

# Watcher manual

11 December 2018

Introduction to Flussonic Watcher

System requirements for Watcher

Installing Watcher Single

Installing Watcher Cluster

Updating the Database Structure

Settings

Reset password

Failover

Motion Detection Events processing

License Plate Detection Events

Auto-login

User Interface

Mobile Applications

Interface customization

Email customization

Watcher API

User import API

Camera Import API

Client billing integration

Backend for user authorization

RADIUS authentication

API for mobile applications

Camera management

Creating a client mosaic

User Management

Groups

Sharing cameras

Serial Numbers

Import Cameras and Search

# Introduction to Flussonic Watcher

## Flussonic Watcher

is a complete video surveillance software system that works with a distributed IP camera network. This system can be used for video streaming, recording and managing video archives.

Flussonic Watcher is a scalable and ready-to-use system with flexible integration options. This web and mobile-oriented solution can solve various business tasks: from launching a corporate cloud video monitoring to a municipal video surveillance system that covers the entire city or state.

## Flussonic Watcher presentation

Internet providers — to launch a dedicated cloud-based video surveillance service and offer it to their customers for an additional fee. Watcher can be easily integrated with any provider and its existing billing system.

Management companies — for live video broadcasts from socially significant facilities and construction monitoring.

Production — to perform audio-video monitoring on factories, courts, polling stations, etc.

Municipal and federal projects — to provide free access to public cameras, as well as limited access to security organizations.

The control module is a component that has a database of all users and cameras, user access management, archive quotas, cluster servers, provisioning and camera activation, as well as APIs for integration with client billing or custom module development.

Web interface is the main interface for users and administrators. It works in all modern browsers and mobile devices. The interface includes a dashboard with cameras and archive, favorites, groups, map, user management tools and cameras, settings, interface branding tools and much more.

iOS and Android mobile applications give you access to cameras and user archive from mobile devices.

Firmware for cameras (agent) is an optional component. It can be installed on the cameras to provide access behind NAT with stable encryption and direct video delivery to Watcher.

Flussonic Watcher comes in two editions: Single and Cluster.

Single is good for smaller projects that have no more than 500 cameras and do not need interface branding (company logo, custom colors, etc.).

The Cluster edition is truly scalable with no limitations on the number of cameras. It also includes branding tools for adding a logo, corporate colors, captions, as well as the ability to organize a server cluster with backup tools.

# System requirements for Watcher

Flussonic Watcher works in two versions:

Single - Suitable for those who do not need branding of the interface (their logo, colors), for projects of up to 500 cameras, not more than 1 server.

Cluster - Advanced version. Includes branding tools (changing logo, color, text), the ability to expand to tens of thousands of cameras, including the ability to build a cluster of dozens of servers with redundant streams. This configuration requires a minimum of 2 servers. Suitable for those who do subscriber service or a large video surveillance system.

Management + streaming server:

Operating system: Ubuntu 14.04 or later, Debian 7 or later;

Database: PostgreSQL or SQLite;

CPU: CPU: Xeon E-3 1230v5 3.4 GHz and higher;

Memory: 32GB RAM;

Dedicated server: Yes.

Hard drive type: HDD / SSD;

Hard drive size: depends on the camera network size and video archive storage requirements.

10 GB of free disk space per 1 Mbps camera per day.

20 GB of free disk space per 2 Mbps camera per day.

70 GB of free disk space per 1 Mbps camera 7 days archive;

These system requirements are suitable for a network of:

500 1Mbps cameras

500 users

Failover turned off

Mosaic turned off

Or

250 2Mbps cameras

500 users

Failover turned off

Mosaic turned off

Servers should be dedicated, so no other software should run on them.

Management server (endpoint — Runs Flussonic Watcher and a DB):

Operating system: Ubuntu 14.04 or later, Debian 7 or later;

CPU: CPU: 2-core CPU;

Memory: 8Gb RAM;

Virtual server support: Yes;

Database: PostgreSQL;

Hard drive type: SSD;

Hard drive size: 64GB of free disk space;

Dedicated server: Yes.

Video streaming server (streamer — Video streaming and video archive storage):

Operating system: Ubuntu 14.04 or later, Debian 7 or later;

CPU: CPU: Xeon E-3 1230v5 3.4 GHz and higher;

Memory: 32GB RAM;

Dedicated server: Yes.

Hard drive type: HDD / SSD;

Hard drive size: depends on the camera network size and video archive storage requirements.

10 GB of free disk space per 1 Mbps camera per day.

20 GB of free disk space per 2 Mbps camera per day.

70 GB of free disk space per 1 Mbps camera 7 days archive;

# Installing Watcher Single

Watcher Single is suitable for small and medium-sized projects, where the maximum number of cameras does not exceed 500. In this mode, all cameras work with one server, where the database is installed, the web interface, streams and archive.

Flussonic Watcher works with two databases: SQLite and PostgreSQL.

Use SQLite only for tests.

This database is included by one command in the administrator interface or configuration file. Has limitations in the form of low speed and the inability to make a backup.

For full work, we recommend using PostgreSQL.

Contents

- The first installation method (for SQLite)

- The second method of installation (for PostgreSQL)

- Watcher Single — First run

## The first installation method (for SQLite)

This method is implemented through the administrator interface. To run Watcher Single:

- 1) Install Flussonic Media Server.
- 2) Go to admin panel: `http://flussonic:8080`.
- 3) Go to IP cameras.
- 4) In "Database path" specify the path to the database:  
`sqlite:///opt/flussonic/priv/vsaas.db`
- 5) Click Save.

At the address `http://flussonic:8080` the Flussonic Watcher interface will open.

To return to Flussonic Media Server admin panel, please visit `http://flussonic:8080/admin`.

## The second method of installation (for PostgreSQL)

This method is implemented using the console and configuration file.

To install PostgreSQL, in the console under root-access execute the command:

```
apt install postgresql
```

In the case of PostgreSQL, the user and the database need to be created independently. Type the commands in the console:

```
sudo -u postgres createuser -P vsaas
```

```
Answer: Enter password for new role: (enter super admin password)
```

```
Answer: Enter it again: (enter super admin password)
```

```
sudo -u postgres createdb -O vsaas -e -E UTF8 -T template0 vsaas_production
```

Answer: CREATE DATABASE vsaas\_production OWNER vsaas ENCODING 'UTF8' TEMPLATE template0

Go to the IP cameras in the administrative interface and specify the path to the database in "Database path" section.

