline for processing deferred events. Events are written to the database using the [Scheduler](#) pipe with a specified execution time. The Schedule pipeline periodically scans the database and launches accumulated events into its pipe chain when the scheduled time arrives.

## Features

- Event scanning occurs every `SCHEDULER_PIPE_ROUTER_PERIOD` seconds (default — 10).
- The event passed to Schedule replaces the data in `$origin` with the value of the Scheduler pipe's `params.template`.
- In `$origin` of the Schedule pipeline, the system field `_schedule_try_count` is available — a counter of event execution attempts (starting from 1). Useful for implementing retry logic with increasing delays.

### CAUTION

In Schedule, `$origin` is replaced with the Scheduler pipe's `params.template`. The original data is not available — pass everything needed (including `send_message_id`) through the template.

## Usage example in Message

In the Message pipeline, the Scheduler pipe passes an event to Schedule with a delay:

```
{
 "id": 801,
 "type": "scheduled",
 "params": {
 "pipe_id": 1,
 "delay": 600,
 "template": {
 "camp_id": "$origin.c_id",
 "send_message_id": "$origin.send_message_id",
 "external_id": "$#in.data.externalId"
 }
 },
 "outs": {
 "success": 802,
 "error": 803
 }
}
```

After 600 seconds, the event will enter pipe `1` of the Schedule pipeline.

## Schedule pipeline example

Typical scenario: checking delivery status 10 minutes after sending.

```
[
 {
 "id": 1,
 "type": "log",
 "params": {
 "message": "[Schedule] try_count: $origin._schedule_try_count, external_id: $origin.external_id",
 "level": "debug"
 },
 "outs": {
 "success": 2
 }
 },
 {
 "id": 2,
 "type": "http_request",
 "params": {
 "method": "GET",
 "timeout": 10,
 "content_type": "json",
 "url": "https://api.example.com/status?id=$origin.external_id",
 "success_codes": [200]
 },
 "outs": {
 "success": 3,
 "error": 5
 }
 },
 {
 "id": 3,
 "type": "unpack",
 "outs": {
 "success": 4,
 "error": 5
 }
 },
 {
 "id": 4,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100002,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 },
 {
 "id": 5,
 "type": "log",
 "params": {
 "message": "[Schedule Error] code: $error.code, message: $error.message",
 "level": "error"
 }
 }
]
```

Here:

1. **Log** (id: 1) — logs the attempt number and external\_id
2. **HTTP Request** (id: 2) — checks delivery status by external\_id
3. **Unpack** (id: 3) — unpacks the JSON response
4. **Eventer** (id: 4) — registers a delivery event (deliv, action\_type 100002 )
5. **Log** (id: 5) — logs an error if the request fails

# Listener

**Listener** is a pipeline for processing incoming events from external services. Events are received in two ways: via the platform's tracking domain (HTTP) or via RabbitMQ.

## Features

- An event can land on any pipe in the pipeline. The `id` of the initial pipe is specified in the URL to which the external service sends data.
- In the Listener pipeline settings, there is an additional field **Response body** — a string that the platform will use to respond to incoming HTTP requests.

## Event sources

The Listener can receive events from two sources:

### HTTP via tracking domain

The external service sends an HTTP request to the platform's tracking domain. Nginx automatically proxies the request to the internal `trklistener` service.

### RabbitMQ

If the external service publishes events to RabbitMQ, the Listener can pull them directly from the queue. To do this, specify a connector and queue name in the Listener pipeline settings:

Field	Description
Type	Connector type. Default is <code>Not chosen</code>
Connector shortname	A connector created in the <a href="#">External Data Configuration</a> section. Default is <code>Not chosen</code>
Queue name	RabbitMQ queue name from which the Listener will pull events

#### Type

 RabbitMQ

#### Connector shortname

#4 RMQ Connector

#### Queue name

#### TIP

Connector and queue settings are only needed if events arrive via RabbitMQ. If events come via HTTP through the tracking domain, all three fields remain at their defaults (`Not chosen` / empty).

## Data transfer

### HTTP request

The external service sends data to the URL:

```
https://<tracking_domain>/fbp/v1/<custom_channel_sid>/<pipe_id>
```

Where:

- `<tracking_domain>` — the tracking domain of your platform (for example, `track.yourplatform.com`). Confirm with the platform administrators.
- `<custom_channel_sid>` — the channel's string identifier (String ID).
- `<pipe_id>` — the `id` of the pipe in the Listener pipeline where the event will land.

GET parameters of the request are written to `$in.form`. The request body is written to `$in.body` as a byte array. To convert bytes into an object, use the [Unpack](#) pipe.

### RabbitMQ

The RabbitMQ message body is written to `$in.body` as a byte array. Message headers are written to `$in.headers` (hyphens in keys are replaced with underscores).

## Listener pipeline example

Typical scenario: an external service sends a webhook about delivery status.

```

[
 {
 "id": 1,
 "type": "unpack",
 "params": {
 "type": "JSON"
 },
 "outs": {
 "success": 2,
 "error": 4
 }
 },
 {
 "id": 2,
 "type": "selector",
 "params": {
 "outs": [
 {
 "query": "('$in.data.status' = 'delivered')",
 "out": 3
 },
 {
 "query": "('$in.data.status' = 'failed')",
 "out": 5
 }
]
 },
 "outs": {
 "success": 6,
 "error": 4
 }
 },
 {
 "id": 3,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100002,
 "event": {
 "send_message_id": "$in.form.send_message_id"
 }
 }
 },
 {
 "id": 5,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100003,
 "event": {
 "send_message_id": "$in.form.send_message_id",
 "message": "$error.code $error.message"
 }
 }
 },
 {
 "id": 6,
 "type": "log",
 "params": {
 "message": "[Listener] status unknown: $in.data.status",
 "level": "warn"
 }
 },
 {
 "id": 4,
 "type": "log",
 "params": {
 "message": "[Listener Error] code: $error.code, message: $error.message",
 "level": "error"
 }
 }
]

```



Here:

1. **Unpack** (id: 1) — unpacks the webhook body from `$in.body` into `$in.data`
2. **Selector** (id: 2) — branches by status: `delivered` → pipe 3, `failed` → pipe 5
3. **Eventer** (id: 3) — registers delivery (deliv)
4. **Eventer** (id: 5) — registers non-delivery (undeliv) with error text
5. **Log** (id: 6) — logs unknown status
6. **Log** (id: 4) — logs unpacking error

# Moderate

**Moderate** is a pipeline for processing mailing moderation actions. It is available only for **indirect** channels that have the **Enable moderation** (With moderate) option enabled in their main settings.

## How moderation works

From a technical perspective, moderation is a separate launch of a mailing, where the message is sent not to the [Message](#) pipeline, but to **Moderate**. In this pipeline, you can implement the logic for sending content for approval, processing the moderation result, and registering the corresponding events.

### User workflow

1. **Creation and activation.** The user creates a placement campaign on a channel with moderation enabled and activates it. The campaign receives the **ModerationPending** status. The edit form is locked, and only the Abort button is available.
2. **Sending for moderation.** The platform launches a binary campaign process with the `SendForModerate=true` flag. Messages are routed to the **Moderate** pipeline via the `piper_moderate` queue instead of being sent directly. The pipeline processes the content (e.g., sends it to an external moderation service via HTTP Request).
3. **Moderation result.** The external service returns a result processed by the [Eventer](#) pipe:
  - `moderate_pass` — moderation passed. Campaign status changes to **ModerationPassed**. The mailing will launch on schedule if configured.
  - `moderate_error` — moderation failed. Campaign status changes to **ModerationError**. A red warning icon and the error message from the Eventer `message` field are displayed in the interface.
4. **Fix and resubmit.** With **ModerationError** status, the user can fix the campaign content and save it again. The campaign is automatically sent for re-moderation (returns to **ModerationPending** status).
5. **Launch the mailing.** After successful moderation (**ModerationPassed**), the mailing launches on schedule, and the message is delivered through the regular [Message](#) pipeline.

### Limitations

- A campaign in **ModerationPending** or **ModerationPassed** status cannot be re-activated — it can only be deactivated.
- Immediate campaign launch is not available for channels with moderation: the “Start now” checkbox is hidden when creating a campaign, and the “Run Now” button is unavailable for campaigns in **ModerationPending** and **ModerationError** statuses.

## Pipeline configuration

To manage moderation status, use the [Eventer](#) pipe with event types `moderate_pass` and `moderate_error`.

### NOTE

For `moderate_pass`, `moderate_error`, and `stop` events, the `event_type` parameter is specified in Eventer, and `action_type` is not required. For all other event types, `event_type` is not used — instead, `action_type` (the event ID from the channel's Events section) is specified.

## Minimal example

A pipeline that sends content for moderation via HTTP and processes the result:

```
[
 {
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON",
 "template": {
 "content": "$template.text",
 "campaign_id": "$origin.c_id"
 }
 },
 "outs": {
 "success": 2
 }
 },
 {
 "id": 2,
 "type": "http_request",
 "params": {
 "method": "POST",
 "timeout": 10,
 "content_type": "json",
 "url": "https://api.example.com/moderate",
 "success_codes": [200]
 },
 "outs": {
 "success": 3,
 "error": 4
 }
 },
 {
 "id": 3,
 "type": "event",
 "params": {
 "event_type": "moderate_pass",
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 },
 {
 "id": 4,
 "type": "event",
 "params": {
 "event_type": "moderate_error",
 "event": {
 "send_message_id": "$origin.send_message_id",
 "message": "$in.body.error"
 }
 }
 }
]
```

In this example:

- **Pipe 1 (Pack)** forms the request body with message content and campaign ID.
- **Pipe 2 (HTTP Request)** sends the request to the external moderation service. Depending on the result, it routes to pipe 3 (success) or pipe 4 (error).
- **Pipe 3 (Eventer)** registers a `moderate_pass` event — the campaign gets **ModerationPassed** status.
- **Pipe 4 (Eventer)** registers a `moderate_error` event with the error text from the service response ( `$in.body.error` ) — the campaign gets **ModerationError** status, and the error text will be displayed in the interface.

# Stop

**Stop** is a pipeline for processing the completion of a placement mailing. If configured, the mailing can be stopped manually or on schedule.

From a technical perspective, completion is a separate launch of a mailing, where messages are sent not to the [Message](#) pipeline, but to **Stop**. In this pipeline, you can implement the logic for stopping the placement on the external service side — for example, send a request to remove a banner or cancel a campaign.

The pipeline runs in two cases:

- **Manually** — when the user deactivates the placement campaign
- **By schedule** — when the stop time specified in placement parameters (`deadline_obj`) is reached

To implement the stop action, use the [Eventer](#) pipe with the event type `stop`.

## Minimal example

A pipeline that sends a request to stop the placement and registers a `stop` event:

```
[
 {
 "id": 1,
 "type": "http_request",
 "params": {
 "method": "POST",
 "timeout": 10,
 "content_type": "json",
 "url": "https://api.example.com/campaign/stop",
 "success_codes": [200]
 },
 "outs": {
 "success": 2,
 "error": 3
 }
 },
 {
 "id": 2,
 "type": "event",
 "params": {
 "event_type": "stop",
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
 },
 {
 "id": 3,
 "type": "log",
 "params": {
 "message": "Stop failed: $error.code $error.message",
 "level": "error"
 }
 }
]
```

# HTTP Request

**HTTP Request** is a **Pipe** for executing HTTP requests to external services. It supports GET, POST, PUT, DELETE methods, various request body formats, and basic authentication.

## Request Body Format

Depending on the value of `params.content_type`, data is transmitted differently:

<code>content_type</code>	Where data comes from
<code>form</code>	From <code>params.form</code> — sent as URL GET parameters
<code>form-data</code>	From <code>params.form</code> — sent as <code>multipart/form-data</code>
<code>json</code>	From <code>\$in.body</code> (byte array, usually prepared by the <a href="#">Pack</a> pipe)
<code>xml</code>	From <code>\$in.body</code> (byte array, usually prepared by the <a href="#">Pack</a> pipe)

## Parameters

Field	Type	Required	Default	Description
<code>url</code>	string	No	<code>""</code>	Request URL. Supports <a href="#">templating</a>
<code>method</code>	string	No	<code>GET</code>	HTTP method: <code>GET</code> , <code>POST</code> , <code>PUT</code> , <code>DELETE</code>
<code>content_type</code>	string	No	<code>form</code>	Body format: <code>form</code> , <code>form-data</code> , <code>json</code> , <code>xml</code>
<code>timeout</code>	number	No	<code>PIPE_HTTP_TIMEOUT</code>	Request timeout in seconds
<code>success_codes</code>	number[]	No	<code>[200, 201, 202, 203, 204, 205, 206, 207, 208]</code>	HTTP codes that route the event to <code>outs.success</code>
<code>headers</code>	object	No	—	Request headers. Supports templating
<code>form</code>	object	No	—	Form data for <code>content_type: form</code> and <code>form-data</code> . Supports templating
<code>auth_type</code>	string	No	<code>basic</code>	Authentication type. The only available value is <code>basic</code> . Activated if <code>auth_username</code> is set. Sets the header <code>Authorization: Basic &lt;username&gt;:&lt;password&gt;</code>
<code>auth_username</code>	string	No	<code>""</code>	Login for basic authentication
<code>auth_password</code>	string	No	<code>""</code>	Password for basic authentication
<code>pool_size</code>	number	No	<code>PIPE_HTTP_MAX_CONN</code>	Connection pool per host within the channel. Increasing <code>pool_size</code> may not improve speed without increasing <code>PIPE_WORKER_SIZE</code> in the configuration file
<code>limit</code>	object	No	—	Rate limit for requests (see below)

### Rate Limit (limit)

Allows limiting the number of requests within a given time interval.

