FindFace Multi

Release 1.0

NtechLab

Jan 18, 2024

CONTENTS

FindFace Multi is a multifunctional multi-object video analytics software, based on FindFace Enterprise Server, a cutting-edge AI recognition technology. FindFace Multi is a turnkey solution that you can harness in such areas as retail, banking, social networking, entertainment, sports, event management, dating services, video surveillance, public safety, homeland security, and others.

FindFace Multi can detect, identify, and analyze the following objects in the video:

- Human faces, along with recognition of such facial attributes as gender, age, emotions, glasses, face mask, beard, and many others. The integrated 2D anti-spoofing system ensures that it is a live person in front of a camera and eliminates the possibility of fraud using images on paper or mobile device screens.
- Human bodies (silhouettes), along with recognition of clothing color (top and bottom).
- Cars, with recognition of such car attributes as make, model, body style, color, license plate number, and others.

After FindFace Multi identifies and analyzes an object, it notifies responsible officials about its appearance within fractions of a second. Additional information about the object, such as a person's gender, age, car license plate number, etc., is displayed in the relevant recognition event.

FindFace Multi supports the integration of third-party solutions via *HTTP API* and *webhooks*, so you can enhance your current system or application with multi-object recognition functionality.

Core features

- AI-based platform.
- Comprehensive dossiers that can accommodate aggregated information about a person: photos of their face, body, and car, and license plate number.
- Fast and robust real-time video monitoring against dossier databases.
- Multi-object identification and analytics: faces, bodies (silhouettes), cars.
- Support for live video and archives, most video formats and codecs that FFmpeg can decode.
- Advanced camera management, including ONVIF support, the possibility of changing video orientation, and fine-tuning a camera for each object type.
- Multi-object verification: faces, bodies, cars.
- AI recognition of gender, age, emotions, glasses, beard, face mask, and other face attributes.
- AI face liveness detector.
- AI recognition of a person.
- AI recognition of clothing color.
- AI recognition of a car's make, model, body style, color, and license plate number.
- Database search for faces, bodies, cars.
- Possibility of counting faces and bodies on connected cameras (single- and multi-camera counting).
- Video surveillance.

Environment

- Developer-friendly installer and user-friendly interface.
- Single- and multi-host deployment.
- Increased performance and fault-tolerance in high load systems with numerous cameras and clients.
- Possibility of distributing dossier database among several hosts with synchronization and replication.
- Network or on-premise licensing.
- CPU- and GPU-based acceleration for your choice.
- Mobile app.

Security

- Advanced user management.
- Authentication based on a password, certificate, and face recognition for guaranteed system protection.
- Dossier security.
- Comprehensive, friendly, searchable audit logs.
- Backup and recovery utilities.

Make the most of your system

- Social interaction analysis.
- Know your customer analytics (KYC).
- Detailed reports on face recognition events, episodes, search events, persons, counters, cameras, dossiers, audit logs, and KYC analytics.
- Face liveness detector as a standalone service.

Useful little things

- Quick dossier database creation.
- Complete dossier customization.
- Deduplication support for events and dossiers.
- Personal data protection (GDPR and similar laws).
- Extended set of search filters.
- Scheduled database cleanup.

Integration

- Integration via HTTP API and webhooks.
- Integrations with favored vendors.

CHAPTER

ONE

SYSTEM ADMINISTRATOR'S GUIDE

This chapter is all about FindFace Multi deployment and further updates and maintenance during exploitation.

1.1 Architecture

Though you mostly interact with FindFace Multi through its web interface, be sure to take a minute to learn the FindFace Multi architecture. This knowledge is essential for the FindFace Multi deployment, integration, maintenance, and troubleshooting.

In this chapter:

1.1.1 Recognition Objects and Recognition Process

FindFace Multi can recognize the following objects and their features:

- human faces
- human bodies (silhouettes)
- cars

Note: The face recognition functionality is enabled by default. Make changes to configuration files to enable the *body* and *car* recognition.

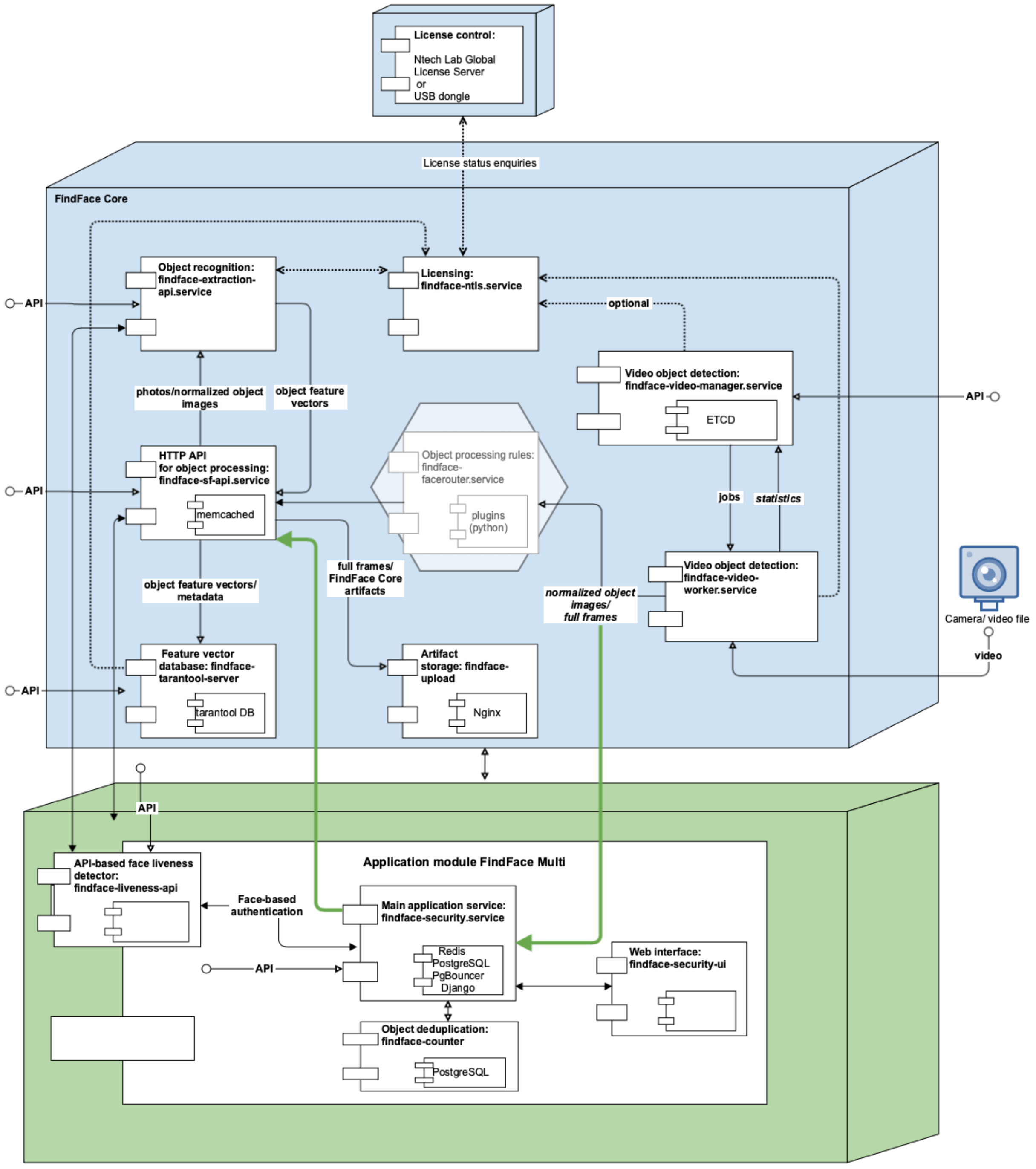
FindFace Multi detects an object in the photo or video and prepares its image through normalization. The normalized image is then used for extracting the object's feature vector (an n-dimensional vector of numerical features that represent the object). Object feature vectors are stored in the database and further used for verification and identification purposes.

1.1.2 Architectural Elements

FindFace Multi consists of the following fundamental architectural elements:

- FindFace core, a cutting-edge AI-based recognition technology that can be used as a separate product FindFace Enterprise Server.
- FindFace Multi, which is a turnkey application module for FindFace Enterprise Server.

Architecture scheme


License control:
Ntech Lab Global License Server or USB dongle
License status enquiries
FindFace Core
Object recognition: findface-extraction-api.service
Licensing: findface-ntls.service
optional
API
photos/normalized object images
object feature vectors
Video object detection: findface-video-manager.service
ETCD
HTTP API for object processing: findface-sf-api.service
memcached
Object processing rules: findface-facerouter.service
plugins (python)
jobs
statistics
object feature vectors/ metadata
full frames/ FindFace Core artifacts
normalized object images/ full frames
Video object detection: findface-video-worker.service
Camera/ video file
video
Feature vector database: findface-tarantool-server
tarantool DB
Artifact storage: findface-upload
Nginx
Application module FindFace Multi
API-based face liveness detector: findface-liveness-api
Face-based authentication
Main application service: findface-security.service
Redis
PostgreSQL
PgBouncer
Django
Web interface: findface-security-ui
Object deduplication: findface-counter
PostgreSQL

FindFace Core

The FindFace core includes the following components:

Component	Ports in use	Description	Vendor
findface-extraction-api	18666	Service that uses neural networks to detect an object in an image and extract its feature vector. It also recognizes object attributes (for example, gender, age, emotions, beard, glasses, face mask - for face objects). CPU- or GPU-acceleration.	NtechLab own deployment
findface-sf-api	18411	Service that implements the internal HTTP API for object detection and recognition.	
findface-tarantool-server	32001, shard ports (default 330xx, 81xx)	Service that provides interaction between the `findface-sf-api` service and the feature vector database (the Tarantool-powered database that stores object feature vectors.	
findface-upload	3333	NginX-based web server used as a storage for original images, thumbnails and normalized object images.	
findface-facerouter	18820	Service used to define processing directives for detected objects. In FindFace Multi, its functions are performed by `findface-security` (see *FindFace Multi Application Module*). If necessary, you can still deploy and enable this component for integration purposes (see *findface-facerouter and Custom Plugins*).	
findface-video-manager	18810, 18811	Service, part of the video object detection module, that is used for managing the video object detection functionality, configuring the video object detector settings and specifying the list of to-be-processed video streams.	
findface-video-worker	18999	Service, part of the video object detection module, that recognizes an object in the video and posts its normalized image, full frame and metadata (such as detection time) to the `findface-facerouter` service for further processing according to given directives. Provides *face liveness detection* if enabled. CPU- or GPU-acceleration.	
findface-ntls	443 (TCP), 3133, 3185	License server which interfaces with the NtechLab Global License Server or a USB dongle to verify the *license* of your FindFace Multi instance.	
Tarantool	Shard ports (default 330xx, 81xx)	Third-party software which implements the feature vector database that stores extracted object feature vectors and identification events. The system data, dossiers, user accounts, and camera settings are stored in PostgreSQL (part of the FindFace Multi application module).	Tarantool
etcd	2379	Third-party software that implements a distributed key-value store for `findface-video-manager`. Used as a coordination service in the distributed system, providing the video object detector with fault tolerance.	etcd
NginX	80; SSL: 8002, 8003, 443, 80	Third-party software which implements the system web interfaces.	nginx
memcached	11211	Third-party software which implements a distributed memory caching system. Used by `findface-extraction-api` as a temporary storage for extracted object feature vectors before they are written to the feature vector database powered by Tarantool.	memcached

FindFace Multi Application Module

The FindFace Multi application module includes the following components:

Component	Ports in use	Description	Vendor
findface-security	Configurable	Component that serves as a gateway to the FindFace core. Provides interaction between the FindFace Core and the web interface, the system functioning as a whole, HTTP and web socket, object monitoring, event notifications, episodes, webhooks, and counters. Includes the following internal services: NTLS checker, Counter manager, Webhooks manager, Persons clusterizator, Event episodes manager, and Video archive queue manager. The last four can be enabled and disabled via the `/etc/findface-security/config.py` configuration file.	NtechLab own deployment
findface-security-ui	Configurable	Main web interface that is used to interact with FindFace Multi. Allows you to work with object identification events, search for objects, manage cameras, users, dossiers, and watch lists, collect real-time statistics, and many more.	
findface-counter	18300	Service used for event deduplication.	
findface-liveness-api	18301	Besides the embedded functionality provided by `findface-video-worker`, face liveness detection can also be harnessed as a standalone service `findface-liveness-api`. The service takes a specific number of frames from a video fragment and returns the best quality face, and decimal liveness result averaged across the taken frames. The service is also used for authentication based on facial recognition. See *Liveness Detection as Standalone Service* and *Authentication and Session Monitoring*.	
PostgreSQL	5432	Third-party software which implements the main system database that stores detailed and categorized dossiers on particular objects (faces, bodies, cars), and data for internal use such as user accounts and camera settings. The object feature vectors and object identification events are stored in Tarantool (part of the FindFace core).	PostgreSQL
Pgbouncer	5439	Third-party software, a lightweight connection pooler for PostgreSQL. Optional, used to increase the database performance under high load.	PgBouncer
Redis	6379	Third-party software which implements a message broker inside `findface-security`.	Redis
Django	5439	Third-party software which implements a web framework for the FindFace Multi web interface.	Django

See also:

Components in Depth

1.1.3 Single- and Multi-Host Deployment

You can deploy FindFace Multi on a single host or in a cluster environment. If you opt for the latter, we offer you one of the following deployment schemes:

- Deploy FindFace Multi standalone and distribute additional `findface-video-worker` components across multiple hosts.
- Distribute the FindFace Multi components across multiple hosts. If necessary, set up load balancing.

See *Guide to Typical Cluster Installation* for details.

1.1.4 CPU- and GPU-acceleration

The `findface-extraction-api` and `findface-video-worker` services can be either CPU- or GPU-based. During installation from the developer-friendly *installer*, you will have an opportunity to choose the acceleration type you need.

If you opt to install FindFace Multi from the *repository package*, deploy the `findface-extraction-api` and `findface-video-worker-cpu` packages on a CPU-based server, and the `findface-extraction-api-gpu` and/or `findface-video-worker-gpu` packages on a GPU-based server.

Important: Refer to *Requirements* when choosing hardware configuration.

Important: If the resolution of a camera(s) in use is more than 1280x720px, it is strongly recommended to use the GPU-accelerated package `findface-video-worker-gpu`.

Note: The *liveness detector* is much slower on CPU than on GPU.

1.2 Requirements

In this chapter:

1.2.1 System Requirements for Basic Configuration

To calculate the FindFace Multi host(s) characteristics, use the requirements provided below.

Tip: Be sure to learn about the FindFace Multi *architecture* first.

Important: If the resolution of a camera(s) in use is more than 1280x720px, it is strongly recommended to use the GPU-accelerated package `findface-video-worker-gpu`.

Important: On AMD CPU servers, the full functionality of the CPU-accelerated `findface-extraction-api` service is not guaranteed. Use the GPU-accelerated service `findface-extraction-api-gpu` along with the GPU-version of neural networks instead.

Note: In the case of a high-load system (~> 15 events per second), we recommend using an SSD.

	Minimum	Recommended
CPU	Intel Core i5 CPU with 4+ physical cores 3+ GHz. AVX2 support	Intel Xeon Silver/Gold with 6+ physical cores
	The own needs of FindFace Multi require 2 cores HT > 2.5 GHz. The characteristics also depend on the number of cameras in use. A single camera 720p@25FPS requires 2 cores >2.5 GHz. AVX2 support	
GPU (optional)	Nvidia Geforce® GTX 1060 6 GB	Nvidia Geforce® GTX 1080Ti+ 11+ GB RAM
	Supported series: GeForce (Maxwell, Pascal, Turing, and above), Tesla (Maxwell, Pascal, Volta v100, Turing, and above)	
RAM	10 Gb	16+ Gb
	The own needs of FindFace Multi require 8 Gb. The RAM consumption also depends on the number of cameras in use. A single camera 720p@25FPS requires 2 GB RAM	
HDD (SSD for best performance)	16 Gb	16+ Gb
	The own needs of the operating system and FindFace Multi require 15 GB. The total volume is subject to the required depth of the event archive in the database and in the log, at the rate of 1.5 Mb per 1 event	
Operating system	Ubuntu 18.04, x64 only	

Note: You can also use an Intel-based VM if there is AVX2 support, and eight physical cores are allocated exclusively to the VM.

Tip: For more accurate hardware selection, contact our support team by support@ntechlab.com.

1.2.2 Required Administrator Skills

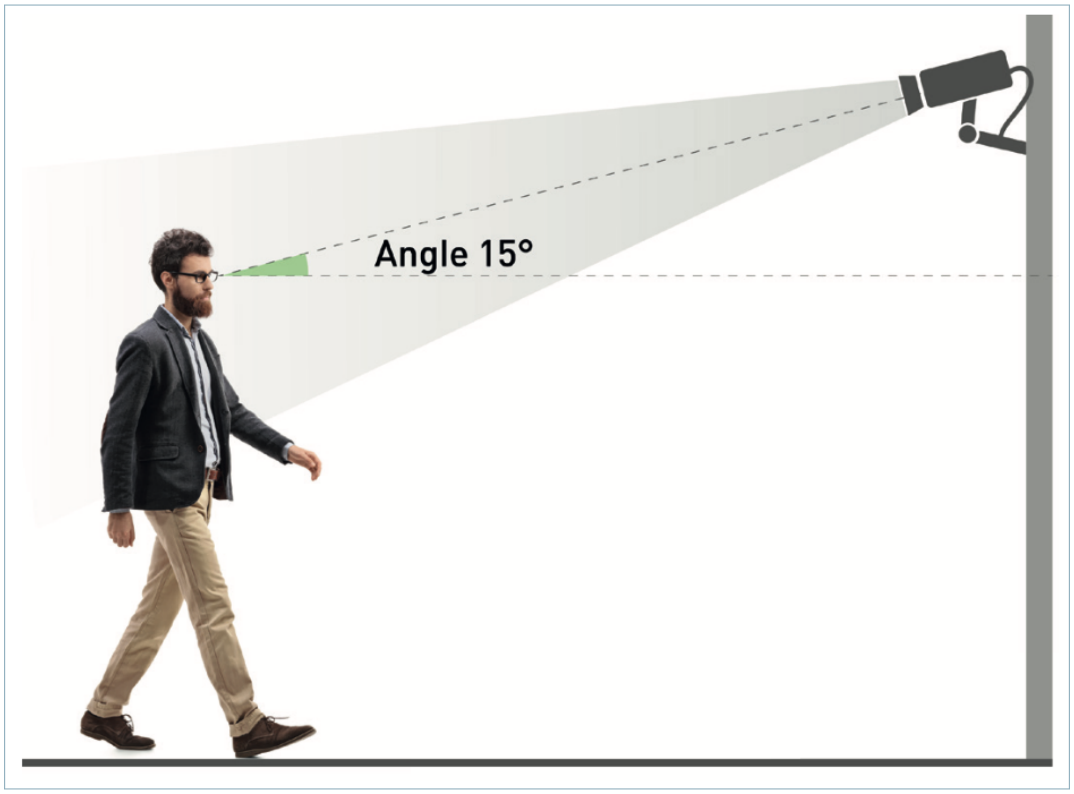
A FindFace Multi administrator must know and understand OS Ubuntu at the level of an advanced user.

1.2.3 Requirements for CCTV Cameras

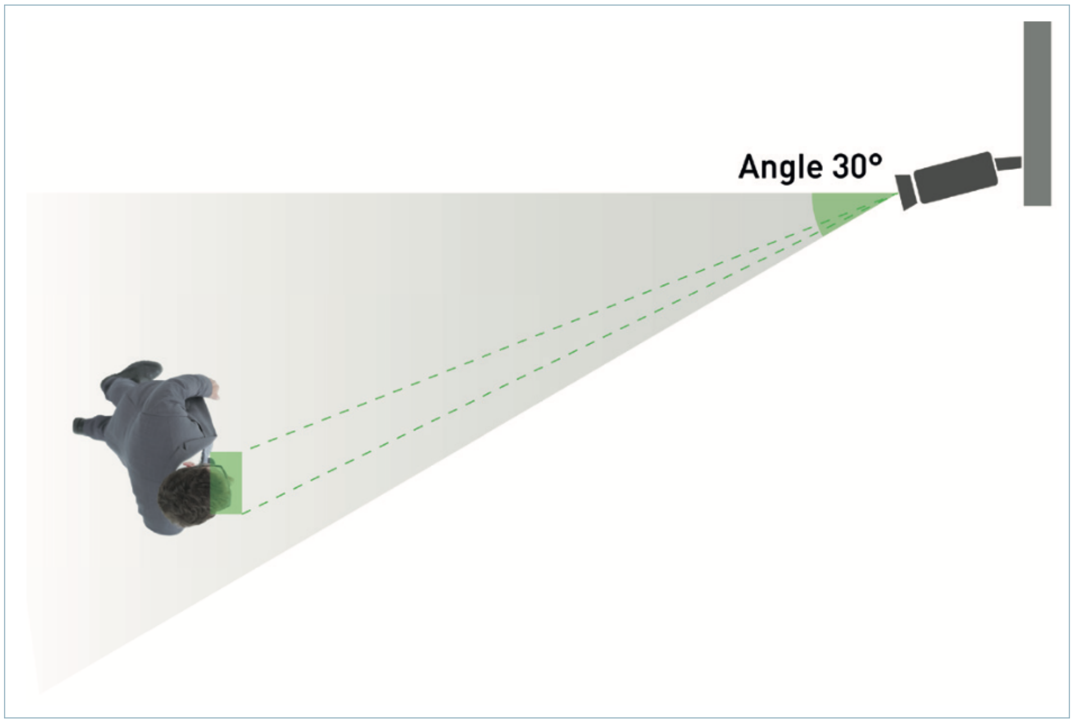
Face Recognition

The primary requirements for installation and characteristics of CCTV cameras in your FindFace Multi-based face recognition system are the following:

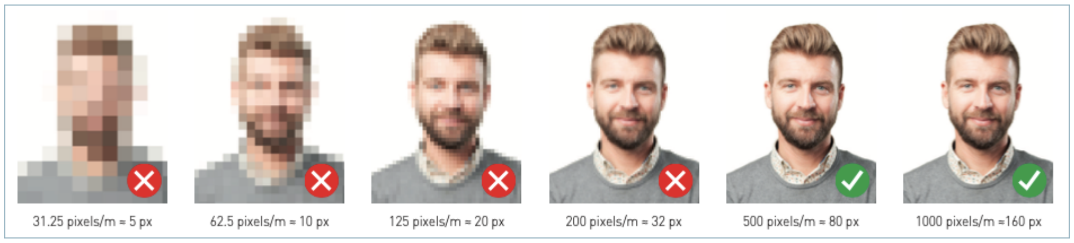
1. For correct face detection in a video stream, mount the camera so that the face of each individual entering the monitored area surely appears in the camera field of view.
2. The vertical tilt angle of the camera should not exceed 15°. The vertical tilt is a deviation of the camera's optical axis from the horizontal plane, positioned at the face center's level for an average height person (160 cm).



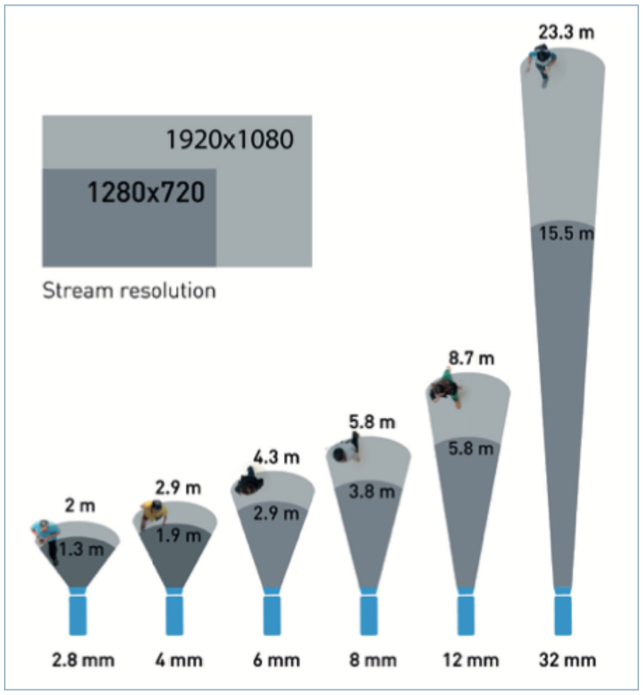
3. The horizontal deflection angle should not exceed 30°. The horizontal deflection is a deviation of the camera's optical axis from the motion vector of the main flow of objects subject to recognition.



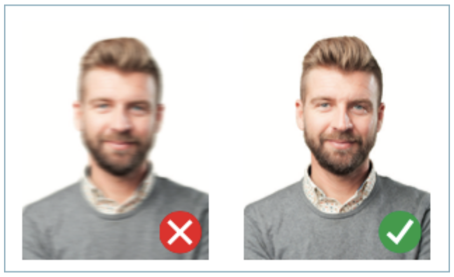
4. The minimum pixel density required for identification is 500 pixels/m (roughly corresponds to a face width of 80 pixels).