Do not forget to replace vsaas\_password with a real password of the vsaas user that you created above.

At the address <http://flussonic:8080> the Flussonic Watcher interface will open.

To return to Flussonic Media Server admin panel, go to <http://flussonic:8080/admin>.

## Watcher Single — First run

Enter <http://flussonic:8080/> in your internet browser address bar to bring up an administrator control panel.

On the very first launch, the system will ask you to enter an email and come up with a password to create the first administrator account.

After you add the first administrator, you will see the Camera Control page, where you can add cameras and manage system settings.

# Installing Watcher Cluster

Flussonic Watcher works in cluster mode as a service. The cluster mode requires at least two servers to operate.

The server configuration in a cluster is the following:

- The control server (endpoint). It has Flussonic Media Server, the web interface to Watcher, the business logic, and the PostgreSQL database engine.

- Streaming servers (streamers). A streamer is a server machine that has Flussonic Media Server installed. It stores DVR archives and handles camera streams. You can add from 1 to 100 streaming servers.

The following picture shows the parts of Watcher cluster and the flow of video streams:

Flussonic Watcher in cluster mode supports only the PostgreSQL database engine. It initializes the database on the first run.

Prerequisite: Flussonic Media Server must be installed on the managing server and on all the streamers.

To install Watcher in a cluster:

- Install PostgreSQL on the managing server and specify the path to it in the administrator UI of Flussonic Media Server.

- Open the Watcher's administrator UI and create a user with administrator privileges.

- In the Watcher's administrator UI create a cluster and add streamers.

All these steps are described later on this page.

## Comparing Watcher Cluster with Watcher Single

The cluster mode is suitable for medium and large projects with unlimited number of cameras (more than 500, limited by hardware).

Larger projects require cluster organizations and stream reservation tools. In this mode, cameras work with two and more servers.

## Installing PostgreSQL

Before you start installing Watcher Cluster, install Flussonic Media Server and then launch PostgreSQL.

To install PostgreSQL is necessary in the console under root-access to execute the command:

```
apt install postgresql
```

In the case of PostgreSQL, the user and the database need to be created independently. Type the commands in the console:

```
sudo -u postgres createuser -P vsaas
```

Answer: Enter password for new role: (enter super admin password)

Answer: Enter it again: (enter super admin password)

```
sudo -u postgres createdb -O vsaas -e -E UTF8 -T template0 vsaas_production
```

Answer: CREATE DATABASE vsaas\_production OWNER vsaas ENCODING 'UTF8' TEMPLATE template0;

Go to the IP cameras in the administrative interface and specify the path to the database in "Database path" section.

Do not forget to replace vsaas\_password with a real password of the vsaas user that you created above.

Specify mode cluster option in the configuration file /etc/flussonic/flussonic.conf:

```
vsaas {  
    database postgresql://vsaas:vsaas_password@localhost/vsaas_production;  
    endpoint enabled;  
    mode cluster;  
}
```

Save and restart Flussonic Media Server with the command:

```
service flussonic restart
```

At the address <http://flussonic:8080> the Flussonic Watcher interface will open.

To return to Flussonic Media Server admin panel, please visit <http://flussonic:8080/admin>.

IMPORTANT! To work correctly, Watcher must be added to the DNS record hostname Watcher server.

After installation, the administrator will be created.

## Admin creation

Enter <http://flussonic:8080/admin> in your internet browser's address bar to bring up an administrator control panel. On the very first launch, the system will ask you to enter an email and come up with a password to create the first administrator account.

Next, go to Settings — Streamers — Edit and create a cluster.

## Create a cluster

Log in as administrator, then go to Settings — Streamers — Edit. This settings page is essential for the entire system, and here you can create a cluster. Add at least one stream server to make cameras function.

Prior to adding a new streamer in the Watcher settings, install Flussonic Media Server on a separate server with an external IP-address. Also, enter cluster\_key that matches the Watcher key.

When the camera server is ready to go, create a new streamer in Settings — Streamers — Edit, and then fill out the New streamer:

Hostname — Streamer's URL. (e.g., 192.168.1.2:8080, or a domain name).

Cluster key — Cluster key. If the streamer's and Watcher cluster keys are the same, there is no need to fill this field.

The path to the archive — It is important to fill in this field, otherwise the archive will not work (e.g., /dvr).

When you have multiple streamers, you can select a default one. Click on the streamer from the list and click Set as default. In this case, all new cameras will be added in the default stream server.

Important!

In section "Streamers", you don't need to add a host that has Watcher deployed.

All servers must have the same time.

You can enable streams reservation for each streamer (failover).

How to do it see this section «Failover».

## Updating Watcher

Important. Whenever you need to install Flussonic Watcher updates, it's necessary to migrate your database. Therefore we recommend that you back up your database before updating Watcher.

# Updating the Database Structure

Some Flussonic Watcher updates include structural changes to the database. In this case, you will see the following message:

The script `/opt/flussonic/contrib/watcher_db_migrate.sh` adds changes to the database structure.

Run this script on the server side manually.

Here is an example of the success log:

```
# /opt/flussonic/contrib/watcher_db_migrate.sh
INFO [alembic.runtime.migration] Context impl SQLiteImpl.
INFO [alembic.runtime.migration] Will assume non-transactional DDL.
INFO [alembic.runtime.migration] Running upgrade dfd74e510414 -
> 1822b8f25e20, agent:model, agent:camera
INFO [alembic.runtime.migration] Running upgrade 1822b8f25e20 -> 1a71a9477bbb, streamer: cluster_key
INFO [alembic.runtime.migration] Running upgrade 1a71a9477bbb -
> 7a3ab2550cab, streamer_fkey cascade
Restarting Watcher
```

After the Flussonic Watcher restart, you will be able to use the Watcher GUI.

If while running `watcher_db_migrate.sh` will be similar error to:

```
alembic.util.exc.CommandError: Can't locate revision identified by 'ebdce5515b6d'
```

Then install previous version (version before update/downgrade) and run:

```
cd /opt/flussonic/apps/vsaas
/opt/flussonic/bin/python -m manage db downgrade
```

Important! Backup your database before running `watcher_db_migrate.sh` script.

