SED Server

Manual

v 4.0

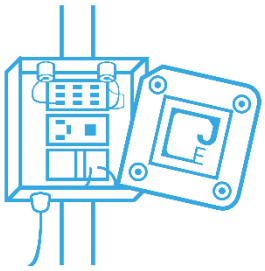


Table of Contents

Introduction

The SED Server serves as a central point for communication between devices and users. The server implements various communication standards so that it can communicate with as many security systems available on the market as possible. At the same time, it offers additional functionality in the form of so-called plug-ins.

Installation

An installation manual how to start SED server on your own host system.

Prerequisities

1. A Linux machine (physical or virtual)
    - 2 Cores
    - 4 GiB RAM
    - 20 GiB storage (40+ GiB if you plan to store events & sounds)
    - Tested with Ubuntu 24.04 LTS

Installation using script

1. Download and unpack the installation script in that directory

   ```
   curl https://soundeventdetector.eu/shared-files/11986/?2026-04-24_SED_Wizard_installer_pack.tar | tar -x
   ```

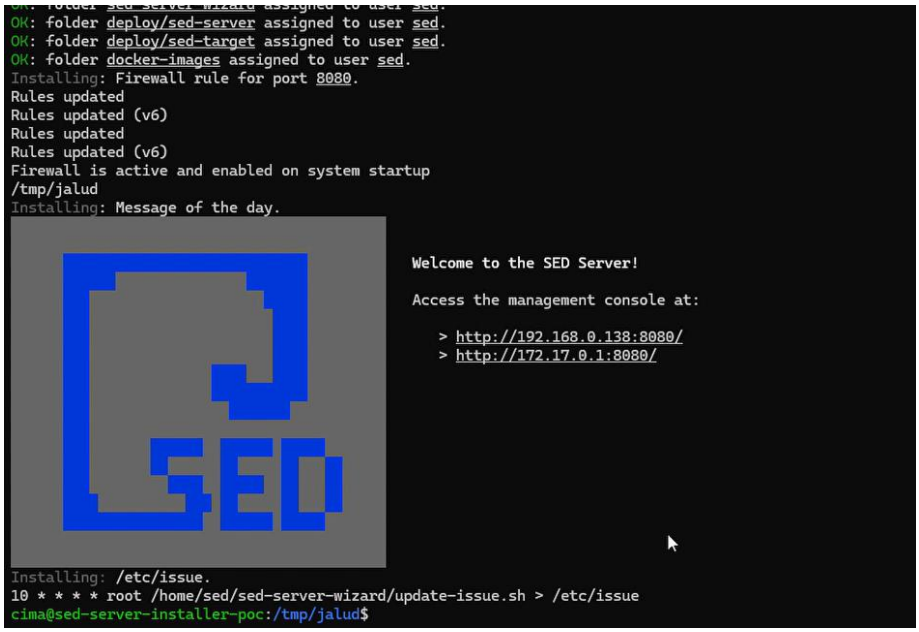
2. Run the installation script

   ```
   ./install-over-ssh.sh
   ```

3. You'll be prompted for password to elevate privilegies so your account must be able to use `sudo`.

   Usually the account created during Linux installation is granted such permission so no extra action is needed.

4. A sucessfull script run shall end with a splash screen with SED logo and a list of URL's at which you can access the web console for installation continuation (and further system monitoring).

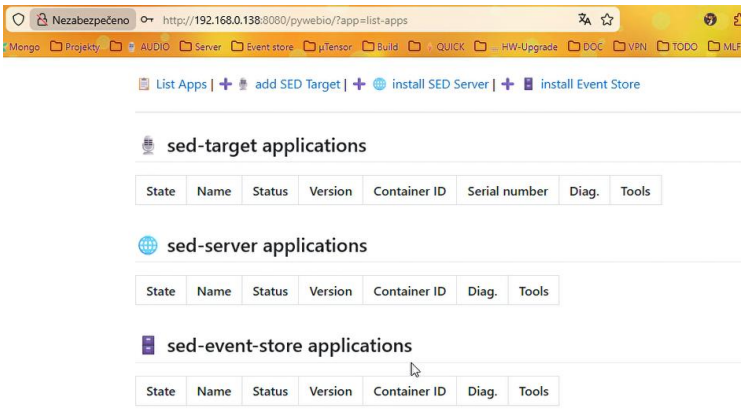


5. Open web browser and use one of displayed URLs.
   e.g. http://192.168.0.115:8080/

6. When prompted enter credentials for Linux user *sed* created during the script run. Initially these are *soundeventdetector*

   We recommend to change them in linux using command `sudo passwd sed`

7. In the top menubar is a link ➕🌐 *install SED Server* which opens dialog for changing hostname, providing license certificate and key and password for that key.