Field	Type	Required	Default	Description
<code>period_sec</code>	number	Yes	—	Interval in seconds
<code>count_requests</code>	number	Yes	—	Maximum number of requests per interval. The limit applies within the <code>key</code> scope in the channel
<code>key</code>	string	No	random UUID	Limit key. Supports templating. One key = one shared limit across multiple HTTP Request pipes

#### TIP

By default, the request rate depends on the **Pipeline** configuration and is approximately 200–300 requests per second.

## Possible Errors

- Template processing error in pipe fields
- `$in.body` not found (Pack pipe is missing before HTTP Request with `content_type: json` or `xml`)
- `$in.body` is not a byte array
- Invalid value of `params.content_type`
- Invalid value of `params.method`
- HTTP request error (network error, timeout, etc.)

## Examples

### Form Request

```

{
 "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
 }
}

```

### JSON Body Request

Before this pipe in the chain, there must be a Pack pipe that builds the JSON and writes it to `$in.body`.

```

{
 "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
 }
}

```

# Pack

**Pack** is a **Pipe** for packing data into JSON or XML. The result is written to `$in.body` as a byte array, ready to be passed to the [HTTP Request](#) pipe.

## Parameters

Field	Type	Required	Default	Description
type	string	No	JSON	Packing format: JSON or XML
template	object	Yes	—	Structure of the resulting JSON or XML. Values support <a href="#">templating</a>

### CAUTION

XML is described in JSON format using the notation of the [clbanning/mxj](#) library:

- Element attributes are written with a `-` prefix (e.g., `"-type": "MESSAGE"` )
- Text content is written as `"#text"`
- JSON arrays are converted into repeating XML elements

## Possible Errors

- Invalid value of `params.type`
- Error processing `params.template` — incorrect template or field not found

## Examples

### JSON

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 in `$.body` :

```
{
 "acceptUrl": "http://example.com/accept_url",
 "backgroundColor": "#ffffff",
 "backgroundImage": "http://example.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://example.com/merchantDomain",
 "merchantLogoPng": "http://example.com/merchantLogoPng",
 "merchantLogoSvg": "http://example.com/merchantLogoSvg",
 "merchantName": "merchantName",
 "offerEnd": "2021-08-09T18:31:42",
 "partnerCampaignId": 101,
 "blockIds": ["adfadf", "dadasdasdas", "aaaaaa"],
 "segments": ["aaaa", "bbbb"],
 "test_obj": {
 "field1": 1,
 "field2": ""
 }
}
```

## XML

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": "$subscription.id"
 }
]
 }
 }
 },
 "templates": {
 "template": [
 {
 "#text": "$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 in `$in.body` :

```

<?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

**Unpack** is a **Pipe** for unpacking JSON or XML from `$in.body` (byte array) into a structured object `$in.data`. It is typically used after the [HTTP Request](#) pipe to process the response from an external service.

## Parameters

Field	Type	Required	Default	Description
type	string	No	JSON	Unpacking format: JSON or XML

## Possible Errors

- `$in.body` not found
- `$in.body` is not a byte array
- Invalid value of `params.type`
- Demarshalling error (incorrect data format)

## Examples

### Unpacking XML

```
{
 "id": 10,
 "type": "unpack",
 "params": {
 "type": "XML"
 },
 "outs": {
 "success": 20,
 "error": 1000
 }
}
```

### Unpacking JSON (default)

```
{
 "id": 10,
 "type": "unpack",
 "outs": {
 "success": 15,
 "error": 1001
 }
}
```

# Eventer

**Eventer** is a **Pipe** for registering channel events in the platform. It is used to record delivery, non-delivery events, as well as to manage moderation and stopping of placement mailings.

## Parameters

Field	Type	Required	Default	Description
event_type	string	No	—	Event type. Required for <code>stop</code> , <code>moderate_pass</code> , <code>moderate_error</code> . Not used for other event types
channel_id	int	No	—	Numeric channel identifier. If not set, taken from the event context. Not required for <code>stop</code> , <code>moderate_pass</code> , <code>moderate_error</code>
action_type	int	No	—	Numeric event identifier within the channel (Event ID). Required for all event types except <code>stop</code> , <code>moderate_pass</code> , <code>moderate_error</code>
event	object	Yes	—	Event parameters (see below)

### Event Parameters (event)

Field	Type	Required	Description
send_message_id	string	Yes	Message identifier in the platform. Supports <a href="#">templating</a>
time	string	No	Event time in RFC3339 format (e.g., <code>2024-05-15T10:30:00Z</code> ). If specified, the event is assigned this date instead of the current time
message	string	No	Text message attached to the event. Supports <a href="#">templating</a> . Useful for recording error text in a bounce: <code>"message": "\$error.code \$error.message"</code>

#### CAUTION

In the [Schedule Pipeline](#), `$origin.send_message_id` may be missing because `$origin` is replaced with the `params.template` value of the [Scheduler](#) pipe. In this case, `send_message_id` must be passed explicitly through the Scheduler pipe's template and accessed as `$origin.<field_name>`.

## Possible Errors

- Template processing error in `params.event.send_message_id`
- Invalid format of `params.event.send_message_id`

## Examples

### Delivery Event Registration

```
{
 "id": 100,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100021,
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 },
 "outs": {
 "error": 101
 }
}
```

### Stop Placement Mailing

```
{
 "id": 150,
 "type": "event",
 "params": {
 "event_type": "stop",
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 }
}
```

### Moderation Error

```
{
 "id": 18,
 "type": "event",
 "params": {
 "event_type": "moderate_error",
 "event": {
 "send_message_id": "$origin.send_message_id"
 }
 },
 "outs": {
 "success": 19,
 "error": 20
 }
}
```

### Recording Error in Bounce

The `message` field allows saving the error text from a previous pipe (e.g., HTTP code and description from `http_request`) into the event:

```
{
 "id": 4,
 "type": "event",
 "params": {
 "channel_id": 10001,
 "action_type": 100003,
 "event": {
 "send_message_id": "$origin.send_message_id",
 "message": "$error.code $error.message"
 }
 }
}
```

# Scheduler

**Scheduler** is a **Pipe** for deferred event execution. It writes the event to the database with a specified execution time. When the scheduled time arrives, the [Schedule Pipeline](#) reads the event and starts processing it.

### CAUTION

The event passed to the Schedule **Pipeline** replaces `$origin` with the value of `params.template` of this pipe. The original `$origin` data will not be available — all required fields must be passed in advance through `template` .

### Parameters

Field	Type	Required	Default	Description
<code>pipe_id</code>	int	Yes	—	<code>id</code> of the pipe in the Schedule <b>Pipeline</b> where the event will land
<code>delay</code>	int	No	<code>PIPE_SCHEDULER_DELAY</code> (60 sec)	Execution delay in seconds
<code>template</code>	object	Yes	—	Data that will be written to <code>\$in</code> and <code>\$origin</code> of the event in Schedule. Supports <a href="#">templating</a>

### Possible Errors

- Template processing error in `params.template`

### Example

```
{
 "id": 801,
 "type": "scheduled",
 "params": {
 "pipe_id": 1,
 "delay": 10,
 "template": {
 "camp_id": "$origin.c_id",
 "send_message_id": "$origin.send_message_id",
 "ym_cmp_id": "$#in.data.ymCampaignId"
 }
 },
 "outs": {
 "success": 802,
 "error": 803
 }
}
```

After 10 seconds, the event will reach pipe `1` in the Schedule **Pipeline** with the data from `template` in `$origin` .

# Selector

**Selector** is a **Pipe** for conditional **Pipeline** branching. It checks conditions using JsonQL queries and routes the event to the corresponding pipe. If no condition is met, the event goes to `outs.success` (if it is defined).

## Parameters

Field	Type	Required	Description
outs	array	Yes	Array of conditions with target pipes

### Outs Array Element

Field	Type	Required	Description
query	string	Yes	JsonQL query returning <code>true</code> or <code>false</code> . Supports <a href="#">templating</a>
out	number	Yes	<code>id</code> of the pipe the event will go to when the condition is met

## Possible Errors

- Error processing `params.outs.query`
- JsonQL query execution error

## Examples

### Branching by Template Field

```
{
 "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
 }
}
```

### Branching by API Response Status

```
{
 "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

**SQL** is a **Pipe** for executing SQL queries against a database. The result is written to `$in.sql_query` as an object (first row) or an array of objects. It is commonly used in **Pipelines** of [External Objects](#) operations to read, create, update, and delete data in an external database.

## Parameters

Field	Type	Required	Default	Description
<code>connector_id</code>	number	Yes	—	SQL connector ID
<code>query</code>	string	Yes	—	SQL query. Supports <a href="#">templating</a>
<code>map</code>	bool	No	<code>false</code>	If <code>true</code> — return the first result as an object. If <code>false</code> — return an array of objects
<code>connection_timeout</code>	number	No	<code>PIPE_SQL_CONNECTION_TIMEOUT</code>	Connection timeout in seconds. Overrides <code>PIPE_SQL_CONNECTION_TIMEOUT</code> in the configuration file
<code>query_timeout</code>	number	No	<code>PIPE_SQL_QUERY_TIMEOUT</code>	Query execution timeout in seconds. Overrides <code>PIPE_SQL_QUERY_TIMEOUT</code> in the configuration file

### CAUTION

For the **Read** operation of external objects, you must use `map: true`. Without this parameter, the pipe will return an array `[]map[string]interface{}`, even if the database contains only one row. The [Result](#) pipe expects an object `map[string]interface{}`, and a type mismatch will cause an error.

## Possible Errors

- Error processing the `params.query` template
- Error executing the SQL query

## Examples

### Reading a Single Record (Read)

The Input Field `id` is used as a filter. The result is passed to Result via ``${in.sql_query}``.

```
[
 {
 "id": 1,
 "type": "log",
 "params": {
 "message": "[Read] Start",
 "level": "info"
 },
 "outs": {
 "success": 100
 }
 },
 {
 "id": 100,
 "type": "sql_pipe",
 "params": {
 "connector_id": 1,
 "connection_timeout": 15,
 "query_timeout": 15,
 "query": "SELECT * FROM my_table WHERE id = $in.id",
 "map": true
 },
 "outs": {
 "success": 200,
 "error": 150
 }
 },
 {
 "id": 200,
 "type": "result",
 "params": {
 "path": "$#in.sql_query"
 }
 },
 {
 "id": 150,
 "type": "error",
 "params": {
 "code": 500,
 "message": "SQL error: $error.code $error.message"
 }
 }
]
```

### Getting a List (List)

Without `map` — an array of all records is returned.



```
[
 {
 "id": 1,
 "type": "log",
 "params": {
 "message": "[List] Start",
 "level": "info"
 },
 "outs": {
 "success": 100
 }
 },
 {
 "id": 100,
 "type": "sql_pipe",
 "params": {
 "connector_id": 1,
 "connection_timeout": 15,
 "query_timeout": 15,
 "query": "SELECT id, name, description FROM my_table"
 },
 "outs": {
 "success": 200,
 "error": 150
 }
 },
 {
 "id": 200,
 "type": "result",
 "params": {
 "path": "$#in.sql_query"
 }
 },
 {
 "id": 150,
 "type": "error",
 "params": {
 "code": 500,
 "message": "SQL error: $error.code $error.message"
 }
 }
]
```

### Creating a Record (Create)

Values from Input Fields are substituted via `$in.<shortname>`.

```
[
 {
 "id": 1,
 "type": "log",
 "params": {
 "message": "[Create] Start",
 "level": "info"
 },
 "outs": {
 "success": 100
 }
 },
 {
 "id": 100,
 "type": "sql_pipe",
 "params": {
 "connector_id": 1,
 "connection_timeout": 15,
 "query_timeout": 15,
 "query": "INSERT INTO my_table (name, description) VALUES ('$in.name', '$in.description')"
 },
 "outs": {
 "success": 200,
 "error": 150
 }
 },
 {
 "id": 200,
 "type": "result",
 "params": {
 "path": "$#in.sql_query"
 }
 },
 {
 "id": 150,
 "type": "error",
 "params": {
 "code": 500,
 "message": "SQL error: $error.code $error.message"
 }
 }
]
```

#### Updating a Record (Update)

```
[
 {
 "id": 1,
 "type": "log",
 "params": {
 "message": "[Update] Start",
 "level": "info"
 },
 "outs": {
 "success": 100
 }
 },
 {
 "id": 100,
 "type": "sql_pipe",
 "params": {
 "connector_id": 1,
 "connection_timeout": 15,
 "query_timeout": 15,
 "query": "UPDATE my_table SET name = '$in.name', description = '$in.description' WHERE id = $in.id"
 },
 "outs": {
 "success": 200,
 "error": 150
 }
 },
 {
 "id": 200,
 "type": "result",
 "params": {
 "path": "$#in.sql_query"
 }
 },
 {
 "id": 150,
 "type": "error",
 "params": {
 "code": 500,
 "message": "SQL error: $error.code $error.message"
 }
 }
]
```

#### Deleting a Record (Delete)

```
[
 {
 "id": 1,
 "type": "log",
 "params": {
 "message": "[Delete] Start",
 "level": "info"
 },
 "outs": {
 "success": 100
 }
 },
 {
 "id": 100,
 "type": "sql_pipe",
 "params": {
 "connector_id": 1,
 "connection_timeout": 15,
 "query_timeout": 15,
 "query": "DELETE FROM my_table WHERE id = $in.id"
 },
 "outs": {
 "success": 200,
 "error": 150
 }
 },
 {
 "id": 200,
 "type": "result",
 "params": {
 "path": "$#in.sql_query"
 }
 },
 {
 "id": 150,
 "type": "error",
 "params": {
 "code": 500,
 "message": "SQL error: $error.code $error.message"
 }
 }
]
```

# Store Set

**Store Set** is a **Pipe** for saving data to storage (MongoDB) by a given key. It supports specifying a time-to-live (TTL) after which the data is automatically deleted.