The database type and location is stored in the configuration file `/etc/flussonic/flussonic.conf`.

SQLite:

```
vsaas {
    database sqlite:///opt/flussonic/priv/vsaas.db;
}
```

Copy file `/opt/flussonic/priv/vsaas.db`.

PostgreSQL:

```
vsaas {
    database postgresql://vsaas:vsaas_password@localhost/vsaas_production;
}
```

Make a reserve copy using the `pg_dump` utility:

```
pg_dump -h localhost -U vsaas -d vsaas_production > /path/to/backup/vsaas.sql
```

Ответ: Password: enter password `vsaas_password`, that you specified for vsaas user

# Settings

Administrator setting is available Flussonic Watcher.

Operating mode – a multi-server cluster mode or a plugin mode that works on one server.

Streamers - become active in a cluster mode.

Archive path — the archive path storage. It can be a Swift-storage, available in a standalone mode, with a single server launch. In the cluster mode, the DVR path can be configured in the stream server settings.

API key — token used for mobile access. You need to use it on the site Erlivideo to activate access to mobile applications.

Guest access – allow to log in unregistered users.

Registration allowed — turn on the self-registration option.

Homepage — select what users will see upon login: map or dashboard.

Guest Homepage — select what guests will see: map or dashboard.

Dashboard — select what items to show on a dashboard: camera or groups.

External Authentication – External authorization. Specify http address or address of RADIUS server.

Show map in navigation — show or hide the map from the side menu.

Map Center — geographic coordinates for map centering.

Map Language – map language.

Map zoom — set the default map zoom.

Map Provider — select the map provider: Google, Openstreetmaps or OpenStreetMaps Offline.

Cameras serial numbers management – control camera serial IDs in Flussonic Watcher. It is necessary when you use Agent.

Custom header text — more in the section «Interface customization»

Custom page title — more in the section «Interface customization»

Custom logo — more in the section «Interface customization»

Custom color theme — more in the section «Interface customization»

# Reset password

A user can use the

Forgot password

option. A password recovery link will be automatically sent via email. Read how to configure an SMTP server.

An administrator can change the user password with the `reset_password` utility.

Usage:

```
reset_password
```

Example:

```
/opt/flussonic/contrib/reset_password email.address@example.com pwd123xyz
```

You will see this message if everything went OK:

```
Changing password for email.address@example.com
```

# Failover

A failover cluster is a group of servers that work together to maintain the overall service stability and exclude any possible downtime of any part of the system. If one of the servers fails, another cluster server takes over its workload. This process is called failover.

In case of a stream server failure, the camera traffic to this server will automatically redistribute between other cluster servers, also called donors. The video archive becomes unavailable on the stream server that had a failure. The new video archive will be temporarily recorded on the donor server that is being used.

When the connection to the server that had an issue is renewed, the traffic is automatically redirected back. Provided that the storage on the primary server wasn't damaged during the failover, the access to the main video archive will be established. However, the temporary video archive will be deleted from the donor server.

Turn on the failover option for each stream server individually in the "Streamer" section and change mode in `/etc/flussonic/flussonic.conf` file on Watcher server:

```
vsas {  
    mode cluster+failover=30;  
    ...  
}
```

Where 30 - streamer check interval in seconds. To apply new settings restart Watcher:  
`/etc/init.d/flussonic restart`

# Motion Detection Events processing

## Motion Detection Events processing

Flussonic Watcher can receive events by email, for this purpose it has a special SMTP server.

It receives emails with motion events from cameras and add corresponding marks in the archive recordings in the places when motion was detected.

You can configure motion detection in two steps:

- 1) Enable receiving emails in Watcher.
- 2) Configuring a camera to send emails.

How motion recording works:

Flussonic Watcher maintains a permanent recording with the specified recording depth. When an e-mail arrives with traffic, Flussonic Watcher marks the time interval in the database to reflect the tags in the archive player.

The most important! The record is protected from deletion.

The duration of a protected recording is determined by two timestamps, the first of which is calculated as the current time minus 10 seconds, and the second timestamp is the current time plus 30 seconds.

You just need to set the depth of the archive: for example, 6 hours. Then enable the reception of events.

You will receive 6 hours of continuous archive + sensor entries will be stored as long as there is free space on the disk.

Recording new events will delete the old ones.

Calculating the necessary disk space based on the bit rate of the cameras and the frequency of movements, you can save up to 50-90% of the disk space compared to the normal recording without events.

## Enabling event capture in Flussonic Watcher

Add the lines with the camera\_alarm plugin into /etc/flussonic/flussonic.conf:

```
plugin camera_alarm {  
    catch motion;  
    listen smtp://0.0.0.0:1025;  
}
```

The catch parameter specifies the word that Flussonic Watcher will search for in the subject of the message. Most cameras send letters with the default subject like this: "Camera 123 Motion Detected at 14:21 27-10-2017".

If your camera sends a letter with a different subject or has the possibility to specify your own subject, then you configure catch as you like.

The listen parameter specifies the interface and port for the built-in SMTP server. You can set login and password for SMTP:

```
listen smtp://username:password@0.0.0.0:1025;
```

Then restart Flussonic Media Server.

The link to the event list will appear on the page of cameras control.

## Configure a camera

The next step requires that you set email notifications for Flussonic Media Server in the camera interface.

Important! Specify a sender and recipient as a full camera names (camera name and ID). The full name can be found in the Watcher UI.

Example: cam1-abcdefg@example.com, where cam1 is a cam's name in Watcher and abcdefg is an ID in Watcher.

The motion marks will appear in the camera archive interface.

# License Plate Detection Events

Flussonic can detect license plates and recognize Russian car numbers on the video recorded by an IP camera. This functionality is known as ANPR (automatic number plate recognition).

ANPR supported only in the new UI for Watcher, starting from version 18.11. To configure the feature, use the web interface and the configuration file, but some options can be set only through the file.

Then, you can view registered events in the Watcher UI and watch the recorded video of each event.

## Pre-requisites

First, prepare hardware and software for the Flussonic server (streamer) that will carry out number recognition. Learn more in Flussonic documentation.