8. When installation succeeds you see gren checkmark and few log lines from the running SED Served docker container.

9. Return to overview using 📋 *List Apps* link in top menu bar. Server container logs can be inspected from overview screen by clicking on logs icon 📄. ⚙️ Grear and 👷 User icons just show configuration files.

Offline Scenario

It is always easiest to install everything by the time you have access to the Internet. But sometimes restrictions do apply and you have to pre-download everything in advance. Below you can find list of things to do before offline installation can begin:

1. Download the file 2026-03-11_SED_Wizard_installer_pack.tar
2. Have Linux with Docker installed
3. Download following docker images by the instruction in the first section of Offline docker image
    - jaluddevelopment/sed-server:4.0
    - jaluddevelopment/sed-target:4.0
    - nginx:1.23

- mongo:5
- jaluddevelopment/sed-event-store-adaptor:1.3.4

Download docker image

Manipulation with docker image to locations without internet connectiity can be achieved by Saving image at location with Internet

```
docker image pull jaluddevelopment/sed-server:4.0
```

```
docker image save --output sed-server-4-0-docker-image.tar.gz \
jaluddevelopment/sed-server:4.0
```

Upload docker image

Once you transfer *sed-server-4-0-docker-image.tar.gz* and other files to the disconnected place you can import it to local docker image store by loading like below

```
docker image load --input sed-server-4-0-docker-image.tar.gz
```

You can now verify the brought image presece

```
docker image ls
```

Trouble shooting

1. Check the container is running

   ```
   sudo docker ps
   ```

   You shoud see a container named *sed-sed-server-1*

2. Check logs

   ```
   sudo docker logs sed-sed-server-1 2>&1 | less
   ```

   There shall be no obvious errors.

3. Check configuration and certificates had been create into mapped directory of *~/SED/Config/*

   ```
   docker exec sed-sed-server-1 ls -la \
       Config/config.json \
       Config/map.json \
       Config/certificates/server.crt \
   ```

```
        Config/certificates/server.key \
        Config/certificates/user


    ls -la \
        ./SED/Config/certificates/server.crt \
        ./SED/Config/certificates/server.key \
        ./SED/Config/certificates/user
```

All files shall be reported on size and date modified.

Update SED server

Set working directory to where you installed the SED Server (e.g. /opt/jalud/SED)

```
cd ~/deploy/sed-server/SED
```

```
./update-sed.sh
```

Edit, Backup & Restore of configuration

To backup configuration copy the configuration file from container to the host system.

```
docker cp \
  sed-sed-server-1:/root/DEPLOY/SED/SEDServer/Config/config.json \
  ./my_config.bak.json
```

To restore the configuration copy the JSON file from host system to the SED server container.

```
docker cp \
  ./my_config.bak.json \
  sed-sed-server-1:/root/DEPLOY/SED/SEDServer/Config/config.json
```

Map - device & user defnitions

Snapshots of mapping files are available at container's address `sed-sed-server-1:/root/DEPLOY/SED/SEDServer/Data/MapBackups/map_*.json` which is currently being mapped to host system so You can back them up by host system tools.

However the restoration of map must be done by copying file from host to container by below command

```
docker cp \
  ./SED/Data/MapBackups/map_250808.json \
  sed-sed-server-1:/root/DEPLOY/SED/SEDServer/Config/map.json
```

Note: change `map_250808.json` to actual file that You are restoring.

Edit configuraion

As configuration files can be modified at runtime and SED Server picks up notification from filesystem, these files can't be mapped into container. These files are located on Docker persisten volume and thus their modification mus be done via a running container. For convenience we have two scripts

```
./edit-config.sh
```

```
./edit-map.sh
```

that interacts with running container and retain information where are those files.

Server configuration

The server configuration is in the *config.json* file in JSON format. Modifications can be made manually directly in the file or using the SEDControl application - see the **SEDControl_Manual_v3.11** manual for more information.

Communication protocols

For integration into diverse security and monitoring systems, SED Server implements several key communication protocols simultaneously. This approach ensures a high level of compatibility with most technologies commonly used in the market, thus greatly facilitating deployment in a variety of environments and scenarios.