5. Select such a focal length of the camera's lenses that provides the required pixel density at a predetermined distance to the recognition objects. The picture below demonstrates how to calculate the focal length subject to the distance between the camera and recognition objects. Estimating the focal length for a particular camera requires either calculators or a methodology provided by the camera manufacturer.



6. The exposure must be adjusted so that the face images are sharp ("in focus"), non-blurred, and evenly lit (not overlit or too dark).



7. For imperfect lighting conditions such as flare, too bright or too dim illumination, choose cameras with WDR hardware (Wide Dynamic Range) or other technologies that provide compensation for backlight and low illumination. Consider BLC, HLC, DNR, high optical sensitivity, Smart infrared backlight, AGC, and such.

8. Video compression: most video formats and codecs that FFmpeg can decode.
9. Video stream delivery protocols: RTSP, HTTP.

Tip: To calculate the precise hardware configuration tailored to your purposes, contact our experts by support@ntechlab.com.

Body and Car Recognition

Since the body and car recognition are new features freshly implemented in the current version, we still need a great deal of testing in the field to formulate universal requirements for CCTV cameras applicable to the operational phase. However, we are always happy to individually assist you with hardware selection and deployment tailored to your business needs. Please do not hesitate to contact our experts (support@ntechlab.com).

1.3 Licensing Info

In this chapter:

1.3.1 Licensing Principles

The FindFace Multi licensing is granted using the following criteria:

1. The overall number of extracted feature vectors, regardless of the object type (face, body, car).

   Note: The feature vectors are extracted from objects detected in the video and from dossier photos and when building so-called *person* centroids.

   The licensing scheme is the following:

   - Events: 1 event of video object detection = 1 object in a license.

   - Dossier: 1 photo in a dossier = 1 object in a license.
   - Persons: 1 person = 1 object in a license.
2. The number of video sources currently in use (i.e., active video processing jobs for cameras and video files).
3. The number of model instances in use in the `findface-extraction-api` component.
4. Face attribute recognition: gender/age/emotions/glasses/beard/face mask.
5. Body attribute recognition: clothing color.
6. Car attribute recognition: make/model/color/body style.
7. License plate recognition.
8. Face liveness detection.
9. Integration with partners.

You can choose between the online and on-premise (aka offline) licensing:

- The online licensing is provided by interaction with the NtechLab Global License Manager `license.ntechlab.com` and requires a stable internet connection, DNS, and open port 443 TCP. Upon being disconnected from the internet, the system will continue working off-grid for 4 hours.

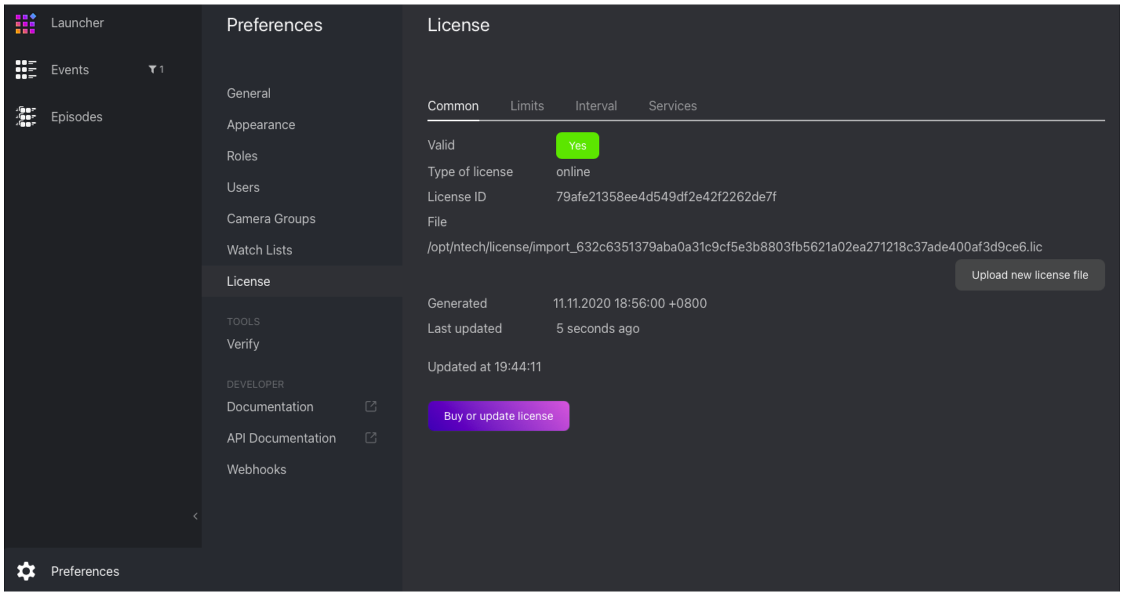
Note: It is possible to prolongate the off-grid period for up to 2 days. Inform your manager if you need that.

- The on-premise (offline) licensing requires a USB port on the physical server with the `findface-ntls` component (license server in the *FindFace core*), that you will use to plug in a provided USB dongle.

Important: For the system to function, a single instance of `findface-ntls` should be enough. If your system requires more license servers, contact your NtechLab manager beforehand to prevent your system from being blocked.

1.3.2 View and Update License

After installing FindFace Multi, upload the license file you obtained from the manager into the system. To do so, navigate to *Preferences -> License*.



License

Valid	Yes
Type of license	online
License ID	79afe21358ee4d549df2e42f2262de7f
File	

/opt/ntech/license/import_632c6351379aba0a31c9cf5e3b8803fb5621a02ea271218c37ade400af3d9ce6.lic

Generated	11.11.2020 18:56:00 +0800
Last updated	5 seconds ago

Updated at 19:44:11

Use the same tab to consult current licensing information and upgrade your license.

See also:

Troubleshoot Licensing and findface-ntls

1.4 Deploy FindFace Multi

FindFace Multi provides the following deployment options:

- from a console installer
- step-by-step from an APT repository

Important: Starting services `findface-extraction-api-gpu` and `findface-video-worker-gpu` for the first time after deployment may take up a considerable amount of time, up to 45 minutes, due to the caching process.

Important: Although FindFace Multi provides *tools* to ensure its protection from unauthorized access, they are not replacing a properly configured firewall. Be sure to use a firewall to heighten the FindFace Multi network protection.

1.4.1 Deploy from Console Installer

To deploy FindFace Multi, use a developer-friendly console installer.

Tip: Before deployment, be sure to consult the *system requirements*.

Important: The FindFace Multi host must have a static IP address in order to be running successfully. To make the IP address static, open the `/etc/network/interfaces` file and modify the current primary network interface entry as shown in the case study below. Be sure to substitute the suggested addresses with the actual ones, subject to your network specification.

```
sudo vi /etc/network/interfaces

iface eth0 inet static
address 192.168.112.144
netmask 255.255.255.0
gateway 192.168.112.254
dns-nameservers 192.168.112.254
```

Restart networking.

```
sudo service networking restart
```

Be sure to edit the `etc/network/interfaces` file with extreme care. Please refer to the Ubuntu guide on networking before proceeding.

To deploy FindFace Multi from the console installer, do the following:

1. Download the installer file `findface-multi-1.0-and-server-5.0.1.run`.
2. Put the `.run` file into some directory on the designated host (for example, `/home/username`).
3. From this directory, make the `.run` file executable.

   ```
   chmod +x findface-multi-1.0-and-server-5.0.1.run
   ```

4. Execute the `.run` file.

   ```
   sudo ./findface-multi-1.0-and-server-5.0.1.run
   ```

   The installer will ask you a few questions and perform several automated checks to ensure that the host meets the system requirements. Fill out the prompts appropriately once requested. The questions are the following:

   1. Product to install: FindFace Multi.
   2. Installation type:

      - 1: install FindFace Multi standalone.
      - 2: install FindFace Multi and configure it to interact with additional remote `findface-video-worker` instances.

        Tip: To install only `findface-video-worker` on a host, refer to *Additional findface-video-worker deployment on remote hosts*.

      - 3: install only the apt repository that can be further used for the *step-by-step deployment*.

        Important: This installation type doesn't provide installation of neural network models essential for the `findface-extraction-api` functioning. Be sure to *manually install* them on the host(s) with `findface-extraction-api`.

      - 4: fully customized installation.

Important: Be sure to *manually install* neural network models on the host(s) with `findface-extraction-api`.

3. Type of `findface-video-worker` package: CPU or GPU.
4. Type of `findface-extraction-api` package: CPU or GPU.

Once all the questions answered, the answers will be saved to a file `/tmp/<findface-installer-*>.json`. You can edit this file and use it to install FindFace Multi on other hosts without having to answer the questions again.

Should you choose to install FindFace Multi standalone, its components will be automatically installed, configured and/or started in the following configuration:

Important: In the case of a clean install, the installer will automatically configure `findface-extraction-api` to use the `kiwi_320` neural network. Otherwise, you will be able to choose between `kiwi_320` and the previous model. It is strictly not recommended to use the installer to update the system. See *Upgrade FindFace Security to FindFace Multi* for the instructions.

Service	Configuration
postgresql-10	Installed and started.
redis-server	Installed and started.
etcd	Installed and started.
pg-bouncer	Installed and started.
mem-cached	Installed and started.
nginx	Installed and started.
django	Installed and started as a web framework for the FindFace Multi web interface.
findface-ntls	Installed and started.
findface-tarantool-server	Installed and started. The number of instances (shards) is calculated using the formula: `N = min(max(min(mem_mb // 2000, cpu_cores), 1), 16 * cpu_cores)`. I.e., it is equal to the RAM size in MB divided by 2000, or the number of CPU physical cores (but at least one shard), or the number of CPU physical cores multiplied by 16, should the first obtained value be greater.
findface-extraction-api	Installed and started.
findface-sf-api	Installed and started.
findface-upload	Installed.
findface-video-manager	Installed and started (CPU/GPU-acceleration).
findface-video-worker-*	Installed and started.
findface-data-*	Neural network models for object and object attribute recognition. Installed.
findface-gpudetector-data	NTechLab gpudetector data. Installed.
python3-ntech.ffsecurity-client	NtechLab FindFace Multi API python client library. Installed.
findface-security	Installed and started.
findface-counter	Installed and started.
findface-liveness-api	Installed and started.
jq	Installed. Used to pretty-print API responses from FindFace Multi.

After the installation is complete, the following output is shown on the console:

Tip: Be sure to save this data: you will need it later.

```
#############################################################################
#                       Installation is complete                            #
#############################################################################
- upload your license to http://172.20.77.17/#/license/
- user interface: http://172.20.77.17/
  superuser:      admin
  password:       admin
  documentation:  http://172.20.77.17/doc/
```

5. Specify your time zone in the `/etc/findface-security/config.py` configuration file either in the `Region/Country/City` or `Etc/GMT+H` format. The time zone determines the time in reports, logs, and names of FindFace Multi artifacts such as event full frames and thumbnails, counter screenshots, etc.

   Tip: See this table for reference.

   ```
   sudo vi /etc/findface-security/config.py

   # time zone
   TIME_ZONE = 'America/Argentina/Buenos_Aires'
   ```

6. Upload the FindFace Multi license file via the main web interface `http://<Host_IP_address>/#/license`. To access the web interface, use the provided `admin` credentials.

   Note: The host IP address is shown in the links to FindFace web services in the following way: as an external IP address if the host belongs to a network, or `127.0.0.1` otherwise.

   Important: Do not disclose the superuser (Super Administrator) credentials to others. To administer the system, create a new user with administrator privileges. Whatever the role, the Super Administrator cannot be deprived of its rights.

7. To automatically install FindFace Multi on another host without answering the installation questions, use the `/tmp/<findface-installer-*>.json` file. Execute:

   ```
   sudo ./findface-multi-1.0-and-server-5.0.1.run -f /tmp/<findface-installer-*>.json
   ```

   Tip: You can find an example of the installation file in *Installation File*.

Important: To preserve the FindFace Multi compatibility with the installation environment, we highly recommend you to disable the Ubuntu automatic update. In this case, you will be able to update your OS manually, fully controlling which packages to update.

To disable the Ubuntu automatic update, execute the following commands:

```
sudo apt-get remove unattended-upgrades
sudo systemctl stop apt-daily.timer
sudo systemctl disable apt-daily.timer
sudo systemctl disable apt-daily.service
sudo systemctl daemon-reload
```

Important: The FindFace Multi services log a large amount of data, which can eventually lead to disc overload. To prevent this from happening, we advise you to disable `rsyslog` due to its suboptimal log rotation scheme and use the appropriately configured `systemd-journal` service instead. See *Service Logs* for the step-by-step instructions.

1.4.2 Deploy Step-by-Step from Repository

This section will guide you through the FindFace Multi step-by-step deployment process. Follow the instructions below minding the sequence.

In this section:

Install APT Repository

First of all, install the FindFace apt repository as follows:

1. Download the installer file `findface-multi-1.0-and-server-5.0.1.run`.
2. Put the `.run` file into some directory on the designated host (for example, `/home/username`).
3. From this directory, make the `.run` file executable.

   ```
   chmod +x findface-multi-1.0-and-server-5.0.1.run
   ```

4. Execute the `.run` file.

   ```
   sudo ./findface-multi-1.0-and-server-5.0.1.run
   ```

   The installer will ask you a few questions and perform several automated checks to ensure that the host meets the system requirements. Fill out the prompts appropriately once requested. The questions are the following:

   1. Product to install: `FindFace Multi`.
   2. Installation type: `repo:  Don't install anything, just set up the APT repository`.

3. Neural network models to install if necessary. To select a model(s), deselect all those on the list by entering `-*` in the command line first, then select the required model by entering its sequence number (keyword): for example, `1 3`. Enter done to save your selection and proceed to another step.

   Important: At least one model for face biometry has to be installed.

After that, the FindFace apt repository will be automatically installed.

Prerequisites

FindFace Multi requires such third-party software as PostgreSQL, Pgbouncer, Redis, etcd, and memcached. Do the following:

1. Install the prerequisite packages as such:

```
sudo apt update
sudo apt install -y postgresql-10 redis-server etcd memcached pgbouncer
```

2. Open the `/etc/memcached.conf` configuration file. Set the maximum memory to use for items in megabytes: `-m 1024`. Set the max item size: `-I 16m`. If one or both of these parameters are absent, add them in the file.

```
sudo vi /etc/memcached.conf

-m 1024
-I 16m
```

3. Give a strong password to the `ntech` user (`9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3` in the example below). Output the credentials to the `pgbouncer` user list.

```
echo '"ntech" "9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3"' | sudo tee -a /etc/pgbouncer/
↪userlist.txt
```

4. Configure `pgbouncer`. In `/etc/pgbouncer/pgbouncer.ini`, add the `ffsecurity` database to the `databases` section. Configure named parameters, as shown in the example below. Parameters other than those must be commented out.

```
sudo vi /etc/pgbouncer/pgbouncer.ini

[databases]
ffsecurity = dbname=ffsecurity host=localhost port=5432 user=ntech
[pgbouncer]
pidfile = /var/run/postgresql/pgbouncer.pid
listen_addr = 127.0.0.1
listen_port = 5439
unix_socket_dir = /var/run/postgresql
auth_type = plain
auth_file = /etc/pgbouncer/userlist.txt
pool_mode = transaction
server_reset_query = DISCARD ALL
max_client_conn = 16384
default_pool_size = 20
syslog = 1
```

5. Enable the prerequisite services autostart and launch the services:

```
sudo systemctl enable postgresql@10-main.service redis-server etcd.service␣
↪memcached.service pgbouncer.service
sudo systemctl restart postgresql@10-main.service redis-server etcd.service␣
↪memcached.service pgbouncer.service
```

Provide Licensing

See also:

Licensing Info

You receive a license file from your NTechLab manager. If you opt for the on-premise licensing, we will also send you a USB dongle.

To provide the FindFace Multi licensing, deploy `findface-ntls`, license server in the FindFace core.

Important: There must be only one `findface-ntls` instance in each `FindFace Multi` installation.

```
sudo apt update
sudo apt install -y findface-ntls
sudo systemctl enable findface-ntls.service && sudo systemctl start findface-ntls.service
```

Deploy Main Database

In FindFace Multi, the main system database is based on PostgreSQL. To deploy the main database, do the following:

1. Open the `pgbouncer` list of users `/etc/pgbouncer/userlist.txt`. Copy the `ntech` user's password (`9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3` in the example below).

```
sudo cat /etc/pgbouncer/userlist.txt

"ntech" "9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3"
```

2. Using the **`PostgreSQL`** console, create a new user `ntech` with the copied password, and databases `ffsecurity` and `ffcounter` in PostgreSQL.

```
sudo -u postgres psql

postgres=# CREATE ROLE ntech WITH LOGIN PASSWORD '9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3';

postgres=# CREATE DATABASE ffsecurity WITH OWNER ntech ENCODING 'UTF-8' LC_COLLATE=
↪'en_US.UTF-8' LC_CTYPE='en_US.UTF-8' TEMPLATE template0;

postgres=# CREATE DATABASE ffcounter WITH OWNER ntech ENCODING 'UTF-8' LC_COLLATE='C.
↪UTF-8' LC_CTYPE='C.UTF-8' TEMPLATE template0;
```

Tip: To quit from the **`PostgreSQL`** console, type `\q` and press `Enter`.

3. Allow authentication by UID of a socket client in **`PostgreSQL`**. Restart **`PostgreSQL`**.

```
echo 'local all ntech peer' | sudo tee -a /etc/postgresql/10/main/pg_hba.conf

sudo systemctl restart postgresql@10-main.service
```

Deploy FindFace Core

To deploy the FindFace core, do the following:

Tip: You can find the description of the FindFace core components and their configuration parameters in *Architecture* and *Components in Depth*.

1. For FindFace Multi on GPU, *install NVIDIA drivers*.
2. Install the FindFace core components:

```
sudo apt update
sudo apt install -y findface-tarantool-server findface-extraction-api findface-sf-
↪api findface-upload findface-video-manager findface-video-worker-cpu findface-
↪liveness-api
```

Note: To install the GPU-accelerated `findface-extraction-api` component, use `findface-extraction-api-gpu` instead of `findface-extraction-api` in the command.

Note: To install the GPU-accelerated `findface-video-worker` component, use `findface-video-worker-gpu` instead of `findface-video-worker-cpu` in the command. If you have several video cards on your server, see *Multiple Video Cards Usage*.

Important: Be sure to *manually install* neural network models on the host(s) with `findface-extraction-api`.

3. In the `/etc/findface-sf-api.ini` configuration file, enable the `allow-return-facen` parameter.

```
sudo vi /etc/findface-sf-api.ini

...
limits:
  ...
  allow-return-facen: true
...
```

4. Open the `/etc/findface-video-worker-cpu.ini` (`/etc/findface-video-worker-gpu.ini`) configuration file. In the `mgr-static` parameter, specify the `findface-video-manager` host IP address, which provides `findface-video-worker` with settings and the video stream list. In the `capacity` parameter, specify the maximum number of video streams to be processed by `findface-video-worker`. In the `streamer` section, specify the IP address and port to access the *video wall*. The streamer `port` must be set to `18999`.

```
sudo vi /etc/findface-video-worker-cpu.ini
sudo vi /etc/findface-video-worker-gpu.ini

mgr-static=127.0.0.1:18811

capacity=10

[streamer]
#------------------------------
## streamer/shots webserver port, 0=disabled
## type:number env:CFG_STREAMER_PORT longopt:--streamer-port
port = 18999
## streamer url - how to access this worker on streamer_port
## type:string env:CFG_STREAMER_URL longopt:--streamer-url
url = 127.0.0.1:18999
```

5. Enable the FindFace core services autostart and launch the services.

```
sudo systemctl enable findface-extraction-api findface-sf-api findface-video-
↪manager findface-video-worker-cpu findface-liveness-api
sudo systemctl start findface-extraction-api findface-sf-api findface-video-manager␣
↪findface-video-worker-cpu findface-liveness-api
```

Deploy FindFace Multi Application Module and Feature Vector Database

To deploy the FindFace Multi application module, do the following:

1. Install the `findface-security`, `findface-security-ui`, and `findface-counter` components. Enable the `findface-counter` autostart and launch the service.

```
sudo apt update
sudo apt install -y findface-security findface-security-ui findface-counter
sudo systemctl enable findface-counter && sudo systemctl start findface-counter
```

2. Migrate the database architecture from FindFace Multi to **`PostgreSQL`**, create user groups with *predefined* rights and the first user with administrator rights (a.k.a. Super Administrator).

Important: Super Administrator cannot be deprived of its rights, whatever the role.

```
sudo findface-security migrate
sudo findface-security create_groups
sudo findface-security create_default_user
```

3. Create a structure of the Tarantool-based feature vector database by executing the command below.

```
sudo findface-security make_tnt_schema | sudo tee /etc/findface-security/tnt_schema.
↪lua
```

4. Open the `/etc/tarantool/instances.available/FindFace.lua` configuration file. Check whether it contains the `dofile` command, `meta_indexes` and `meta_scheme` definitions, as in the example below.

```
sudo vi /etc/tarantool/instances.available/FindFace.lua

dofile("/etc/findface-security/tnt_schema.lua")
-- host:port to bind, HTTP API
FindFace = require("FindFace")
FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    meta_indexes=meta_indexes,
    meta_scheme = meta_scheme
})
```

Important: The IP address and port number specified in the `shards` section of the `/etc/findface-sf-api.ini` configuration file must be identical to those in the `FindFace.start` section.

```
sudo vi /etc/tarantool/instances.available/FindFace.lua

...

FindFace.start("127.0.0.1", 8101...)
```

```
sudo vi /etc/findface-sf-api.ini

storage-api:
  ...
  shards:
  - master: http://127.0.0.1:8101/v2/
  ...
```

Important: If you change the `/etc/findface-sf-api.ini` configuration file, be sure to restart the `findface-sf-api` service:

```
sudo systemctl restart findface-sf-api.service
```

5. Enable the `findface-tarantool-server` service autostart and launch the service.

```
sudo systemctl enable tarantool@FindFace.service && sudo systemctl start␣
↪tarantool@FindFace.service
```

6. Open the `/etc/findface-security/config.py` configuration file. Specify the following parameters:

Tip: You can find the `/etc/findface-security/config.py` default version here.

- `SERVICE_EXTERNAL_ADDRESS`: FindFace Multi IP address or URL prioritized for the Genetec integration and webhooks. Once this parameter not specified, the system uses `EXTERNAL_ADDRESS` for these purposes. To use Genetec and webhooks, be sure to specify at least one of those parameters: `SERVICE_EXTERNAL_ADDRESS`, `EXTERNAL_ADDRESS`.
- `EXTERNAL_ADDRESS`: (Optional) IP address or URL that can be used to access the FindFace Multi web interface. Once this parameter not specified, the system auto-detects it as the external IP address. To access

FindFace Multi, you can use both the auto-detected and specified IP addresses.

- `VIDEO_DETECTOR_TOKEN`: to authorize the video face detection module, come up with a token and specify it here.
- `VIDEO_MANAGER_ADDRESS`: IP address of the `findface-video-manager` host.
- `NTLS_HTTP_URL`: IP address of the `findface-ntls` host.
- `ROUTER_URL`: IP address of the `findface-security` host that will receive detected faces from the `findface-video-worker` instance(s). Specify either external or internal IP address, subject to the network through which `findface-video-worker` interacts with `findface-security`. Change the default port, subject to the *redirect settings* from HTTP to HTTPS, or omit it leaving only the IP address.
- `SF_API_ADDRESS`: IP address of the `findface-sf-api` host.
- `DATABASES` (section): fill it in as such: `'PORT': 5439, 'USER': 'ntech', 'PASSWORD': '<password from /etc/pgbouncer/userlist.txt>'` (see *Prerequisites*).

Tip: If necessary, ensure data security by enabling *SSL*.

Tip: If necessary, set `'IGNORE_UNMATCHED': True` to disable logging events for faces which have no match in the dossiers (negative verification result). Enable this option if the system has to process a large number of faces. The face similarity threshold for verification is defined by the `CONFIDENCE_THRESHOLD` parameter.

Tip: It is NOT recommended to change the `MINIMUM_DOSSIER_QUALITY` default value. This parameter determines the minimum quality of a face in a dossier photo. Photos containing faces of worse quality will be rejected when uploading to a dossier. Upright faces in frontal position are considered the best quality.

7. Generate a signature key for the session encryption (used by Django) by executing the command below. Specify this key as `SECRET_KEY`.

```
pwgen -sncy 50 1|tr "'" "."
```

8. Start the services.

```
sudo systemctl enable findface-security
sudo systemctl start findface-security
```

9. Disable the default nginx server and add the `findface-security` server to the list of enabled servers. Restart nginx.

```
sudo rm /etc/nginx/sites-enabled/default

sudo ln -s /etc/nginx/sites-available/ffsecurity-nginx.conf /etc/nginx/sites-
↪enabled/

sudo nginx -s reload
```

10. Provide licensing. Use the FindFace Multi main web interface to *upload the license file* you have prior received from your manager (*Preferences* -> *License*). For the on-premise licensing, insert the USB dongle into a USB port.

Important: To log in for the first time, use the default Super Administrator account `admin:admin`. To create more users or change the Super Administrator password, refer to *User Management*.

Important: To preserve the FindFace Multi compatibility with the installation environment, we highly recommend you to disable the Ubuntu automatic update. In this case, you will be able to update your OS manually, fully controlling which packages to update.