## Setting up ANPR in the UI

### Turning on plate detection for a camera

Important. The camera for which you want to recognize car numbers must be connected to a streamer with at least one GPU NVIDIA with at least 3 Gb video memory. Learn more in Flussonic documentation

Before you configure cameras, enable the recognition system. Add the line `plugin vision;` to the configuration file `/etc/flussonic/flussonic.conf`.

To turn on plate detection and number recognition for a camera:

In the Watcher UI, go to Cameras. Find the camera in the list and open its settings by clicking the icon in the upper right corner of the player.

Select the ANPR check box and click Save.

Now Flussonic will recognize car numbers that appear in the frame of this camera, and mark the time when the car entered and left the scene.

Flussonic modifies the stream settings in the configuration file `/etc/flussonic/flussonic.conf`.

You may need to edit manually the GPU number. Details

## Viewing ANPR events in Watcher

Flussonic creates events of two types:

enter – a car number appeared in the field of view of a camera

leave – a car number left the field of view.

To see detected car numbers for a camera:

In the Watcher UI, go to Notifications. The list of all events opens.

To find certain events, use filtering and search capabilities on the right:

In Source, select Plate detector.

In From and To, select the date and time of beginning and end of the period when events were detected.

In Search, type the car number.

To delete the specified search criteria, click Clear Filter.

In Source, select Plate detector.

In From and To, select the date and time of beginning and end of the period when events were detected.

In Search, type the car number.

To delete the specified search criteria, click Clear Filter.

The list of events is filtered as you enter search parameters.

To view the recording of an event, click in the line with this event. The player will appear on the bottom right to play the recording. To enlarge the player, just double-click it.

# Auto-login

Flussonic Watcher allows its users to login by a special URL (auto-login link), without entering a password. This may be necessary if you want to simplify the access to Flussonic Watcher for your users or prevent the transfer of passwords to third parties.

The auto-login link is issued to an authorized client.

In order to generate a URL for auto-login, you must first request a token by using the link `/vsaas/api/generate-autologin-token`. After that, the user can be authorized via POST to `/vsaas/autologin`.

Follow the steps:

1) First, generate an administrator's token. To do so, you'll need the KEY API. The KEY API is shown in the Flussonic Watcher administrative interface (admin panel, Watcher UI) on the Settings page.

```
sign = md5(salt + ":" + utc + ":" + api_key)
```

```
admin_token = salt + ":" + utc + ":" + sign
```

Where:

salt — any random string;

utc — the current UTC time in seconds;

api\_key — API key from the settings page of the Flussonic Watcher admin panel.

The generated token has a finite lifetime and is bound to the specified time in UTC. For example, if

```
salt = "20a666"
```

```
utc = "1487258700"
```

```
api_key = "HELLO"
```

then the final resulting admin\_token will be: "20a666: 1487258700: 4b60f36de708e5b3472155db2fea990a"

2) So, the admin token is ready, now you need to get a token for the autologin of a particular user. To do this, you'll need to make a POST request in JSON-format:

```
curl --header "X-Vsaas-Api-Key: ADMIN_TOKEN" --header "Content-Type: application/json" --request POST --data '{"login": LOGIN, "valid_till": VALID_TILL, "lifetime": LIFETIME}' "http://watcher.com/vsaas/api/generate-autologin-token"
```

ADMIN\_TOKEN — the token generated in step 1. This token is transmitted in the HTTP header `X-Vsaas-Api-Key`.

LOGIN — login (the same as an email) of the user to whom you want to give access. Line. Required.

VALID\_TILL — the UTC time in seconds until this token is valid for autologin. Integer. Optional parameter.

LIFETIME — the duration of the session opened through the autologin, in seconds. Integer. Optional parameter.

The response will be in the JSON format:

```
{
  "autologin_token": "demo:1487258314:f8b1:b4bdaac58cbe94638e5b14a3728b8e6d633f3c6e",
  "success": true
}
```

```
}
```

You'll need this `autologin_token`.

3) The `autologin_token` received in step 2 is used for the POST request to Flussonic Watcher. For example, this way:

At the click on the submit button, the user will be logged in automatically into the Flussonic Watcher web interface.

# User Interface

A user can log into Flussonic Watcher using an email and password after adding to the system.

New users can access cameras that were assigned to them by an administrator. Cameras may be public, available after authorization, and private.

A user interface has two screens: the first one shows cameras on a map and the second one features a list of cameras.

The following menus are available:

- Dashboard.

- Map.

- Favorite.

- Cameras.

- Users and Groups. (For Administrator)

- Settings. (For Administrator)

## Dashboard

Dashboard — is a primary webpage for users to access cameras.

The following access filters are available:

- Any.

- Public.

- Authorized.

- Private.

Camera display modes:

- Medium Thumbnails.

- Large Thumbnails.

- List.

You can also toggle Compact and Hide inactive cameras.

The list below shows automatically updated screenshots from all cameras. In order to watch a certain camera or show its video archive, click on the respective image or Play button.

You can also click on the little arrow button on the top right corner of each block to access the video archive. To open camera in full screen, click on a camera name:

Mosaic shows up to 8 selected cameras on a single screen. To use a multi-camera view, create a client mosaic in the Cameras section.

## Map

The map shows cameras that have coordinates configured.

## Favorites

A user can mark a camera with a star. After that, it'll show up in the «Favorite» section.

## Cameras

Here you can perform all operations with cameras:

- Add a camera.
- Camera search via ONVIF.
- Import Flussonic Cameras.

Also you can do the following:

- Export a list of files in CSV format.
- View camera serial numbers list.
- Create a Mosaic.

More information on adding cameras: «Camera management» section.

## Users and Groups

In this menu, you can add and modify Flussonic Watcher users. For more information see «User Management».

Use groups to simplify user and camera navigation by making logical grouping (for example, by floors, departments, regions, neighborhoods, etc.), as well as simplify the user right management to provide camera visibility. See the «Groups» section for more information.

## Settings

This menu is used to configure the Flussonic Watcher. More in the section «Settings».

# Mobile Applications

## IMPORTANT!

Mobile applications need to have the address of your Watcher system to work properly. Get your Operator ID that stores the URL of your Watcher system beforehand.

By default, the application tries to reach an Erlyvideo server, so a mobile app can't authenticate without a personal Operator ID.