These communication protocols can be activated, deactivated or configured within the server in the **protocols** section. Some settings are the same for all communication protocols:

- **name** - name of the communication protocol
- **active** - this setting activates/deactivates the communication protocol. When the protocol is deactivated, it cannot be used for communication with the user application.
- **microservice** - setting if the communication protocol will be run in a separate process or if it will be part of the main server process. The main advantage of running in a separate process is higher system stability.

TCP Protocol (JALUD)

This is the main communication protocol used by all SED devices to communicate with the server and also by proprietary SED applications. Unlike the others, this protocol cannot be manually disabled but instead shuts itself down when the license expires.

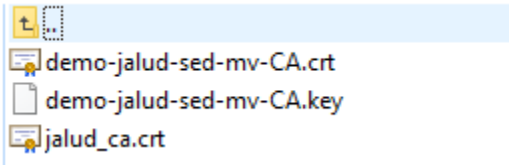
In addition to communication, it is also used to generate user certificates, which each user automatically downloads to the SED applications after the first login and does not need to continue to log in with a name and password.

The TCP protocol can use both secure (HTTPS) and non-secure (HTTP) versions of communication. Secure communication is recommended and enabled by default.

Name: **TCPProtocol**

Settings:

- HTTP
- active - enables/disables the unsecured version of the communication protocol (default false)
- port - port number (default 8001)
- HTTPS
- active - activates/deactivates the secure version of the communication protocol (default true)
- port - port number (default 8002)
- server_crt - path to the server certificate for secure communication. The certificate is generated automatically based on the Certificate Authority supplied when the SEDServer is purchased and the Hostname of the machine on which the server is running. The root folder for evaluating the relative path is the mapped folder *~/SED/Config/* (default "certificates/server.crt").
- server_key - path to the private key for secure communication - see server_crt (default "certificates/server.key") for more information.
- server_key_passwd - password of the private key (default "").
- server_ca_path - the path to the CAs that will consider the server secure. The license files supplied when installing the SED Server are placed in this folder (default "certificates/ca").



- server_ca_key_passwd - the password to the Certification Authority key (supplied when the SED Server was purchased).

ONVIF (Open Network Video Interface Forum)

ONVIF is an open standard designed for communication and integration between devices and systems in the video surveillance industry. In our case, ONVIF is primarily used to connect the server to a VMS (Video Management System), allowing easy integration with camera systems and surveillance platforms from various manufacturers.

Name: **onvifProtocol**

Manual: **SED_ONVIF_Driver_1.6.0**

Settings:

- HTTP
- active - activates/deactivates the unsecured version of the communication protocol (default true)
- address - ip address of the machine. Used in responses to some client requests according to the ONVIF specification. The application automatically adds the first IP address from the system that is not a localhost. If it is not the correct IP address, it needs to be changed manually.
- port - port number (default 8004)
- HTTPS
- active - activates/deactivates the secure version of the communication protocol (default true)
- port - port number (default 8005)
- address - see HTTP settings
- server_crt - see TCP Protocol
- server_key - see TCP Protocol
- server_key_passwd - - see TCP Protocol
- server_ca_path - - see TCP Protocol
- RTSP
- port - port number for RTSP streaming - video, audio and metadata (default 8006)
- Discovery
- discoverable - enables/disables ONVIF functionality to discover the server on the local network (default true)
- end_point - the uuid that is used to identify the SED Server (generated on first boot)

REST Server

The REST API provides a modern and flexible way to communicate that is now the standard in software integration. The server provides a REST interface for easy

connection to other systems - whether for device management, event transfer, or data collection.

Name: RestProtocol

Manual: **SED_REST_Driver_1.1.1**

Settings:

- HTTP
- active - activates/deactivates the insecure version of the communication protocol (default false)
- port - port number (default 8008)
- HTTPS
- active - activates/deactivates the secure version of the communication protocol (default true)
- port - port number (default 8009)
- server_crt - see TCP Protocol
- server_key - see TCP Protocol
- server_key_passwd - - see TCP Protocol
- server_ca_path - - see TCP Protocol
- storage_name - address of Firestore storage for storing media - audio samples and photos (default "")

SIA DC-09

We implement SIA DC-09 protocol for communication with centralized protection consoles (PCO). This is a widely used protocol used by security agencies and monitoring centers to receive alarm events. The SIA DC-09 implementation ensures reliable and standardized transmission of event information to PCOs.