## Parameters

Field	Type	Required	Default	Description
key	string	Yes	—	The key by which the value is saved. Supports <a href="#">templating</a>
value	any	Yes	—	Data to save. Supports templating
ttl	number	Yes	—	Record lifetime in seconds. The record is deleted after expiration
table	string	No	""	Virtual collection name (tag). Does not require additional database configuration

## Possible Errors

- Error processing the template in `params.key` , `params.value` , `params.table` , or `params.ttl`
- 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
 }
}
```

# Store Get

**Store Get** is a **Pipe** for reading data from storage (MongoDB) by a given key. The result is written to `$in.<name>`. It supports branching in case the record is not found.

## Parameters

Field	Type	Required	Default	Description
key	string	Yes	—	Key for lookup. Supports <a href="#">templating</a>
name	string	No	value of <code>key</code>	Field name in <code>\$in</code> where the result will be written ( <code>\$in.&lt;name&gt;</code> )
delete	bool	No	false	If <code>true</code> — the record is deleted immediately after reading
table	string	No	""	Virtual collection name (tag)
not_found	number	No	—	<b>Pipe</b> <code>id</code> to which the event will transition if the record is not found

## Possible Errors

- Error processing the template in `params.key` or `params.table`
- Database query error (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
 }
}
```

# Log

**Log** is a **Pipe** for writing messages to the `procpiper.log` log file. It is used for debugging and monitoring **Pipeline** execution.

## Parameters

Field	Type	Required	Default	Description
message	string	No	""	Message text. Supports <a href="#">templating</a>
prefix	string	No	<channel id> <pipe id>	Message prefix in the log
level	string	No	info	Log level: <code>info</code> , <code>debug</code> , <code>warn</code> , <code>error</code> , <code>trace</code>

## Possible Errors

- Error processing the `params.message` template — invalid template or field not found

## Examples

### Logging an HTTP Request Error

```
{
 "id": 10,
 "type": "log",
 "params": {
 "message": "error http request $error.code $error.message"
 },
 "outs": {
 "success": 20
 }
}
```

### Logging with Trace Level

```
{
 "id": 10,
 "type": "log",
 "params": {
 "message": "some message",
 "level": "trace"
 },
 "outs": {
 "success": 15
 }
}
```

# Result

### CAUTION

Available only in **Pipelines** of [External Objects \(entities\)](#) operations.

**Result** is a **Pipe** for finalizing the result of a synchronous External Object operation **Pipeline**. The result is returned to the platform interface and displayed to the user.

### Features

- An empty result is not an error.
- Result does not have to be the last **Pipe** in the chain — after finalizing the result, you can execute additional logic (for example, close a session).
- A **Pipeline** can contain multiple Result pipes, but none of them should be a child of another Result (the result is finalized only once per branch).
- The result must be an object or an array of objects.

### Parameters

Field	Type	Required	Description
path	string	No	Path to the field from which the result is taken. Supports <a href="#">templating</a> . If not specified — the result is taken from <code>\$.data</code>

### CAUTION

To return an array or an object, use the `$.#` prefix. The `$.` prefix will return the value as a string, which will cause an error.

### Possible Errors

- `params.path` is specified, but the format is incorrect or the field is not found
- `params.path` is not specified, and the `data` key is missing in `$.in`

### Examples

#### Correct Example

`$.in.data.categories` contains an array of objects:

```
{
 "id": 10,
 "type": "result",
 "params": {
 "path": "$$.in.data.categories"
 }
}
```

#### Result from an SQL Query

When used with the [SQL](#) pipe, the query result is available in `$.in.sql_query`. Use the `$.#` prefix to preserve the original type (object or array):

```
{
 "id": 300,
 "type": "result",
 "params": {
 "path": "$$.in.sql_query"
 }
}
```

This pattern is used in all External Objects operations with SQL: Read, List, Create, Update, Delete.



# Error

### CAUTION

Available only in **Pipelines** of [External Objects \(entities\)](#) operations.

**Error** is a **Pipe** for terminating the execution of a synchronous External Object operation **Pipeline** with an error. It allows you to specify an error code and message that will be displayed in the interface.

### Features

- The pipe has no outputs — the event ends immediately after its execution.
- `params.message` must be text.

### Parameters

Field	Type	Required	Description
<code>code</code>	number	No	Error code
<code>message</code>	string	No	Error message. Supports <a href="#">templating</a> . Cannot be empty if specified

### Possible Errors

- Error processing `params.message` — template is invalid or field is not found

### Examples

#### Error with Code

```
{
 "id": 10,
 "type": "error",
 "params": {
 "code": 500
 }
}
```

#### Error with Code and Text from API Response

```
{
 "id": 10,
 "type": "error",
 "params": {
 "code": 500,
 "message": "${in.data.error}"
 }
}
```

# RMQ Publisher

**RMQ Publisher** — a pipe for publishing messages to RabbitMQ. Used to send data to external services via a RabbitMQ broker. Typical scenario: sending push notifications through an external delivery service.

## Input data

The pipe expects `$in.body` — the message body to publish:

- If `$in.body` is a byte array ( `[]byte` ), it is used directly
- If `$in.body` is a string, it is decoded from base64 to `[]byte`

Typically, an [Pack](#) pipe is placed before RMQ Publisher, which forms JSON or XML and writes the result to `$in.body`.

## Parameters

Field	Type	Required	Default	Description
connector_shortcode	string	Yes	—	String name of the RMQ connector created in <a href="#">External Data Configuration</a>
content_type	string	No	json	Content type: <code>json</code> or <code>xml</code>
delivery_mode	number	No	—	Delivery mode: <code>1</code> (transient — message is not persisted to disk) or <code>2</code> (persistent — message is saved to the broker disk)
headers	object	No	—	Additional message headers ( <code>map[string]string</code> ). Supports <a href="#">templating</a>
target	object	Yes	—	Publication target object (see below)

### Target object (target)

The pipe supports two target options:

#### Exchange:

Field	Type	Required	Description
type	string	Yes	Value: <code>exchange</code>
exchange_name	string	Yes	Exchange name in RabbitMQ
exchange_type	string	No (default: <code>direct</code> )	Exchange type: <code>direct</code> , <code>topic</code> , <code>fanout</code> , <code>headers</code>
routing_key	string	No	Routing key

#### Queue:

Field	Type	Required	Description
type	string	Yes	Value: <code>queue</code>
queue_name	string	Yes	Queue name in RabbitMQ

## Possible errors

- `$in.body` not found (missing [Pack](#) pipe before RMQ Publisher)
- `$in.body` is not a byte array or string (base64)
- Invalid `content_type` value (only `json` or `xml` )
- Connector not found by `connector_shortcode`
- Connector is not an RMQ connector
- RabbitMQ publication error (network error, broker unavailable, etc.)

## Examples

### Publish to exchange

A [Pack](#) pipe should precede this pipe in the chain to form the message body.

```
{
 "id": 2,
 "type": "rmq_publish",
 "params": {
 "connector_shortcode": "ens_push",
 "content_type": "json",
 "delivery_mode": 2,
 "target": {
 "type": "exchange",
 "exchange_name": "fintechiq.push-manager.send",
 "exchange_type": "topic",
 "routing_key": "fintechiq.push-manager.push.by-text.send"
 }
 },
 "outs": {
 "success": 4,
 "error": 100
 }
}
```

### Publish to queue

```
{
 "id": 2,
 "type": "rmq_publish",
 "params": {
 "connector_shortcode": "my_rmq",
 "content_type": "json",
 "delivery_mode": 2,
 "headers": {
 "message_id": "$origin.send_message_id"
 },
 "target": {
 "type": "queue",
 "queue_name": "notifications.inbox"
 }
 },
 "outs": {
 "success": 3,
 "error": 4
 }
}
```

## Migrating from HTTP Request to RMQ Publisher

If the external service accepts messages via RabbitMQ instead of an HTTP API, replace the **HTTP Request** pipe with **RMQ Publisher**. The rest of the chain (Pack, Selector, Log) remains unchanged.

### What changes

Pipe `http_request` (id:3) is replaced with `rmq_publish` using the same id:

```
{
 "id": 3,
 "type": "rmq_publish",
 "params": {
 "connector_shortcode": "cas_rmq",
 "content_type": "json",
 "delivery_mode": 2,
 "target": {
 "type": "exchange",
 "exchange_name": "cas.notifications",
 "exchange_type": "topic",
 "routing_key": "cas.message.send"
 }
 },
 "outs": {
 "success": 10,
 "error": 12
 }
}
```

Parameters `url`, `method`, `timeout`, `success_codes` are removed. Instead — `connector_shortcode` and `target`. The Pack pipe before RMQ Publisher works the same way — it forms `$in.body`, which is published to RabbitMQ.

Creating an RMQ connector

Before using the pipe, create a connector in **External Data Configuration**:

Field	Description
Name	Arbitrary name
Short Name	Unique identifier — used in the pipe's <code>connector_shortcode</code>
Host	RabbitMQ broker address
Port	Port (default <code>5672</code> )
Username	Login
Password	Password
Vhost	Virtual host
SSL	SSL and PEM certificates (if needed)

Routing messages with Selector

To direct messages to different queues or exchanges based on event data, use a **Selector** before multiple RMQ Publisher pipes.

Scheme:

```
Pack → Selector → RMQ Publisher (priority)
 ↘ RMQ Publisher (normal)
```

Both pipes can use the same connector — the connection is cached and shared between pipes.

Example: priority and normal notifications

The `$template.priority` field determines the routing key:

```
[
 {
 "id": 1,
 "type": "pack",
 "params": {
 "type": "JSON",
 "template": {
 "messageId": "$in.send_message_id",
 "priority": "$template.priority",
 "body": "$template.text",
 "recipient": "$subscription.client_id"
 }
 },
 "outs": {
 "success": 2,
 "error": 99
 }
 },
 {
 "id": 2,
 "type": "selector",
 "params": {
 "outs": [
 {
 "query": "('$template.priority' = 'high')",
 "out": 10
 },
 {
 "query": "('$template.priority' = 'low')",
 "out": 20
 }
]
 },
 "outs": {
 "success": 20,
 "error": 99
 }
 },
 {
 "id": 10,
 "type": "rmq_publish",
 "params": {
 "connector_shortcode": "cas_rmq",
 "content_type": "json",
 "delivery_mode": 2,
 "target": {
 "type": "exchange",
 "exchange_type": "topic",
 "exchange_name": "cas.notifications",
 "routing_key": "cas.priority.send"
 }
 },
 "outs": {
 "success": 30,
 "error": 99
 }
 },
 {
 "id": 20,
 "type": "rmq_publish",
 "params": {
 "connector_shortcode": "cas_rmq",
 "content_type": "json",
 "delivery_mode": 2,
 "target": {
 "type": "exchange",
 "exchange_type": "topic",
 "exchange_name": "cas.notifications",
 "routing_key": "cas.normal.send"
 }
 },
 "outs": {
 "success": 30,

```

```

 "error": 99
 }
},
{
 "id": 30,
 "type": "log",
 "params": {
 "message": "published successfully"
 }
},
{
 "id": 99,
 "type": "log",
 "params": {
 "message": "error: $error.code $error.message",
 "level": "error"
 }
}
}
]

```

Here:

1. **Pack** (id:1) forms the JSON message body
2. **Selector** (id:2) checks `$template.priority` :
  - `high` → pipe 10 (routing key `cas.priority.send`)
  - `low` → pipe 20 (routing key `cas.normal.send`)
  - no match → `outs.success` → pipe 20 (fallback)
3. Each **RMQ Publisher** publishes to its routing key
4. Both converge on pipe 30 (success) or 99 (error)

Selector conditions are checked in order — the first match wins. If no condition matches, the event goes to `outs.success`, which is convenient for a fallback route. All event data is available in `query`: `$template.*`, `$subscription.*`, `$campaign.*`, `$in.*`, `$origin.*`.