To get your Operator ID, log in to your account, click on the installation key on the main portal screen and specify the public Watcher URL in the following format: `http://APIKEY@watcher-host`. The host address should be discoverable from the Internet. It's recommended to use a domain name, not just IP-address.

To start using the application, download it from the Apple Store or Google Play, and then authenticate using your Watcher login, password and the Operator ID that you received from `manage.erlyvideo.org`.

**APIKEY** — an authentication key that you can obtain in the Watcher settings UI.

**Watcher-Hostname** — a publicly visible path to Watcher.

**IMPORTANT!** Make sure that your server is visible from the Internet and that it has a white IP-address. If your server is using NAT or a Firewall, configure the 1935 port forwarding to the server (RTMP protocol), so the application can work.

# Interface customization

The Watcher Cluster version provides all necessary tools for interface customization. Here is what you can do from the

Settings

menu:

Custom header text. Add arbitrary text to the title of the interface.

Custom page title. Change the title.

Custom Logo. File requirements: .png image, height 50 px. A white or gray-colored logo with a transparent background will work the best.

Login page custom logo.

Custom color theme. You can select from three demo templates or fine-tune CSS. Save your custom CSS rule-sets to the file `/vsaas/static/css/user.css`. The file `user.css.example` can be used as an example.

Also you can customize the email password recovery template. Read more in the article [Email customization](#).

# Email customization

## Setting the SMTP server

Configure your mailbox settings to make password recovery email reach your users. Add the mail server parameters to the config file.

For example, use these parameters below:

E-mail: email.address@example.com;

Password: xyz123;

Mail server address: smtp.example.com;

Port: 465;

Connection security: SSL

Config file:

```
vsaas {  
  database sqlite://opt/flussonic/priv/vsaas.db;  
  smtp_server smtp.example.com:465;  
  smtp_login email.address:xyz123;  
  email_from "Flussonic Watcher ";  
  smtp_opts ssl;  
}
```

## Forgot Password Email Template Change

You can change the template for password recovery email. Follow these steps:

Go to /opt/flussonic/apps/vsaas/watcher/templates.

Use the password\_reset\_request.email and password\_changed.email text files as an examples.

Add the custom\_ file prefix to your custom template versions:

custom\_password\_reset\_request.email and custom\_password\_changed.email.

If you want to use custom hypertext templates, add the .html extension to these files as follows:

custom\_password\_reset\_request.email.html and custom\_password\_changed.email.html. Save these files in the same directory.

The template consists of two parts, the header and the body of the. You can also add a subject to the header.

In addition, you can use variables {{data.base\_url}} and {{data.token}} in the message body.

Text template example:

custom\_password\_reset\_request.email:

---

subject: "ABC surveillance password reset"

---

Thank you for using the forgot password option. Follow this link to reset your password to the ABC surveillance system:

{{data.base\_url}}/vsaas/forgot-password/{{data.token}}

HTML template example:

custom\_password\_reset\_request.email.html:

---

subject: "ABC surveillance password reset"

---

Thank you for using the forgot password option.to the ABC surveillance system:

Reset your password

# Watcher API

You can integrate the Watcher video surveillance system into your system and create custom mobile applications using the flexible API.

Flussonic Watcher API let you import or export users and cameras individually or in bulk. API has the advanced integration options with client billing. It allows you manage the availability of cameras, private archives and billing, change passwords, check camera status and solve other tasks that are available in the GUI.

Also, the API let you configure authentication via RADIUS-server or use an authorization backend.

User Import API    Camera Import API    integration with billing    Change password    Backend for  
user authorization    RADIUS authorization

# User import API

You can import users and settings from third-party systems, databases or spreadsheets using API. It requires API authorization. To do that, include the APIKEY that can be found in the Watcher settings in the

`x-vsaa-api-key`

HTTP header.

To import a list of users, create a CSV file and send it to:

`http(s)://YOUR_WATCHER_URL/vsaa/api/import/users`

CSV file fields:

`email` — the main user ID for authentication that is also used for password recovery.

`password` — plain text password.

`is_active` — 1: active user; 0: blocked user.

`is_admin` — 1: administrator user; 0: regular user.

`groups` — a list of groups to which the user belongs. Delimiter is a semicolon ';'.  
Example: `user1;grp1;grp2`

`note` — a comment to this user.

Command-line import:

```
curl --data-binary @mydata.csv -H 'Content-type:text/plain; charset=utf-8' -H 'x-vsaa-api-key: API_KEY_БАШЕГО БОТЧЕРА' http://WATCHER-HOSTNAME/vsaa/api/import/users
```

mydata.csv example:

```
email,password,is_active,is_admin,note,groups
ivanov@domain.tld,CergitMig,1,0,user1,grp1;grp2
petrov@domain.tld,LajQuolOy,0,1,user2,grp2
```

One string example:

`echo -`

```
e "email,password,is_active,is_admin,note,groups\nivanov@domain.tld,CergitMig,1,0,user1,grp1;grp2\npetrov@domain.tld,LajQuolOy,0,1,user2,grp2" | curl http://127.0.0.1:8080/vsaa/api/import/users?type=csv --data-binary @- -H 'Content-type:text/plain; charset=utf-8' -H 'x-vsaa-api-key: 3a7d9386-6c3a-440d-a75d-e6b3fdc3368e'
```

Response:

```
{"success": true}
```

# Camera Import API

```
dd { font-style: italic; /* cursive <em>/ } dt { margin-top: 1em; /</em> top margin */ }
```

A POST request is used to import the cameras to the following URL:

http(s)://YOUR\_WATCHER\_URL/vsaas/api/interchange/usercameras

Console:

```
curl http://127.0.0.1:8080/vsaas/api/interchange/usercameras --data-binary @mydata.csv -H 'Content-type:text/csv' -H 'x-vsaas-api-key: '
```

mydata.csv example:

```
stream_url,substream_url,thumbnails,onvif_url,onvif_profile,ptz,dvr_depth,dvr_path,enabled,access,title
rtsp://127.0.0.1:554,,,http://127.0.0.1:8899,000,0,3,storage,1,private,office_cam1
rtsp://127.0.0.2:554,,,http://127.0.0.2:8899,000,0,3,storage,1,private,office_cam2
```