To disable the Ubuntu automatic update, execute the following commands:

```
sudo apt-get remove unattended-upgrades
sudo systemctl stop apt-daily.timer
sudo systemctl disable apt-daily.timer
sudo systemctl disable apt-daily.service
sudo systemctl daemon-reload
```

1.4.3 Additional `findface-video-worker` deployment on remote hosts

Important: Before deploying `findface-video-worker` instances on remote hosts, do the following:

1. Allow accessing the `findface-ntls` license server from any IP address. To do so, open the `/etc/findface-ntls.cfg` configuration file on the server with `findface-ntls` and set `listen = 0.0.0.0:3133`. Restart the `findface-ntls` service.

   ```
   sudo vi /etc/findface-ntls.cfg

   ## Address to accept incoming client connections (IP:PORT)
   ## type:string env:CFG_LISTEN longopt:--listen
   listen = 0.0.0.0:3133
   ```

   ```
   sudo systemctl restart findface-ntls.service
   ```

2. Allow accessing the `findface-video-manager` service from any IP address. To do so, open the `/etc/findface-video-manager.conf` configuration file on the server with `findface-video-manager` and set `listen:  0.0.0.0:18810` and `rpc:listen:  0.0.0.0:18811`. Restart the `findface-video-manager` service.

   ```
   sudo vi /etc/findface-video-manager.conf

   listen: 0.0.0.0:18810
   ...
   rpc:
     listen: 0.0.0.0:18811
   ```

   ```
   sudo systemctl restart findface-video-manager.service
   ```

3. On the FindFace Multi server, open the `/etc/findface-security/config.py` configuration file and make sure that the `ROUTER_URL` parameter contains the external IP address of the FindFace Multi server and not the localhost. The `findface-video-worker` instances on the remote hosts will be using this address for posting objects.

```
sudo vi /etc/findface-security/config.py

...

'ROUTER_URL': 'http://192.168.0.12',

...
```

To install only a `findface-video-worker` service, do the following:

Tip: Before deployment, be sure to consult the *system requirements*.

Tip: If you have several video cards on your server, see *Multiple Video Cards Usage* before deploying `findface-video-worker-gpu`.

1. Download the installer file `findface-security-and-server-5.0.run`.
2. Put the `.run` file into some directory on the designated host (for example, `/home/username`).
3. From this directory, make the `.run` file executable.

   ```
   chmod +x findface-security-and-server-5.0.run
   ```

4. Execute the `.run` file.

   ```
   sudo ./findface-security-and-server-5.0.run
   ```

   The installer will ask you a few questions and perform several automated checks to ensure that the host meets the system requirements. Fill out the prompts appropriately once requested. The questions are the following:

   1. Product to install: FindFace Video Worker.
   2. Type of `findface-video-worker` package: CPU or GPU.
   3. IP address of the `findface-security` host.

   After that, the installation process will automatically begin.

   Note: The answers will be saved to a file `/tmp/<findface-installer-*>.json`. You can edit this file and use it to install FindFace Multi on other hosts without having to answer the questions again.

Note: If you chose to install `findface-ntls` and/or `findface-video-manager` on different hosts than that with `findface-security`, specify their IP addresses in the `/etc/findface-video-worker-cpu.ini` (`/etc/findface-video-worker-gpu.ini`) configuration file after the installation.

```
sudo vi /etc/findface-video-worker-cpu.ini
sudo vi /etc/findface-video-worker-gpu.ini
```

In the `ntls-addr` parameter, specify the `findface-ntls` host IP address.

```
ntls-addr=127.0.0.1:3133
```

In the `mgr-static` parameter, specify the `findface-video-manager` host IP address, which provides `findface-video-worker` with settings and the video stream list.

```
mgr-static=127.0.0.1:18811
```

Tip: To automatically install `findface-video-worker` on another host without answering the installation questions, use the `/tmp/<findface-installer-*>.json` file. Execute:

```
sudo ./findface-security-and-server-5.0.run -f /tmp/<findface-installer-*>.json
```

You can find an example of the installation file in *Installation File*.

Important: To preserve the FindFace Multi compatibility with the installation environment, we highly recommend you to disable the Ubuntu automatic update. In this case, you will be able to update your OS manually, fully controlling which packages to update.

To disable the Ubuntu automatic update, execute the following commands:

```
sudo apt-get remove unattended-upgrades
sudo systemctl stop apt-daily.timer
sudo systemctl disable apt-daily.timer
sudo systemctl disable apt-daily.service
sudo systemctl daemon-reload
```

Important: The FindFace Multi services log a large amount of data, which can eventually lead to disc overload. To prevent this from happening, we advise you to disable `rsyslog` due to its suboptimal log rotation scheme and use the appropriately configured `systemd-journal` service instead. See *Service Logs* for the step-by-step instructions.

1.4.4 Installation of Neural Network Models

To detect and recognize objects and object attributes, `findface-extraction-api` uses neural networks.

If you want to manually initiate the installation of neural network models, use the console installer as follows:

1. Execute the prepared `findface-multi-1.0-and-server-5.0.1.run` file.

   ```
   sudo ./findface-multi-1.0-and-server-5.0.1.run
   ```

2. Select the installation type: `Fully customized installation`.
3. Select a FindFace Multi component to install: `findface-data`. To do so, first deselect all the listed components by entering `-*` in the command line, then select the required component by entering its sequence number (keyword). Enter `done` to save your selection and proceed to another step.
4. In the same manner, select models to install. After that, the installation process will automatically begin.

You can find installed models for the object and object attribute recognition at `/usr/share/findface-data/models/`. See *Neural Network Models*.

1.4.5 Fully Customized Installation

The FindFace Multi developer-friendly installer provides you with quite a few installation options, including the fully customized installation. This option is mostly used when deploying FindFace Multi in a highly distributed environment.

To initiate the fully customized installation, answer the installer questions as follows:

- Product to install: `FindFace Multi`.
- Installation type: `Fully customized installation`.
- FindFace Multi components to install: whenever you have to make a selection, first deselect all the listed components by entering `-*` in the command line, then select required components by entering their sequence number (keyword), for example: `1 7 13`, etc. Enter `done` to save your selection and proceed to another step.
- Related questions such as about the acceleration type: CPU or GPU.

1.4.6 Guide to Typical Cluster Installation

This section is all about deploying FindFace Multi in a cluster environment.

Tip: If after having read this section, you still have questions, do not hesitate to contact our experts by support@ntechlab.com.

The reasons for deploying FindFace Multi in a cluster are the following:

- The necessity to distribute the video processing high load.
- The necessity to process video streams from a group of cameras in the place of their physical location.

  Note: The most common use cases where such need comes to the fore are hotel chains, chain stores, several security checkpoints in the same building, etc.

  See also:

  Allocate findface-video-worker to Camera Group

- The necessity to distribute the feature vector extraction high load.
- Large number of objects to search through, that requires implementation of a distributed object database.

Before you start the deployment, outline your system architecture, depending on its load and allotted resources (see *Requirements*). The most common distributed scheme is as follows:

- One principal server with the following components: `findface-ntls`, `findface-security`, `findface-sf-api`, `findface-video-manager`, `findface-upload`, `findface-video-worker`, `findface-extraction-api`, `findface-tarantool-server`, and third-parties.
- Several additional video processing servers with installed `findface-video-worker`.
- (If needed) Several additional extraction servers with installed `findface-extraction-api`.
- (If needed) Additional database servers with multiple Tarantool shards.

This section describes the most common distributed deployment. In high load systems, it may also be necessary to distribute the API processing (`findface-sf-api` and `findface-video-manager`) across several additional servers. In this case, refer to *Fully Customized Installation*.

To deploy FindFace Multi in a cluster environment, follow the steps below:

Deploy Principal Server

To deploy the principal server as part of a distributed architecture, do the following:

1. On the designated physical server, *install* FindFace Multi from installer as follows:

   - Product to install: `FindFace Multi`.
   - Installation type: `Single server, multiple video workers`. In this case, FindFace Multi will be installed and configured to interact with additional remote `findface-video-worker` instances.
   - Type of the `findface-video-worker` acceleration (on the principal server): CPU or GPU, subject to your hardware configuration.
   - Type of the `findface-extraction-api` acceleration (on the principal server): CPU or GPU, subject to your hardware configuration.

   After the installation is complete, the following output will be shown on the console:

```
#############################################################################
#                       Installation is complete                            #
#############################################################################
- upload your license to http://172.20.77.17/#/license/
- user interface: http://172.20.77.17/
  superuser:      admin
  password:       admin
  documentation:  http://172.20.77.17/doc/
```

2. Upload the FindFace Multi license file via the main web interface `http://<Host_IP_address>/#/license`. To access the web interface, use the provided `admin` credentials.

   Note: The host IP address is shown in the links to FindFace web services in the following way: as an external IP address if the host belongs to a network, or `127.0.0.1` otherwise.

   Important: Do not disclose the superuser (Super Administrator) credentials to others. To administer the system, create a new user with the administrator privileges. Whatever the role, Super Administrator cannot be deprived of its rights.

3. Allow the licensable services to access the `findface-ntls` license server from any IP address, To do so, open the `/etc/findface-ntls.cfg` configuration file and set `listen = 0.0.0.0:3133`. Restart `findface-ntls.service`.

```
sudo vi /etc/findface-ntls.cfg

## Address to accept incoming client connections (IP:PORT)
## type:string env:CFG_LISTEN longopt:--listen
listen = 0.0.0.0:3133
```

```
sudo systemctl restart findface-ntls.service
```

4. Allow accessing the `findface-video-manager` service from any IP address. To do so, open the `/etc/findface-video-manager.conf` configuration file and set `listen: 0.0.0.0:18810` and `rpc:listen: 0.0.0.0:18811`. Restart the `findface-video-manager` service.

```
sudo vi /etc/findface-video-manager.conf

listen: 0.0.0.0:18810
...
rpc:
  listen: 0.0.0.0:18811
```

```
sudo systemctl restart findface-video-manager.service
```

Deploy Video Processing Servers

On an additional video processing server, install only a `findface-video-worker` instance following the *step-by-step instructions*. Answer the installer questions as follows:

- Product to install: FindFace Video Worker.
- Type of the `findface-video-worker` acceleration: CPU or GPU, subject to your hardware configuration.
- FindFace Multi IP address: IP address of the principal server.

After that, the installation process will automatically begin. The answers will be saved to a file `/tmp/<findface-installer-*>.json`. Use this file to install `FindFace Video Worker` on other hosts without having to answer the questions again, by executing:

```
sudo ./findface-multi-1.0-and-server-5.0.1.run -f /tmp/<findface-installer-*>.
↪json
```

Note: If `findface-ntls` and/or `findface-video-manager` are installed on a different host than that with `findface-security`, specify their IP addresses in the `/etc/findface-video-worker-gpu.ini` (`/etc/findface-video-worker-cpu.ini`) configuration file after the installation.

```
sudo vi /etc/findface-video-worker-cpu.ini
sudo vi /etc/findface-video-worker-gpu.ini
```

In the `ntls-addr` parameter, specify the `findface-ntls` host IP address.

```
ntls-addr=127.0.0.1:3133
```

In the `mgr-static` parameter, specify the `findface-video-manager` host IP address, which provides `findface-video-worker` with settings and the video stream list.

```
mgr-static=127.0.0.1:18811
```

Deploy Extraction Servers

On an additional extraction server, install only a `findface-extraction-api` instance from the console installer. Answer the installer questions as follows:

- Product to install: `FindFace Multi`.
- Installation type: `Fully customized installation`.
- FindFace Multi components to install: `findface-extraction-api` and `findface-data`. To make a selection, first deselect all the listed components by entering `-*` in the command line, then select `findface-extraction-api` and `findface-data` by entering their sequence number (keyword): `1 7`. Enter `done` to save your selection and proceed to another step.
- Type of `findface-extraction-api` acceleration: CPU or GPU.
- Modification of the `/etc/findface-extraction-api.ini` configuration file: specify the IP address of the `findface-ntls` server.
- Neural network models to install: CPU or GPU model for face biometrics (mandatory), and (optional) CPU/GPU models to recognize face attributes, car and car attributes, and body and body attributes. To make a selection, first deselect all the listed models by entering `-*` in the command line, then select required models by entering their sequence number (keyword), for example, `8 2 7`. Enter `done` to save your selection and proceed to another step. Be sure to choose the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models.

  Tip: See *Neural Network Models*, *Enable Face Attribute Recognition*, *Enable Car and Car Attribute Recognition*, *Enable Body and Body Attribute Recognition* for details.

After that, the installation process will automatically begin. The answers will be saved to a file `/tmp/<findface-installer-*>.json`. Use this file to install `findface-extraction-api` on other hosts without having to answer the questions again.

```
sudo ./findface-multi-1.0-and-server-5.0.1.run -f /tmp/<findface-installer-*>.
↪json
```

After all the extraction servers are deployed, distribute load across them by using a *load balancer*.

Distribute Load across Extraction Servers

To distribute load across several extraction servers, you need to set up load balancing. The following step-by-step instructions demonstrate how to set up `nginx` load balancing in a round-robin fashion for 3 `findface-extraction-api` instances located on different physical hosts: one on the FindFace Multi principal server (`172.168.1.9`), and 2 on additional remote servers (`172.168.1.10`, `172.168.1.11`). Should you have more extraction servers in your system, load-balance them by analogy.

Tip: You can use any load balancer according to your preference. Please refer to the relevant official documentation for guidance.

To set up load balancing, do the following:

1. Designate the FindFace Multi principal server (recommended) or any other server with nginx as a gateway to all the extraction servers.

   Important: You will have to specify the gateway server IP address when configuring the FindFace Multi *network*.

   Tip: You can install nginx as such:

   ```
   sudo apt update
   sudo apt install nginx
   ```

2. On the gateway server, create a new nginx configuration file.

   ```
   sudo vi /etc/nginx/sites-available/extapi
   ```

3. Insert the following entry into the just created configuration file. In the `upstream` directive (`upstream extapibackends`), substitute the exemplary IP addresses with the actual IP addresses of the extraction servers. In the `server` directive, specify the gateway server listening port as `listen`. You will have to enter this port when configuring the FindFace Multi *network*.

   ```
   upstream extapibackends {
           server 172.168.1.9:18666; ## ``findface-extraction-api`` on principal server
           server 172.168.1.10:18666; ## 1st additional extraction server
           server 127.168.1.11:18666; ## 2nd additional extraction server
   }
   server {
           listen 18667;
           server_name extapi;
           client_max_body_size 64m;
           location / {
                   proxy_pass http://extapibackends;
                   proxy_next_upstream error;
           }
           access_log /var/log/nginx/extapi.access_log;
           error_log /var/log/nginx/extapi.error_log;
   }
   ```

4. Enable the load balancer in nginx.

   ```
   sudo ln -s /etc/nginx/sites-available/extapi /etc/nginx/sites-enabled/
   ```

5. Restart nginx.

   ```
   sudo service nginx restart
   ```

6. On the principal server and each additional extraction server, open the `/etc/findface-extraction-api.ini` configuration file. Substitute localhost in the `listen` parameter with the relevant server address that you have specified in `upstream extapibackends` (`/etc/nginx/sites-available/extapi`) before. In our example, the address of the 1st additional extraction server has to be substituted as such:

```
sudo vi /etc/findface-extraction-api.ini

listen: 172.168.1.10:18666
```

7. Restart the `findface-extraction-api` on the principal server and each additional extraction server.

```
sudo systemctl restart findface-extraction-api.service
```

The load balancing is now successfully set up. Be sure to specify the actual gateway server IP address and listening port, when configuring the FindFace Multi *network*.

Distribute Database

The `findface-tarantool-server` component connects the Tarantool database and the `findface-sf-api` component, transferring search results from the database to `findface-sf-api` for further processing. To increase search speed, multiple `findface-tarantool-server` shards can be created on each Tarantool host. Their running concurrently leads to a remarkable increase in performance. Each shard can handle up to approximately 10,000,000 feature vectors. When deploying `findface-tarantool-server` from installer, shards are created automatically given the server hardware.

To distribute the face database, install only a `findface-tarantool-server` instance on each additional database server. Answer the installer questions as follows:

- Product to install: `FindFace Multi`.
- Installation type: `Fully customized installation`.
- FindFace Multi components to install: `findface-tarantool-server`. To make a selection, first deselect all the listed components by entering `-*` in the command line, then select `findface-tarantool-server` by entering its sequence number (keyword). Enter `done` to save your selection and proceed to another step.

After that, the installation process will automatically begin. The answers will be saved to a file `/tmp/<findface-installer-*>.json`. Use this file to install `findface-tarantool-server` on other hosts without having to answer the questions again.

```
sudo ./findface-multi-1.0-and-server-5.0.1.run -f /tmp/<findface-installer-*>.json
```

As a result of the installation, `findface-tarantool-server` shards will be automatically installed in the amount of `N = min(max(min(mem_mb // 2000, cpu_cores), 1), 16 * cpu_cores)`. I.e., it is equal to the RAM size in MB divided by 2000, or the number of CPU physical cores (but at least one shard), or the number of CPU physical cores multiplied by 16, should the first obtained value be greater.

Be sure to specify the shards IP addresses and ports, when configuring the FindFace Multi *network*. To learn the port numbers, execute on each database server:

```
sudo cat /etc/tarantool/instances.enabled/*shard* | grep -E ".start|(listen =)"`
```

You will get the following result:

```
    listen = '127.0.0.1:33001',
FindFace.start("127.0.0.1", 8101, {
    listen = '127.0.0.1:33002',
FindFace.start("127.0.0.1", 8102, {
```

You can find the port number of a shard in the `FindFace.start` section, for example, `8101`, `8102`, etc.

Configure Network

After all the FindFace Multi components are deployed, configure their interaction over the network. Do the following:

1. Open the `/etc/findface-sf-api.ini` configuration file:

```
sudo vi /etc/findface-sf-api.ini
```

Specify the following parameters:

Parameter	Description
`extraction-api -> extraction-api`	IP address and listening port of the *gateway extraction server* with set up load balancing.
`storage-api -> shards -> master`	IP address and port of the `findface-tarantool-server` master shard. Specify each shard by analogy.
`upload_url`	WebDAV NginX path to send original images, thumbnails and normalized face images to the `findface-upload` service.

```
...
extraction-api:
  extraction-api: http://172.168.1.9:18667

...
webdav:
  upload-url: http://127.0.0.1:3333/uploads/

...
storage-api:
  ...
  shards:
  - master: http://172.168.1.9:8101/v2/
    slave: ''
  - master: http://172.168.1.9:8102/v2/
    slave: ''
  - master: http://172.168.1.12:8101/v2/
    slave: ''
  - master: http://172.168.1.12:8102/v2/
    slave: ''
  - master: http://172.168.1.13:8102/v2/
    slave: ''
  - master: http://172.168.1.13:8102/v2/
    slave: ''
```

2. Open the `/etc/findface-security/config.py` configuration file.

```
sudo vi /etc/findface-security/config.py
```

Specify the following parameters:

Parameter	Description
SERVICE_EXTERNAL_ADDRESS	FindFace Multi IP address or URL prioritized for the Genetec integration and webhooks. Once this parameter not specified, the system uses EXTERNAL_ADDRESS for these purposes. To use Genetec and webhooks, be sure to specify at least one of those parameters: SERVICE_EXTERNAL_ADDRESS, EXTERNAL_ADDRESS.
EXTERNAL_ADDRESS	(Optional) IP address or URL that can be used to access the FindFace Multi web interface. Once this parameter not specified, the system auto-detects it as the external IP address. To access FindFace Multi, you can use both the auto-detected and specified IP addresses.
VIDEO_DETECTOR_TOKEN	To authorize the video object detection module, come up with a token and specify it here.
VIDEO_MANAGER_ADDRESS	IP address of the findface-video-manager host.
NTLS_HTTP_URL	IP address of the findface-ntls host.
ROUTER_URL	External IP address of the findface-security host that will receive detected objects from the findface-video-worker instance(s).
SF_API_ADDRESS	IP address of the findface-sf-api host.

```
sudo vi /etc/findface-security/config.py

...
# SERVICE_EXTERNAL_ADDRESS prioritized for webhooks and genetec
SERVICE_EXTERNAL_ADDRESS = 'http://localhost'
EXTERNAL_ADDRESS = 'http://127.0.0.1'


...
FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '7ce2679adfc4d74edcf508bea4d67208',
    ...
    'VIDEO_MANAGER_ADDRESS': 'http://127.0.0.1:18810',
    ...
    'NTLS_HTTP_URL': 'http://127.0.0.1:3185',
    'ROUTER_URL': 'http://172.168.1.9',
    ...
    'SF_API_ADDRESS': 'http://127.0.0.1:18411',
    ...
}
```

The FindFace Multi components interaction is now set up.

Important: To preserve the FindFace Multi compatibility with the installation environment, we highly recommend you to disable the Ubuntu automatic update. In this case, you will be able to update your OS manually, fully controlling which packages to update.

To disable the Ubuntu automatic update, execute the following commands:

```
sudo apt-get remove unattended-upgrades
sudo systemctl stop apt-daily.timer
sudo systemctl disable apt-daily.timer
sudo systemctl disable apt-daily.service
sudo systemctl daemon-reload
```

Important: The FindFace Multi services log a large amount of data, which can eventually lead to disc overload. To prevent this from happening, we advise you to disable `rsyslog` due to its suboptimal log rotation scheme and use the appropriately configured `systemd-journal` service instead. See *Service Logs* for the step-by-step instructions.

1.4.7 Add NVIDIA Repository and Install Drivers (GPU only)

FindFace Multi on GPU requires the prior installation of NVIDIA drivers.

To add the NVIDIA repository and install the drivers, do the following:

1. Download the installer file `findface-multi-1.0-and-server-5.0.1.run`.
2. Put the `.run` file into some directory on the designated host (for example, `/home/username`).
3. From this directory, make the `.run` file executable.

```
chmod +x findface-multi-1.0-and-server-5.0.1.run
```

4. Execute the `.run` file.

```
sudo ./findface-multi-1.0-and-server-5.0.1.run
```

5. Choose the product to install: `NVIDIA CUDA drivers`.

1.5 Maintenance and Troubleshooting

1.5.1 Upgrade FindFace Security to FindFace Multi

If you are already using our product FindFace Security, you can upgrade it to FindFace Multi.

Important: Your current license will determine the scope of the features activated in FindFace Multi after the upgrade.

Warning: When updating from FindFace Security 4.2 or earlier, be aware that *offline videos* will be lost and cannot be re-processed in FindFace Multi with different parameters. If you use this functionality, first delete the videos in the FindFace Security web interface. Then, after the system update, re-upload and re-process the videos that you want to keep in FindFace Multi.

If you don't do so, then after the update, you will still be able to see the video-related events but without association with the specific video.

You can find the videos at `/var/lib/findface-security/uploads/videos/` (default path). The actual path to the directory is specified in the `MEDIA_ROOT` parameter in `/etc/findface-security/conf.py`. We recommend you delete the videos if you no longer need them.

To upgrade FindFace Security from any previous version to FindFace Multi, do the following:

1. Open the `/etc/findface-security/config.py` (`/etc/ffsecurity/config.py`) configuration file. Save the values of the following parameters for later use: `EXTERNAL_ADDRESS`, `SECRET_KEY`, `VIDEO_DETECTOR_TOKEN`, `ROUTER_URL`.

```
# FindFace Security 4.2 and earlier

sudo vi /etc/ffsecurity/config.py

# FindFace Security 4.3 and later

sudo vi /etc/findface-security/config.py

EXTERNAL_ADDRESS = "http://172.20.77.58"

...
# use pwgen -sncy 50 1|tr "'" "." to generate your own unique key
SECRET_KEY = 'c8b533847bbf7142102de1349d33a1f6'

FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '381b0f4a20495227d04185ab02f5085f',
    ...
    'ROUTER_URL': 'http://172.20.77.58',
    ...
}
```

2. Stop the `findface-security` service.

```
sudo systemctl stop findface-security*.service
```

3. Create a backup of the Tarantool-based feature vector database in any directory of your choice, for example, `/etc/findface_dump`.

Tip: See *Back Up and Recover Data Storages* for details.

```
sudo mkdir -p /etc/findface_dump
cd /etc/findface_dump
sudo findface-storage-api-dump -config /etc/findface-sf-api.ini
```

4. Install the apt repository with FindFace Multi, using the console installer as described in *this section*.

5. If you are upgrading from FindFace Security version 4.1.2 or earlier, do the following:

   1. Give a strong password to the `ntech` user (`9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3` in the example below). Output the credentials to the `pgbouncer` user list.