### Different brokers for different routes

If queues are on different RabbitMQ brokers, create two connectors and use different `connector_shortname` values in each RMQ Publisher pipe.

# Personnel requirements

Specialists who maintain Altcraft, administer and support the infrastructure must have the following knowledge and skills:

## Mandatory requirements

- Technical education;
- Knowledge of TCP/IP protocols, ability to configure firewalls and diagnose network issues;
- At least one year of Linux administration experience (Ubuntu/CentOS/Debian);
- Command-line skills: process management, file permissions, system limits ( `ulimit` , `systemd` );
- Knowledge of Altcraft platform functionality (see [Architecture Overview](#) and [Component Description](#));

## Databases and storage

The platform administrator works with the following databases and storage systems:

### MongoDB

- Installation, configuration, running as a systemd service;
- User and access rights management;
- Index creation and management;
- Replica set configuration;
- Backup and recovery ( `mongodump` / `mongorestore` );
- Performance diagnostics (slow query log, profiler);
- MongoDB is used for Control DB (system settings), FileDB (files and media), and each account's databases (subscribers, campaigns, templates, segments, etc.);

### ClickHouse

- Installation and basic configuration;
- User and database management;
- Understanding table structure and partitions;
- Backup and recovery;
- ClickHouse is used for analytics, event history, segmentation, and caching external SQL queries;

### Kvrocks (Redis-compatible storage)

- Installation, running as a systemd service;
- Basic `redis-cli` commands (keys, memory, connections);
- Backup (RDB/AOF);
- Kvrocks is used as a system cache, for campaign deduplication, workflow cache, and loyalty program cache;

### SSDB

- Installation from source code, running as a systemd service;
- Configuration management (tab indentation in config files);
- Managing multiple instances (hardbounces, notifications, MTA statistics);
- SSDB is used for email stop lists, notifications, and hourly statistics;

### RabbitMQ

- Installation, running as a systemd service;
- User, vhost, and permissions management ( `rabbitmqctl` );
- Queue monitoring via web interface (port 15672);
- Disk free limit configuration;
- RabbitMQ is the main message broker for communication between all platform services;

### PostgreSQL (optional)

- Installation and basic administration;
- PostgreSQL is used for the Market module (products, orders) and as DataDB for accounts;

## Web servers and proxies

- **Nginx** — understanding configuration structure, SSL termination, reverse proxy;
- The platform automatically generates Nginx configurations via `akd` , but the administrator must be able to diagnose proxy issues;

## Object storage (optional)

- **S3-compatible storage** (MinIO, AWS S3, Yandex Cloud Storage, or others) — basic understanding of the S3 protocol (buckets, access, endpoint);
- Used for storing platform files (attachments, media, import/export);

## Platform tools

- `akd` — platform service management: `start` , `stop` , `restart` , `status` , `list` ;

- **aktool** — administrative utility: database migrations, account management, segments, licenses, MTA nodes;
- Understanding the **main.json** configuration file structure and ability to make changes;

## Monitoring and diagnostics

- **Prometheus** — collecting metrics from platform processes (each service exports metrics on a separate port), Push Gateway configuration;
- Monitoring key indicators:
  - Disk space (MTA spool, logs, segments, databases);
  - RabbitMQ queues (length, consumption rate);
  - Kvrocks (memory, connections);
  - MongoDB (pool connections, query time);
  - ClickHouse (insert speed, table size);
  - SMTP (delivery, bounce rates, complaints);

## Authentication (optional)

- **LDAP** — configuring connection to an LDAP server for admin panel authentication;
- **Keycloak** — configuring Keycloak integration for alternative authentication;

## Containerization (optional)

- **Docker / Docker Compose** — running the platform in a container, volume management, image updates;
- The platform is delivered as a monolithic Docker image; databases and brokers are deployed separately;

## System utilities

- **cutycapt + xvfb** — HTML email preview generation (headless rendering);
- Understanding system limits ( **nofile** , **memlock** ) and their impact on performance;

### NOTE

Elimination of software errors in the platform operation is carried out only by specialists of Altcraft LLC, after the collected diagnostic information is provided to them.



# 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 purchased.

## Regular platform updates

For the entire period of the right to use the platform, the customer is granted the right to receive regular free updates.

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, after the collected diagnostic information is provided to them.

All diagnostic information can be shared via a personal support chat or by email to [team@altcraft.com](mailto:team@altcraft.com).

## Platform modification

Creation, modification, modernization of platform components / services is carried out only by specialists of Altcraft LLC.

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 solve his business problems by agreeing on the specifications with Altcraft LLC.

To agree on specifications, please contact [customers.department@altkraft.com](mailto: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 Platform 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 Platform instance. If it is found, a backup copy of platform directories will be created and all platform processes will be stopped. After the update the processes will be restarted. Creating a backup copy may take some time,

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 Platform base directory.

```
tar -xzf AKD.tar.gz
```

No configuration changes are required.

# RabbitMQ migration

If your RabbitMQ broker starts to consume too much hardware resources (CPU or memory) it is a good practice to move it to a separate server.

This guide will show you how to transfer RabbitMQ messages to another instance with the help of [Shovel Plugin](#).

## 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 Controlling server

Update RabbitMQ location and credentials in `main.json` configuration file and restart the platform with `./akd restart`.

```
"RABBITMQ_HOST": "192.168.*.*",
"RABBITMQ_PASS": "abcdefghijklmnopqrstuvwxyzABCDEFGH",
"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 queues that have messages you want to transfer:

```
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_results"` queues.

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. After the transfer is complete you can disable your old 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 command, you must make sure that there is enough free space in the current directory to save the archive.

## 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 environment necessary for the module to operate

## 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 method using snapshots allows you to save a lot of time when restoring "heavy" collections with a large number of indexes.

### 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

Recover 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. Importing and exporting keys in large SSDB databases can take up to several days. Databases structure example.

```
/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 the instructions: [Manual installation](#)

**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, notifications and stats).

## NOTE

All of the backup methods create an archive in a user's current directory. Before executing the command, you must make sure that there is enough free space in the current directory to save the archive.

## Control module

### Backup

The recommended way to create a backup is to place the directory with the control module in an archive, excluding logs, sockets and PID files:

```
tar -czvf akd-backup.tar.gz /opt/akd --exclude=*.log --exclude=*.sock --exclude=*.pid
```

### Recovery

Unpack the archive and move its contents:

```
tar -xzf 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 environment necessary for the module to operate (see the page [Preparation, database configuration, launching the platform])(/admin-guide/platform-installation/9406483).

## 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://docs.mongodb.com/manual/tutorial/backup-with-filesystem-snapshots/#back-up-and-restore-with-filesystem-snapshots>). The method using snapshots allows you to save a lot of time when restoring large collections with a large number of indexes.

### Backup

For small instances, you can use the `utilitymongodump` <https://docs.mongodb.com/database-tools/mongodump>:

```
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. Importing and exporting keys in large SSDB databases can take up to several days.

## Backup

For small instances you can use utility `ssdb-dump` (cm. <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:procrresume
AK:procsenderev
AK:procsmsev
AK:proctrigger
AK:procworkflow
AK:prochhook
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 --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trkimage --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trkread --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trksms --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trkwebctrl --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trkpush --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trkcustom --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trklisten --rpcAddr <old_host_ip>:8972
/opt/akd/ak/bin/trk_amazon_sns --rpcAddr <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 tables with a replicated engine to properly move data from regular tables later on. You can create and delete tables for storing statistics, or populate data (based on triggers in MV) using the aktool 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
```

## Parameters of commands

List of parameters (tables) for `delete`, `fill`, `init` commands:

Parameter	Table	Description
all	All tables for storing statistics	
summary	summary_report summary_MV_%версия	Summary report
pixel	pixel_report pixelUndefMV_%версия pixelMV_%версия	Pixels
engagement	engagement_report engagement_MV	Audience growth report
click_map	click_map_report click_mapMV	Click map
forms	total_form_report formMV  form_pages_view_report form_pages_viewMV  group_form_report group_formMV	Forms
undelive	undelive_report undeliveMV	Undelivered report
campaigns	camp_stats	Real-time mailing statistics
notify	notify_sender_stats	Daily and weekly user reports

## 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 waiting time can be configured using the parameter `database_atomic_delay_before_drop_table_sec` in the ClickHouse server configuration.

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 through the terminal. With `aktool` you can perform various operations such as migration, data cleansing, account management, user management, tariffs and other system entities.

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 you to losslessly update your data to the new platform version. In the description of a release that requires such a migration, we describe in detail which commands can be used to perform it.

## List of aktool commands

For more information about a specific command, type the following:

```
./aktool [command] --help
```

Command	Description
accounts	Managing accounts
admins	Managing administrators
akmta	AKMTA management
cleanup	Cleaning up data for an instance
completion	Generate an autocomplete script for the specified shell
configs	Configuration management
field_migrator	Migrating one field to another
help	Get help on any command
history_storage	Actions with history storage tables
indexes	Creating indexes in the platform database
keys	Managing keys
migrate	Perform data migration
nodes	Managing nodes
printconf	Display current values of platform configs
profile_history	Manage profile history tables
rabbitmq	Manage RabbitMQ
resource_system_fields	Adding system fields to a resource
roles	Actions with roles in accounts
smid	[SMID] decoder( <a href="#">en/admin-guide/extra/56257642</a> )
stats	<a href="#">Manage statistics tables</a>
tariffs	Manage tariffs
users	Manage account users
workflow	Scenarios management