One string example:

echo -

```
e "stream_url,substream_url,thumbnails,onvif_url,onvif_profile,ptz,dvr_depth,dvr_path,enabled,access,title\nrtsp://127.0.0.1:554,,,http://127.0.0.1:8899,000,0,3,storage,1,private,office_cam1\nrtsp://127.0.0.2:554,,,http://127.0.0.2:8899,000,0,3,storage,1,private,office_cam2" | curl http://127.0.0.1:8080/vsaas/api/interchange/usercameras?type=csv --data-binary @- -H 'Content-type:text/csv' -H 'x-vsaas-api-key: 3a7d9386-6c3a-440d-a75d-e6b3fdc3368e'
```

Response:

```
{"zu": 0, "cameras": [{"name": "office_cam2-689f1b1548", "created": true}, {"name": "office_cam1-c0ce3faa10", "created": true}], "users": [], "success": true, "zc": 2}}
```

CSV or JSON fields:

title: camera name.

name: stream name. Default is 'title' with a random suffix.

static: 1: constantly working stream, 0: on-demand stream.

stream\_url: main stream RTSP URL.

substream\_url: secondary stream RTSP URL (for multi-bitrate). Turned off by default.

thumbnails (string): camera snapshot URL. Send 0 instead of the URL to reset thumbnails. If you don't know the camera snapshot URL, use 1. Watcher will turn it on automatically.

Note: we recommend that you use a direct URL. Otherwise, the server load will increase at 10%. If you specify the URL, a direct communication with the camera is established, saving your server processing power.

onvif\_url: camera URL to access via the onvif protocol. By default, it's set to 'no.'

Onvif\_profile — UNFIV profile.

onvif\_profile: ONFIV profile

ptz: (0 or 1) — turn PTZ off/on (if camera supports).

access: camera access type (private / public / authorized). Public will be accessible to all users, private — to a camera owner only. Private is a default value.

owner: camera owner login.

enabled: (0 or 1) turns camera on / off.

dvr\_path: path to save the archive. Default is no archive.

dvr\_depth: (integer in days) — number of days to store the archive. 0 — disables the archive.

coordinates: geographical coordinates.

postal\_address: camera's postal address.

comment: camera comment.

agent\_model (string): camera model.

agent\_serial (string): camera serial number.

agent\_id (string): camera agent unique number.

agent\_key (string): special field used for Watcher authorization.

agent\_pin (string): special field used for Watcher authorization.

# Client billing integration

Camera provisioning automation (i.e., billing redirect) requires a firmware update from Flussonic Agent. Flussonic Agent is an optional module within the Watcher software system that automates the camera activation process through the client billing. This module establishes a secure tunnel between cameras and Watcher that allows connecting through NAT, and also it improves the video transport stability. Firmware modification can be customized for each client's camera model.

Learn how Agent works

When the very first connection is established between the camera and Agent, a request is sent to the provisioning server (i.e., activation server). It redirects Agent to the client's server that has Watcher up and running.

After sending a request to the activation server, Agent generates a CSV and exports it to a client billing system. The export URL should be predefined.

The list of CSV attributes, sent from the Flussonic activation server to the client billing:

agent\_model (string) - camera model.

agent\_serial (string) - camera's serial number.

agent\_id (string) - Agent UID on the camera.

agent\_key (string) - special field used for Watcher authorization.

agent\_pin (string) - special field used for Watcher authorization.

stream\_url (string) - primary stream's RTSP-URL.

substream\_url (string) - secondary stream's RTSP-URL.

thumbnails (string) - URL snapshots from the camera.

onvif\_url (string) - URL at which the camera can be accessed via onvif protocol.

onvif\_profile (string) -service field.

ptz (0 or 1) - turns PTZ on / off.

The activation server sends information to the client billing to create new objects for each camera. It will append new cameras to others currently registered in the billing system. Another camera registration system can also be used instead of the standard billing. The agent\_serial parameter is used for camera registration.

After the registration, other attributes can be added per camera object on the client billing side.

Client-side billing CSV Attributes:

owner (string) — login of the connected account.

dvr\_depth (integer) - number of days to store video archives for the camera. 0 — disables the archive.

enabled (0 or 1) - turns camera on or off.

access (private / public) - camera visibility. It can be either accessible by the camera owner or my all users.

Note: These attributes are not transmitted.

The user should develop the mechanism of billing plans and other logic that will affect access to the file and settings, and to a certain video archive's depth.

After you apply these settings (owner, dvr\_depth, enabled, access), the data on all cameras and users will be automatically pushed to a CSV file here by the following:

[http\(s\)://YOUR-WATCHER-URL/vsaas/api/interchange/usercameras?x-vsaas-api-key=\(obtain the key in the Watcher settings\)](http(s)://YOUR-WATCHER-URL/vsaas/api/interchange/usercameras?x-vsaas-api-key=(obtain the key in the Watcher settings))

User information also should be exported along with the camera information. User CSV Attributes:

[http\(s\)://YOUR-WATCHER-URL/vsaas/api/import/users](http(s)://YOUR-WATCHER-URL/vsaas/api/import/users)

login (string) — the user's Watcher login.

is\_active (0 limits watcher access) — cameras work, but the user can not enter Watcher.

dvr\_allowed (0 limits video archive access) — cameras write to the archives, but users cannot have access to them.

It's possible to add cameras with users before activation. The process is described below:

1) Add a camera to the client billing (or another registration system) as an item with the attribute: agent\_serial (serial number on a camera box). The following attributes should be set on the camera purchasing step:

owner — client login

dvr\_depth — archive depth

enabled — camera status

access — access type

2) Then, when a technician turns on the camera in the client's home, the camera sends a request to the activation server, receives necessary activation attributes, then pushes the gathered information to the CSV file in the client billing. It will append new items based on the agent\_serial attribute for previously added cameras.

3) When you have the camera information, it will be updated automatically.

# Backend for user authorization

How it works:

An operator needs to realize the HTTP request handler, that has the logic to authenticate users.

The operator enters the path to the Flussonic Watcher authentication backend (Settings — Authentication backend).

A user logs into Flussonic Watcher using login/password.

Watcher transfers this data to the backend operator in a request payload.

Backend checks the incoming data.

If the user can be authenticated, the system returns the 200 HTTP code.

If the authentication data is incorrect, it returns the 403 HTTP code.

If the user is not found, the system gives 404.

Along with the login permissions, the backend can return the list of gorups in the JSON body.