   ```
   echo '"ntech" "9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3"' | sudo tee -a /etc/pgbouncer/
   ↪userlist.txt
   ```

   2. Install the `pgbouncer` package as such:

   ```
   sudo apt update
   sudo apt install -y pgbouncer
   ```

   3. Configure `pgbouncer`. In `/etc/pgbouncer/pgbouncer.ini`, add `ffsecurity` to the `databases` section. Configure named parameters, as shown in the example below. Parameters other than those must be commented out.

```
sudo vi /etc/pgbouncer/pgbouncer.ini

[databases]
ffsecurity = dbname=ffsecurity host=localhost port=5432 user=ntech
[pgbouncer]
pidfile = /var/run/postgresql/pgbouncer.pid
listen_addr = 127.0.0.1
listen_port = 5439
unix_socket_dir = /var/run/postgresql
auth_type = plain
auth_file = /etc/pgbouncer/userlist.txt
pool_mode = transaction
server_reset_query = DISCARD ALL
max_client_conn = 16384
default_pool_size = 20
syslog = 1
```

4. Copy the password of the `ntech` user (`9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3` in the example). In **`PostgreSQL`**, set the copied password for the `ntech` role. Open the **`PostgreSQL`** interactive terminal. You will see the line `postgres=#` appear. After the `#` sign, enter the following command: `ALTER ROLE ntech PASSWORD '<copied password>'`.

```
sudo -u postgres psql

postgres=# ALTER ROLE ntech PASSWORD '9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3';
```

5. Using the **`PostgreSQL`** interactive terminal, create a database `ffcounter` in PostgreSQL.

```
postgres=# CREATE DATABASE ffcounter WITH OWNER ntech ENCODING 'UTF-8' LC_
↪COLLATE='C.UTF-8' LC_CTYPE='C.UTF-8' TEMPLATE template0;
```

6. Restart **`PostgreSQL`**.

```
sudo systemctl restart postgresql@10-main.service
```

7. Create and configure `pgbouncer.service`.

```
sudo touch /etc/systemd/system/pgbouncer.service
sudo vi /etc/systemd/system/pgbouncer.service
```

Insert the following code:

```
[Unit]
Description=Pgbouncer service
After=postgresql.service
Before=findface-security.service
[Service]
User=postgres
Group=postgres
ExecStart=/usr/sbin/pgbouncer "/etc/pgbouncer/pgbouncer.ini"
[Install]
WantedBy=multi-user.target
```

8. Enable the `pgbouncer.service` autostart and restart it:

```
sudo systemctl enable pgbouncer.service
sudo systemctl restart pgbouncer.service
```

6. Install the FindFace Multi services from the repository, following your architecture outline.

   CPU-version:

```
sudo apt update
sudo apt install findface-security findface-security-ui findface-extraction-api␣
↪findface-ntls findface-sf-api findface-tarantool-server findface-upload findface-
↪video-manager findface-video-worker-cpu findface-counter findface-liveness-api
```

   GPU-version:

```
sudo apt update
sudo apt install findface-security findface-security-ui findface-extraction-api-gpu␣
↪findface-ntls findface-sf-api findface-tarantool-server findface-upload findface-
↪video-manager findface-video-worker-gpu findface-counter findface-liveness-api
```

Important: FindFace Multi on GPU requires the prior installation of *NVIDIA drivers*.

Important: At some moment, you will be prompted to choose which version of the `findface-security` configuration file to keep. Opt for `Install the packages maintainer's version`.

7. Enable the `findface-counter` and `findface-liveness-api` services autostart.

```
sudo systemctl enable findface-counter.service findface-liveness-api.service
```

8. Open the `/etc/findface-security/config.py` configuration file and paste the saved `EXTERNAL_ADDRESS`, `SECRET_KEY`, `VIDEO_DETECTOR_TOKEN`, and `ROUTER_URL` into it. Fill in the `DATABASES` section by analogy: `'PORT': 5439, 'USER': 'ntech', 'PASSWORD': '9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3'` (password from `/etc/pgbouncer/userlist.txt`).

```
sudo vi /etc/findface-security/config.py
...
# Database is used by FindFace Security to store cameras,
# camera groups, watchlists and so on. Only PostgreSQL is supported.
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'DISABLE_SERVER_SIDE_CURSORS': True,
        'NAME': 'ffsecurity',
        'PORT': 5439, 'USER': 'ntech', 'PASSWORD': '9T3g1nXy9yx3y8MIGm9fbef3dia8UTc3
↪',
    }
}
...
# Use pwgen -sncy 50 1|tr "'" "." to generate your own unique key
SECRET_KEY = '002231ccb690586f4d33e98322c591bb'
...
SERVICE_EXTERNAL_ADDRESS = 'http://172.20.77.58'
```

(continues on next page)

(continued from previous page)

```
# EXTERNAL_ADDRESS is used to access objects created inside FFSecurity via external␣
↪links.
EXTERNAL_ADDRESS = 'http://172.20.77.58'
...
    # findface-video-worker authorization token
    'VIDEO_DETECTOR_TOKEN': '8977e1b0067d43f6c908d0bf60363255',
...
    # findface-video-worker face posting address,
    # it must be set to either FFSecurity EXTERNAL_ADDRESS (by default)
    # or findface-facerouter url (in some specific cases)
    'ROUTER_URL': 'http://127.0.0.1:80',
```

9. Open the old version of the `findface-ntls` configuration file available at `/etc/findface-ntls.cfg.dpkg-old` and check it against the new version `/etc/findface-ntls.cfg`. Make sure that all the custom parameters from the old version are present in the new one. Do the same for other components, e.g. for `findface-extraction-api`, check `/etc/findface-extraction-api.ini.ucf-old` against `/etc/findface-extraction-api.ini`, etc.

```
sudo vi /etc/findface-ntls.cfg.dpkg-old
sudo vi /etc/findface-ntls.cfg
sudo vi /etc/findface-extraction-api.ini.ucf-old
sudo vi /etc/findface-extraction-api.ini

...
```

10. Restart the services.

```
sudo systemctl restart findface-ntls findface-extraction-api findface-video-worker*␣
↪findface-video-manager findface-sf-api findface-counter findface-liveness-api
```

11. Modify the Tarantool database structure by applying the `tnt_schema.lua` file from FindFace Multi.

```
sudo findface-security make_tnt_schema | sudo tee /etc/findface-security/tnt_schema.
↪lua
```

12. Stop the `findface-tarantool-server` shards. Purge data from all the directories relevant to active shards.

```
sudo systemctl stop 'tarantool@*'

sudo rm /opt/ntech/var/lib/tarantool/shard-*/{index,snapshots,xlogs}/*
```

13. Navigate to the directory with Tarantool configuration file(s) `/etc/tarantool/instances.available/`. Check whether each configuration file `shard-*.lua` contains the `dofile` command, `meta_indexes` and `meta_scheme` definitions, as in the example below.

```
sudo vi /etc/tarantool/instances.available/shard-*.lua

...
dofile("/etc/findface-security/tnt_schema.lua")
...
FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    meta_indexes=meta_indexes,
```

(continues on next page)

(continued from previous page)

```
    meta_scheme = meta_scheme
})
```

14. Restart the `findface-tarantool-server` shards.

```
TNT=$(ls /etc/tarantool/instances.enabled/ | cut -c 7,8,9)
for i in $TNT; do sudo systemctl restart tarantool@shard-$i.service ; done
```

15. Restore the Tarantool database from the backup.

```
cd /etc/findface_dump

for x in *.json; do sudo findface-storage-api-restore -config /etc/findface-sf-api.
↪ini < "$x"; done
```

16. Migrate the old data from the `v2` to `v3` API version of the `findface-sf-api` service.

```
sudo findface-sf-api-migrate-v2-v3 -config /etc/findface-sf-api.ini
```

17. Migrate the main database architecture from FindFace Multi to **`PostgreSQL`**, re-create user groups with *predefined* rights, and the first user with administrator rights.

```
sudo findface-security migrate
sudo findface-security create_groups
sudo findface-security create_default_user
```

18. Restart the `findface-security` service.

```
sudo systemctl restart findface-security.service
```

Important: To preserve the FindFace Multi compatibility with the installation environment, we highly recommend you to disable the Ubuntu automatic update. In this case, you will be able to update your OS manually, fully controlling which packages to update.

To disable the Ubuntu automatic update, execute the following commands:

```
sudo apt-get remove unattended-upgrades
sudo systemctl stop apt-daily.timer
sudo systemctl disable apt-daily.timer
sudo systemctl disable apt-daily.service
sudo systemctl daemon-reload
```

1.5.2 Back Up and Recover Data Storages

This section is all about the backup and recovery of the FindFace Multi data storages, which are the following:

- Tarantool-based feature vector database that stores feature vectors and object identification events.
- Main system database `ffsecurity` based on PostgreSQL, that stores internal system data, dossiers, user accounts, and camera settings.
- Directory `/var/lib/findface-security/uploads` that stores uploaded dossier photos, video files, and such event artifacts as full frames, object thumbnails, and normalized object images.
- Directory `/var/lib/ffupload/` that stores only such event artifacts as object thumbnails.

In this section:

Feature Vector Database Backup and Recovery

There are the following galleries in the Tarantool-based feature vector database:

- `ffsec_body_events`: feature vectors extracted from bodies detected in the video.
- `ffsec_body_objects`: feature vectors extracted from bodies in dossier photos.
- `ffsec_car_events`: feature vectors extracted from cars detected in the video.
- `ffsec_car_objects`: feature vectors extracted from cars in dossier photos.
- `ffsec_face_events`: feature vectors extracted from faces detected in the video.
- `ffsec_face_objects`: feature vectors extracted from faces in dossier photos.
- `ffsec_user_face`: feature vectors extracted from photos of FindFace Multi users, used for face-based authentication.
- `ffsec_persons`: centroids of persons (virtual feature vectors averaged across all person's faces) and metadata.

The database backup/recovery functionality allows you to fully restore all the galleries when needed.

To avoid data loss, we recommend you back up a feature vector database at least once a week. Overall, the backups' frequency depends on the number of dossiers and object recognition events, and available disk space.

Be sure to back up the database before *migrating* your system to another neural network model.

Utilities

To back up and recover the FindFace Multi feature vector database, the following utilities are needed:

1. backup: `findface-storage-api-dump`,
2. recovery: `findface-storage-api-restore`.

These utilities are automatically installed along with `findface-sf-api`.

Back Up Database

To back up the feature vector database, use the `findface-storage-api-dump` utility as follows:

Important: The following services must be active: `findface-tarantool-server`, `findface-sf-api`.

Note: The backup functionality can be applied to a distributed database. In this case, the `findface-storage-api-dump` utility will back up galleries on all the shards specified in `/etc/findface-sf-api.ini`.

1. On the server with `findface-sf-api`, create a directory to store the backup files (`/etc/findface_dump` in the example below).
2. Launch the `findface-storage-api-dump` utility by executing:

   ```
   sudo findface-storage-api-dump -output-dir=/etc/findface_dump -config /etc/findface-
   ↪sf-api.ini
   ```

   The utility will back up at once all the galleries into the files with corresponding names (`ffsec_body_events.json`, `ffsec_face_events`, etc.) and save them into the directory. These files contain all the data needed to restore the entire database.

Recover Database

To recover the feature vector database from the backup, launch the `findface-storage-api-restore` utility for all the files in the backup folder:

```
sudo findface-storage-api-restore -config /etc/findface-sf-api.ini /etc/findface_dump/*.
↪json
```

The recovery process can be interrupted and resumed whenever necessary. To resume the process after the interruption, launch the `findface-storage-api-restore` utility again.

See also:

- *Backup Options*
- *Restore Options*

Main Database Backup and Recovery

To back up the main database `ffsecurity` based on PostgreSQL, execute:

```
sudo -u postgres pg_dump ffsecurity > ffsecurity_postgres_backup.sql
```

To recover the main database, do the following:

1. Stop the `findface-security` service.

   ```
   sudo systemctl stop findface-security.service
   ```

2. Stop the `pgbouncer` service to delete its active sessions with the `ffsecurity` database.

   ```
   sudo systemctl stop pgbouncer.service
   ```

3. Open the PostgreSQL interactive terminal.

   ```
   sudo -u postgres psql
   ```

4. Remove the old `ffsecurity` database.

   ```
   DROP DATABASE ffsecurity;
   ```

5. Create a new `ffsecurity` database. Leave the PostgreSQL interactive terminal.

   ```
   CREATE DATABASE ffsecurity WITH OWNER ntech ENCODING 'UTF-8' LC_COLLATE='C.UTF-8'␣
   ↪LC_CTYPE='C.UTF-8' TEMPLATE template0;
   ```

6. Start the `pgbouncer` service.

   ```
   sudo systemctl start pgbouncer.service
   ```

7. Recover the database content from the backup.

   ```
   sudo -u postgres psql -d ffsecurity -f ffsecurity_postgres_backup.sql
   ```

8. Migrate the database architecture from FindFace Multi to **`PostgreSQL`**, re-create user groups with *predefined* rights and the first user with administrator rights.

   ```
   sudo findface-security migrate
   sudo findface-security create_groups
   sudo findface-security create_default_user
   ```

9. Start the `findface-security` service.

   ```
   sudo systemctl start findface-security.service
   ```

Artifacts Backup and Recovery

The FindFace Multi artifacts, such as uploaded dossier photos, video files, and such event artifacts as full frames, object thumbnails, and normalized object images, are stored in the following directories:

- `/var/lib/findface-security/uploads`
- `/var/lib/ffupload/`

To back up the artifacts, execute:

```
sudo tar -cvzf /home/some_directory/var_lib_ffsecurity_uploads.tar.gz /var/lib/findface-
↪security/uploads/
sudo tar -cvzf /home/some_directory/var_lib_ffupload.tar.gz /var/lib/ffupload/
```

To recover the artifacts, execute the following commands from the root directory:

```
cd /
sudo tar -xvf /home/some_directory/var_lib_ffsecurity_uploads.tar.gz
sudo tar -xvf /home/some_directory/var_lib_ffupload.tar.gz
```

1.5.3 Migrate Face Data to Different Neural Network Model

Tip: Do not hesitate to contact our experts on migration by support@ntechlab.com.

Sometimes you have to migrate the face biometric data to another neural network model, such as when you decide to update to the latest version of the product that uses a different set of neural networks.

To migrate to a different neural network model, do the following:

1. Create a backup of the Tarantool-based feature vector database in any directory of your choice, for example, `/etc/findface_dump`.

   Tip: See *Back Up and Recover Data Storages* for details.

   ```
   mkdir -p /etc/findface_dump
   cd /etc/findface_dump
   sudo findface-storage-api-dump -config /etc/findface-sf-api.ini
   ```

2. Stop the `findface-sf-api` service.

   ```
   sudo systemctl stop findface-sf-api.service
   ```

3. Create new shards that will host regenerated feature vectors.

   1. Open the `/etc/tarantool/instances.available/` directory and find out the number of shards by counting the number of configuration files `shard-*.lua`.

      Note: There are four shards in the example below.

```
cd /etc/tarantool/instances.available/

ls -l

shard-001.lua
shard-002.lua
shard-003.lua
shard-004.lua
```

2. Create the same number of new shards by copying the configuration files `shard-*.lua`.

Note: For convenience, the second digit in the new names is 1: `shard-01*.lua`.

```
sudo cp shard-001.lua shard-011.lua
sudo cp shard-002.lua shard-012.lua
sudo cp shard-003.lua shard-013.lua
sudo cp shard-004.lua shard-014.lua
```

3. Modify the following lines in each new shard's configuration file, depending on its name (`shard-011`, `shard-012`, etc., in our example):

Old value	New value
listen = '127.0.0.1:32001'	Listen = '127.0.0.1:32011'
vinyl_dir = '/opt/ntech/var/lib/tarantool/shard-001'	vinyl_dir = '/opt/ntech/var/lib/tarantool/shard-011'
work_dir = '/opt/ntech/var/lib/tarantool/shard-001'	work_dir = '/opt/ntech/var/lib/tarantool/shard-011'
memtx_dir = '/opt/ntech/var/lib/tarantool/shard-001/snapshots'	memtx_dir = '/opt/ntech/var/lib/tarantool/shard-011/snapshots'
wal_dir = '/opt/ntech/var/lib/tarantool/shard-001/xlogs'	wal_dir = '/opt/ntech/var/lib/tarantool/shard-011/xlogs'
FindFace.start("127.0.0.1", 8101, {	FindFace.start("127.0.0.1", 8111, {

4. Create symbolic links to the new shards.

```
cd /etc/tarantool/instances.enabled/

sudo ln -s /etc/tarantool/instances.available/shard-01*.lua /etc/tarantool/
↪instances.enabled/
```

5. Create directories that will host files of the new shards. Assign permissions for the created directories.

```
cd /opt/ntech/var/lib/tarantool/

mkdir -p shard-01{1..4}/{index,snapshots,xlogs}

chown tarantool:tarantool shard-01* shard-01*/*
```

4. Open the `/etc/findface-extraction-api.ini` configuration file and replace the old neural network model with the new one (`kiwi_320.cpu.fnk` in the example).

```
sudo vi /etc/findface-extraction-api.ini

face: face/jackfruit_480.cpu.fnk -> face: face/kiwi_320.cpu.fnk
```

Restart the `findface-extraction-api` service.

```
sudo systemctl restart findface-extraction-api.service
```

5. Start the new shards.

```
for i in {11..14}; do sudo systemctl start tarantool@shard-0$i; done
```

6. Create a configuration file with migration settings <`migration.ini`> based on the example below.

```
extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 0s
  extraction-api: http://127.0.0.1:18666
storage-api-from: # current location of the gallery
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  shards:
    - master: http://127.0.0.1:8101/v2/
      slave: ""
    - master: http://127.0.0.1:8102/v2/
      slave: ""
    - master: http://127.0.0.1:8103/v2/
      slave: ""
    - master: http://127.0.0.1:8104/v2/
      slave: ""
storage-api-to:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  shards:
    - master: http://127.0.0.1:8111/v2/
      slave: ""
    - master: http://127.0.0.1:8112/v2/
      slave: ""
    - master: http://127.0.0.1:8113/v2/
      slave: ""
    - master: http://127.0.0.1:8114/v2/
      slave: ""
```

(continues on next page)

(continued from previous page)

```
workers_num: 3
faces_limit: 100
extraction_batch_size: 8
normalized_storage:
  type: webdav
  enabled: True
  webdav:
    upload-url: http://127.0.0.1:3333/uploads/
  s3:
    endpoint: ""
    bucket-name: ""
    access-key: ""
    secret-access-key: ""
    secure: False
    region: ""
    public-url: ""
    operation-timeout: 30
```

In the `storage-api-from` section, specify the old shards to migrate the data from.

```
storage-api-from: # current location of the gallery
  ...
  shards:
    - master: http://127.0.0.1:8101/v2/
      slave: ""
    - master: http://127.0.0.1:8102/v2/
      slave: ""
    - master: http://127.0.0.1:8103/v2/
      slave: ""
     - master: http://127.0.0.1:8104/v2/
  ...
```

In the `storage-api-to` section, specify the new shards that will host migrated data.

```
storage-api-to:
  ...
  shards:
    - master: http://127.0.0.1:8111/v2/
      slave: ""
    - master: http://127.0.0.1:8112/v2/
      slave: ""
    - master: http://127.0.0.1:8113/v2/
      slave: ""
    - master: http://127.0.0.1:8114/v2/
      slave: ""
  ...
```

7. Launch the `findface-sf-api-migrate` utility with the `-config` option and provide the `<migration.ini>` configuration file.

```
findface-sf-api-migrate -config migration.ini
```

Note: The migration process can take up a significant amount of time if there are many events and dossiers in the system.

8. After the migration is complete, stop the old shards and disable their autostart in OS (do not remove them).

```
for i in {01..04}; do sudo systemctl stop tarantool@shard-0$i.service ; done

for i in {01..04}; do sudo systemctl disable tarantool@shard-0$i.service ; done
```

9. Open the `/etc/findface-sf-api.ini` configuration file and adjust the shards ports, subject to the new shards settings. Restart the `findface-sf-api` service.

```
sudo vi /etc/findface-sf-api.ini

shards:
  - master: http://127.0.0.1:8111/v2/
    slave: ""
  - master: http://127.0.0.1:8112/v2/
    slave: ""
  - master: http://127.0.0.1:8113/v2/
    slave: ""
  - master: http://127.0.0.1:8114/v2/
    slave: ""

sudo systemctl start findface-sf-api.service
```

10. Import the new database structure from the `tnt_schema.lua` file.

```
sudo findface-security make_tnt_schema | sudo tee /etc/findface-security/tnt_schema.
↪lua
```

See also:

Modify Feature Vector Database Structure.

11. Migrate the main database architecture from FindFace Multi to **PostgreSQL**, re-create user groups with *predefined* rights, and the first user with administrator rights.

```
sudo findface-security migrate
sudo findface-security create_groups
sudo findface-security create_default_user
```

12. Restart the services.

```
sudo systemctl restart findface-security.service
sudo systemctl restart findface-extraction-api findface-video-worker* findface-
↪video-manager findface-sf-api
```

1.5.4 Modify Feature Vector Database Structure

Sometimes it may be necessary to apply a new structural schema to your Tarantool-based feature vector database, for example, when updating to the latest version of the product, or when you want to enhance the default database structure with additional parameters, advanced object metadata, and so on.

In this section:

About Database Structure

In FindFace Multi, the database structure is set via the `/etc/findface-security/tnt_schema.lua` file.

The structure is created as a set of fields. Each field is described with the following parameters:

- id: field id;
- name: field name, must be the same as the name of a relevant object parameter;
- field_type: data type;
- default: field default value. If a default value exceeds '1e14 – 1', use a string data type to specify it, for example, `"123123.."` instead of `123123...`

You can find the default `tnt_schema.lua` file `here`.

Structure Modification

To modify the database structure, do the following:

1. Stop the `findface-security` service.

```
sudo systemctl stop findface-security.service
```

2. Create a backup of the Tarantool-based feature vector database in any directory of your choice, for example, `/etc/findface_dump`.

   Tip: See *Back Up and Recover Data Storages* for details.

```
mkdir -p /etc/findface_dump
cd /etc/findface_dump
sudo findface-storage-api-dump -config /etc/findface-sf-api.ini
```

3. Prepare the `tnt_schema.lua` file containing the new database structure.
4. Modify the database structure by applying the new `tnt_schema.lua` file.

```
sudo findface-security make_tnt_schema | sudo tee /etc/findface-security/tnt_schema.
↪lua
```

5. Navigate to the directory with Tarantool configuration file(s) `/etc/tarantool/instances.available/`. For each shard, make sure that there is a line `dofile("/etc/findface-security/tnt_schema.lua")` before the `FindFace.start` section and `meta_scheme` and `meta_indexes` are defined in the `FindFace.start` parameters.

```
sudo vi /etc/tarantool/instances.available/<shard_00N>.lua

dofile("/etc/findface-security/tnt_schema.lua")

FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    meta_indexes=meta_indexes,
    meta_scheme = meta_scheme
})
```

6. Purge data from all the directories relevant to active shards.

```
sudo rm /opt/ntech/var/lib/tarantool/shard-*/{index,snapshots,xlogs}/*
```

7. Restore the Tarantool database from the backup.

Important: If some fields were removed from the new database structure, you have to first manually delete the corresponding data from the backup copy.

```
cd /tmp/dump
for x in *.json; do curl -X POST "http://127.0.0.1:18411/v2/galleries/${x%%.json}";␣
↪done
for x in *.json; do sudo findface-storage-api-restore -config /etc/findface-sf-api.
↪ini < "$x"; done
```

8. Start the `findface-security` service.

```
sudo systemctl start findface-security.service
```

See also:

Dossier Face Custom Metadata in Tarantool

1.5.5 Remove FindFace Multi Instance

You can automatically remove FindFace Multi along with the database by using the `ffsec_uninstall.sh` script. The FindFace Multi configuration files and database will be backed up.

Do the following:

1. Download the `ffsec_uninstall.sh` script to some directory on a designated host (for example, to `/home/username/`).
2. From this directory, make the script executable.

```
chmod +x ffsec_uninstall.sh
```

3. Run the script.

```
sudo ./ffsec_uninstall.sh
```

4. Answer **all** to completely remove FindFace Multi along with the database.

1.5.6 Check Component Status

Check the status of components once you have encountered a system problem.

Component	Command to view service status
`findface-extraction-api`	sudo systemctl status findface-extraction-api.service
`findface-sf-api`	sudo systemctl status findface-sf-api.service
`findface-tarantool-server`	sudo systemctl status tarantool.service
`findface-tarantool-server` shards	sudo systemctl status tarantool@shard-00*
`findface-video-manager`	sudo systemctl status findface-video-manager.service
`findface-video-worker`	sudo systemctl status findface-video-worker*.service
`findface-ntls`	sudo systemctl status findface-ntls
`findface-security`	sudo systemctl status findface-security.service
`findface-counter`	sudo systemctl status findface-counter.service
`findface-liveness-api`	sudo systemctl status findface-liveness-api.service
`etcd`	sudo systemctl status etcd.service
`NginX`	sudo systemctl status nginx.service
`memcached`	sudo systemctl status memcached.service
`postgresql`	sudo systemctl status postgresql*
`redis`	sudo systemctl status redis.service
`pgbouncer`	sudo systemctl status pgbouncer.service

1.5.7 Service Logs

Service logs provide a complete record of each FindFace Multi component activity. Consulting logs is one of the first things you should do to identify a cause for any system problem.

In this section:

- *Configure Logging*
- *Consult Service Logs*

Configure Logging

The FindFace Multi services log a large amount of data, which can eventually lead to disc overload. To prevent this from happening, we advise you to disable `rsyslog` due to its suboptimal log rotation scheme and use the appropriately configured `systemd-journal` service instead.