Name: **SIADCProtocol**

Manual: SED_SIA-DC09_Driver_1.2.2 and SED_Alarm_Table

Settings: storage_name - Firestore storage address for storing media - audio samples and photos (default "")

Contact ID

Another protocol used for communication with PCO is Contact ID. This older but still commonly used protocol allows the transmission of basic event types in the form of numeric codes. Thanks to its support, the server can also be used in environments where older technology is deployed.

Name: **ContactID**

Manual: SED_ContactID_Driver_1.3.0 and SED_Alarm_Table

FCM (Firebase Cloud Messaging

We use FCM technology for the purpose of notifications to mobile devices. This service enables secure and reliable message transmission directly to end devices, which is crucial for quickly informing operators or users about system events.

Name: **FCMProtocol**

Plugins

SED Server is primarily used to route communication between users and SED devices. The plugins are used to extend this basic functionality by, for example, storing multimedia on a local disk, event localization, event verification, generating user statistics, etc.

The list of plugins can be gradually expanded as the server functionality increases, or a plugin can be customised for the customer. The following is a list of plugins that are available by default.

Plugins can be activated, deactivated or set in the **plugins** section. Some settings are the same for all communication protocols

- **name** - name of the plugin
- **active** - this setting activates/deactivates the plugin.
- **microservice** - setting whether the plugin will be run in a separate process or if it will be part of the main server process. The main advantage of running in a separate process is higher system stability.
- **exclude_devices** - the plugin functionality will not be applied to devices with this id. This setting is used when, for example, you don't want to save multimedia from some devices to the local disk, etc.

Event Localization

If the same event is detected by at least 3 SED devices, the SED Server uses triangulation to calculate the exact location of the event (shooter in case of a gunshot, victim in case of a scream, etc.) based on the difference in time stamps.

To increase the localization accuracy, it is recommended to deploy the devices in the shape of an equilateral triangle with a side length of at least 80 m. The event location is suitable for larger areas such as squares or parks, but not for narrow streets, etc.

Name: **EventsLocalizationPlugin**

Record storage

This plugin is used to store multimedia and event metadata for future use. The plugin can store events on a local disk in a **~/SED/Data/EventsStorage** folder and/or in a Mongo database, which then serves as an event history for users in GUI applications - see SEDPresentation Manual v3.11 and SED Control Manual v3.11. Plans include storing data in Firebase Storage.

Name: **RecordsStoringPlugin**

Settings:

- store_local
- active - enables/disables storing media and metadata to local disk
- max_usage[MB] - maximum volume of data stored
- min_space_left[MB] - minimum disk space that must remain free
- store_alerts - activates/deactivates storage of media and alarm metadata
- store_events - activates/deactivates storage of multimedia and event metadata
- store_measurements - activates/deactivates storage of noise measurement metadata
- store_pictures - activates/deactivates the storage of images
- store_mongo
- active - activates/deactivates saving media and metadata to the mongo database
- collection_name - settings for saving the record to the database
- database_name - the name of the database to store the data
- uri - database address
- store_alerts - see store_local
- store_events - see store_local
- store_measurements **- see** store_local
- store_pictures - see store_local

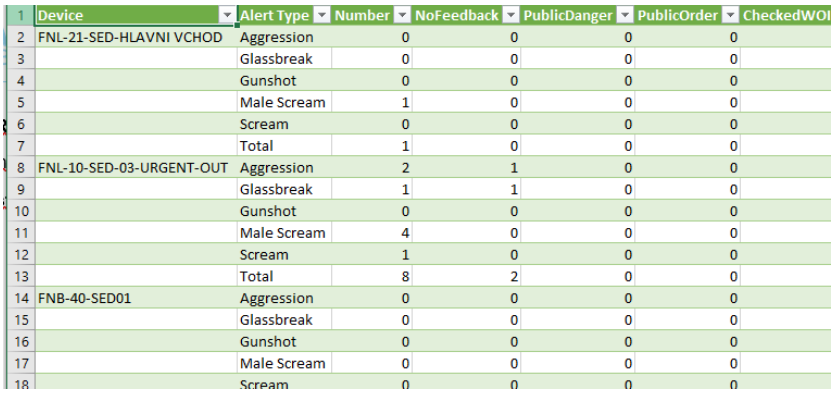
Feedback

This plugin is used to store user feedback on individual alarms. When the user checks in an alarm via **SEDPresentation**, the user assigns the alarm to one of the following categories:

- **Danger**: The alert indicated a dangerous situation, such as a shooting, an assault, panic, or something similar.
- **Incident**: A detected act of vandalism or disturbance, such as breaking glass, painting graffiti, unauthorised motor vehicle entry, or disturbing the peace at night.
- **No fault**: The sound was correctly detected as an alert but did not constitute a public order offence - for example, shouting during a game, breaking glass in a glass container, etc.
- **False Alarm**: An incorrectly triggered alert where the sound does not correspond to the detected event.