Flussonic Watcher checks the information about the groups in the database.

If the permission is granted, the password is passed as an coded hash string and the group belongings are checked.

In case of decline, the user password is reset.

If the authentication backed was out of reach or send the response less than 2 seconds. The user check if being done in the Watcher's database.

```
import falcon, json
```

```
class AuthResource:
```

```
    def on_get(self, req, resp):
```

```
        print "GET %r\n%r" % (req.uri, req.params)
```

```
        login = req.params.get('login', None)
```

```
        password = req.params.get('password', None)
```

```
        if not login or not password:
```

```
            print 'incorrect request login: %r, pass: %r' % (login, password)
```

```
            resp.status = falcon.HTTP_400
```

```
            return
```

```
        if login == 'user0':
```

```
            if password == 'letmein':
```

```
                return
```

```
            resp.status = falcon.HTTP_403
```

```
            return
```

```
        if login == 'user1':
```

```
            if password == 'letmein':
```

```
                resp.body = json.dumps(dict(groups=['a', 'b']))
```

```
                return
```

```
            resp.status = falcon.HTTP_403
```

```
            return
```

```
resp.status = falcon.HTTP_404
```

```
app = falcon.API()  
ad = AuthResource()
```

```
app.add_route('/auth', ad)
```

## Examples

A user can get through with groups A and B:

```
curl -vvv http://localhost:8001/auth?login=user1&password=letmein
```

```
* Trying 127.0.0.1...  
* Connected to localhost (127.0.0.1) port 8001 (#0)  
> GET /auth?login=user1&password=letmein HTTP/1.1  
> Host: localhost:8001  
> User-Agent: curl/7.47.0  
> Accept: */*  
>
```

A user can get through without groups:

```
curl -vvv http://localhost:8001/auth?login=user0&password=letmein
```

```
* Trying 127.0.0.1...  
* Connected to localhost (127.0.0.1) port 8001 (#0)  
> GET /auth?login=user0&password=letmein HTTP/1.1  
> Host: localhost:8001  
> User-Agent: curl/7.47.0  
> Accept: */*  
>
```

A user can't get through:

```
curl -vvv http://localhost:8001/auth?login=user0&password=wrong
```

```
* Trying 127.0.0.1...  
* Connected to localhost (127.0.0.1) port 8001 (#0)  
> GET /auth?login=user0&password=wrong HTTP/1.1  
> Host: localhost:8001  
> User-Agent: curl/7.47.0  
> Accept: */*  
>
```

A user is not found:

```
curl -vvv http://localhost:8001/auth?login=user10&password=wrong
```

```
* Trying 127.0.0.1...  
* Connected to localhost (127.0.0.1) port 8001 (#0)  
> GET /auth?login=user10&password=wrong HTTP/1.1
```

> Host: localhost:8001  
> User-Agent: curl/7.47.0  
> Accept: \*/\*  
>

# RADIUS authentication

RADIUS server can be used to authenticate Watcher users. It is especially useful, if you have a large number of users.

The setting can be enabled via admin interface:

The address 'radius://ldap.example.com:1812/secret' consists of 3 parts: host, port and secret. Change it according to your RADIUS server settings. Now, when a user try to login, Watcher redirects to the server via RADIUS protocol. Watcher sends User-Name and User-Password in the Access-Request query. The RADIUS server responds by giving the group list for this user. An attribute Filter-Id(11) is used here. Each group is stored in a separate attribute.

Watcher redirects to RADIUS on every user log in.

If the RADIUS answers Access-Accept, Watcher logs user in and saves the HEX password and group belongings to the database.

If the RADIUS answers Access-Reject, the user becomes locked in the database.

If the RADIUS did not answer, Watcher searches a user in the database.

It is necessary to bear in mind that RADIUS should know about all users, including administrators. The administrator user attribute can not be transferred to the RADIUS response and it can be assigned through Watcher only.

# API for mobile applications

POST: /vsaas/api/login

Example:

```
`curl -H 'Content-Type: application/json' -  
d '{"email": "email@example.com", "password": "passwwd"}' http://localhost:8080/vsaas/api/login`
```

Parameters:

HTTP request payload  
{"email": , "password": }

Reply: JSON

```
{  
  success: boolean,  
  session:  
}
```

## Adding a camera to Favorites

POST: /vsaas/my/fav/cameras

Parameters:

HTTP request payload  
{"id": } // camera id

Reply: JSON

```
{"success": true}
```

## Removing a camera from Favorites

DELETE: /vsaas/my/fav/cameras/{cam\_id}

Parameters:

cam\_id camera id

Reply: JSON

```
{"success": true}
```

## All the available cameras

GET: /vsaas/api/my/cameras

Parameters:

HTTP request headers  
x-vsaas-session: <session\_token>

## A list of favourites cameras

GET: /vsaas/api/my/cameras?filter=favorite

Parameters:

HTTP request headers

x-vsaa-session: <session\_token>

Reply: JSON

```
{
  "id": string
  "title": string
  "access": string // ["private", "public", "authorized"]
  "comment": string
  "postal_address": string
  "coordinates": string // "55.7512 37.6184"
  "streamer_url": string // "http://demo-watcher.flussonic.com"
  "dvr_depth": int // the depth of the archive (per days)
  "permissions": { // the user rights status
    "dvr": boolean // archive status
    "ptz": boolean // PTZ control
    "view": boolean // live-view
  }
  "ptz": boolean // PTZ control status
  "server": string // demo-watcher.flussonic.com
  "static": boolean // stream type: static or on-demand
  "status": { // information about the stream condition
    "source_error": string
    "alive": boolean
  }
  "thumbnails": boolean // the camera's snapshots status
  "urls": { // the links of any protocols receiving the stream
    "hls": string
    "hds": string
    "jpg": string
    "rtmp": string
    "rtsp": string
    "media_info": string // URL information of the stream
    "recording_status": string // URL archive status
  }
}
```

# Camera management

After the module is installed, you will see no camera items, even if they are present on Flussonic Media Server:

To add a camera to the system, click New camera :

Camera settings:

Title. Must be filled with English letters and numbers, as it will be used in URLs.