Do the following:

1. Check whether the `/var/log/journal` directory already exists. If not, create it by executing the following command:

```
sudo mkdir /var/log/journal
sudo chmod 2755 /var/log/journal
```

2. Open the `/etc/systemd/journald.conf` configuration file. Enable saving `journald` logs to your hard drive by uncommenting the `Storage` parameter and changing its value to `persistent`. Disable filtering in `systemd-journal` as well:

```
sudo vi /etc/systemd/journald.conf

[Journal]
...
Storage=persistent
...
RateLimitInterval=0
RateLimitBurst=0
...
```

   If necessary, uncomment and edit the `SystemMaxUse` parameter. This parameter determines the maximum volume of log files on your hard drive. Specify its value in bytes or use K, M, G, T, P, E as units for the specified size (equal to 1024, 1024^2, $\ldots$ bytes).

```
...
SystemMaxUse=3G
```

3. Restart the `journald` service.

```
sudo systemctl restart systemd-journald.service
```

4. Stop and disable the `syslog` service.

```
sudo systemctl stop syslog.socket rsyslog.service
sudo systemctl disable syslog.socket rsyslog.service
```

5. If necessary, delete the existing log files created through `syslog`, and the kernel logs.

```
sudo rm /var/log/syslog*
sudo rm /var/log/kern.log*
```

Consult Service Logs

Use the `journalctl -u <component>` command to consult a component log, for example as follows:

```
journalctl -u findface-extraction-api
```

See also:

Audit Logs

1.5.8 Troubleshoot Licensing and `findface-ntls`

When troubleshooting licensing and `findface-ntls` (see *Licensing Info*), the first step is to retrieve the licensing information and `findface-ntls` status. You can do so by sending an API request to `findface-ntls`. Necessary actions are then to be undertaken, subject to the response content.

Tip: Please do not hesitate to contact our experts on troubleshooting by support@ntechlab.com.

Note: The online licensing is done via the NtechLab Global License Manager `license.ntechlab.com`. Check its availability. A stable internet connection and DNS are required.

To retrieve the FindFace Multi *licensing* information and `findface-ntls` status, execute on the `findface-ntls` host console:

```
curl http://localhost:3185/license.json -s | jq
```

The response will be given in JSON. One of the most significant parameters is `last_updated`. It indicates in seconds how long ago the local license has been checked for the last time.

Interpret the `last_updated` value as follows:

- [0, 5] — everything is alright.
- (5, 30] — there may be some problems with connection, or with the local drive where the license file is stored.
- (30; 120] — almost certainly something bad happened.
- (120; ∞) — the licensing source response has been timed out. Take action.
- `"valid": false`: connection with the licensing source was never established.

```
curl http://localhost:3185/license.json -s | jq
{
  "name": "NTLS",
  "time": 1565186356,
  "type": "online",
  "license_id": "61063ce4b86945e1b70c3bdbedea453b",
  "generated": 1514467939,
  "last_updated": 5,
  "valid": {
    "value": true,
    "description": ""
  },
  "source": "/opt/ntech/license/import_
↪b68d7b7ec9a7310d18832035318cff0c9ddf11e3a9ab0ae962fbe48645e196d1.lic",
  "limits": [
    {
      "type": "time",
      "name": "end",
      "value": 1609161621
    },
    {
      "type": "number",
      "name": "faces",
```

(continues on next page)

(continued from previous page)

```
      "value": 9007199254740991,
      "current": 0
    },
    {
      "type": "number",
      "name": "cameras",
      "value": 4294967295,
      "current": 0
    },
    {
      "type": "number",
      "name": "extraction_api",
      "value": 256,
      "current": 0
    },
    {
      "type": "boolean",
      "name": "gender",
      "value": true
    },
    {
      "type": "boolean",
      "name": "age",
      "value": true
    },
    {
      "type": "boolean",
      "name": "emotions",
      "value": true
    },
    {
      "type": "boolean",
      "name": "fast-index",
      "value": true
    },
    {
      "type": "boolean",
      "name": "sec-genetec",
      "value": false
    },
    {
      "type": "boolean",
      "name": "beard",
      "value": false
    },
    {
      "type": "boolean",
      "name": "glasses",
      "value": false
    },
    {
      "type": "boolean",
```

(continues on next page)

(continued from previous page)

```
        "name": "liveness",
        "value": false
      }
    ],
    "services": [
      {
        "name": "video-worker",
        "ip": "127.0.0.1:53276"
      },
      {
        "name": "FindFace-tarantool",
        "ip": "127.0.0.1:53284"
      },
      {
        "name": "FindFace-tarantool",
        "ip": "127.0.0.1:53288"
      }
    ]
}
```

1.5.9 Automatic Tarantool Recovery

If your system architecture doesn't imply uninterrupted availability of Tarantool servers, it is recommended to enable automatic database recovery. In this case, each time an error occurs while reading a snapshot or xlog file, Tarantool will skip invalid records, read as much data as possible, and re-build the file.

To enable automatic database recovery, do the following:

1. For each Tarantool shard, open the configuration file `/etc/tarantool/instances.available/shard-*.lua` and uncomment `force_recovery = true`.

   ```
   sudo vi /etc/tarantool/instances.available/shard-*.lua


   box.cfg{

       force_recovery = true,
   }
   ```

2. Restart the shards.

   ```
   systemctl restart tarantool@shard-*
   ```

1.5.10 Manually Purge Old Data from Database

Tip: To schedule automatic database cleanup, see *Automatic Event And Episode Cleanup*.

To manually remove old data from the FindFace Multi database, use the `cleanup` utility. You can separately remove the following data:

- matched events (faces, bodies, cars) and related episodes (currently only faces),
- unmatched events (faces, bodies, cars) and related episodes (faces),
- full frames of matched events (faces, bodies, cars),
- full frames of unmatched events (faces, bodies, cars),
- counter records,
- person events,
- audit-logs.

To invoke the `cleanup` help message, execute:

```
sudo findface-security cleanup --help
usage: findface-security cleanup [-h] [--as-configured]
                                 [--car-events-matched-age CAR_EVENTS_MATCHED_AGE]
                                 [--car-events-unmatched-age CAR_EVENTS_UNMATCHED_AGE]
                                 [--car-events-fullframe-matched-age CAR_EVENTS_
↪FULLFRAME_MATCHED_AGE]
                                 [--car-events-fullframe-unmatched-age CAR_EVENTS_
↪FULLFRAME_UNMATCHED_AGE]
                                 [--face-events-matched-age FACE_EVENTS_MATCHED_AGE]
                                 [--face-events-unmatched-age FACE_EVENTS_UNMATCHED_AGE]
                                 [--face-events-fullframe-matched-age FACE_EVENTS_
↪FULLFRAME_MATCHED_AGE]
                                 [--face-events-fullframe-unmatched-age FACE_EVENTS_
↪FULLFRAME_UNMATCHED_AGE]
                                 [--body-events-matched-age BODY_EVENTS_MATCHED_AGE]
                                 [--body-events-unmatched-age BODY_EVENTS_UNMATCHED_AGE]
                                 [--body-events-fullframe-matched-age BODY_EVENTS_
↪FULLFRAME_MATCHED_AGE]
                                 [--body-events-fullframe-unmatched-age BODY_EVENTS_
↪FULLFRAME_UNMATCHED_AGE]
                                 [--counter-records-age COUNTER_RECORDS_AGE]
                                 [--person-events-age PERSON_EVENTS_AGE]
                                 [--audit-logs-age AUDIT_LOGS_AGE]
                                 [--configuration CONFIGURATION] [--version]
                                 [-v {0,1,2,3}] [--settings SETTINGS]
                                 [--pythonpath PYTHONPATH] [--traceback]
                                 [--no-color] [--force-color] [--skip-checks]
Delete FFSecurity entities
optional arguments:
  -h, --help            show this help message and exit
  --as-configured       Apply config age options for events, counter records
                        and persons. Can't be used with other arguments.
  --car-events-matched-age CAR_EVENTS_MATCHED_AGE
```

(continues on next page)

(continued from previous page)

```
                        Minimum age in days of matched car events to clean up
  --car-events-unmatched-age CAR_EVENTS_UNMATCHED_AGE
                        Minimum age in days of unmatched car events to clean
                        up
  --car-events-fullframe-matched-age CAR_EVENTS_FULLFRAME_MATCHED_AGE
                        Minimum age in days of matched car events fullframes
                        to clean up
  --car-events-fullframe-unmatched-age CAR_EVENTS_FULLFRAME_UNMATCHED_AGE
                        Minimum age in days of unmatched car events fullframes
                        to clean up
  --face-events-matched-age FACE_EVENTS_MATCHED_AGE
                        Minimum age in days of matched face events to clean up
  --face-events-unmatched-age FACE_EVENTS_UNMATCHED_AGE
                        Minimum age in days of unmatched face events to clean
                        up
  --face-events-fullframe-matched-age FACE_EVENTS_FULLFRAME_MATCHED_AGE
                        Minimum age in days of matched face events fullframes
                        to clean up
  --face-events-fullframe-unmatched-age FACE_EVENTS_FULLFRAME_UNMATCHED_AGE
                        Minimum age in days of unmatched face events
                        fullframes to clean up
  --body-events-matched-age BODY_EVENTS_MATCHED_AGE
                        Minimum age in days of matched body events to clean up
  --body-events-unmatched-age BODY_EVENTS_UNMATCHED_AGE
                        Minimum age in days of unmatched body events to clean
                        up
  --body-events-fullframe-matched-age BODY_EVENTS_FULLFRAME_MATCHED_AGE
                        Minimum age in days of matched body events fullframes
                        to clean up
  --body-events-fullframe-unmatched-age BODY_EVENTS_FULLFRAME_UNMATCHED_AGE
                        Minimum age in days of unmatched body events
                        fullframes to clean up
  --counter-records-age COUNTER_RECORDS_AGE
                        Minimum age in days of counter records to clean up
  --person-events-age PERSON_EVENTS_AGE
                        Minimum age in days of person events to clean up
  --audit-logs-age AUDIT_LOGS_AGE
                        Minimum age in days of audit logs to clean up
  --configuration CONFIGURATION
                        The name of the configuration class to load, e.g.
                        "Development". If this isn't provided, the
                        DJANGO_CONFIGURATION environment variable will be
                        used.
  --version             show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings SETTINGS   The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath PYTHONPATH
```

(continues on next page)

(continued from previous page)

```
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback           Raise on CommandError exceptions
  --no-color            Don't colorize the command output.
  --force-color         Force colorization of the command output.
  --skip-checks         Skip system checks.
```

To entirely remove events and episodes older than a given number of days, use the `--*-events-matched-age`/`--*-events-unmatched-age` options, subject to the object type. For example, to remove unmatched car events older than 5 days, execute:

```
sudo findface-security cleanup --car-events-unmatched-age 5
```

To remove only matched car events older than 5 days, execute:

```
sudo findface-security cleanup --car-events-matched-age 5
```

The following commands remove only full frames of matched/unmatched body events:

```
sudo findface-security cleanup --body-events-fullframe-matched-age 5
sudo findface-security cleanup --body-events-fullframe-unmatched-age 5
```

To remove only counter records, execute:

```
sudo findface-security cleanup --counter-records-age 5
```

To remove only person events, execute:

```
sudo findface-security cleanup --person-events-age 5
```

To remove only audit logs, execute:

```
sudo findface-security cleanup --audit-logs-age 5
```

Important: You must provide at least one of the mentioned arguments.

1.5.11 Disable Services

You can disable the following FindFace Multi services should you no longer need them:

- episodes
- video archive queue manager
- webhooks
- persons

To do so, open the `/etc/findface-security/config.py` configuration file and modify the `SERVICES` section, setting `False` for the services that are no longer in use. Restart the `findface-security` service.

```
sudo vi /etc/findface-security/config.py

# disable unused services to increase
# overall system performance in some cases.
SERVICES = {
    "ffsecurity": {
        "episodes": True,
        "webhooks": True,
        # use queue manager to prevent drops of video archive events
        "video_archive_events_manager": True,
        "persons": False,
    }
}
```

```
sudo systemctl restart findface-security.service
```

After that, the corresponding tabs will disappear from the web interface.

Note: A tab will remain if there are some entities on it (for example, webhooks on the *Webhooks* tab). However, new artifacts will cease to arrive.

1.5.12 Hide Menu Items

To hide specific menu items, do the following:

1. Open the `/etc/findface-security/config.py` configuration file.

   ```
   sudo vi /etc/findface-security/config.py
   ```

2. Into the `FFSECURITY_UI_CONFIG` section, insert the `menu` section, as shown in the example below. List the menu items that you want to hide.

   ```
   FFSECURITY_UI_CONFIG = {
       "menu": {
           "disabled_items": ["video-wall", "know_your_customer"]
       },
   ```

   You can hide the following items:

Menu item	Configure as follows
Search	`"search"`
Events	`"events"`
Episodes	`"episodes"`
Persons	`"persons"`
Relations	`"contacts"`
Counters	`"counters"`
Cameras	`"cameras"`
Video Wall	`"video-wall"`
Reports	`"reports"`
Audit Logs	`"audit_logs"`
Dossiers	`"dossiers"`
Videos	`"videos"`
Analytics	`"know_your_customer"`
Preferences	`"preferences"`
General Preferences	`"settings"`
Appearance	`"appearance"`
Roles	`"roles"`
Users	`"users"`
Camera Groups	`"camera-groups"`
Watch Lists	`"dossier-lists"`
License	`"license"`
Tools	`"tools"`
Verify	`"verify"`
Developer	`"developer"`
Documentation	`"documentation"`
API documentation	`"api_doc"`
Webhooks	`"webhooks"`
menu items activated by custom plugins	Contact our support team for details about your plugin.

3. Restart the `findface-security` service.

```
sudo systemctl restart findface-security.service
```

1.5.13 Reset Password

To reset a user password to the FindFace Multi web interface, execute the following command:

```
findface-security changepassword %username
```

1.5.14 Migrate Data to Another Disk

High disk load may lead to delays in event arrivals. In severe cases, it might result in complete inoperability of FindFace Multi. One of the means for reducing the disk load is to migrate the FindFace Multi data storages to another disk.

In this section:

Prepare Disk

To prepare a disk for the data migration, do the following:

1. Create a new mount point (`/mnt/ffdata` in our example).

```
sudo mkdir /mnt/ffdata
sudo chown ntech:ntech /mnt/ffdata
```

2. Create a partition.

```
sudo parted /dev/sdb
mklabel gpt
mkpart primary ext4 1MiB 100%
q
sudo mkfs.ext4 /dev/sdb1
```

3. Learn the UUID of the partition (`sdb1` in our example).

```
sudo blkid | grep sdb1
/dev/sdb1: LABEL="data" UUID="0638ebe0-853e-43ea-8f35-bfae305695d1" TYPE="ext4"␣
↪PARTUUID="8cebaacc-77d7-4757-b4c6-14147e92646c"
```

4. Add the partition to fstab to make it automatically mount on booting.

```
sudo vi /etc/fstab
------------
#DATA mount
UUID=0638ebe0-853e-43ea-8f35-bfae305695d1 /mnt/ffdata/    ext4    auto,user,rw    ␣
↪0       2
-------------
```

5. Mount all the filesystems.

```
sudo mount -a
```

Migrate Photo Storage

To migrate the FindFace Multi photo storage, do the following:

1. Stop the `findface-security` service to prevent the data loss.

   ```
   sudo systemctl stop findface-security
   ```

2. By default, the photo data are stored at `/var/lib/`. Migrate the photo storage to the *new disk*.

   ```
   sudo cp -ax /var/lib/findface-security/ -R /mnt/ffdata/
   sudo rm -r /var/lib/findface-security/
   sudo cp -ax /var/lib/ffupload/ -R /mnt/ffdata/
   sudo rm -r /var/lib/ffupload/
   ```

3. Create symbolic links for the new directories.

   ```
   sudo ln -s /mnt/ffdata/findface-security/ /var/lib/
   sudo ln -s /mnt/ffdata/ffupload/ /var/lib/
   ```

4. Ensure that the rights are correctly assigned.

   ```
   sudo chown ntech:ntech /mnt/ffdata/findface-security/
   ```

5. Start the `findface-security` service.

   ```
   sudo systemctl start findface-security
   ```

Migrate Main Database (PostgreSQL)

To migrate the PostgreSQL database, do the following:

1. Learn the current database directory.

   ```
   postgres=# SHOW data_directory;

           data_directory
   ------------------------------
    /var/lib/postgresql/10/main
   ```

2. Stop PostgreSQL.

   ```
   sudo systemctl stop postgresql
   ```

3. Create a new directory that will hold the database and assign it to the ntech user.

   ```
   mkdir postgres_data_dir
   chown ntech postgres_data_dir
   ```

4. Migrate the database and backup the old one.

   ```
   sudo rsync -av /var/lib/postgresql /test/postgres_data_dir/
   sudo mv /var/lib/postgresql/10/main /backups/pg_backup
   ```

5. Substitute the directory in the PostgreSQL configuration file `<PostgreSQL directory>/postgresql.conf`.

```
data_directory = '/test/postgres_data_dir/postgresql/10/main'
```

6. Start PostgreSQL.

```
sudo systemctl start postgresql
```

7. Check if the directory has successfully been changed.

```
postgres=# SHOW data_directory;

        data_directory
--------------------------------------------
/test/postgres_data_dir/postgresql/10/main
```

1.6 Appendices

1.6.1 Components in Depth

`findface-extraction-api`

The `findface-extraction-api` service uses neural networks to detect an object in an image, extract the object feature vector, and recognize object attributes (for example, the clothing color for bodies).

It interfaces with the `findface-sf-api` service as follows:

- Gets original images with objects and normalized object images.
- Returns the object bounding box coordinates and, if requested by `findface-sf-api`, feature vector and object attribute data.

Functionality:

- object detection in an original image (with a return of the bbox coordinates),
- object normalization,
- feature vector extraction from a normalized image,
- object attribute recognition (a person's gender, age, emotions; clothing color; car color, car model, etc.).

The `findface-extraction-api` service can be based on CPU (installed from the `findface-extraction-api` package) or GPU (installed from the `findface-extraction-api-gpu` package). For both CPU- and GPU-accelerated services, configuration is done through the `/etc/findface-extraction-api.ini` configuration file. You can find its default content `here for CPU` and `here for GPU`.

When configuring `findface-extraction-api` (on CPU or GPU), refer to the following parameters:

Parameter	Description
`cheetah -> min_object_size`	The minimum size of a face (bbox) guaranteed to be detected. The larger the value, the less resources required for face detection.
`gpu_device`	(Only for GPU) The number of the GPU device used by `findface-extraction-api-gpu`.
`license_ntls_server`	The `ntls` license server IP address and port.

If necessary, you can also enable recognition models for face attributes, body and body attributes, car and car attributes, and liveness detection. You can find the detailed step-by-step instructions in the following sections:

- *Enable Face Attribute Recognition*.
- *Real-time Face Liveness Detection*
- *Liveness Detection as Standalone Service*
- *Enable Body and Body Attribute Recognition*
- *Enable Car and Car Attribute Recognition*

Important: The acceleration type for each model must match the acceleration type of `findface-extraction-api`: CPU or GPU. Note that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models.

Tip: To disable an extractor model, simply pass an empty value to a relevant parameter. Do not remove the parameter itself as in this case the system will be searching for the default model.

```
...
extractors:
...
  models:
    body_color: ''
    body_emben: ''
    body_quality: ''
    car_color: ''
    car_description: ''
    car_emben: ''
    ...
```

`findface-sf-api`

The `findface-sf-api` service implements HTTP API to access the FindFace core functions such as object detection and object recognition.

Note: The mentioned functions themselves are provided by `findface-extraction-api`.

The `findface-sf-api` interfaces with the following FindFace core components:

- feature vector database powered by Tarantool via the `findface-tarantool-server` service
- `findface-extraction-api` that provides object detection and object recognition
- `findface-upload` that provides a storage for original images and FindFace core artifacts

To detect an object in an image, you need to send the image in an API request to `findface-sf-api`. The `findface-sf-api` will then redirect the request to `findface-extraction-api` for object detection and recognition.

If there is a configured video object detection module in the system (like in FindFace Multi), `findface-sf-api` also interfaces with the `findface-facerouter` service. It receives data of detected objects and processing directives from `findface-facerouter` and executes the received directives (for example, saves objects into a specific database gallery).

Note: In FindFace Multi, `findface-facerouter` functions are performed by `findface-security`.

Functionality:

- HTTP API implementation (object detection and object recognition methods, performed via `findface-extraction-api`).
- saving object data to the feature vector database (performed via `findface-tarantool-server`),
- saving original images, object thumbnails and normalized object images to an NginX-powered web server (via `findface-upload`).
- provides interaction between all the FindFace core components.

The `findface-sf-api` configuration is done through the `/etc/findface-sf-api.ini` configuration file. You can find its default content `here`.

When configuring `findface-sf-api`, refer to the following parameters:

Parameter	Description
`extraction-api` -> `extraction-api`	IP address of the `findface-extraction-api` host.
`limits` -> `body-image-length`	The maximum size of an image in an API request, bytes.
`normalized-storage` -> `webdav` -> `upload_url`	WebDAV NginX path to send original images, thumbnails and normalized object images to the `findface-upload` service.
`storage-api` -> `shards` -> `master`	IP address of the `findface-tarantool-server` master shard.
`storage-api` -> `shards` -> `slave`	IP address of the `findface-tarantool-server` replica shard.

`findface-tarantool-server`

The `findface-tarantool-server` service provides interaction between the `findface-sf-api` service and the Tarantool-based feature vector database in the following way:

Tip: See Tarantool official documentation for details.

- From `findface-sf-api`, `findface-tarantool-server` receives data, such as information of detected objects, to write into the feature vector database.
- By request from `findface-sf-api`, `findface-tarantool-server` performs database searches and returns search results.

Multiple `findface-tarantool-server` shards can be created on each Tarantool host to increase search speed. Their running concurrently leads to a remarkable increase in performance (70x-100x).

Functionality:

- saving object data to the feature vector database,
- database search,
- implementation of direct API requests to the database (see *Direct API Requests to Tarantool*).

The `findface-tarantool-server` configuration is done through the `/etc/tarantool/instances.available/<shard-*>.lua` configuration file. You can find its default content `here`.

Important: In a multi-shard environment, the configuration has to be done for each shard.

When configuring `findface-tarantool-server`, refer to the following parameters:

Parameter	Description
`force_recovery`	Enables automatic database recovery. In this case, each time an error occurs while reading a snapshot or xlog file, Tarantool will skip invalid records, read as much data as possible, and re-build the file.
`license_ntls_server`	IP address and port of the `findface-ntls` license server.
`memtx_memory`	Maximum RAM that can be used by a Tarantool shard. Set in bytes, depending on the number of objects the shard handles. Consult our experts by support@ntechlab.com before setting this parameter.
`meta_scheme`	A database structure to store the object recognition results. The structure is created as a set of fields. Describe each field with the following parameters: `id`: field id; `name`: field name, must be the same as the name of a relevant object parameter; `field_type`: data type; `default`: field default value, if a default value exceeds '1e14 – 1', use a string data type to specify it, for example, `"123123.."` instead of `123123..`

Default database structure is passed from `/etc/findface-security/tnt_schema.lua` to the `meta_scheme` parameter. See *Modify Feature Vector Database Structure* for details.

`findface-upload`

The `findface-upload` component is an NginX-based web server used as a storage for original images, thumbnails and normalized object images which it receives from the `findface-sf-api` component.

By default the original images, thumbnails and normalized images are stored at `/var/lib/ffupload/uploads/`.

The `findface-upload` component is automatically configured upon installation. Custom configuration is not supported.

Video object detection: `findface-video-manager` and `findface-video-worker`

Note: The `findface-video-worker` is delivered in a CPU-accelerated (`findface-video-worker-cpu`) and a GPU-accelerated (`findface-video-worker-gpu`) packages.

In this section:

Functions of `findface-video-manager`

The `findface-video-manager` service is the part of the video object detection module that is used for managing the video object detection functionality.

The `findface-video-manager` service interfaces with `findface-video-worker` as follows:

- It supplies `findface-video-worker` with settings and the list of to-be-processed video streams. To do so, it issues a so-called *job*, a video processing task that contains configuration settings and stream data.
- In a distributed system, it distributes video streams (jobs) across vacant `findface-video-worker` instances.

Note: The configuration settings passed via jobs have priority over the `/etc/findface-video-manager.conf` configuration file.

The `findface-video-manager` service functioning requires ETCD, third-party software that implements a distributed key-value store for `findface-video-manager`. In the FindFace core, ETCD is used as a coordination service, providing the video object detector with fault tolerance.

Functionality:

- allows for configuring video object detection parameters
- allows for managing the list of to-be-processed video streams

Functions of `findface-video-worker`

The `findface-video-worker` service (on CPU/GPU) is the part of the video object detection module, that recognizes objects in the video. It can work with both live streams and files, and supports most video formats and codecs that can be decoded by FFmpeg.

The `findface-video-worker` service interfaces with the `findface-video-manager` and `findface-facerouter` services as follows:

- By request, `findface-video-worker` gets a job with settings and the list of to-be-processed video streams from `findface-video-manager`.
- The `findface-video-worker` posts extracted normalized object images, along with the full frames and meta data (such as bbox, camera ID and detection time) to the `findface-facerouter` service for further processing.

Note: In FindFace Multi, the `findface-facerouter` functions are performed by `findface-security`.