- **Other**: sounds that do not fall into any of the above categories - for example, a system test. A feedback note is mandatory for this type of feedback.

No feedback: This option is mainly for the case where the application is installed on multiple devices, and it is not necessary for multiple operators to enter feedback. Based on this feedback, alarms are further sorted to further refine the detection models. At the same time, monthly statistics are sent to the user, where he can see the number of alarms for each device, sorted by these categories.



	Device	Alert Type	Number	NoFeedback	PublicDanger	PublicOrder	CheckedWOI
2	FNL-21-SED-HLAVNI VCHOD	Aggression	0	0	0	0	
3		Glassbreak	0	0	0	0	
4		Gunshot	0	0	0	0	
5		Male Scream	1	0	0	0	
6		Scream	0	0	0	0	
7		Total	1	0	0	0	
8	FNL-10-SED-03-URGENT-OUT	Aggression	2	1	0	0	
9		Glassbreak	1	1	0	0	
10		Gunshot	0	0	0	0	
11		Male Scream	4	0	0	0	
12		Scream	1	0	0	0	
13		Total	8	2	0	0	
14	FNB-40-SED01	Aggression	0	0	0	0	
15		Glassbreak	0	0	0	0	
16		Gunshot	0	0	0	0	
17		Male Scream	0	0	0	0	
18		Scream	0	0	0	0	

Name: **EventsFeedbackPlugin**

Condition of the device

This plugin logs device statuses, logs the times when a device was connected or disconnected, and generates monthly statistics for users.

Device	State	Days	Time	Percentage
Městský park	Disconnected	0	00:01:30.582	0.00%
	Connected	27	23:57:17.402	100.00%
	Signal Lost	0	00:01:01.098	0.00%
Centrál	Disconnected	0	00:04:45.603	0.00%
	Connected	27	23:52:11.169	100.00%
	Signal Lost	0	00:02:52.258	0.00%
Hrnčířská	Disconnected	0	11:56:44.467	1.80%
	Connected	27	11:56:15.351	98.20%
	Signal Lost	0	00:06:49.157	0.00%

Name: **DevicesStatePlugin**

Alarm Verification

This plugin is used to verify incoming alarms to minimise the number of false alarms. Incoming alarms are verified based on their type by another (usually more sophisticated) detection model

The plugin configuration creates rules based on a numeric type that can be looked up in the Label column of the **SED_Alarm_Table** document.

Name: **AlarmsVerificationPlugin**

Settings:

- **Model_2F** - a list of rules for verifying alarms based on their type
- **active** - activates/deactivates verifications for the event type
- **exclude_devices** - devices excluded from verification for a specific event type. This way you can set the device to have active verifications for, for example, gunshot, but not for glass breakage
- **labels** - list of alert types for which this rule is valid
- **min_loudness_db** - minimum required event volume
- **min_margin_db** - the minimum required distance of the event from the ambient noise
- **required_confidence** - the minimum required level of confidence with which the verification model must verify the event
- **model_path** - path to the verification file from the root directory **~/SED/Config**

User and Device Configuration

The user and device configurations are located in the **~/SED/Config** folder as the *map.json* file. The configuration can be changed manually or by using the **SEDControl** application. Changes to the configuration are reflected immediately after saving, no need to restart the server.

Device Configuration

The SED Server handles the device uniformly regardless of whether it is a dedicated SED device or whether it is a software integration into a CCTV camera or mobile phone, or whether it is a simulation in a Windows application.

The device settings are in the **devices** section. The configuration contains device properties such as ID, description, coordinates, etc., which are displayed to the user within the user applications.