URL. The camera URL. If you have the camera URL, you can import the camera from the stream server (Flussonic) or use auto-search. For example: `rtsp://mycam.local/stream0`

Substreams. Additional address. For example: `rtsp://mycam.local/stream1; rtsp://mycam.local/stream2`

Substreams. Additional address. For example: `rtsp://mycam.local/stream1; rtsp://mycam.local/stream2`

Thumbnails URL. Address the removal of screenshots from the camera.

Location. Location of the camera. Shows the camera placement on the map. You can change its placement. We recommend that you place all camera objects to their actual locations on the map to help users finding them.

Address. The postal address of the camera.

Owner. Choice of camera owner. Owners are configured in the [Users and Groups] menu (manage-users);

Access. The camera availability: to all users, only to registered users or to specific users:

Public (public) - available to all users of Flussonic Watcher, regardless of the settings of groups and users. When setting the value of public source appears in the list of cameras for all users of your server. Also, without restrictions on access, the camera can be published on a web page with permanent access without authorization.

For customers (authorized) — also available to all users, but you need to authorize on your Flussonic Watcher.

Private — is available to the user only when adding access to the camera in the Users and Groups section .

Public (public) - available to all users of Flussonic Watcher, regardless of the settings of groups and users. When setting the value of public source appears in the list of cameras for all users of your server. Also, without restrictions on access, the camera can be published on a web page with permanent access without authorization.

For customers (authorized) — also available to all users, but you need to authorize on your Flussonic Watcher.

Private — is available to the user only when adding access to the camera in the Users and Groups section .

Server. Selecting the Flussonic Media Server.

DVR. Recording depth. In other words, it's a number of days after recording when the video archive

is available. To enable this option, configure the repository in the System settings or Streamer settings.

Options.

Turn the camera on and off.

Constantly working camera or on request.

Image available for preview or not.

Restrict access to the camera archive.

Turn the camera on and off.

Constantly working camera or on request.

Image available for preview or not.

Restrict access to the camera archive.

Comment. Text comment. Auxiliary text that describes camera positioning and other information.

Press Save. After that, the camera will appear on the management page and in the dashboard:

# Creating a client mosaic

Mosaic is the ability to show up to 16 selected cameras on a single screen.

## Creating a mosaic

To create a mosaic, log into Flussonic Watcher as an administrator. Then do the following:

- 1) Go to the «Cameras» section.
- 2) Press «Mosaic».
- 3) Click the «Add» button.
- 4) In the window opened:
  - Enter the mosaic name.
  - Mark the «enabled» checkbox.
  - Select the number of cameras.
  - Transfer the selected cameras into the Internet/Ethernet. Use filters if there are lots of cameras.
- 5) Click «Save».

Edit or delete mosaic in the «Cameras» > «Mosaic» section.

All created mosaics are available in «Dashboard» > «Mosaic».

# User Management

After the module installation, you will see one administrator user on the user management page:

## Add a user

To add a new user, you must enter:

Login

Password

Internal gray IP for auto login

Email

Checkboxes note:

Active. Check to have the account active.

Administrator. Mark if the user needs administrative rights.

DRV access. Mark if the user needs access to the archive of records.

You can also leave a note for the user.

## Adding cameras

To add a camera user, click on the tag «0 shared cameras», then «Add» and in the opened list select those cameras that are available to the user.

Click «Delete» if you want to close the user's access to the camera.

By tag «0 groups» you can add a group of cameras to the user. More in the article Groups.

After adding cameras, they will be available to the user in the dashboard.

# Groups

Use groups to simplify user and camera navigation by making logical grouping (for example, by floors, departments, regions, neighborhoods, etc.), as well as simplify the user right management to provide camera visibility. To do that, create a group with a few cameras and then add users to it. A user will have access to all private cameras of the group in a few clicks.

To create camera and user groups, go to the

Users and Groups

>

Panel

>

Add

. Next, enter the group name and description (if necessary), and click «Save».

Now, in the group list, you will see the new group.

By clicking on the «0 cameras» tag, you can add cameras from the list, and if you click on the «0 users» tag > «Add».

You can perform the same task if you click on the group name > «Cameras» > «Add».

You can turn on the file archive visibility (DVR) for each user in the list, as well as use an option to control the camera PTZ (certain cameras may not support it).

Important! If you use groups, it's necessary to change the camera status to Private. It'll give camera access to a limited number of users. A Public status makes a camera visible to all users, regardless of belonging to any group.

# Sharing cameras

With the camera sharing tool in Flussonic Watcher, an administrator can give private camera access to certain users without creating a group.

To do this, go to the «Users and Groups» and click «Add» next to a selected user where it says «0 shared cameras».

Select cameras you want to add from the list and click on the «Add» button. After that, the selected user will have access to these private cameras without being their owner.

You can perform the same operation via the user card. Click on «0 cameras» at the top and use the option to add one or more cameras that will be shared with this user.

Camera sharing is available for administrators only, and there is no such an option for users yet.

# Serial Numbers

If you purchased Watcher with camera agents, you need to create a user in the Users and Groups section to set a user auto-snap for this camera.

Then go to the Camera > Serial Numbers and add the serial number from the camera box (this camera will go to the end user).

Then, select the depth for the archive, assign this serial number to the end user, and provide login / password to the user.

When you turn the camera on for the first time, it will go through the activation procedure, provisioning to your server, and binding it to a certain end user using the serial number. The DVR setting that determines the archive depth will be applied as well.

From the user's perspective, the process will require these steps:

- Turn on the camera.

- Connect the camera to the Internet.

- Log in to Watcher using the login/password.

After these steps, the camera is up and running, and the user can see the video stream from the web-application.

# Import Cameras and Search

## Import cameras from Flussonic Media Server

On a Flussonic Watcher launch, cameras will be available on the network. You can use either Onvif search or import streams from Flussonic Media Server.

You can import a Flussonic Watcher camera that was previously added to Flussonic Media Server.

## Camera Search

Flussonic Watcher uses the WS-Discovery mechanism to search Onvif-compatible cameras. It can detect Ubiquity, Samsung and other cameras.

Some cameras require login and password for authorization. If you cannot find a camera, enter your login and password on the camera and search again.

Many cameras has two or more H264 streams, so you can add all of them as show on the video:

After you selected the camera profile, the correct value will appear in the RTSP URL. Enter a name that will be used for this camera.

See how to set the DVR path in the settings.