Functionality:

- detects objects in the video,
- extracts normalized object images,
- searches for the best object snapshot,
- snapshot deduplication (only one snapshot per object detection event).

When processing a video, `findface-video-worker` consequently uses the following algorithms:

- **`Motion detection`**. Used to reduce resource consumption. Only when the motion detector recognizes the motion of certain intensity that the object tracker can be triggered.

- `Object tracking`. The object tracker traces, detects, and captures objects in the video. It can simultaneously be working with several objects. It also searches for the best object snapshot using the embedded neural network. After the best object snapshot is found, it is posted to `findface-facerouter`.

The best object snapshot can be found in one of the following modes:

- Real-time
- Offline

Real-Time Mode

In the real-time mode, `findface-video-worker` posts an object on-the-fly after it appears in the camera field. The following posting options are available:

- If `realtime_post_every_interval: true`, the object tracker searches for the best object snapshot within each time period equal to `realtime_post_interval` and posts it to `findface-facerouter`.
- If `realtime_post_every_interval: false`, the object tracker searches for the best face snapshot dynamically:
  1. First, the object tracker estimates whether the quality of an object snapshot exceeds a pre-defined internal threshold. If so, the snapshot is posted to `findface-facerouter`.
  2. The threshold value increases after each post. Each time the object tracker gets a higher quality snapshot of the same object, it is posted.
  3. When the object disappears from the camera field, the threshold value resets to default.
- If `realtime_post_first_immediately: true`, the object tracker doesn't wait for the first `realtime_post_interval` to complete and posts the first object from a track immediately after it passes through the quality, size, and ROI filters. The way the subsequent postings are sent depends on the `realtime_post_every_interval` value. If `realtime_post_first_immediately: false`, the object tracker posts the first object after the first `realtime_post_interval` completes.

Offline Mode

The offline mode is less storage intensive than the real-time one as in this mode `findface-video-worker` posts only one snapshot per track but of the highest quality. In this mode, the object tracker buffers a video stream with an object until the object disappears from the camera field. Then the object tracker picks up the best object snapshot from the buffered video and posts it to `findface-facerouter`.

Configure Video Object Detection

The video object detector configuration is done through the following configuration files:

1. The `findface-video-manager` configuration file `/etc/findface-video-manager.conf`. You can find its default content `here`

   When configuring `findface-video-manager`, refer to the following parameters:

Option	Description
etcd -> endpoints	IP address and port of the etcd service. Default value: 127.0.0.1:2379.
ntls -> enabled	If true, findface-video-manager will send a job to findface-video-worker only if the total number of processed cameras does not exceed the allowed number of cameras from the license. Default value: false.
ntls -> url	IP address and port of the findface-ntls host. Default value: http://127.0.0.1:3185/.
router_url	IP address and port of the findface-facerouter host to receive detected faces from findface-video-worker. In FindFace Multi, findface-facerouter functions are performed by findface-security. Default value: http://127.0.0.1:18820/v0/frame.
play_speed	If less than zero, the speed is not limited. In other cases, the stream is read with the given play_speed. Not applicable for live streams.
disable_drops	Enables posting all appropriate objects without drops. By default, if findface-video-worker does not have enough resources to process all frames with objects, it drops some of them. If this option is active, findface-video-worker puts odd frames on the waiting list to process them later. Default value: false.
imotion_threshold	Minimum motion intensity to be detected by the motion detector. The threshold value is to be fitted empirically. Empirical units: zero and positive rational numbers. Milestones: 0 = detector disabled, 0.002 = default value, 0.05 = minimum intensity is too high to detect motion.
router_timeout_ms	Timeout for a findface-facerouter (or findface-security in the standard FindFace Multi configuration) response to a findface-video-worker API request, in milliseconds. If the timeout has expired, the system will log an error. Default value: 15000.
router_verify_ssl	Enables a https certificate verification when findface-video-worker and findface-facerouter (or findface-security in the standard FindFace Multi configuration) interact over https. Default value: true. If false, a self-signed certificate can be accepted.
router_headers	Additional header fields in a request when posting an object: ["key = value"]. Default value: headers not specified.
router_body	Additional body fields in a request body when posting an object: ["key = value"]. Default value: body fields not specified.
ffmpeg_params	List of a video stream ffmpeg options with their values as a key=value array: ["rtsp_transpotr=tcp", .., "ss=00:20:00"]. Check out the FFmpeg web site for the full list of options. Default value: options not specified.
ffmpeg_format	Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.
use_stream_timestamp	If true, retrieve and post timestamps from a video stream. If false, post the actual date and time.
start_stream_timestamp	Add the specified number of seconds to timestamps from a stream.
rot	Enables detecting and tracking objects only inside a clipping rectangle WxH+X+Y. You can use this option to reduce findface-video-worker load. Default value: rectangle not specified.
video_transform	Change a video frame orientation right after decoding. Values (case insensitive, JPEG Exif Orientation Tag in brackets): None (1), FlipHorizontal (2), Rotate180 (3), FlipVertical (4), Transpose (5), Rotate90 (6), Transverse (7), Rotate270 (8). Default value: not specified.

The following parameters are available for configuration for each detector type (face, body, car):

Option	Description
`filter_min_quality`	Minimum threshold value for an object image quality. Default value: subject to the object type. Do not change the default value without consulting with our technical experts (support@ntechlab.com).
`filter_min_size`	Minimum size of an object in pixels. Undersized objects are not posted. Default value: 1.
`filter_max_size`	Maximum size of an object in pixels. Oversized objects are not posted. Default value: 8192.
`roi`	Enable posting objects detected only inside a region of interest WxH+X+Y. Default value: region not specified.
`fullframe_crop_rot`	Crop posted full frames by ROT. Default value: false.
`fullframe_use_png`	Send full frames in PNG and not in JPEG as set by default. Do not enable this parameter without supervision from our team as it can affect the entire system functioning. Default value: false (send in JPEG).
`jpeg_quality`	Quality of an original frame JPEG compression, in percents. Default value: 95%.
`overall_only`	Enables the offline mode for the best object search. Default value: true (CPU), false (GPU).
`realtime_post_first_immediately`	Enable posting an object image right after it appears in a camera field of view (real-time mode). Default value: false.
`realtime_post_interval`	Only for the real-time mode. Defines the time period in seconds within which the object tracker picks up the best snapshot and posts it to `findface-facerouter`. Default value: 1.
`realtime_post_every_interval`	Only for the real-time mode. Post best snapshots obtained within each `realtime_post_interval` time period. If false, search for the best snapshot dynamically and send snapshots in order of increasing quality. Default value: false.
`track_interpolate_bboxes`	Interpolate the missed bboxes of objects in track. For example, if frames #1 and #4 have bboxes and #2 and #3 do not, the system will reconstruct the absent bboxes #2 and #3 based on the #1 and #4 data. Enabling this option allows you to increase the detection quality on account of performance. Default value: true.
`track_miss_interval`	The system closes a track if there has been no new object in the track within the specified time (seconds). Default value: 1.
`track_max_duration_frames`	The maximum approximate number of frames in a track after which the track is forcefully completed. Enable it to forcefully complete “eternal tracks,” for example, tracks with objects from advertisement media. The default value: 0 (option disabled).
`track_send_history`	Send track history. Default value: false.
`post_best_track_frame`	Send full frames of detected objects. Default value: true.
`post_best_track_normalize`	Send normalized images for detected objects. Default value: true.
`post_first_track_frame`	Post the first frame of a track. Default value: false.
`post_last_track_frame`	Post the last frame of a track. Default value: false.

1. The `findface-video-worker` configuration file `/etc/findface-video-worker-cpu.ini` or `/etc/findface-video-worker-gpu.ini`, subject to the acceleration type in use.

   When configuring `findface-video-worker` (on CPU/GPU), refer to the following parameters:

CPU	GPU	Description
`batch_size`		Post faces in batches of the given size.
`capacity`		Maximum number of video streams to be processed by `findface-video-worker`.
N/a	`cpu`	If necessary, decode video on CPU.
N/a	`device_number`	GPU device number to use.
`exit_on_first_finished`		(Only if `input` is specified) Exit on the first finished job.
`input`		Process streams from file, ignoring stream data from `findface-video-manager`.
`labels`		Labels used to allocate a video object detector instance to a certain group of cameras. See *Allocate findface-video-worker to Camera Group*.
`mgr-cmd`		(Optional, instead of the `mgr-static` parameter) A command to obtain the IP address of the `findface-video-manager` host.
`mgr-static`		IP address of the `findface-video-manager` host to provide `findface-video-worker` with settings and the list of to-be-processed streams.
`metrics_port`		HTTP server port to send metrics. If 0, the metrics are not sent.
`min_size`		Minimum object size to be detected.
`ntls-addr`		IP address and port of the `findface-ntls` host.
`resize_scale`		Rescale video frames with the given coefficient.
`resolutions`		Preinitialize `findface-video-worker` to work with specified resolutions. Example: “640x480;1920x1080”.
`save_dir`		(For debug) Save detected objects to the given directory.
`streamer` -> `port`, `url`		IP address and port to access the *video wall*.
`use_time_from_sei`		(For MPEG-2) Use SEI (supplemental enhancement information) timestamps.

If necessary, you can also enable neural network models and normalizers to detect bodies, cars, and liveness. You can find the detailed step-by-step instructions in the following sections:

- *Real-time Face Liveness Detection*
- *Enable Body and Body Attribute Recognition*
- *Enable Car and Car Attribute Recognition*

Jobs

The `findface-video-manager` service provides `findface-video-worker` with a so-called job, a video processing task that contains configuration settings and stream data.

You can find a job example `here`.

Each job has the following parameters:

- `id`: job id.
- `enabled`: active status.
- `stream_url`: URL/address of video stream/file to process.
- `labels`: tag(s) that will be used by the `findface-facerouter` component (`findface-security` in the standard FindFace Multi configuration) to find processing directives for faces detected in this stream.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).
- `router_url`: IP address and port of the `findface-facerouter` component (`findface-security` in the standard FindFace Multi configuration) to receive detected faces from the `findface-video-worker` component for processing.

- `stream_settings`: video stream settings that duplicate *those* in the `/etc/findface-video-manager.conf` configuration file (while having priority over them).
- `status`: job status.
- `status_msg`: additional job status info.
- `statistic`: job progress statistics (progress duration, number of posted and not posted objects, processing fps, the number of processed and dropped frames, job start time, etc.).
- `worker_id`: id of the `findface-video-worker` instance executing the job.

findface-ntls

The `findface-ntls` service is to be installed on a designated host to verify the FindFace license. For verification purposes, `findface-ntls` uses one of the following sources:

- Ntech Lab global license center if you opt for the online licensing, direct or via a proxy server.
- USB dongle if you opt for the on-premise licensing.

Use the FindFace Multi web interface (*Preferences* -> *License*) to manage `findface-ntls` in the following way:

- view the list of purchased features,
- view license limitations,
- upload a license file,
- view the list of currently active components.

The following components are licensable:

- `findface-tarantool-server`,
- `findface-extraction-api`,
- `findface-video-manager`,
- `findface-video-worker`.

Important: After connection between `findface-ntls` and a licensable component, or between `findface-ntls` and the global license server is broken, you will have 4 hours to restore it before the licensable components will be automatically stopped. It is possible to prolongate the off-grid period for up to 2 days. Inform your manager if you need that.

The `findface-ntls` configuration is done through a configuration file `/etc/findface-ntls.cfg`. You can find its default content `here`.

When configuring `findface-ntls`, refer to the following parameters:

Parameter	Description
`license_dir`	Directory to store a license file.
`listen`	IP address from which licensable services access `findface-ntls`. To allow access from any IP address, use `0.0.0.0:3133`.
`proxy`	(Optional) IP address and port of your proxy server.
`ui`	IP address from which accessing the `findface-ntls` web interface must originate. To allow access from any remote host, set "0.0.0.0".

findface-security

The findface-security component serves as a gateway to the FindFace core. It provides interaction between the FindFace Core and the web interface, the system functioning as a whole, HTTP and web socket (along with Django), database update, and *webhooks*.

The findface-security component also performs the functions of findface-facerouter (part of the FindFace Core), setting processing directives for detected objects. It accepts an object bbox and normalized image along with the original image and other data (for example, the detection date and time) from the findface-video-worker service and redirect them to findface-sf-api for further processing.

The findface-security configuration is done through the /etc/findface-security/config.py configuration file. You can find its default content here.

The /etc/findface-security/config.py file has detailed comments for each setting. Refer to them when configuring findface-security.

findface-facerouter and Custom Plugins

Important: The findface-facerouter is not included in the FindFace Multi standard configuration. Use it for integration if necessary.

The findface-facerouter service sets processing directives for objects detected in the video. The directives are set through custom plugins.

The findface-facerouter service accepts an object bbox and normalized image along with the original image and other data (for example, the detection date and time) from the findface-video-worker service.

In general, findface-facerouter allows you to apply arbitrary processing directives to the received objects, including directly sending objects to a partner application. In FindFace Multi, the findface-facerouter functions are performed by the findface-security service that redirects the objects to findface-sf-api.

Functionality:

- sets processing directives for objects detected in the video,
- redirects objects detected in the video to findface-sf-api or other service (including a third-party application) for further processing.

The findface-facerouter configuration is done through a configuration file /etc/findface-facerouter.py. You can find its default content here.

When configuring findface-facerouter, refer to the following parameters:

Parameter	Description
sfapi_url	IP address and port of the findface-sf-api host.
plugin_dir	Directory with plugins to define object processing directives.

Deploy `findface-facerouter` in FindFace Multi

To deploy the `findface-facerouter` component, do the following:

1. Install `findface-facerouter` either from the *console installer* or from the apt repository as such:

```
sudo apt update
sudo apt install -y findface-facerouter
```

2. Open the `/etc/findface-facerouter.py` configuration file.

```
sudo vi /etc/findface-facerouter.py
```

3. If the `findface-facerouter` and `findface-sf-api` components are installed on different hosts, uncomment the `sfapi_url` parameter and specify the `findface-sf-api` host IP address.

```
sfapi_url                    = 'http://localhost:18411'
```

4. Open the `/etc/findface-security/config.py` configuration file. In the `ROUTER_URL` parameter, actualize the `findface-facerouter` IP address and port (18820 by default). Specify either external or internal IP address, subject to the network through which `findface-video-worker` interacts with `findface-facerouter`.

```
sudo vi /etc/findface-security/config.py

...
FFSECURITY = {
   'ROUTER_URL': 'http://127.0.0.1:18820/v0/frame?',
```

5. Enable the `findface-facerouter` service autostart and launch the service.

```
sudo systemctl enable findface-facerouter.service && sudo systemctl start findface-
↪facerouter.service
```

6. Restart the `findface-security` service.

```
sudo systemctl restart findface-security.service
```

Configure `findface-facerouter` to Use Plugins

Important: Be sure to *change* the Tarantool database structure prior, according to the processing directive in the plugin.

Important: The `findface-facerouter` component must be *installed and configured*.

To configure `findface-facerouter` to use plugins, do the following:

1. Put a plugin into a directory of your choice. All plugins in use have to be in the same directory.
2. Open the `/etc/findface-facerouter.py` configuration file. Uncomment the `plugin_dir` parameter and specify the plugin directory.

Warning: The `findface-facerouter.py` content must be correct Python code.

```
sudo vi /etc/findface-facerouter.py

plugin_dir                    = '/etc/findface/plugins/'
```

3. Restart `findface-facerouter`.

```
sudo systemctl restart findface-facerouter.service
```

Plugin Basics

In this section:

Plugin Architecture

After the `findface-video-worker` component detects a face, the face is posted to the `findface-facerouter` component via an HTTP API request. To process this request, each `findface-facerouter` plugin must export the `activate(app, ctx, plugin_name, plugin_source)` function.

The `activate` function has the following parameters:

- `app`: a `tornado.web.Application` entity of the `findface-facerouter` component.
- `ctx`: data context to be passed to a plugin upon activation.
- `plugin_name`: the name of the plugin to be activated.
- `plugin_source`: source object to load the plugin from.

Upon activation, a plugin is passed the following data context:

1. `request.ctx.sfapi`: a set up `ntech.sfapi_client.Client` instance that can be invoked directly to process the result of video face detection (for example, to create a new gallery, add a face to a gallery, etc.).
2. `plugins`: `OrderedDict` with all the plugins as (`key`: plugin name, `value`: the result returned by the `activate` function).
3. `idgen`: id generator that can be invoked as `ctx.idgen()`.

The `activate(app, ctx, plugin_name, plugin_source)` function must return an object with the following methods:

1. `preprocess`,
2. `process`,
3. `shutdown` (optional).

The `preprocess` method

In this method, a `findface-facerouter` plugin decides if it is interested in the face received from the `findface-video-worker` component. If so, it returns a tuple or a list that contains one or several strings `'facen'`, `'gender'`, `'age'`, `'emotions'`. This means that it is necessary to extract a feature vector, recognize gender, age, emotions respectively. If the returned tuple/list is non-empty, the `findface-facerouter` redirects the face to the `findface-sf-api` in a `/detect POST` request with relevant query string parameters (`facen=on`, `gender=on`, `age=on`, `emotions=on`).

The basic `preprocess` method to inherit from has the following syntax (see the `Plugin` class):

`preprocess`(*self*, *request: FrHTTPRequest*, *labels: Mapping[str, str]*) → Tuple[str]

Parameters

- **`FrHTTPRequest`** (*`tornado.httpserver.HTTPRequest`*) – a HTTP API request that includes an extra argument `params`
- **`labels`** (*`dictionary`*) – a custom set of a frame labels, which are initially specified in a job parameters for `findface-video-worker` and then assigned to the frame

The `params` argument of FrHTTPRequest includes the following fields:

Parameters

- **`photo`** (*`bytes`*) – JPEG video frame featuring a detected face
- **`face0`** (*`bytes`*) – normalized face image
- **`bbox`** (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – coordinates of the face region in the video frame
- **`cam_id`** (*`string`*) – camera id
- **`timestamp`** (*`datetime.datetime`*) – video frame timestamp
- **`detectorParams`** (*`dictionary`*) – debug information from the video face detector
- **`bs_type`** (*`string`*) – best face search mode. Available options: overall (the `findface-video-worker` posts only one snapshot per track, but of the highest quality.), realtime (the `findface-video-worker` posts the best snapshot within each of consecutive time intervals).
- **`labels`** (*`dictionary`*) – (duplicates params.labels) a custom set of a frame labels, which are specified in a job parameters for `findface-video-worker` and then assigned to the frame

The decision about face processing is made based on the data in the `request.params`, including the custom set of labels, as well as for any other reasons.

The `process` method

This method is called if the `preprocess` method returns a non-empty tuple or list (i.e. with 'facen', 'gender', 'age', an/or 'emotions' strings). After the `findface-sf-api` returns a response with the result of face detection (see the `/detect POST` request) with all the requested face features, the `findface-facerouter` component calls the `process` method of the plugin in order to the perform face processing itself.

To process a face, a plugin uses `request.ctx.sfapi`.

The basic `process` method to inherit from has the following syntax (see the `Plugin` class):

`process`(*self*, *request: FrHTTPRequest*, *photo: bytes*, *bbox: List[int]*, *event_id: int*, *detection: DetectFace*)

The `shutdown` method

This method is only called before the `findface-facerouter` shutdown.

The basic `shutdown` method to inherit from has the following syntax (see the `Plugin` class):

`shutdown`(*self*)

Plugin Classes and Methods

In this section:

Basic Classes

`class facerouter.plugin.`**`Plugin`**

Provides the basic methods for writing a plugin (see *Plugin Basics*). A custom class that wraps a plugin must inherit from the `Plugin` class.

`preprocess`(*self*, *request: FrHTTPRequest*, *labels: Mapping[str, str]*) → Tuple[str]

Returns a tuple that contains one or several strings `'facen'`, `'gender'`, `'age'`, `'emotions'`. This means that `findface-facerouter` must request `findface-extraction-api` to extract a biometric sample, recognize gender, age, emotions respectively.

Parameters

- **`FrHTTPRequest`** (*`tornado.httpserver.HTTPRequest`*) – a HTTP API request that includes an extra argument `params`
- **`labels`** (*`dictionary`*) – a custom set of a frame labels from `request.params`

Returns
one or several strings `'facen'`, `'gender'`, `'age'`, `'emotions'`

Return type
tuple

The `params` argument of `FrHTTPRequest` includes the following fields:

Parameters

- **photo** (*bytes*) – JPEG video frame featuring a detected face
- **face0** (*bytes*) – normalized face image
- **bbox** (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – coordinates of the face region in the video frame
- **cam_id** (*string*) – camera id
- **timestamp** (*datetime.datetime*) – video frame timestamp
- **detectorParams** (*dictionary*) – debug information from the video face detector
- **bs_type** (*string*) – best face search mode. Available options: overall (the `findface-video-worker` posts only one snapshot per track, but of the highest quality.), realtime (the `findface-video-worker` posts the best snapshot within each of consecutive time intervals).
- **labels** (*dictionary*) – (duplicates params.labels) a custom set of a frame labels, which are specified in a job parameters for `findface-video-worker` and then assigned to the frame

process(*self*, *request: FrHTTPRequest*, *photo: bytes*, *bbox: List[int]*, *event_id: int*, *detection:* DetectFace)

Accepts the detected face features.

Parameters

- **request** (*tornado.httpserver.HTTPRequest*) – a HTTP API request from `findface-video-worker`
- **photo** (*bytes*) – JPEG video frame featuring a detected face, from `request.params`
- **bbox** (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – coordinates of the face region in the video frame, from `request.params`
- **event_id** (*uint64*) – id of the face automatically set by `findface -facerouter` upon receiving it from `findface-video-worker`. Can be used as a face custom identifier in the biometric database.
- **detection** (`objects.DetectFace`) – detection result received from `findface-sf-api`, that contains requested face features such as faces, gender, age and emotions.

Returns
n/a

Return type
n/a

`shutdown`(*self*)

This method is invoked before the `findface-facerouter` shutdown.

Parameters
n/a

Returns
n/a

Object Classes

class `objects.BBox`

Represents coordinates of the rectangle around a face.

class `objects.DetectFace`

Represents a detection result with the following fields:

Parameters

- **id** (*string*) – id of the detection result in memcached
- **bbox** (*objects.Bbox*) – coordinates of the rectangle around a face
- **features** (*dictionary*) – (optional) information about gender, age and emotions

class `objects.DetectResponse`

Represents a list of `objects.DetectionFace` objects with an additional field `orientation` featuring information about the face EXIF orientation in the image.

Parameters
orientation (*EXIF orientation*) – orientation of a detected face

class `objects.FaceId`(*namedtuple('FaceId', ('gallery', 'face'))*)

Represents a custom face identifier object in the gallery.

Parameters

- **gallery** (*string*) – gallery name
- **face** (*integer*) – custom face identifier in the gallery

class `objects.Face`

Represents a result of database search by biometric sample

Parameters

- **id** (`objects.FaceId`) – FaceId object.
- **features** (*dictionary*) – information about gender, age and emotions
- **meta** (*dictionary*) – face meta data
- **confidence** (*float*) – similarity between the biometric sample and a face in the search result

class `objects.ListResponse`

Represents a list of `objects.Face` objects (i.e. a list of biometric sample search results) with an additional field `next_page` featuring the cursor for the next page with search results.

Parameters
next_page (*string*) – cursor for the next page with search results

Face Detection and Gallery Management

class `ntech.sfapi_client.client.`**`Client`**

Represents basic methods to detect faces in images and work with galleries.

`detect`(*self*, ***, *url=None*, *image=None*, *facen=False*, *gender=False*, *age=False*, *emotions=False*, *return_facen=False*, *autorotate=False*, *detector: str = None*, *timeout=None*) → *DetectResponse*

Detects a face and returns the result of detection.

Parameters

- **`url`** (*`URL`*) – image URL if you pass an image that is publicly accessible on the internet
- **`image`** (*`bytes`*) – PNG/JPG/WEBP image file is you pass an image as a file
- **`facen`** (*`boolean`*) – extract a biometric sample from the detected face. To save the detection result in memcached pass `facen=True`
- **`gender`** (*`boolean`*) – extract and return information about gender
- **`age`** (*`boolean`*) – extract and return information about age
- **`emotions`** (*`boolean`*) – extract and return information about emotions
- **`return_facen`** (*`boolean`*) – return facen in the method result
- **`autorotate`** (*`boolean`*) – automatically rotate the image in 4 different orientations to detect faces in each of them. Overlapping detections with IOU > 0.5 will be merged
- **`detector`** (*`boolean`*) – nnd or normalized. The `normalized` detector is used to process normalized images, for example, those which are received from `fkvideo_worker`.
- **`timeout`** (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns

Detection result

Return type

`DetectorResponse` object.

`gallery`(*self*, *name*)

Returns a gallery object `sfapi_client.Gallery` to refer to it later (for example, to list gallery faces).

Parameters

`name` (*`string`*) – gallery name

Returns

a gallery object

Return type

sfapi_client.Gallery

`list_galleries(self, timeout=None):`

Returns the list of galleries.

Parameters

`timeout` (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns

list of galleries with the fields `name` (a gallery name, string) and `number` (the number of faces in the gallery, number)

Return type
list of GalleryListItem

`class ntech.sfapi_client.gallery.Gallery`

Provides methods to work with galleries and faces.