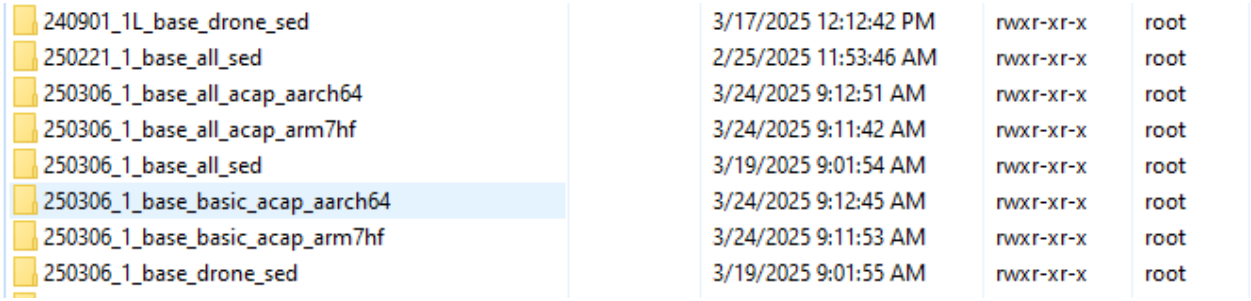
Properties:

- **device_id** - the device ID that is assigned by the administrator. This ID is used for communication with user applications and is used to uniquely identify the device. There cannot be 2 devices with the same ID in the system. The ID consists of 3 parts - Example 2052019
- **Main ID** [0 - 999].
- **Customer ID** [0 - 999]: Number of the customer or some larger administrative unit (see example: 52)
- **Object ID** [0 - 999]: Device number (see example: 19)
- **Serial** - the serial number of the device. The form of the serial number varies by device type, but uniquely identifies the device in the system.
- **location** - description of the location of the device. This information is usually used by the end user to navigate between devices.
- **latitude, longitude** - GPS coordinates of the device
- **model** - version of the detection model. When the model is changed, the device is automatically updated - for more information, see Update firmware
- **interior** - device coordinates when using custom maps
- **x, y** - position on the map
- **floor** - the floor where the device is located

- **building** - identification of the building in which the device is located

Update firmware

SED Server allows remote update of the device firmware. When a new version is released, simply upload the firmware folder to the *~/SED/Config/models* folder



The firmware name consists of the following parts:

- model - example: 250306_1_base
- SW version - basic, full, drone
- platform - sed, acap_arm7hf, heop_g7, etc.

To upload the firmware, just change the **model** property in the SED Control application or manually in the device configuration to the new firmware name. The SED Server detects that the name does not correspond to the current firmware on the device and updates it.

If a non-existent name is entered, then the update will not occur. It is important to select the firmware with the correct platform (in future versions, the server will select the platform automatically)

Configuring Users

The user configuration contains a list of users who have access to information to a certain extent. The extent of this access is mainly determined by the type of user and the number of devices assigned to them.

There are 3 types of users

- **Common User**: This type of user account belongs mostly to end users who have a set of devices to monitor. Its permissions are very limited. A common user can change his own password and description of the devices assigned to him.
- **Super User**: A Super User has the same permissions as a Common User and can also manage subordinate Common Users accounts. Managing subordinate

user accounts means that the Super User can create, edit, or remove accounts and select the set of devices that the subordinate user will have access to. Super User accounts usually belong to integrators who manage multiple end user accounts and assign devices to them based on their ownership.

- **Admin User**: The Admin user has rights to create, edit, or remove user accounts of all types and is the only type of user that can create or remove devices. The Admin User has access to all devices registered on the server and all user accounts. This user type should only be available to the server owner.

The device settings are in the **users** section. The configuration includes user properties such as ID, type, name, assigned devices, etc.

Properties:

- **user_id** - ID that uniquely identifies the user within the system. There cannot be 2 devices with the same ID in the system. The ID consists of 3 parts - Example 20520. For clarity, it is recommended that the MainID and CustomerID correspond between devices and assigned user accounts.
- **Main ID** [0 - 999]: Main part, the number of the administrator, for example distributor or integrator. SuperUser and its subordinate accounts always have the same Main ID (see example: 2)
- **Customer ID** [0 - 999]: Number of a customer or some larger administrative unit (see example: 52)
- **Personal ID** [0 - 9]: Personal number if the customer needs to have multiple accounts (see example: 0)
- **user_name** - user name
- **user_type** - type of user - see above
- **password_hash** - the hash of the user's password, which is used for authentication when authenticating the user.
- **devices** - list of assigned devices
- **protocols** - activated communication protocols of the user - for more, see Communication settings
- **latitude, longitude** - coordinates of the user's location (optional)

Communication settings

TBD