### 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=mb1_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 Mounting 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_RUN_AS_GROUP` parameters (by default, **altcraft**), provided that such user and group exist in the system. These settings can be overridden by adding the corresponding parameters in the main platform 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 following command:

```
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 services may use the `/tmp` directory to store some temporary files.

To create screenshots of templates (for example, to display a preview of a template in the interface), the `cutycapt` utility is used, which works in conjunction with `xvfb-run` to emulate the graphical environment. These tools are run by the platform via the command line, they are executed **at the OS level**.



# Platform service monitoring

Depending on the monitoring system used in the organization, a specialist must choose a method for monitoring the status of platform processes.

The page describes all the processes that need to be kept in working order for the platform to fully operate.

We recommend checking for the existence of a process by name using the tools built into the monitoring system or using the [pidof](#) utility.

The following information must be collected for each host (or virtual machine):

- RAM usage and available memory;
- CPU usage and available CPU time;
- Available disk space and disk utilization information (speed, latency, SMART);
- Number of open files and connections.

## Databases and services

	Default IP	Default port	Importance	Alert
MongoDB	127.0.0.1	27017	Critical	Unable to establish TCP connection
ClickHouse	127.0.0.1	9000	Critical	Unable to establish TCP connection
SSDB (hb)	127.0.0.1	4420	Critical	Unable to establish TCP connection
SSDB (notify)	127.0.0.1	4430	Critical	Unable to establish TCP connection
RabbitMQ	0.0.0.0	5672	Critical	Unable to establish TCP connection

## RabbitMQ queues

Virtual host	Queue name	Producer	Consumer	Description	Importance	Alert
altcraft	oxy_triggers	*	AK:proctrigger	Triggers (mailings)	High	Minimum number of messages in queue >= 10000 over 10 minutes
altcraft	oxy_triggers_prior	*	AK:proctrigger	Priority triggers (mailings)	High	Minimum number of messages in queue >= 5000 over 10 minutes
altcraft	trk_*	AK:trk* AK:cookie_saver	procactions procpixel	Events processed by trackings	High	Minimum number of messages in queue >= 5000 over 10 minutes
altcraft	akmta_*	AK:proctrigger AK:webcontrol campaign AK:procworkflow	AK:akmtad	Messages for sending	High	Number of messages does not decrease over 50 minutes
altcraft	geo_akmta_*	AK:proctrigger AK:webcontrol campaign AK:procworkflow	AK:akmtad	Messages for sending by time zones	High	Number of messages does not decrease over 50 minutes
altcraft	prior_akmta_*	AK:proctrigger AK:webcontrol campaign AK:procworkflow	AK:akmtad	Priority messages for sending	High	Number of messages does not decrease over 50 minutes

## AKD processes

To verify platform operability, it is sufficient to ensure the presence of the process in the virtual filesystem `/proc`.

The full list of processes can be viewed with the command – `<BASEDIR>/akd --processes`

The default PID file location is – `<BASEDIR>/pids/<Executable file name>.pid`

Executable file name	Process name	Description	Importance	Alert
webadmin	AK:webadmin	Admin panel	Medium	Process not found in the virtual filesystem <code>/proc</code>
akmtad	AK:akmtad	Mail transfer agent, AKMTA	High	Process not found in the virtual filesystem <code>/proc</code>
api	AK:api	API	High	Process not found in the virtual filesystem <code>/proc</code>
cookie_saver	AK:cookie_saver	User cookie processing	High	Process not found in the virtual filesystem <code>/proc</code>
proctask	AK:proctask	Task execution, mailing launch	High	Process not found in the virtual filesystem <code>/proc</code>
procactions	AK:procactions	Event processing for statistics (clicks, opens, subscriptions, etc.)	High	Process not found in the virtual filesystem <code>/proc</code>
procevent	AK:procevent	Event processing and writing to ClickHouse	High	Process not found in the virtual filesystem <code>/proc</code>
prochook	AK:prochook	Capturing various events from the platform	High	Process not found in the virtual filesystem <code>/proc</code>
procpixel	AK:procpixel	Pixel events processing	High	Process not found in the virtual filesystem <code>/proc</code>
proctintegras	AK:proctintegras	Integration with external systems (AppMetrica, etc.)	High	Process not found in the virtual filesystem <code>/proc</code>
procleadsaver	AK:procleadsaver	Import statistics processing	Medium	Process not found in the virtual filesystem <code>/proc</code>
procnotify	AK:procnotify	Notification processing  <i>The process was removed in versions v2024.2.68.2.2206 and later; its functionality was migrated to <code>proctask</code></i>	Medium	Process not found in the virtual filesystem <code>/proc</code>

Executable file name	Process name	Description	Importance	Alert
procpush	AK:procpush	Push events processing	High	Process not found in the virtual filesystem /proc
procrresume	AK:procrresume	Processing and database scanning to restore profile status	Medium	Process not found in the virtual filesystem /proc
procrpc	AK:procrpc	RPC client for processing RPC connections with processes	High	Process not found in the virtual filesystem /proc
procsenderev	AK:procsenderev	Event processing	High	Process not found in the virtual filesystem /proc
procsmsev	AK:procsmsev	Requests information on SMS sendings	High	Process not found in the virtual filesystem /proc
procsmslisten	AK:procsmslisten	Processing responses from SMS gateways	High	Process not found in the virtual filesystem /proc
proctrigger	AK:proctrigger	Trigger mailing processing	High	Process not found in the virtual filesystem /proc
procwebver	AK:procwebver	Web versions processing	Medium	Process not found in the virtual filesystem /proc
procworkflow	AK:procworkflow	Scenarios processing	High	Process not found in the virtual filesystem /proc
tariffcontroller	AK:tariffcontroller	Sending limit control (tariffs)  The process was removed in versions v2024.2.68.2.2206 and later; its functionality was migrated to proctask	Medium	Process not found in the virtual filesystem /proc
trkaction	AK:trkaction	Tracking event registration	High	Process not found in the virtual filesystem /proc
trk_amazon_sns	AK:trk_amazon_sns	Amazon sender event registration	High	Process not found in the virtual filesystem /proc
trkmandrill	AK:trkmandrill	Mandrill sender event registration	High	Process not found in the virtual filesystem /proc
trkimage	AK:trkimage	Pixel events registration	High	Process not found in the virtual filesystem /proc
trkpush	AK:trkpush	Push tracking events registration	High	Process not found in the virtual filesystem /proc
trkread	AK:trkread	Email message read events registration	High	Process not found in the virtual filesystem /proc
trksms	AK:trksms	SMS message read events registration	High	Process not found in the virtual filesystem /proc
trkwebversion	AK:trkwebversion	Web version read events registration	High	Process not found in the virtual filesystem /proc
webcontrol	AK:webcontrol	User web interface	High	Process not found in the virtual filesystem /proc

CPU time

[https://en.wikipedia.org/wiki/CPU\\_time](https://en.wikipedia.org/wiki/CPU_time)

Metric type	Description	Importance	Alert
CPU system time	CPU usage by process in percent ( system )	Info	-
CPU iowait time	CPU usage by process in percent ( iowait )	High	> 15% * number of vCPU cores
CPU user time	CPU usage by process in percent ( user )	Info	-
CPU utilization	CPU usage by process in percent ( total )	High	> 50% * number of vCPU cores

Memory

Metric type	Description	Importance	Alert
RSS (resident set size)	<a href="https://en.wikipedia.org/wiki/Resident_set_size">https://en.wikipedia.org/wiki/Resident_set_size</a>	High	> 20% of total memory
SWAP	<a href="https://en.wikipedia.org/wiki/Paging#Unix_and_Unix-like_systems">https://en.wikipedia.org/wiki/Paging#Unix_and_Unix-like_systems</a>	High	> 5% of total SWAP

Network

To identify network issues, it is recommended to monitor the number of connections for each state.

[https://en.wikipedia.org/wiki/Transmission\\_Control\\_Protocol#Protocol\\_operation](https://en.wikipedia.org/wiki/Transmission_Control_Protocol#Protocol_operation)

Metric type	Description	Importance	Alert
CLOSE	Closed. Socket is not in use.	Info	-
CLOSE_WAIT	Remote side has disconnected; waiting for socket closure.	Medium	Number of connections > 5000
CLOSING	Socket closed, then remote side disconnected; waiting for acknowledgment.	Info	-
ESTABLISHED	Connection established.	Medium	Number of connections > 25000
FIN_WAIT1	Socket closed; connection disconnecting.	Info	-
FIN_WAIT2	Socket closed; waiting for remote side disconnect.	Info	-
LAST_ACK	Remote side disconnected, then socket closed; waiting for acknowledgment.	Info	-
LISTEN	Waiting for incoming connections.	Info	-
SYN_RECV	Initial connection synchronization in progress.	Info	-
SYN_SENT	Actively attempting to establish connection.	High	Number of connections > 5000
TIME_WAIT	Socket closed, but waiting for packets still in the network for processing	High	Number of connections > 5000

# Process and mailing monitoring via Prometheus

Process and mailing monitoring via Prometheus is used to control performance, processing stability, and message queue status.

Collecting metrics allows you to:

- track mailing processing speed and individual lead processing stages;
- identify delays in scenario execution and data processing;
- monitor message publishing status in RabbitMQ;
- detect queue growth, re-sends, and lost events;
- analyze load on `campaign`, `procworkflow`, `proctrigger`, and `procpiper` processes;
- build dashboards and configure alerts in Prometheus and Grafana;
- find bottlenecks during performance degradation or mailing processing errors.

## Which processes support metrics

Currently, metrics are supported by the `campaign`, `procworkflow`, `proctrigger`, and `procpiper` processes. Mailing metrics can be delivered in two ways:

- via Pushgateway when `campaign` runs separately;
- via the pull model inside `procworkflow` and `proctrigger`, if campaigns are executed within these processes.

Monitoring type	Processes	Collection method
Pull	<code>procworkflow</code> , <code>proctrigger</code> , <code>procpiper</code>	Prometheus scrapes <code>/metrics</code>
Push	<code>campaign</code>	Metrics are sent to Pushgateway

## Configuring pull metrics

The pull model is used to collect metrics from the `procworkflow`, `proctrigger`, and `procpiper` processes. In this mode, Prometheus periodically queries the HTTP endpoint `/metrics` exposed by the corresponding process.

Metrics from `procworkflow` and `proctrigger` also include mailing metrics if mailings are executed within these processes.

### Pull metrics configuration example

Example platform configuration for enabling pull metrics for `procworkflow`, `proctrigger`, and `procpiper` simultaneously:

```
{
 "PROMETHEUS_METRICS": {
 "ENABLE": true,
 "PROCESSES": [
 "procworkflow",
 "proctrigger",
 "procpiper"
]
 },
 "WF_METRIC_HOST": "0.0.0.0",
 "WF_METRIC_PORT": 8911,
 "PROC_TRIGGER_METRIC_HOST": "0.0.0.0",
 "PROC_TRIGGER_METRIC_PORT": 8912,
 "PROC_PIPER_METRIC_HOST": "0.0.0.0",
 "PROC_PIPER_METRIC_PORT": 8917
}
```

If the `PROCESSES` array is empty ( `[]` ), metrics are automatically enabled for all supported processes.

Add metric scrape jobs to the Prometheus configuration:

```
scrape_configs:
- job_name: 'procworkflow'
 metrics_path: /metrics
 static_configs:
 - targets:
 - '<procworkflow-server>:8911'

- job_name: 'proctrigger'
 metrics_path: /metrics
 static_configs:
 - targets:
 - '<proctrigger-server>:8912'

- job_name: 'procpiper'
 metrics_path: /metrics
 static_configs:
 - targets:
 - '<procpiper-server>:8917'
```

After starting the processes, verify that the metric services are listening on the specified ports:

```
netstat -tln | grep 8911
netstat -tln | grep 8912
netstat -tln | grep 8917
```

Example output:

```
tcp6 0 0 ::::8911 :::* LISTEN
tcp6 0 0 ::::8912 :::* LISTEN
tcp6 0 0 ::::8917 :::* LISTEN
```

Check the availability of the `/metrics` endpoint:

```
curl http://<procworkflow-server>:8911/metrics
curl http://<proctrigger-server>:8912/metrics
curl http://<procpiper-server>:8917/metrics
```

Replace `<procworkflow-server>`, `<proctrigger-server>`, and `<procpiper-server>` with the actual IP addresses or hostnames of the servers running the respective processes.

If the services are configured correctly, the endpoint will return a list of Prometheus metrics for the `procworkflow`, `proctrigger`, and `procpiper` processes.

Pull metric parameters

Parameter	Description
PROMETHEUS_METRICS.ENABLE	Global enable of Prometheus metrics
PROMETHEUS_METRICS.PROCESSES	List of processes for which metric collection is enabled
WF_METRIC_HOST	Address on which <code>procworkflow</code> publishes metrics
WF_METRIC_PORT	Port of the <code>procworkflow</code> metrics service
PROC_TRIGGER_METRIC_HOST	Address on which <code>proctrigger</code> publishes metrics
PROC_TRIGGER_METRIC_PORT	Port of the <code>proctrigger</code> metrics service
PROC_PIPER_METRIC_HOST	Address on which <code>procpiper</code> publishes metrics
PROC_PIPER_METRIC_PORT	Port of the <code>procpiper</code> metrics service

When configuring pull metrics, consider the Prometheus location relative to the platform server.

Value	When to use
127.0.0.1	Prometheus is installed on the same server as the platform
0.0.0.0	Prometheus is installed on a separate server

The `WF_METRIC_HOST` and `PROC_TRIGGER_METRIC_HOST` parameters define the internal address on which the processes will accept requests to the `/metrics` endpoint.

The `WF_METRIC_PORT`, `PROC_TRIGGER_METRIC_PORT`, and `PROC_PIPER_METRIC_PORT` parameters set the metric service ports. You can use ports in the range `1024` to `9999`, excluding ports occupied by other services.

CAUTION

If Prometheus is located on a separate server, specify `0.0.0.0` in the `WF_METRIC_HOST` and `PROC_TRIGGER_METRIC_HOST` parameters.

Do not use the same port for `WF_METRIC_PORT`, `PROC_TRIGGER_METRIC_PORT`, and `PROC_PIPER_METRIC_PORT`. Each process must serve metrics on a separate port.

If the metrics service does not start after changing `WF_METRIC_HOST`, `WF_METRIC_PORT`, `PROC_TRIGGER_METRIC_HOST`, `PROC_TRIGGER_METRIC_PORT`, `PROC_PIPER_METRIC_HOST`, or `PROC_PIPER_METRIC_PORT`, verify that the specified port is free and available on the server.

## Configuring push metrics for `campaign`

The push model is used to send metrics from the `campaign` process to the Prometheus Pushgateway.

In this mode, the `campaign` process independently sends metrics to the Pushgateway, after which Prometheus scrapes them from the gateway server.

### INFO

The push model is supported only for the `campaign` process. Metrics will not appear in Pushgateway until the mailing has been started at least once.

Before configuration, you must deploy and start the Prometheus Pushgateway.

The platform does not start Pushgateway automatically. In the `ADDRESS` parameter, you must specify the address of an already running Pushgateway.

### Pushgateway launch example via `systemd`

Example unit file:

```
[Unit]
Description=Prometheus Pushgateway
Wants=network-online.target
After=network-online.target

[Service]
User=pushgateway
Group=pushgateway
Type=simple
ExecStart=/usr/local/bin/pushgateway

[Install]
WantedBy=multi-user.target
```

After starting Pushgateway, configure metric delivery in the platform configuration:

```
{
 "PROMETHEUS_METRICS_PUSH_GATEWAY": {
 "ENABLE": true,
 "ADDRESS": "<pushgateway-server>:9091",
 "PERIOD_SEC": 5
 }
}
```

Parameter description:

Parameter	Description
<code>ENABLE</code>	Enables sending metrics to Pushgateway
<code>ADDRESS</code>	Address and port of the already running Pushgateway
<code>PERIOD_SEC</code>	Metric send interval in seconds

Add Pushgateway to the Prometheus configuration:

```
scrape_configs:
- job_name: 'pushgateway'
 static_configs:
 - targets:
 - '<pushgateway-server>:9091'
```

Check Pushgateway metrics availability:

```
curl http://<pushgateway-server>:9091/metrics
```

Replace `<pushgateway-server>` with the actual IP address or hostname of your Pushgateway server.

`campaign` metrics start appearing in the Pushgateway after the mailing is launched.

Grouping metrics by campaign ID

The `CAMPAIGN_ID_PROMETHEUS_GROUPING_ENABLE` parameter controls grouping of push metrics by mailing ID.

Example configuration:

```
{
 "CAMPAIGN_ID_PROMETHEUS_GROUPING_ENABLE": true
}
```

Value	Behavior
true	Metrics are grouped by mailing ID
false	All metrics are sent to a single group

By default, the parameter is enabled ( `true` ).

When grouping is enabled, separate metric groups are created in the Pushgateway for each mailing. This simplifies:

- analyzing performance of individual mailings;
- building Grafana dashboards;
- configuring alerting rules;
- finding issues in a specific mailing.

When grouping is disabled, metrics from all mailings are aggregated into a single Pushgateway group.

RabbitMQ publisher metrics

RabbitMQ publisher business metrics are available for the `procworkflow` and `proctrigger` processes.

These metrics are used to monitor message publishing, delivery confirmation time, and the number of retry attempts.

Configuring histogram bucket values

For the `total_duration` and `confirm_duration` metrics, you can configure custom histogram bucket values.

Example configuration:

```
{
 "PROMETHEUS_METRICS_RMQ_PUBLISHER": {
 "MSEC_BUCKETS": {
 "total_duration": [10, 25, 50, 75.5],
 "confirm_duration": [10, 25, 50, 75.5]
 }
 }
}
```

Available metrics

Metric	Description
<code>total_duration</code>	Total message publishing duration
<code>confirm_duration</code>	Publishing confirmation duration
<code>retry_counts</code>	Number of retry attempts
<code>retried_count</code>	Number of messages sent with at least one retry
<code>lost_failed_events_count</code>	Number of messages discarded after exceeding the retry limit

Metric interpretation

When analyzing RabbitMQ publisher metrics, pay attention to the following changes:

Metric	Possible cause
Increase in <code>confirm_duration</code>	RabbitMQ slowdown or network issues
Increase in <code>retry_counts</code>	Unstable message delivery
Increase in <code>retried_count</code>	Elevated number of publishing errors
Non-zero <code>lost_failed_events_count</code>	Event loss after exceeding the retry limit

Procpiper metrics

Procpiper metrics are available for the `procpiper` process and are used to monitor pipeline processing performance in custom channels.

Configuring procpiper metrics collection

To enable procpiper metrics, add the process to the `PROMETHEUS_METRICS.PROCESSES` list and specify the address and port of the HTTP metrics server:

```
{
 "PROMETHEUS_METRICS": {
 "ENABLE": true,
 "PROCESSES": ["procpiper"]
 },
 "PROC_PIPER_METRIC_HOST": "0.0.0.0",
 "PROC_PIPER_METRIC_PORT": 8917
}
```

Add a metric scrape job to the Prometheus configuration:

```
scrape_configs:
- job_name: 'procpiper'
 metrics_path: /metrics
 static_configs:
 - targets:
 - '<procpiper-server>:8917'
```

Available metrics

Metric	Type	Labels	Description
ak_procpiper_pipe_duration_seconds	Histogram	channel_id, pipe_id, pipe_type, pipeline_type	Execution time of Action() for a single pipe
ak_procpiper_http_request_duration_seconds	Histogram	channel_id, pipe_id, status_code, result, pipeline_type	HTTP request duration in httpRequestPipe
ak_procpiper_event_writer_send_duration_seconds	Histogram	channel_id, pipe_id, pipe_type, result, pipeline_type	Event send duration via EventPipe / RmqPublisherPipe
ak_procpiper_event_total_duration_seconds	Histogram	channel_id, pipeline_type	Total time from consumer to pipeline exit

Metric interpretation

When analyzing procpiper metrics, pay attention to the following changes:

Metric	Possible cause
Increase in ak_procpiper_pipe_duration_seconds	Slowing of a specific pipe execution; possible issues with a dependent service
Increase in ak_procpiper_http_request_duration_seconds	Slowing of HTTP requests; network issues or remote API problems
Increase in ak_procpiper_event_writer_send_duration_seconds	Slowing of event sending to RabbitMQ; queue overload
Increase in ak_procpiper_event_total_duration_seconds	Overall pipeline processing slowdown; individual stage analysis required

Mailing metrics

Mailing metrics are used to monitor lead processing performance, individual stage execution time, and error counts during mailing execution.

The tables below list the main metrics. Actual names in Prometheus may contain additional prefixes, suffixes, and labels depending on the platform configuration.

Lag metrics

Metric	Description
cursor_lag_milliseconds	Mailing processing lag relative to the current queue state

An increase in cursor\_lag\_milliseconds may indicate insufficient resources, queue overload, or slowed lead processing.

General lead processing metrics

Metric	Description
lead_prepare_milliseconds	Lead preparation time
lead_processing_milliseconds	Lead processing time
lead_wait_milliseconds	Total wait time
lead_total_milliseconds	Total lead processing time

Stage processing metrics

Metric	Description
lead_suppress_lists_check_milliseconds	Suppress list check
lead_policy_check_milliseconds	Policy check
lead_static_milliseconds	Static data processing
lead_form_milliseconds	Form processing
lead_relation_milliseconds	Relations processing
lead_query_milliseconds	Query execution
lead_loyalty_milliseconds	Loyalty data processing
lead_loyalty_program_milliseconds	Loyalty program processing
lead_site_milliseconds	Site data processing
lead_json_milliseconds	JSON processing
lead_render_milliseconds	Content rendering

Metric	Description
lead_links_milliseconds	Link generation
lead_sends_milliseconds	Message sending

Stage processing metrics are used to find bottlenecks during mailing execution.

Stage wait metrics

Metric	Description
lead_suppress_lists_check_wait_milliseconds	Wait time for suppress list check
lead_policy_check_wait_milliseconds	Wait time for policy check
lead_static_wait_milliseconds	Wait time for static data processing
lead_form_wait_milliseconds	Wait time for form processing
lead_relation_wait_milliseconds	Wait time for Relations processing
lead_query_wait_milliseconds	Wait time for query execution
lead_loyalty_wait_milliseconds	Wait time for loyalty data
lead_loyalty_program_wait_milliseconds	Wait time for loyalty programs
lead_site_wait_milliseconds	Wait time for site data
lead_json_wait_milliseconds	Wait time for JSON processing
lead_render_wait_milliseconds	Wait time for rendering
lead_links_wait_milliseconds	Wait time for link generation

An increase in wait metrics typically indicates insufficient resources, locks, or overloaded dependent services.

Stage error metrics

Metric	Description
lead_suppress_list_check_failure_count	Suppress list check errors
lead_policy_check_failure_count	Policy check errors
lead_static_failure_count	Static data processing errors
lead_form_failure_count	Form processing errors
lead_relation_failure_count	Relations processing errors
lead_query_failure_count	Query execution errors
lead_loyalty_failure_count	Loyalty data processing errors
lead_loyalty_program_failure_count	Loyalty program processing errors
lead_site_failure_count	Site data processing errors
lead_json_failure_count	JSON processing errors
lead_render_failure_count	Rendering errors
lead_links_failure_count	Link generation errors
lead_sends_failure_count	Message sending errors

An increase in failure metrics indicates errors at specific mailing processing stages and can be used to configure alerting rules in Prometheus and Grafana.

Monitoring verification

After configuration, verify that Prometheus receives metrics.

For pull metrics, run the following requests:

```
curl http://<procworkflow-server>:8911/metrics
curl http://<proctrigger-server>:8912/metrics
curl http://<procpiper-server>:8917/metrics
```

For push metrics, check the Pushgateway:

```
curl http://<pushgateway-server>:9091/metrics
```

Replace `<procworkflow-server>`, `<proctrigger-server>`, `<procpiper-server>`, and `<pushgateway-server>` with the actual IP addresses or hostnames of your servers.

Also check the target status in the Prometheus UI. Jobs `procworkflow`, `proctrigger`, `procpiper`, and `pushgateway` should show status `UP`.

Common issues

Issue	Possible cause	What to check
Prometheus does not receive pull metrics	Process listens only on the local interface	Check <code>WF_METRIC_HOST</code> , <code>PROC_TRIGGER_METRIC_HOST</code> , and <code>PROC_PIPER_METRIC_HOST</code> values
<code>/metrics</code> endpoint is unavailable	Metrics service did not start	Check the port with <code>netstat -tln</code>
Metrics service does not start	Port is occupied by another process	Specify a free port in the <code>1024-9999</code> range
No <code>campaign</code> metrics in Pushgateway	Mailing has not been launched yet	Launch the mailing and re-check
All campaign metrics fall into a single group	Mailing ID grouping is disabled	Check <code>CAMPAIGN_ID_PROMETHEUS_GROUPING_ENABLE</code>



# 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 need to put the modified templates in this directory and add `custom.` to the file name.

```
$ 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	Meaning
<code>unsubscribe.html</code>	Page after unsubscribing, if it isn't specified in the message
<code>deprecated.html</code>	The page of the expired web message
<code>not_found.html</code>	Page for invalid links on tracking

## Custom notification templates

Custom notification templates are located in the directory `templates/email/<CountryCode>/`. As they will be overwritten during the upgrade, you can create files for them that override them. `CountryCode` determines the template language. If the language is not found, the default value will be EN (English).

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	Meaning
<code>content.json</code>	all	Applies to all templates, contains headers, <code>subject</code> and <code>reply-to</code> addresses of letters
<code>template_all.html</code>	all	Applies to all templates
<code>welcome.html</code>	account, admin	Appears when the user with access is created
<code>password.html</code>	account, admin	Password change by user or administrator
<code>complains.html</code>	account	In case of exceeding the rates for complaints
<code>hardbounces.html</code>	account	In case of exceeding the <code>hardbounce</code> indicators
<code>softbounces.html</code>	account	In case of exceeding the <code>softbounce</code> indicators
<code>report_weekly.html</code>	account	Weekly report of the user
<code>report_daily.html</code>	account	Daily report of the user
<code>sendlimit.html</code>	account	In case of exceeding the limit (tariff) for sending
<code>leadslimit.html</code>	account	In case of exceeding the limit (tariff) for the amount of data
<code>suspended.html</code>	account	In case the account is inactive in the admin panel
<code>sender_problem.html</code>	admin	During the process of setting up notification for sender in the administration panel

## Customization of form pages

All template files are located in the platform installation directory under `/templates/form/v1/EN` and `/templates/form/v1/RU`. Files in the `.tpl` format contain form templates and components used within those templates. The `.html` format contains standalone tables.

File	Description
<code>button.tpl</code>	Template for displaying buttons. Controls their appearance and placement in the form.
<code>content.json</code>	File containing the form's data and structure — includes field descriptions, transition logic, texts, settings, etc.
<code>user_code.tpl</code>	A custom template designed to override the default form template. It is used to insert custom code into the <code>&lt;head&gt;</code> , <code>&lt;body&gt;</code> , and <code>&lt;footer&gt;</code> sections of the page. It also allows wrapping the main form content with additional HTML markup.
<code>error.html</code>	Page displayed when an error occurs while loading the form or if the link is invalid.
<code>final.tpl</code>	Template for the final page shown to the user after successful form submission.
<code>form_fill.tpl</code>	Template for the main form page where the user fills in the fields.
<code>form.tpl</code>	Main container template for the form. Includes basic HTML layout, structure, and links to other templates.
<code>hidden.html</code>	Displayed when the form is hidden from public access. Usually contains a message like "Form unavailable".

File	Description
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 use it instead of the original configuration.

**Static tracking files**

In some cases it can be useful to place static files on the tracking — the necessary files can be placed in the directory `ak/static/trk/public/`. This way, they will be available at the absolute path `/static/` of your tracking. For example, this directory contains the `cat.html` file.

# Send Message ID

**Send Message ID** is a unique sending identifier used in the platform. It stores the identifiers necessary to determine the profile and objects within which the message was sent. There are several identifier formats in Altcraft Platform.

## Identifier types

Type	Prefix	Description
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 <code>Message-ID</code> and <code>Return-Path</code> headers in email messages.
FSMID	f	Full SendMessageID. Added to the <code>X-AKMTA-FSMID</code> header in email messages. Contains maximum information about the send.
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`:

 Send test event

Select event type

Email: Send

Event structure

```
{
 "event_type": "send",
 "event_date": "2023-11-30T14:40:24.516052354+03:00",
 "list_id": 123456,
 "profile_id": "ffffffffffffffffffff",
 "_xxh": "61be4d0b871502d4",
 "_md5": "a42c44fc77533b43ffd569ed4c83e377",
 "subscription_hash_id": "b7b5bd90",
 "is_test": true,
 "event_data": {
 "campaign_id": 284,
 "launch_id": "2_5U_4Hy116Zuyar",
 "message_id": 9,
 "resource_id": 3,
 "segment_id": 80,
 "send_message_id": "w4Hy117d8rpR_2_5U_a_g_2o_3_4.2Mw2HxVPFzUEwsGR3_5GmMaj"
 },
 "custom_data": {
 "email": "kirk.mckinney83@example.com"
 },
 "subscription": {
 "resource_id": 0,
 "channel": "email",
 "email": "kirk.mckinney83@example.com",
 "priority": 0,
 }
}
```

Send

Also `SendMessageID` is available in the profile history. You can [export profile history](#) from a database or segment. The identifiers are recorded in the `SMID` column.

Segments ⓘ

Search...

🔍

🔖

⏚

+ Create

Access groups Main

😊

Matches: 51

↑↓

ID

First

1

2

3

Last

Export profile history

#54 Active subscribers

Dynamic

Profiles: 8319

🔄

⌚ Less than 1 sec.