`list`(*self*, **, *filters: Iterable[filters.Filter] = None*, *limit: int = 1000*, *sort: str = ''*, *page=None*, *ignore_errors=False*, *timeout=None*) → *ListResponse*

Returns a list-like object with faces from the gallery, that match the given filters. The returned list-like object has an additional property `next_page` which can be used as a value for the `page` parameter in next requests.

Parameters

- **`filters`** (`sfapi_client.filters.Filter`) – list of filters
- **`limit`** (*`integer`*) – maximum number of returned faces
- **`sort`** (*`string`*) – sorting order. Pass one of the following values: `id`: increasing order by id, `-id`: decreasing order by id (sorting by id is used if you have NOT specified a feature vector to search for), `-confidence`: decreasing order by face similarity (only if you have specified a feature vector to search for). By default, the method uses the `id` order (no feature vector specified), or `-confidence` (with feature vector).
- **`page`** – cursor of the next page with search results. The `page` value is returned in the response in the `next_page` parameter along with the previous page results.
- **`ignore_errors`** (*`boolean`*) – By default, if one or several `findface-tarantool-server` shards are out of service during face identification, `findface-sf-api` returns an error. Enable this Boolean parameter to use available `findface-tarantool-server` shards to obtain face identification results.
- **`timeout`** (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns
list with faces from the gallery, that match the given filters.

Return type
ListResponse object

`add`(*self*, *new_id: Union[int, Callable]*, *source: Union[DetectFace, Face, str]*, **, *meta: Dict[str, Union[int, str, List[str]]] = None*, *regenerate_attempts=None*, *timeout=None*) → *Face*

Creates a face in the gallery.

Parameters

- **`new_id`** (*`integer or callable`*) – custom face identifier (Face ID) in the database gallery. May be a (async) callable which returns the id. To generate id, you can use the `ctx.idgen()` function delivered with the context.
- **`source`** (*`sfapi_client.DetectFace, sfapi_client.Face, sfapi_client.FaceId, or string`*) – face source: create a face using another face in the database or a detection result as a source.
- **`meta`** (*`dictionary`*) – face metadata. Keys must be strings and values must be either ints, strings or lists of strings. Metadata keys and types must be previously specified in the storage configuration files (`/etc/tarantool/instances.available/*.lua`).
- **`regenerate_attempts`** – number of attempts to regenerate a unique Face ID with the `ctx.idgen()` function if new_id is callable

- **`timeout`** (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns
representation of the newly created face

Return type
Face object

`delete`(*self*, *face: Union*[Face, *int*], *timeout=None*) → None

Removes a face from the gallery.

Parameters

- **`face`** (*`sfapi_client.Face, sfapi_client.FaceId or id in integer`*) – face to be removed
- **`timeout`** (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns
None

`get`(*self*, *face: Union*[Face, *int*], *timeout=None*) → *Face*

Retrieves a face from the gallery.

Parameters

- **`face`** (*`sfapi_client.Face, sfapi_client.FaceId or id in integer`*) – face to be retrieved
- **`timeout`** (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns
representation of the face

Return type
Face object

`create`(*self*, *timeout=None*) → None

Creates a gallery in `findface-sf-api` as a `sfapi_client.Gallery` object. Being a proxy object, `sfapi_client.Gallery` doesn't require a gallery to be existing on the server.

Parameters
`timeout` (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns
None

`drop`(*self*, *timeout=None*) → None:

Removes a gallery from `findface-sf-api`.

Parameters
`timeout` (*`number`*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns
None

update(*self*, *face: Union[Face, str]*, ***, *meta: Dict[str, Union[int, str, List[str]]] = None*, *timeout=None*) → *Face*

Update face meta data in the gallery.

Parameters

- **face** (*sfapi_client.Face, sfapi_client.FaceId or id in integer*) – face to be updated
- **meta** (*dictionary*) – face meta data to be updated. Keys must be strings and values must be either ints, strings or lists of strings. If a meta string is not passed or passed as null, it won't be updated in the database.
- **timeout** (*number*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns
representation of the updated face

Return type
Face object

Filters for Database Search

class `ntech.sfapi_client.filters.`**Filter**

Generic class. Represents a list of filters (with assigned values) that have to be applied to the gallery content.

serialize(*self*)

Method that passes the list of filters with assigned values to the `findface-sf-api` component.

Returns
filter names and filter values

Return type
tuple ('filtername', ["value1", "value2"]).

class `ntech.sfapi_client.filters.`**Id**

Represents methods for filtering gallery content by id. Don't instantiate, use relevant classmethods to call a filter.

classmethod lte(*cls*, *value: int*) → *Filter*

LTE filter. Select all faces with `id` less or equal to `value`.

Parameters
value (*integer*) – id value

Returns
filter name (LTE) and its value.

Return type
object of `Filter` class.

Example: `Id.lte(1234)` selects faces with id less or equal to `1234`.

classmethod gte(*cls*, *value: int*) → *Filter*

GTE filter. Select all faces with `id` greater or equal to `value`.

Parameters
value (*integer*) – id value

Returns
filter name (GTE) and its value.

Return type
object of `Filter` class.

Example: `Id.lte(1234)` selects faces with id greater or equal to `1234`.

`classmethod oneof`(*cls*, **value: Union[int]*) → *Filter*

IN filter. Select a face(s) with id from a given set.

Parameters
value (*list of integers*) – list of id values

Returns
filter name (IN) and its value.

Return type
object of `Filter` class.

Example: `Id.oneof(1234, 5678)` selects a face(s) with id `1234` and/or `5678`.

`class ntech.sfapi_client.filters.Meta`

Represents methods for filtering gallery content by metadata. Don't instantiate, use relevant classmethods to call a filter.

`classmethod lte`(*self*, *value: Union[str, int]*) → *Filter*

LTE filter. Select all faces with a metastring less or equal to `value`

Parameters
value (*string or integer*) – metastring value

Returns
filter name (LTE) and its value.

Return type
object of `Filter` class.

Example: `Meta('foo').lte(1234)` selects faces with a metastring `foo` less or equal to `1234`.

`classmethod gte`(*self*, *value: Union[str, int]*) → *Filter*

GTE filter. Select all faces with a metastring greater or equal to `value`

Parameters
value (*string or integer*) – metastring value

Returns
filter name (GTE) and its value.

Return type
object of `Filter` class.

Example: `Meta('foo').gte(1234)` selects faces with a metastring `foo` greater or equal to `1234`.

`classmethod oneof`(*self*, **value: Union[str, int]*) → *Filter*

IN filter. Select a face(s) with a metastring from a given set.

Parameters
value (*list of strings or integers*) – list of metastring values

Returns
filter name (IN) and its value.

Return type
object of `Filter` class.

Example: `Meta.oneof(1234, 5678)` selects a face(s) with a metastring `1234` and/or `5678`.

classmethod subset(*self*, **value: str*) → *Filter*

SUBSET filter. Select all faces with a metastring featuring all values from a given set.

Parameters
value (*list of strings or integers*) – list of metastring values

Returns
filter name (SUBSET) and its value.

Return type
object of `Filter` class.

Example: `Meta('foo').subset("male", "angry")` selects face with a metastring `foo` featuring all values from the set [“male”, “angry”].

class ntech.sfapi_client.filters.**Detection**(*Filter*)

Represents a method that identifies a detected face (searches the database for similar faces).

__init__(*self*, *id: Union[str, objects.DetectFace]*, *threshold: float*)

Parameters

- **id** (`objects.DetectFace` or temporary face id in memcached returned by sfapi_client.Client.detect(), string) – face (detection result) to be identified
- **threshold** (*float*) – identification threshold similarity between faces from 0 to 1.

Example: `Detection(det1, 0.77)` selects faces similar to the detection result `det1` with similarity greater or equal to `0.77`.

class ntech.sfapi_client.filters.**Face**(*Filter*)

Represents a method that searches the database for faces similar to a given face from a gallery.

__init__(*self*, *id: Union[str, objects.Face]*, *threshold: float*)

Parameters

- **id** (`objects.Face`, `objects.FaceId` or custom face id in the gallery, string) – face from a gallery to be identified
- **threshold** (*float*) – identification threshold similarity between faces from 0 to 1.

Example: Detection(FaceId(“gal1”, 1234), 0.77) selects faces similar to the face 1234 from the `gal1` gallery with similarity greater or equal than 0.77.

Several Filters Usage Example

```
filters=[filters.Id.gte(123456), filters.Meta('age').gte(45), filters.Meta('camera').
↪oneof('abc', 'def')]
```

Display Error Messages

class sfapi_client.**SFApiRemoteError**

This error message appears if the error occurred for a reason other than a network failure.

The error body always includes at least two fields:

- `code` is a short string in `CAPS_AND_UNDERSCORES`, usable for automatic decoding.
- `reason` is a human-readable description of the error and should not be interpreted automatically.

Common Error Codes

Error code	Description
UNKNOWN_ERROR	Error with unknown origin.
BAD_PARAM	The request can be read, however, some method parameters are invalid. This response type contains additional attributes `param` and``value`` to indicate which parameters are invalid.
CONFLICT	Conflict.
EXTRACTION_ERROR	Error upon a face feature vector extraction.
LICENSE_ERROR	The system configuration does not match license.
MALFORMED_REQUEST	The request is malformed and cannot be read.
OVER_CAPACITY	The `findface-extraction-api` queue length has been exceeded.
SOURCE_NOT_FOUND	The face in the `from` parameter does not exist.
SOURCE_GALLERY_NOT_FOUND	The gallery in the `from` parameter does not exist.
STORAGE_ERROR	The biometric database not available.
CACHE_ERROR	Memcached not available.
NOT_FOUND	Matching faces not found.
NOT_IMPLEMENTED	This functionality not implemented.
GALLERY_NOT_FOUND	Matching galleries not found.

class sfapi_client.**SFApiMalformedResponseError**

This error message appears if the error occurred due to a network failure, or if Client was unable to read an API response from `findface-sf-api`.

1.6.2 Installation File

FindFace Multi installation configuration is automatically saved to a file `/tmp/<findface-installer-*>.json`. You can edit this file and use it to install FindFace Multi on other hosts without having to answer the installation questions again.

Tip: See *Deploy from Console Installer* to learn more about the FindFace Multi installer.

Important: Be sure to remove fields `*.config`, `exp_ip`, and `int_ip` before installing FindFace Multi on a host with a different IP address.

Here is an example of the `installation file`.

1.6.3 Neural Network Models

Here you can see a summary for neural network models created by our Lab and used in FindFace Multi.

You can find installed models at `/usr/share/findface-data/models/`.

Important: The default face biometrics model upon a clean install is `kiwi_320`.

Face detection

```
ls /usr/share/findface-data/models/facedet/

cheetah.cpu.fnk  cheetah_fast.cpu.fnk  cheetah_fast.gpu.fnk  cheetah.gpu.fnk
```

Face and body image normalization

```
ls /usr/share/findface-data/models/facenorm/

ant.v2.cpu.fnk    bee.v2.gpu.fnk                crop2x.v2_no_maxsize.cpu.fnk
ant.v2.gpu.fnk    crop1x.v2_maxsize400.cpu.fnk  crop2x.v2_no_maxsize.gpu.fnk
bee_fast.cpu.fnk  crop1x.v2_maxsize400.gpu.fnk  cropbbox.v2.cpu.fnk
bee_fast.gpu.fnk  crop2x.v2_maxsize400.cpu.fnk  cropbbox.v2.gpu.fnk
bee.v2.cpu.fnk    crop2x.v2_maxsize400.gpu.fnk
```

Face recognition

```
ls /usr/share/findface-data/models/face/

kiwi_160.cpu.fnk  kiwi_320.cpu.fnk
kiwi_160.gpu.fnk  kiwi_320.gpu.fnk
```

Face attribute recognition

```
ls /usr/share/findface-data/models/faceattr/

age.v1.cpu.fnk       gender.v2.cpu.fnk           medmask3.v2.cpu.fnk
age.v1.gpu.fnk       gender.v2.gpu.fnk           medmask3.v2.gpu.fnk
beard.v0.cpu.fnk     glasses3.v0.cpu.fnk         quality.v1.cpu.fnk
beard.v0.gpu.fnk     glasses3.v0.gpu.fnk         quality.v1.gpu.fnk
emotions.v1.cpu.fnk  liveness.alleyn.v2.cpu.fnk
emotions.v1.gpu.fnk  liveness.alleyn.v2.gpu.fnk
```

Car detection

```
ls /usr/share/findface-data/models/cadet/

efreitor.cpu.fnk  efreitor.gpu.fnk
```

Car image normalization

```
ls /usr/share/findface-data/models/carnorm/

anaferon.v0.cpu.fnk  anaferon.v0.gpu.fnk
```

Car recognition

```
ls /usr/share/findface-data/models/carrec/

alonso.cpu.fnk  alonso.gpu.fnk
```

Car attribute recognition

```
ls /usr/share/findface-data/models/carattr/

carattr.license_plate.v0.cpu.fnk          carattr.quality.v0.cpu.fnk
carattr.license_plate.v0.gpu.fnk          carattr.quality.v0.gpu.fnk
carattr.license_plate_quality.v0.cpu.fnk  description.v0.cpu.fnk
carattr.license_plate_quality.v0.gpu.fnk  description.v0.gpu.fnk
```

Body detection

```
ls /usr/share/findface-data/models/pedet/

glenn_005.cpu.fnk  glenn_005.gpu.fnk  glenny_005_fast.cpu.fnk  glenny_005_fast.gpu.fnk
```

Body recognition

```
ls /usr/share/findface-data/models/pedrec/

andariel.cpu.fnk  andariel.gpu.fnk
```

Body attribute recognition

```
ls /usr/share/findface-data/models/pedattr/

pedattr.color.v1.cpu.fnk  pedattr.quality.v0.cpu.fnk
pedattr.color.v1.gpu.fnk  pedattr.quality.v0.gpu.fnk
```

1.6.4 FindFace Multi Data Storages

In this section:

List of Storages

FindFace Multi uses the following data storages:

- Tarantool-based feature vector database that stores object feature vectors and identification events.
- Main system database based on PostgreSQL, that stores internal system data, dossiers, user accounts, and camera settings.
- Directory `/var/lib/findface-security/uploads` that stores uploaded dossier photos, video files, and such event artifacts as full frames, object thumbnails, and normalized object images.
- Directory `/var/lib/ffupload/` that stores only such event artifacts as object thumbnails.

Feature Vector Database Galleries

There are the following galleries in the Tarantool-based feature vector database:

- `ffsec_body_events`: feature vectors extracted from bodies detected in the video.
- `ffsec_body_objects`: feature vectors extracted from bodies in dossier photos.
- `ffsec_car_events`: feature vectors extracted from cars detected in the video.
- `ffsec_car_objects`: feature vectors extracted from cars in dossier photos.
- `ffsec_face_events`: feature vectors extracted from faces detected in the video.
- `ffsec_face_objects`: feature vectors extracted from faces in dossier photos.
- `ffsec_user_face`: feature vectors extracted from the FindFace Multi users' photos, used for face-based authentication.
- `ffsec_persons`: centroids of persons (virtual feature vectors averaged across all person's faces) and metadata.

1.6.5 Backup Options

To backup the feature vector database, you need the `findface-storage-api-dump` utility. It can be launched with the following options:

Note: You can find the detailed information on the `findface-storage-api-dump` usage in *Back Up and Recover Data Storages*.

```
findface-storage-api-dump --help

Usage of findface-storage-api-dump:
  -cache string
        Cache type: inmemory, redis or memcache (default "memcache")
  -cache-inmemory-size int
        Maximum number of items in ARC cache (default 16384)
  -cache-memcache-nodes value
        Comma-separated list of memcache shards (default 127.0.0.1:11211)
  -cache-memcache-timeout duration
        Specifies read/write timeout (default 100ms)
  -cache-redis-addr string
        Host:Port address (default "localhost:6379")
  -cache-redis-db int
        Database to be selected after connecting to the server.
  -cache-redis-network string
        Network type, either tcp or unix (default "tcp")
  -cache-redis-password string
        Optional password. Must match the password specified in the requirepass server␣
↪configuration option.
  -cache-redis-timeout duration
        Specifies dial/read/write timeout (default 5s)
  -config string
        Path to config file
  -config-template
        Output config template and exit
  -continue-on-errors
        Continue on errors instead of exiting
  -cpu-profile string
        Enable CPU profile and set output file
  -extraction-api-extraction-api string
        Extraction API address (default "http://127.0.0.1:18666")
  -extraction-api-timeouts-connect duration
        extraction-api-timeouts-connect (default 5s)
  -extraction-api-timeouts-idle-connection duration
        extraction-api-timeouts-idle-connection (default 10s)
  -extraction-api-timeouts-overall duration
        extraction-api-timeouts-overall (default 35s)
  -extraction-api-timeouts-response-header duration
        extraction-api-timeouts-response-header (default 30s)
  -limits-allow-return-facen
        Allow returning raw feature vectors to detect responses if ?return_facen=true
  -limits-body-image-length int
        Maximum length of image supplied in request body (default 33554432)
```

(continues on next page)

(continued from previous page)

```
  -limits-deny-networks string
        Comma-separated list of subnets that are not allowed to fetch from (default "127.
↪0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8")
  -limits-url-length int
        Maximum supported url length in bytes (default 4096)
  -listen string
        IP:port to listen on (default ":18411")
  -normalized-storage-enabled
        Enables normalize saving (default true)
  -normalized-storage-s3-access-key string
        Access key for the object storage
  -normalized-storage-s3-bucket-name string
        S3 storage bucket name
  -normalized-storage-s3-endpoint string
        S3 compatible object storage endpoint
  -normalized-storage-s3-operation-timeout int
        Storage operations (Get,Put,Delete) timeout in seconds (default 30)
  -normalized-storage-s3-public-url string
        Storage public url
  -normalized-storage-s3-region string
        Storage region
  -normalized-storage-s3-secret-access-key string
        Secret key for the object storage
  -normalized-storage-s3-secure
        If 'true' API requests will be secure (HTTPS), and insecure (HTTP) otherwise␣
↪(default true)
-normalized-storage-webdav-timeouts-connect duration
        normalized-storage-webdav-timeouts-connect (default 5s)
  -normalized-storage-webdav-timeouts-idle-connection duration
        normalized-storage-webdav-timeouts-idle-connection (default 10s)
  -normalized-storage-webdav-timeouts-overall duration
        normalized-storage-webdav-timeouts-overall (default 35s)
  -normalized-storage-webdav-timeouts-response-header duration
        normalized-storage-webdav-timeouts-response-header (default 30s)
  -normalized-storage-webdav-upload-url string
        webdav storage for normalized, disable normalized if empty string (default
↪"http://127.0.0.1:3333/uploads/")
  -normalized_storage string
        Normalized storage type: webdav, s3 (default "webdav")
  -output-dir string
        Output directory (default ".")
  -storage-api-max-idle-conns-per-host int
        storage-api-max-idle-conns-per-host (default 20)
  -storage-api-timeouts-connect duration
        storage-api-timeouts-connect (default 5s)
  -storage-api-timeouts-idle-connection duration
        storage-api-timeouts-idle-connection (default 10s)
  -storage-api-timeouts-overall duration
        storage-api-timeouts-overall (default 35s)
  -storage-api-timeouts-response-header duration
        storage-api-timeouts-response-header (default 30s)
```

1.6.6 Restore Options

To restore the feature vector database from a backup, you need the `findface-storage-api-restore` utility. It can be launched with the following options:

Note: You can find the detailed information on the `findface-storage-api-restore` usage in *Back Up and Recover Data Storages*.

```
findface-storage-api-restore --help

Usage of findface-storage-api-restore:
  -cache string
        Cache type: inmemory, redis or memcache (default "memcache")
  -cache-inmemory-size int
        Maximum number of items in ARC cache (default 16384)
  -cache-memcache-nodes value
        Comma-separated list of memcache shards (default 127.0.0.1:11211)
  -cache-memcache-timeout duration
        Specifies read/write timeout (default 100ms)
  -cache-redis-addr string
        Host:Port address (default "localhost:6379")
  -cache-redis-db int
        Database to be selected after connecting to the server.
  -cache-redis-network string
        Network type, either tcp or unix (default "tcp")
  -cache-redis-password string
        Optional password. Must match the password specified in the requirepass server␣
↪configuration option.
  -cache-redis-timeout duration
        Specifies dial/read/write timeout (default 5s)
  -config string
        Path to config file
  -config-template
        Output config template and exit
  -cpu-profile string
        Enable CPU profile and set output file
  -dont-create-gallery
        Don't create gallery, fail if doesn't exist
  -extraction-api-extraction-api string
        Extraction API address (default "http://127.0.0.1:18666")
  -extraction-api-timeouts-connect duration
        extraction-api-timeouts-connect (default 5s)
  -extraction-api-timeouts-idle-connection duration
        extraction-api-timeouts-idle-connection (default 10s)
  -extraction-api-timeouts-overall duration
        extraction-api-timeouts-overall (default 35s)
  -extraction-api-timeouts-response-header duration
        extraction-api-timeouts-response-header (default 30s)
  -limits-allow-return-facen
        Allow returning raw feature vectors to detect responses if ?return_facen=true
  -limits-body-image-length int
        Maximum length of image supplied in request body (default 33554432)
```

(continues on next page)

(continued from previous page)

```
  -limits-deny-networks string
        Comma-separated list of subnets that are not allowed to fetch from (default "127.
↪0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8")
  -limits-url-length int
        Maximum supported url length in bytes (default 4096)
  -listen string
        IP:port to listen on (default ":18411")
  -normalized-storage-enabled
        Enables normalize saving (default true)
  -normalized-storage-s3-access-key string
        Access key for the object storage
  -normalized-storage-s3-bucket-name string
        S3 storage bucket name
  -normalized-storage-s3-endpoint string
        S3 compatible object storage endpoint
  -normalized-storage-s3-operation-timeout int
        Storage operations (Get,Put,Delete) timeout in seconds (default 30)
  -normalized-storage-s3-public-url string
        Storage public url
  -normalized-storage-s3-region string
        Storage region
  -normalized-storage-s3-secret-access-key string
        Secret key for the object storage
  -normalized-storage-s3-secure
        If 'true' API requests will be secure (HTTPS), and insecure (HTTP) otherwise␣
↪(default true)
  -normalized-storage-webdav-timeouts-connect duration
        normalized-storage-webdav-timeouts-connect (default 5s)
  -normalized-storage-webdav-timeouts-idle-connection duration
        normalized-storage-webdav-timeouts-idle-connection (default 10s)
  -normalized-storage-webdav-timeouts-overall duration
        normalized-storage-webdav-timeouts-overall (default 35s)
  -normalized-storage-webdav-timeouts-response-header duration
        normalized-storage-webdav-timeouts-response-header (default 30s)
  -normalized-storage-webdav-upload-url string
        webdav storage for normalized, disable normalized if empty string (default
↪"http://127.0.0.1:3333/uploads/")
  -normalized_storage string
        Normalized storage type: webdav, s3 (default "webdav")
  -rename string
        Ignore dump header and use this string as gallery name
  -storage-api-max-idle-conns-per-host int
        storage-api-max-idle-conns-per-host (default 20)
  -storage-api-timeouts-connect duration
        storage-api-timeouts-connect (default 5s)
  -storage-api-timeouts-idle-connection duration
        storage-api-timeouts-idle-connection (default 10s)
  -storage-api-timeouts-overall duration
        storage-api-timeouts-overall (default 35s)
  -storage-api-timeouts-response-header duration
        storage-api-timeouts-response-header (default 30s)
```

CHAPTER TWO

USER'S GUIDE

This chapter describes how to work with the FindFace Multi web interface, including its advanced possibilities, and will be of interest to administrators, analysts, operators, and other users.

2.1 First Steps after Deployment

Once you have successfully deployed FindFace Multi, it is time to *to perform the primary configuration*, open the *web interface*, and get started. In this chapter, you can find a recommended sequence of steps that will help you harness your system's complete functionality.

In this chapter:

2.1.1 Gear Up for Work

Perform the primary configuration of your system:

1. *Set up* the left side navigation bar.
2. *Adapt* general preferences.
3. *Choose* the language.

You may also need:

1. *Enable face attribute recognition*
2. *Protect your system from spoofing with the liveness detector*
3. *Enable car and car attribute recognition*
4. *Enable body and body attribute recognition*

2.1.2 Create Users and Ensure System Security

1. Check out the list of *predefined user roles* and *create new roles* if necessary.
2. *Add users* to the system and grant them privileges.
3. *Configure* authentication and user session monitoring. Authentication is possible by password, face, face or password, face and password.

You may also need:

1. *Enable* SSL data encryption.
2. *Enable* dossier security. If the dossier security is disabled, the dossier photos and attachments will be available by direct link regardless of the user rights.
3. *Disable* FindFace Multi ACL if you do not need it, as the constant permission checks consume a significant amount of system resources.

2.1.3 Organize Cameras

1. *Create a new camera group* or use the default one. A camera group is an entity that allows you to group cameras subject to their physical location. For example, cameras at the same entrance to a building can be combined into one camera group.
2. *Add cameras* to the camera group and *check their statuses*.

You may also need:

1. Configure your system to process video from the group of cameras at their physical location. It may come in handy in a distributed architecture. *Learn more*.
2. Consider enabling event deduplication if observation scenes of cameras within the group overlap. This feature allows you to exclude coinciding object recognition events among cameras belonging to the same group. *Learn more*.