📄

✎

📥

🔄

📊

📄

🗑

updated 30 Nov 2023 at 14:39 by user Pol Zak

## Decoding SMID

In the main menu select **Tools** —> **Message ID decoder**:

# Dashboard

That's what we have today:



- ARF decoder
- URL decoder
- SMID decoder
- Tasks
- Jobs

Paste the message ID (SMID, FSMID, ESMID или USMID) and click **Decode**. Depending on the identifier type, the result contains detailed information about platform objects used to generate this message:

- 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 interface.

# SMID Decoder

Paste SMID, FSMID, ESMID or USMID here:

fDbrZmUOCeN1LUPcHsiCJwXAAAEh4CEfEPEBAAAKUBEBEplA8-oUiCJwXYN836j9i\_HE3BfDjZDJmMWM0ODibQ8doA

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 from the root directory of the platform, replacing `<Any SMID>` with the desired SMID. The command will return the same arguments as the decoder in the Admin Panel, but in plain text:

```
USMID: xxxxxxxxxxxxxxxx
Account: 1 - account1
AK Client ID: ak1
Campaign: 2 - campaign1
Database: 3 - database1
Date: 2019-11-09T23:00:00Z
External Id: exid1
Is Indirect: true
Is Test: false
Launch Id Time: 2019-11-10T23:00:00Z
Message: 4 - message1
Node: 5 - node1
Profile: /* profile schema data */
Provider: /* provider schema data */
Segment: 6 - segment1
Sender: 7 - sender1
Series Id: 123
Subscription: /* subscription schema data */
Webversion: /* webversion schema data */
Workflow: 8 - workflow1
```

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 — [User settings](#)).

```
<yandex>
 <users>
 <altcraft>
 <password>abcdefghijklmnopqrstuvwxyzABCDEF</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 "abcdefghijklmnopqrstuvwxyzABCDEF" 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 documentation for more details — [Settings profiles](#)).

```
<?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
 errors.
 -->
 <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 the ClickHouse DBMS.

```
"CLICKHOUSE_SYSTEM": {
 "HOST": "localhost",
 "PORT": 9000,
 "USER": "altcraft",
 "MAX_CONNECTION": 100,
 "CONN_TIMEOUT": 15,
 "PASSWORD": "abcdefghijklmnopqrstuvwxyzABCDEF",
 "IS_DEBUG": false
},
```

**Step 5.** Run migration. For these purposes, a special utility has been prepared — `actions_migrate`. Full list of available options:

Parameter	Default value	Description
--accounts string	All active accounts	List of accounts to import separated by commas Example: "1, 2, 3"
--appname bool	false	Show additional information about the utility
--batch_timeout int	5	Time in seconds after which the insert will be executed
--ch_batch int	100000	Minimum batch of events to insert
--ch_debug bool	false	Enable additional logging level
--ch_host string	The main configuration <code>main.json</code> is used	Host to connect
--ch_password string	The main configuration <code>main.json</code> is used	Password to connect
--ch_port int	The main configuration <code>main.json</code> is used	Connection port
--ch_user string	The main configuration <code>main.json</code> is used	User to connect
--config string	/opt/MarketingPlatform/config/main.json	Path to the main configuration file <code>main.json</code>
--exclude string	""	List of accounts to be excluded from import separated by commas Example: "1, 2, 3"
--level_log string	INFO	Logging level: TRACE, DEBUG, INFO, WARN, ERROR, CRITICAL, FATAL
--log_path string	./actions_migrate.log	Location of the log file
--range string	""	Time range to sample existing history The format is "MM.YY-MM.YY", "-MM.YY", "MM.YY-"
<del>--rmq bool</del>	<del>false</del>	<del>Additional option for inserting via RabbitMQ</del>
--version bool	false	Show utility version
--worker_size int	-1	Number of workers (by default, it corresponds to the number of vCPUs).

#### TIP

Migration can take a long time, up to 3 or more days. To continue the migration after disconnecting from the server, you can use the `screen` utility to work in a separate session. You can find an example of using the utility here: <https://access.redhat.com/articles/5247>.

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 service, it is possible to specify a test address both for the legacy API (`FIREBASE_LEGACY_CUSTOM_SENDING_URL`) and for API v1 (`FIREBASE_APIv1_CUSTOM_SENDING_URL`).

# Utility for importing push subscriptions to Firebase project

## Description

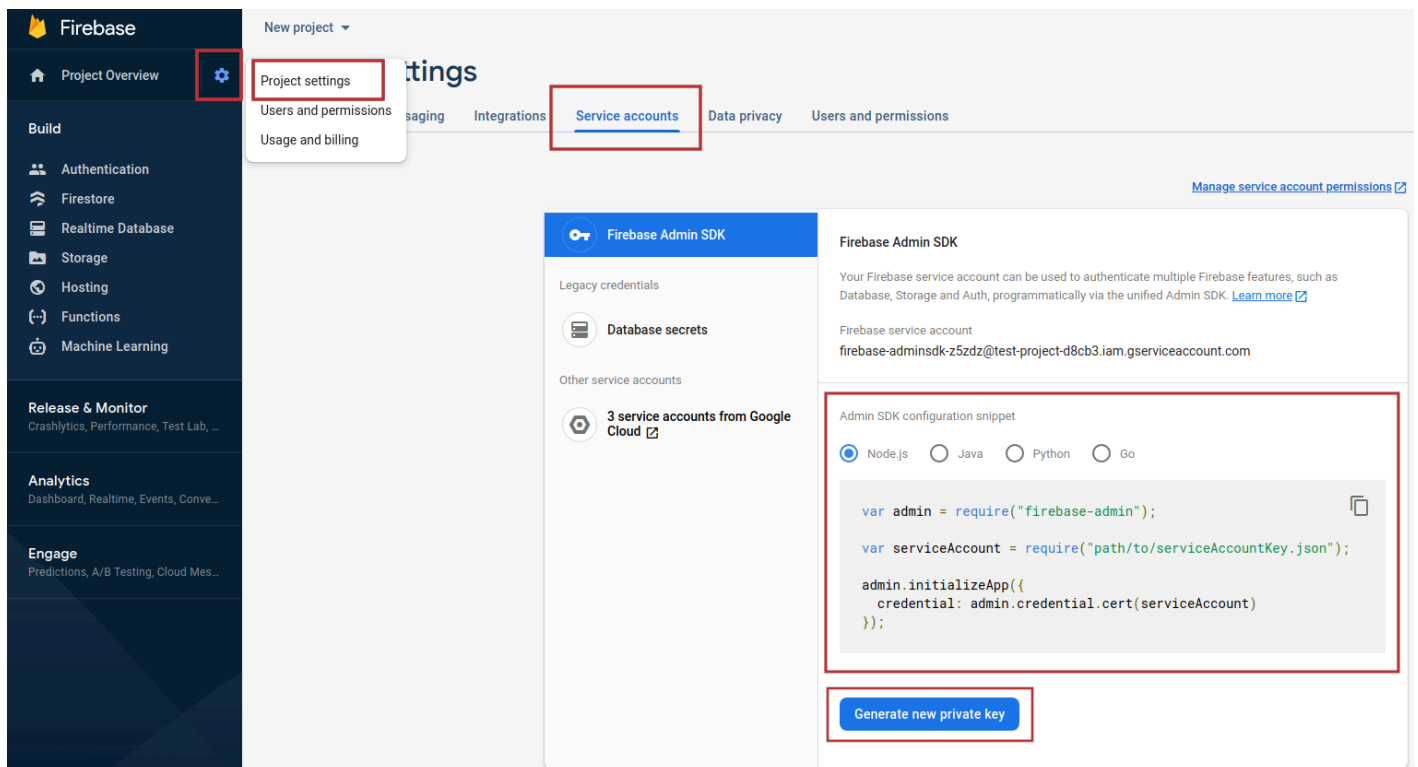
Utility `firebase_import_subs` is a tool for importing [Web Push](#) subscriptions of a third-party push service into a Firebase project.

## How does it work?

1. A **CSV file** is sent as the input data. It contains fields for subsequent matching during the import process and subscription data.
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 column **firebase\_push\_token** is **added**.

## 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.com>. Go to the settings of the project into which you want to import subscriptions: **"Project Settings"** —> **"Service accounts"**. Go to the **"Admin SDK configuration snippet"** section and click **"Generate 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.json --auth_field_name web_auth --p256_field_name web_p256 --endpoint_field_name identifier --publ_vapid_key
BDk08iue5By4UcLUK5SCL0LLtD19RwvSyqr0W0IjiXQTJmwH0yHxcrqidZ-PyJ9QwksTBHkh9wx88YK4xnn17_Y
```

## Parameters

Required parameters	
in	
out	
cred_json	
auth_field_name	
p256_field_name	
endpoint_field_name	
publ_vapid_key	
Additional parameters	
—	
proxy	

Required parameters
config

Utility example:

```
root@dev-pollina-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 Bdk08iue5By4UcLUk5SCL0LLtDL9RWvSyqr0W0IjiXQTJmW0yHxcrgidZ-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:
  - the same data that the initial file contains
  - 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	55428f9-2719-4	https://fcm.goo	Win32	2021-03-03 18	h55Z+Om1vu3s	BMic5wxJH/Z0j0	cpnQpvZ2E1g:APA91bHNymNJJJr7b9fLuR7eR-qJyyZlhk8ZPpTvbQ4Pqr664pai5KCECKv0CkDPZp8
3	77a82d5-d5e7-4	https://fcm.goo	Linux armv8	2021-02-24 12	O+9JVatXnL8k	BO213jqIRNxzx	c90sAUMrbBM:APA91bHWDtl7THwoUIFPEPREQUwHzEHkONIEKMfSqwzdJROoLsvcH4HVN8e_\
4	a04bt05e-80c6-4	https://fcm.goo	Win32	2021-02-24 12	9i+K7OV9U9dr	BP9cNcRUVECX	fcs_lpmNlLw:APA91bHD5-oV1NylPeO9ukjejXZb6kGvGv8W0ocoCLiUCcufFvynfluVUpmmZosN31QNjI
5	d7d51395-6994-4	https://updates	Linux x86_64	2021-02-24 11	xzXGzS4QPXz	BGB4N1OVM0C	dZ_kyI0WIPA:APA91bEopoNjF67KG15gy9JLDKNO5qMQX79LJ61XsyWdFn-oAzQXDVs4_HNy3ksmi
6	e2853901-52c1-4	https://fcm.goo	Linux x86_64	2021-03-10 12	Agx80qyVeVFu	BK8SEpJU9Qq	scrGHstlcBSq:APA91bGHIPIkLTJ9Wm6RF9HddeahVp1sfNhFUVrDvKnge7rpuD_dW1ncjGnwUaF81UH

- Send the resulting csv file to the platform user.

# Administrator API

This part of documentation describes Altcraft Platform administrator API interface.

## NOTE

All administrator API requests use POST method and need a token, created in Altcraft Platform administrative panel.

Some requests are similar to user API requests — the difference comes with multiple accounts in one Altcraft Platform service instance, so `account_id` parameter allows you to specify account you want to address.

# 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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxy <sup>z</sup> "	Yes	API token
format	string	"json"	No	Response data format By default – json
from_id	int	1	No	The first account ID on the list By default –1
limit	int	100	No	Accounts list limit By default – 1000

## Request example

### JSON

```
{
 "token": "abcdefghijklmnopqrstuvwxyz",
 "from_id": 300,
 "limit": 10
}
```

### XML

```
<xml>
 <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text

Parameter	Type	Description
id	int	Account ID
name	string	Account name
timezone	string	Account time zone
tariff_id	int	Account tariff ID
is_active	bool	Active account flag
pixels_enabled	bool	Enabled pixel flag
pushes_enabled	bool	Enabled push flag
is_smart	bool	Smart account flag
is_subscribers_suspend_all	bool	All subscribers suspended flag
is_tariff_exceed	bool	Tariff exceeded flag

# 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	Example	Required	Description
token	string	"abcdefghijklmnpqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format (By default – json)
id	int	1	No	Account ID
name	string	"Account name"	Yes	Account name
title	string	"example title"	Yes	Account title
tz	string	"UTC"	Yes	Account time zone
tariff	int/string	"example tariff"	Yes	Tariff ID or name
web_version_policy	int/string	"0"	No	Web version policy ID or name
account_postgresql	int/string	"2"	No	Postgresql account ID or name
account_cache_clickhouse	int/string	"3"	No	Clickhouse account ID or name
external_data_connectors	int	[1,2,3,4]	No	External data connectors
reports_day	int	4	Yes	Day of the report
reports_hour	int	18	Yes	Time of the report
force_2fa_auth	bool	false	No	Requirement of mandatory use of two-factor authentication
enable_push	bool	true	No	Enabling Push Notifications
custom_push_domain	string	"example.com"	No	Domain of custom push notifications
enable_smart_settings	bool	true	No	Access to smart settings
enable_pixel	bool	true	No	Enabling a pixel
enable_track_email_replies	bool	false	No	Tracking email replies

## Request example

### JSON

```
{
 "token": "abcdefghijklmnpqrstuvwxyz",
 "name": "Steve",
 "title": "ASdf",
 "tz": "UTC",
 "tariff": "example tariff",
 "reports_day": 5,
 "reports_hour": 22
}
```

### XML

```
<xml>
 <token>abcdefghijklmnpqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text
saved_id	int	Account 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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format (By default – json)
id	int	1	Yes	Account ID

## Request example

### JSON

```
{
 "token": "abcdefghijklmnopqrstuvwxyz",
 "format": "json",
 "id": 9
}
```

### XML

```
<xml>
 <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format (By default – json)
account_id	int	1	Yes	Account identifier
password	string	"abcdefghijkl1232313!"	Yes	User password
name	string	"name"	Yes	User name to be used in the login
fname	string	"Fname"	Yes	User name
lname	string	"Lname"	Yes	User surname
lang	string	"RU"	Yes	language
email	string	" <a href="mailto:abcdefghijkl@example.com">abcdefghijkl@example.com</a> "	Yes	User email
master	bool	true	Yes	Access to master settings

### 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": "abcdefghijklmnpqrst",
 "format": "json",
 "account_id": 1,
 "password": "asdfGHJKOP12!_",
 "name": "example",
 "lname": "Example",
 "fname": "Example",
 "lang": "RU",
 "email": "example@gmail.com",
 "master": false
}
```

### XML

```
<xml>
 <token>abcdefghijklmnpqrst</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	Type	Description
error	int	Error code
error_text	string	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	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format (By default – json)
account_id	int	1	Yes	Account identificator
password	string	"abcdefghijklmnopqrstuvwxyz1232313"	Yes	User password
name	string	"name"	Yes	User name to be used in the login. <b>Note:</b> the name must be in lowercase, otherwise the user will not be added.
fname	string	"Fname"	Yes	User name
lname	string	"Lname"	Yes	User surname
lang	string	"RU"	Yes	language
email	string	" <a href="#">abcdefghijklmnopqrstuvwxyz@example.com</a> "	Yes	User email
master	bool	true	Yes	Access to master settings
send_welcome	bool	true	No	Sending a welcome email

### 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": "abcdefghijklmnopqrstuvwxyz",
 "format": "json",
 "account_id": 1,
 "password": "abcdefghijklmnopqrstuvwxyz1232313",
 "name": "example",
 "lname": "Example",
 "fname": "Example",
 "lang": "RU",
 "email": "example@gmail.com",
 "master": false
}
```

### XML

```
<xml>
 <token>abcdefghijklmnopqrstuvwxyz</token>
 <format>json</format>
 <account_id>example@example</account_id>
 <password>abcdefghijklmnopqrstuvwxyz1232313</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxy"	Yes	API token
format	string	"json"	No	Response data format (By default – json)
account_id	int	1	Yes	Account identifier
name	string	"example"	Yes	User name to be used in the login

## Request example

### JSON

```
{
 "token": "abcdefghijklmnopqrstuvwxy",
 "format": "json",
 "account_id": 1,
 "name": "example"
}
```

### XML

```
<xml>
 <token>abcdefghijklmnopqrstuvwxy</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format (By default – json)
account_id	int	9	Yes	Account ID
active	bool	true	No	Acticvate user
master	bool	false	No	Access to master settings
lang	string	RU	No	Language

## Request example

### JSON

```
{
 "token": "abcdefghijklmnopqrstuvwxyz",
 "format": "json",
 "account_id": 9,
 "active": true,
 "master": false,
 "lang": "RU"
}
```

### XML

```
<xml>
 <token>abcdefghijklmnopqrstuvwxyz</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	Description
error	int	Error code
error_text	string	Error text
username	string	User name to be used in the login
first_name	string	User name
last_name	string	User surname
email	string	User email
notify_email	string	User email for notifications
phone	string	Phone number
lang	string	language
active	bool	User activity status
master	bool	Access to master settings



# Sending a Welcome Email

## Description

Sends a welcome email with a link to set a password. The email provides personal data about the initiator of the recovery/registration completion operation:

- 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

# Добро пожаловать!

Спасибо, что присоединились к Altcraft Marketing.

Ваши данные для входа ниже. Обязательно смените автоматически сгенерированный пароль и никому его не показывайте.

Если вы не запрашивали эту ссылку, проигнорируйте это письмо.

С уважением, Altcraft

Логин example@steve

Установить пароль



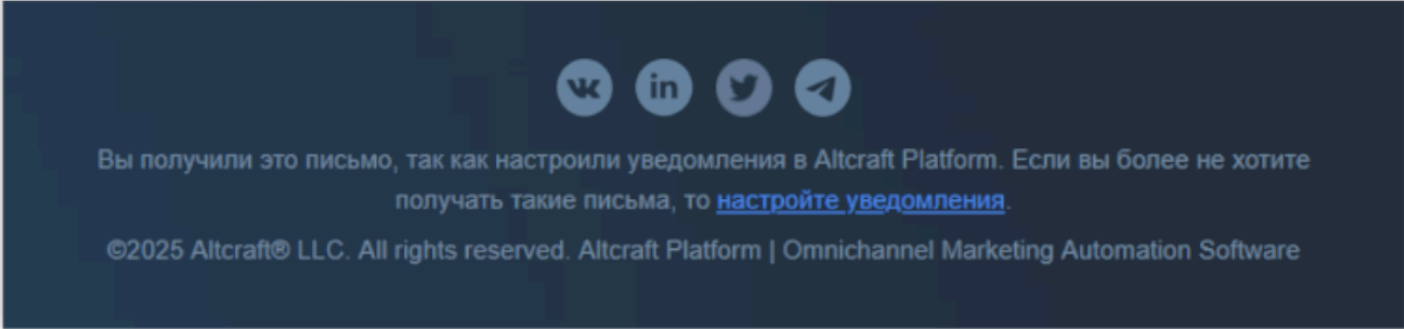
Платформа



Время

15 августа 2025 10:28 UTC

Если у вас возникли вопросы, обратитесь в нашу [службу поддержки](#).



URL Address

Method: POST

https://example.com/api/v1.1/admin/accounts/users/send\_welcome

Request Parameters

Parameter	Type	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
account_id	int	1	Yes	Account identifier
name	string	"example"	Yes	Username for personalization

Request Example

JSON

```
{
 "token": "abcdefghijklmnopqrstuvwxyz",
 "account_id": 7,
 "name": "example"
}
...

XML

```xml
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Да	API token
id	int	10	Да	Node ID
format	string	"json"	Нет	Response data format By default – json

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
from_id	int	1	No	The first node ID on the list By default –1
limit	int	100	No	Nodes list limit By default – 1000

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz"
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text
id	int	Node ID
name	string	Node name
host	string	Node IP:port
mode	string	"HTTPS" or "SSH" mode

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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxy"	Yes	API token
id	int	10	Yes	Node ID
format	string	"json"	No	Response data format By default – json

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxy",
  "id": 10
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxy</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	Description
error	int	Error code
error_text	string	Error text

Parameter	Type	Description
synced	bool	MTA synchronization status
running	bool	MTA status

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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
id	int	10	Yes	Node ID
format	string	"json"	No	Response data format By default – json

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
id	int	10	Yes	Node ID
format	string	"json"	No	Response data format By default – json

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	“abcdefghijklmnopqrstuvwxyz”	Yes	API token
name	string	“AKMTA mail sender”	Yes	Sender name
type	string	“akmta”	Yes	Sender type
assigned_accounts	JSON array	[123, 456]	Yes	Accounts assigned to the sender
verified_domains	JSON array	["mail.roo", "rambler.roo"]	Yes	Verified email damains
akmta_obj	JSON object		Yes	Sender options
id	int	16	Required for update requests	Sender ID for update requests

Request example

JSON

```

{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "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>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnqrstuvwxyz"	Yes	API token
id	int	10	Yes	Sender ID
format	string	"json"	No	Response data format By default – 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": "abcdefghijklmnopqrstuvwxyz",
      "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": "abcdefghijklmnopqrstuvwxyz",
      "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>abcdefghijklmnopqrstuvwxyz</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>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text
id	int	Sender ID
name	string	Sender name
type	string	Sender type
channel_type	string	Communication channel type
assigned_accounts	JSON array	Accounts assigned to the sender
verified_domains	JSON array	Verified email domains
akmta_obj	JSON object	AKMTA 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	Example	Required	Description
token	string	"abcdefghijklmnqrstuvwxyz"	Yes	API token
id	int	10	Yes	Sender ID By default – 1
format	string	"json"	No	Response data format By default – json
account_id	int	1	Yes	Account ID By default – 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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
from_id	int	1	No	The first sender ID on the list By default –1
limit	int	100	No	Senders list limit By default – 1000

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz"
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text
id	int	Sender ID
name	string	Sender name
type	string	Sender type
channel_type	string	Communication channel

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	Example	Required	Description
token	string	“abcdefghijklmnopqrstuvwxyz”	Yes	API token
format	string	“json”	No	Response data format By default – json
id	int	12	Yes	Sender ID By default – 1

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 683
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
id	int	12	Yes	Sender ID By default – 1

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 683
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
id	int	12	Yes	Sender ID By default – 1
account_id	int	12	No	Account ID By default – 1
campaign_id	int	12	No	Mailing ID By default – 0
isp	string	"Gmail"	No	ISP (email provider)
group *	string	"total"	No	Message grouping mode By default – 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": "abcdefghijklmnopqrstuvwxyz",
  "id": 119
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxy"	Yes	API token
format	string	"json"	No	Response data format By default – json
id	int	12	Yes	Sender ID By default –1
account_id	int	12	No	Account ID By default –1
campaign_id	int	12	No	Mailing ID By default –1

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxy",
  "id": 3,
  "account_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxy</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxy"	Yes	API token
format	string	"json"	No	Response data format By default – json
id	int	12	Yes	Sender ID By default –1
account_id	int	12	No	Account ID By default –1
campaign_id	int	12	No	Mailing ID By default –1

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxy",
  "id": 3,
  "account_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxy</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	Type	Description
error	int	Error code
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
id	int	12	Yes	Sender ID By default–1
account_id	int	12	No	Account ID By default–1
campaign_id	int	12	No	Mailing ID By default–1

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 3,
  "account_id": 1,
  "campaign_id": 3
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code

Parameter	Type	Description
error_text	string	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	Example	Required	Description
token	string	"abcdefghijklmnqrstuvwxyz"	Yes	API token
account_id	int	306	Yes	Account ID
from_id	int	11768	No	The first virtual sender ID on the list By default – 1
limit	int	10	No	Virtual senders list limit By default – 1000
format	string	"json"	No	Response data format By default – 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	Type	Description
error	int	Error code
error_text	string	Error text
id	int	Virtual sender ID
name	string	Virtual sender name
sms_channel	bool	true if SMS channel is enabled
email_channel	bool	true if email channel is enabled

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	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
account_id	int	306	Yes	Account ID
id	int	12	Yes	Virtual sender ID
format	string	"json"	No	Response data format By default – json

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 1,
  "account_id": 1
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text
id	int	Virtual sender ID
name	string	Virtual sender name
email_rules	JSON-object	Rules for email sender
sms_rules	JSON-object	Rules for SMS sender

Create virtual sender

Description

Creates a new virtual sender

Request URL

Method: POST

https://example.com/api/v1.1/admin/vsenders/add

Request parameters

Parameter	Type	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
account_id	int	306	Yes	Account ID
name	string	"New virtual sender"	Yes	New virtual sender name
email_rules	JSON-object		Yes	Rules for email senders
sms_rules	JSON-object		Yes	Rules for SMS senders

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "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>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text
saved_id	int	New virtual sender ID

Update virtual sender

Description

Updates an existent virtual sender configuration.

Request URL

Method: POST

https://example.com/api/v1.1/admin/vsenders/update

Request parameters

Parameter	Type	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
account_id	int	306	Yes	Account ID
id	int	12345	Yes	Virtual sender ID
name	string	"New virtual sender"	Yes	Virtual sender name
email_rules	JSON-object		Yes	Rules for email senders
sms_rules	JSON-object		Yes	Rules for SMS senders

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "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>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text

Clone virtual sender

Description

Creates a copy of an existing virtual sender.

Request URL

Method: POST

https://example.com/api/v1.1/admin/vsenders/clone

Request parameters

Parameter	Type	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
account_id	int	306	Yes	Account ID
format	string	"json"	No	Response data format By default – json
id	int	12345	Yes	Virtual sender ID
name	string	"Clone of ..."	No	New virtual sender name By default – "Clone of ..."

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "id": 14,
  "name": "Clone of virtual sender #14"
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text
saved_id	int	New virtual sender ID

Delete virtual sender

Description

Removes a virtual sender entry.

Request URL

Method: POST

https://example.com/api/v1.1/admin/vsenders/delete

Request example

Parameter	Type	Example	Required	Description
token	string	"abcdefghijklmnopqrstuvwxyz"	Yes	API token
format	string	"json"	No	Response data format By default – json
account_id	int	306	Yes	Account ID
id	int	12345	Yes	Virtual sender ID

Request example

JSON

```
{
  "token": "abcdefghijklmnopqrstuvwxyz",
  "account_id": 1,
  "id": 16
}
```

XML

```
<xml>
  <token>abcdefghijklmnopqrstuvwxyz</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	Type	Description
error	int	Error code
error_text	string	Error text