2.1.4 Organize Watch Lists and Dossiers

1. *Create a new watch list* or use the default one. A watch list is an entity that allows you to classify objects (faces, bodies, cars) by arbitrary criteria, e.g., persona non grata, wanted, VIP, staff, etc.
2. Upload dossiers and add them in the watch list either *manually*, *in bulk via the web interface*, or use the *console bulk upload* function.

You may also need:

1. *Distribute dossier database* among several hosts. The dossier database will be available for editing on the `master` server and reading and monitoring on the `slaves`.
2. *Customize dossier content*. Create additional fields, tabs, and search filters.

2.1.5 Start Monitoring Objects

By default, FindFace Multi is monitoring only *unmatched objects*. To enable a watch list monitoring, make this list *active*. You can also turn on sound notifications and request manual acknowledgment for the events associated with the list.

You may also need:

1. Support laws related to the processing of personal data of individuals (GDPR and similar). *Learn more*.

2.1.6 Organize Video Surveillance

Create a camera layout for essential video surveillance.

2.1.7 Start Counting Faces and Bodies

Set up *counters* to count faces and bodies on connected cameras. This functionality can apply to a wide range of situations, such as people counting in queues and waiting areas, monitoring public gatherings, crowding prevention, and more.

2.1.8 Start Analyzing People

FindFace Multi provide a set of people-related analytical tools:

1. *Enable* person recognition to build a person gallery. The system databases will hold a new entity `person event` linked to all *episodes* that feature a person's face. You can work with the person gallery similarly as with events and episodes.
2. *Analyze* social interactions. Examine a circle of people with whom a person has previously been in contact.
3. View 'know your customer' analytics (KYC). It is analytics on the number of visitors, their gender, average age, most frequently visited zones, and the character of visits (first-timers or returners). *Learn more*.

2.1.9 FindFace Multi in Action

1. *Automatically identify objects (faces, bodies, cars) in live video* and check them against watch lists. Work with the event history by using various filters.
2. Harness the *episodes* (only face recognition events). An episode is a set of identification events that feature faces of the same person, detected within a certain period. As events on the *Events* tab show up in an arbitrary order, a large number of miscellaneous events can make the work challenging and unproductive. With the Episodes, the system uses AI to organize incoming events based on the faces similarity and detection time. This allows for the effortless processing of diverse events, even in large numbers.
3. Search for objects in the database of detected objects and dossier database. *Learn more*.
4. *Search archived videos* for objects under monitoring.
5. Manually *compare two objects* and verify that they match.
6. *Build* detailed reports on object recognition events, episodes, search events, persons, counters, cameras, dossiers, KYC analytics, and audit logs.

2.1.10 Basic Maintenance

1. *Configure* automatic cleanup of events, episodes, and full frames.
2. Manually *purge* events, episodes, and full frames.
3. Regularly *backup* the database.
4. *Harness* the FindFace Multi comprehensive and searchable audit logs to enhance your system protection.

2.1.11 Go Further

1. Set up *webhooks* to automatically send notifications about specific events, episodes, and counter records to a given URL. In this case, when such an event occurs, FindFace Multi will send an HTTP request to the URL configured for the webhook. You can use webhooks for various purposes, for example, to notify a user about a particular event, invoke required behavior on a target website, and solve security tasks such as automated access control. *Learn more*.
2. Harness the FindFace Multi functions through *HTTP API*.
3. Check out the list of our *partner integrations*.

See also:

- *Primary Configuration*

- *User Management and System Security*
- *Camera Management*
- *Configure Object Monitoring. Dossier Database*
- *Object Identification in Offline Videos*
- *Face and Body Counters*
- *Person Recognition and People-Related Analytics*
- *Advanced Functionality*
- *Maintenance and Troubleshooting*

2.2 Work with FindFace Multi

Use the web interface to interact with FindFace Multi. To open the web interface, enter its basic address in the address bar of your browser, and log in.

Note: The basic address is set during *deployment*.

Important: To log in for the first time, use the admin account created during *deployment*. To create more users, refer to *User Management*.

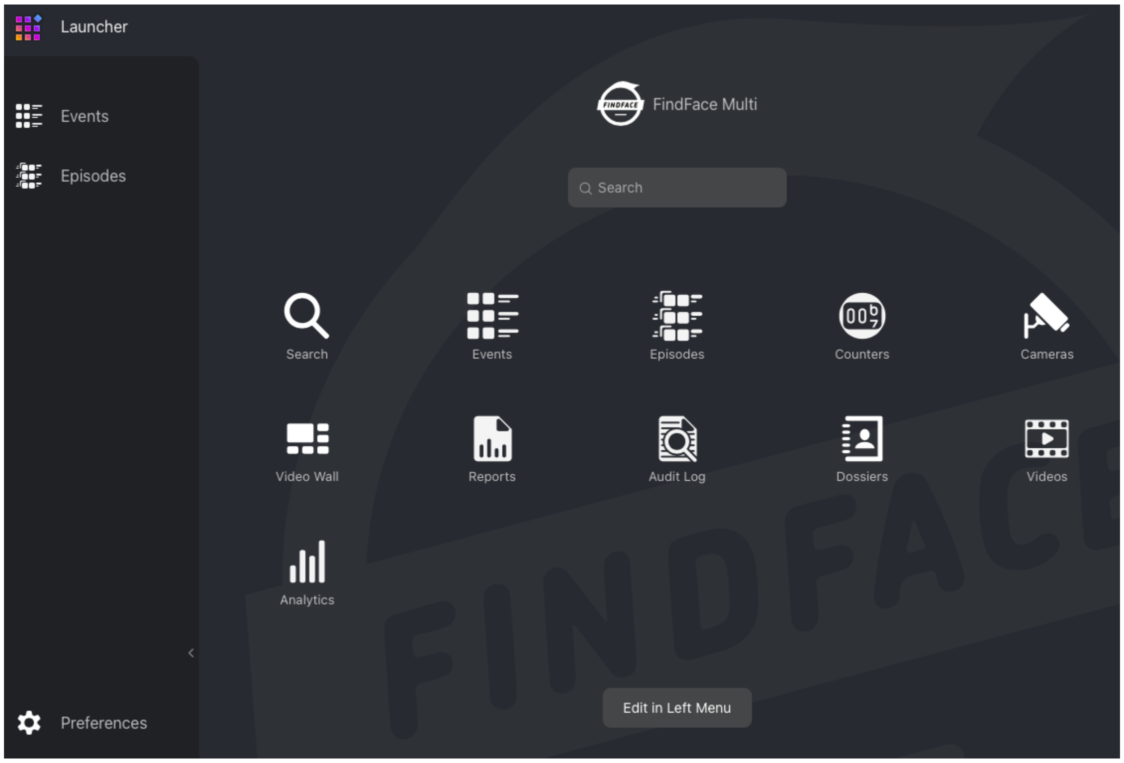
Tip: Take your system security up a notch with *face-based authentication*.

2.2.1 Primary Configuration

This section is about the FindFace Multi primary configuration. Learn how to configure the left side navigation bar, enable additional objects to recognize, schedule the automatic database cleanup, switch the language, and more.

Navigation

By default, there are only two items in the left side navigation bar: *Events* and *Episodes*. Use *Launcher* to get access to the other FindFace Multi tabs.



Through *Launcher*, you can also make your favorite tabs permanently available from the navigation bar. Do the following:

1. Click *Edit in Left Menu*.
2. Check and uncheck navigation items, subject to your needs.
3. Click *Finish Editing*.

Enable Face Attribute Recognition

Subject to your needs, you can enable automatic recognition of such face attributes as gender, age, emotions, glasses, beard, and face mask. This functionality can be activated on both GPU- and CPU-accelerated video object detectors.

To enable automatic recognition of face attributes, do the following:

1. Open the `/etc/findface-extraction-api.ini` configuration file.

```
sudo vi /etc/findface-extraction-api.ini
```

2. Specify the relevant recognition models in the `extractors` section, as shown in the example below. Be sure to indicate the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models.

```
extractors:
  face_age: faceattr/age.v1.gpu.fnk
  face_beard: faceattr/beard.v0.gpu.fnk
  face_emotions: faceattr/emotions.v1.gpu.fnk
  face_gender: faceattr/gender.v2.gpu.fnk
```

(continues on next page)

(continued from previous page)

```
  face_glasses3: faceattr/glasses3.v0.gpu.fnk
  face_medmask3: faceattr/medmask3.v2.gpu.fnk
```

The following models are available:

Face attribute	Acceleration	Configure as follows
age	CPU	`face_age: faceattr/age.v1.cpu.fnk`
	GPU	`face_age: faceattr/age.v1.gpu.fnk`
gender	CPU	`face_gender: faceattr/gender.v2.cpu.fnk`
	GPU	`face_gender: faceattr/gender.v2.gpu.fnk`
emotions	CPU	`face_emotions: faceattr/emotions.v1.cpu.fnk`
	GPU	`face_emotions: faceattr/emotions.v1.gpu.fnk`
glasses	CPU	`face_glasses3: faceattr/glasses3.v0.cpu.fnk`
	GPU	`face_glasses3: faceattr/glasses3.v0.gpu.fnk`
beard	CPU	`face_beard: faceattr/beard.v0.cpu.fnk`
	GPU	`face_beard: faceattr/beard.v0.gpu.fnk`
face mask	CPU	`face_medmask3: faceattr/medmask3.v2.cpu.fnk`
	GPU	`face_medmask3: faceattr/medmask3.v2.gpu.fnk`

Tip: To leave a recognition model disabled, pass the empty value "" to the relevant parameter. Do not remove the parameter itself. Otherwise, the system will be searching for the default model.

```
extractors:
  face_age: ""
  face_beard: ""
  face_emotions: ""
  face_gender: ""
  face_glasses3: ""
  face_medmask3: ""
```

Note: You can find face attribute recognition models at `/usr/share/findface-data/models/faceattr/`.

```
ls /usr/share/findface-data/models/faceattr/
age.v1.cpu.fnk  age.v1.gpu.fnk  beard.v0.cpu.fnk  beard.v0.gpu.fnk  emotions.v1.cpu.
↪fnk  emotions.v1.gpu.fnk  gender.v2.cpu.fnk  gender.v2.gpu.fnk  glasses3.v0.cpu.
↪fnk  glasses3.v0.gpu.fnk medmask3.v2.cpu.fnk medmask3.v2.gpu.fnk liveness.alleyn.
↪v2.gpu.fnk quality.v1.cpu.fnk quality.v1.gpu.fnk
```

3. Restart `findface-extraction-api`.

```
sudo systemctl restart findface-extraction-api
```

4. To display the face attribute recognition results in the event list, open the `/etc/findface-security/config.py` configuration file.

```
sudo vi /etc/findface-security/config.py
```

5. Specify the required models in the following line of the `FFSECURITY` section, subject to the list of enabled models:

```
FFSECURITY = {
    ...
    'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses', 'medmask']
    ...
}
```

6. Restart findface-security.

```
sudo systemctl restart findface-security
```

Real-time Face Liveness Detection

Note: The *liveness detector* is much slower on CPU than on GPU.

To spot fake faces and prevent photo attacks, use the integrated 2D anti-spoofing system that distinguishes a live face from a face image. Due to the analysis of not one, but a number of frames, the algorithm captures any changes in a facial expression and skin texture. This ensures that it is a live person in front of a camera and eliminates the possibility of fraud using images on paper or mobile device screens.

The liveness detector estimates a face liveness with a certain level of confidence and returns the confidence score along with a binary result `real/fake`, depending on the pre-defined liveness threshold.

In this section:

Enable Face Liveness Detector

To enable the face liveness detector, do the following:

1. Open the `/etc/findface-video-worker-gpu.ini` (`/etc/findface-video-worker-cpu.ini`) configuration file. In the `liveness` section, specify the path to the neural network model (`fnk`) and normalizer (`norm`) which are used in the face liveness detector.

```
sudo vi /etc/findface-video-worker-gpu.ini

#------------------------------
[liveness]
#------------------------------
## path to liveness fnk
## type:string env:CFG_LIVENESS_FNK longopt:--liveness-fnk
fnk = /usr/share/findface-data/models/faceattr/liveness.alleyn.v2.gpu.fnk

## path to normalization for liveness
```

(continues on next page)

(continued from previous page)

```
## type:string env:CFG_LIVENESS_NORM longopt:--liveness-norm
norm = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.gpu.fnk
```

```
sudo vi /etc/findface-video-worker-cpu.ini

#------------------------------
[liveness]
#------------------------------
## path to liveness fnk
## type:string env:CFG_LIVENESS_FNK longopt:--liveness-fnk
fnk = /usr/share/findface-data/models/faceattr/liveness.alleyn.v2.cpu.fnk

## path to normalization for liveness
## type:string env:CFG_LIVENESS_NORM longopt:--liveness-norm
norm = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.cpu.fnk
```

2. Restart `findface-video-worker`.

```
sudo systemctl restart findface-video-worker-gpu
sudo systemctl restart findface-video-worker-cpu
```

Configure Liveness Threshold

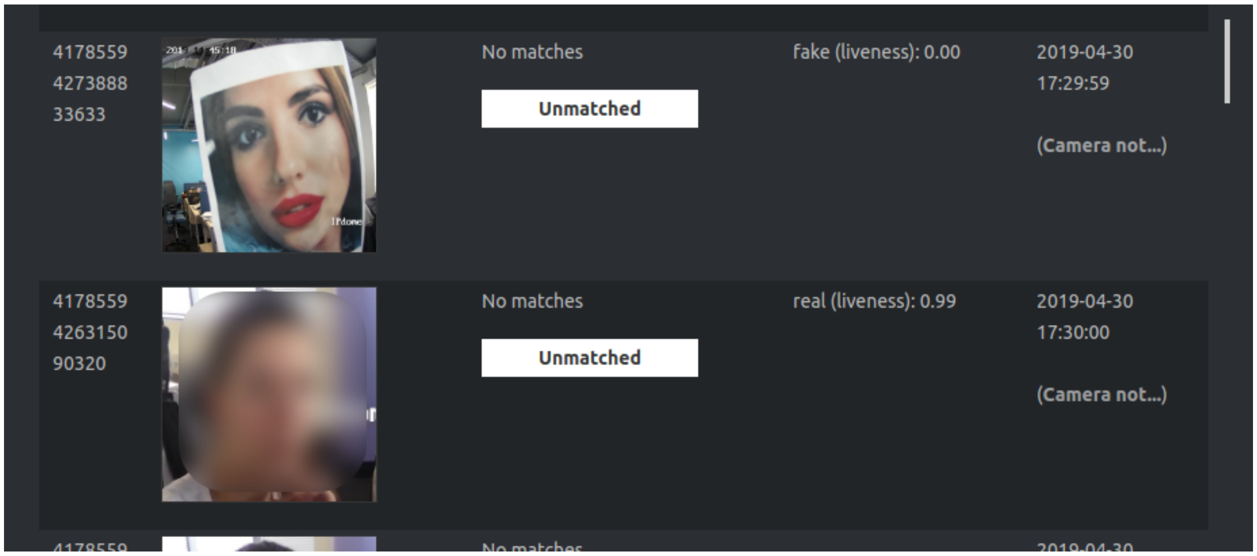
If necessary, you can adjust the liveness `threshold` in the `/etc/findface-security/config.py` configuration file. The liveness detector will estimate a face liveness with a certain level of confidence. Depending on the threshold value, it will return a binary result `real` or `fake`.

Note: The default value is optimal. Before changing the threshold, we recommend you to seek advice from our experts by support@ntechlab.com.

```
sudo vi /etc/findface-security/config.py

'LIVENESS_THRESHOLD': 0.85,
```

Face Liveness in Web Interface

Once the face liveness detector configured, you will see liveness estimation for each event.



4178559
4273888
33633

No matches

Unmatched

fake (liveness): 0.00

2019-04-30
17:29:59

(Camera not...)

4178559
4263150
90320

No matches

Unmatched

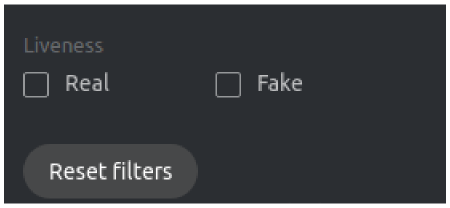
real (liveness): 0.99

2019-04-30
17:30:00

(Camera not...)

Note: The liveness score is `null` when the liveness detector is unable to estimate the face liveness in the provided image.

Use the *Liveness* filter to display only real or only fake faces in the event list.



See also:

Liveness Detection as Standalone Service

Enable Car and Car Attribute Recognition

FindFace Multi allows you to recognize individual cars and car attributes.

The attributes are the following:

- license plate number (for selected countries),
- color,
- make,
- model,
- car body style.

Important: Recognition of individual cars is an experimental feature. Therefore, we highly recommend you enable the additional attribute analysis to improve the recognition quality. In this case, the system compares not only the feature vectors of two cars but also their attributes such as color, body style, make, and model. A conclusion about the cars' match is only made if both the feature vectors and attributes of the cars coincide.

See the detailed description of how to enable the additional attribute analysis in the step-by-step instructions below.

To enable recognition of cars and their attributes, do the following:

1. Specify neural network models for car object and car attribute recognition in the /etc/findface-extraction-api.ini configuration file. Do the following:

   Important: Be sure to choose the right acceleration type for each model, matching the acceleration type of findface-extraction-api: CPU or GPU. Be aware that findface-extraction-api on CPU can work only with CPU-models, while findface-extraction-api on GPU supports both CPU- and GPU-models.

   1. Open the /etc/findface-extraction-api.ini configuration file.

      ```
      sudo vi /etc/findface-extraction-api.ini
      ```

   2. Specify the car detector model in the detectors -> models section by pasting the following code:

      ### GPU

      ```
      detectors:

        ...
        models:
          ...
          efreitor:
            aliases:
            - car
            model: cadet/efreitor.gpu.fnk
            options:
              min_object_size: 32
              resolutions: [256x256, 384x384, 512x512, 768x768, 1024x1024, 1536x1536,␣
      ↪2048x2048]

        ...
      ```

      ### CPU

      ```
      detectors:

        ...
        models:
          ...
          efreitor:
            aliases:
            - car
            model: cadet/efreitor.cpu.fnk
            options:
      ```

(continues on next page)

(continued from previous page)

```
      min_object_size: 32
      resolutions: [256x256, 384x384, 512x512, 768x768, 1024x1024, 1536x1536,␣
↪2048x2048]

  ...
```

3. Specify the extraction models in the `extractors -> models` section, subject to the extractors you want to enable:

GPU

```
extractors:
  ...
  models:
   car_color: ''
   car_description: carattr/description.v0.gpu.fnk
   car_emben: carrec/alonso.gpu.fnk
   car_license_plate: carattr/carattr.license_plate.v0.gpu.fnk
   car_license_plate_quality: carattr/carattr.license_plate_quality.v0.gpu.fnk
   car_make: ''
   car_quality: carattr/carattr.quality.v0.gpu.fnk
```

CPU

```
extractors:
  ...
  models:
   car_color: ''
   car_description: carattr/description.v0.cpu.fnk
   car_emben: carrec/alonso.cpu.fnk
   car_license_plate: carattr/carattr.license_plate.v0.cpu.fnk
   car_license_plate_quality: carattr/carattr.license_plate_quality.v0.cpu.fnk
   car_make: ''
   car_quality: carattr/carattr.quality.v0.cpu.fnk
```

The following extractors are available:

Extractor	Configure as follows
individual car object	`car_emben: carrec/alonso.cpu.fnk`
	`car_emben: carrec/alonso.gpu.fnk`
license plate number	`car_license_plate: carattr/carattr.license_plate.v0.cpu.fnk car_license_plate_quality: carattr/carattr.license_plate_quality.v0.cpu.fnk`
	`car_license_plate: carattr/carattr.license_plate.v0.gpu.fnk car_license_plate_quality: carattr/carattr.license_plate_quality.v0.gpu.fnk`
set of attributes: make / color / model / body style	`car_description: carattr/description.v0.cpu.fnk`
	`car_description: carattr/description.v0.gpu.fnk`
car image quality	`car_quality: carattr/carattr.quality.v0.cpu.fnk`
	`car_quality: carattr/carattr.quality.v0.gpu.fnk`

Tip: To leave a model disabled, pass the empty value `''` to the relevant parameter. Do not remove the parameter itself. Otherwise, the system will be searching for the default model.

```
extractors:
  ...
  models:
    car_color: ''
    car_description: ''
    car_emben: ''
    car_license_plate: ''
    car_license_plate_quality: ''
    car_make: ''
    car_quality: ''
    car_trash: ''
```

4. Specify the normalizers required for the extractors specified in the previous step. For example, if you need license plate recognition, specify the `carlicplate` normalizer.

Normalizer	Normalizer model	Used for extractors
carlicplate	`carnorm/anaferon.v0.gpu.fnk` `carnorm/anaferon.v0.cpu.fnk`	`car_license_plate`
cropbbox	`facenorm/cropbbox.v2.gpu.fnk` `facenorm/cropbbox.v2.cpu.fnk`	`car_license_plate_quality`, `car_description`, `car_quality`

GPU

```
normalizers:
  ...

  models:
    carlicplate:
      model: carnorm/anaferon.v0.gpu.fnk
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
```

CPU

```
normalizers:
  ...

  models:
    carlicplate:
      model: carnorm/anaferon.v0.cpu.fnk
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
```

5. Make sure that the `objects -> car` section contains the `quality_attribute:  car_quality` and `base_normalizer:  facenorm/cropbbox.v2.*.fnk`:

GPU

```
objects:
  ...
  car:
    base_normalizer: facenorm/cropbbox.v2.gpu.fnk
    quality_attribute: car_quality
  ...
```

CPU

```
objects:
  ...
  car:
    base_normalizer: facenorm/cropbbox.v2.cpu.fnk
    quality_attribute: car_quality
```

6. After completing the configuration file modification, make sure it looks similar to the `following example`.

7. Restart `findface-extraction-api`.

```
sudo systemctl restart findface-extraction-api
```

2. Modify the `/etc/findface-video-worker-gpu.ini` (`/etc/findface-video-worker-cpu.ini`) configuration file. In the `car` section, specify the neural network models by analogy with the example below. Restart `findface-video-worker-gpu` (`findface-video-worker-cpu`).

GPU

```
sudo vi /etc/findface-video-worker-gpu.ini


#------------------------------
[car]
#------------------------------
## detector param
## type:number env:CFG_CAR_MIN_SIZE longopt:--car-min-size
min_size = 60

## path to car detector
## type:string env:CFG_CAR_DETECTOR longopt:--car-detector
detector = /usr/share/findface-data/models/cadet/efreitor.gpu.fnk

## path to normalizer (usually crop2x)
## type:string env:CFG_CAR_NORM longopt:--car-norm
norm = /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk

## path to car quality extractor
## type:string env:CFG_CAR_QUALITY longopt:--car-quality
quality = /usr/share/findface-data/models/carattr/carattr.quality.v0.gpu.fnk

## path to car quality normalizer
## type:string env:CFG_CAR_NORM_QUALITY longopt:--car-norm-quality
norm_quality = /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
```

```
sudo systemctl restart findface-video-worker-gpu.service
```

CPU

```
sudo vi /etc/findface-video-worker-cpu.ini


#------------------------------
[car]
#------------------------------
## detector param
## type:number env:CFG_CAR_MIN_SIZE longopt:--car-min-size
min_size = 60

## path to car detector
## type:string env:CFG_CAR_DETECTOR longopt:--car-detector
```

(continues on next page)

(continued from previous page)

```
detector = /usr/share/findface-data/models/cadet/efreitor.cpu.fnk

## path to normalizer (usually crop2x)
## type:string env:CFG_CAR_NORM longopt:--car-norm
norm = /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk

## path to car quality extractor
## type:string env:CFG_CAR_QUALITY longopt:--car-quality
quality = /usr/share/findface-data/models/carattr/carattr.quality.v0.cpu.fnk

## path to car quality normalizer
## type:string env:CFG_CAR_NORM_QUALITY longopt:--car-norm-quality
norm_quality = /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
```

```
sudo systemctl restart findface-video-worker-cpu.service
```

3. Open the `/etc/findface-video-manager.conf` configuration file and make sure it contains the `car` section in `detectors` that looks similar to the example below.

Tip: As a reference value for the `filter_min_quality` parameter, you can take the `MINIMUM_CAR_QUALITY` parameter value from the `/etc/findface-security/config.py` configuration file.

```
sudo vi /etc/findface-video-manager.conf

detectors:
    ...
    car:
      filter_min_quality: 0.73
      filter_min_size: 1
      filter_max_size: 8192
      roi: ''
      fullframe_crop_rot: false
      fullframe_use_png: false
      jpeg_quality: 95
      overall_only: false
      realtime_post_first_immediately: false
      realtime_post_interval: 1
      realtime_post_every_interval: false
      track_interpolate_bboxes: true
      track_miss_interval: 1
      track_overlap_threshold: 0.25
      track_max_duration_frames: 0
      track_send_history: false
      post_best_track_frame: true
      post_best_track_normalize: true
      post_first_track_frame: false
      post_last_track_frame: false
```

4. Enable the recognition of cars and car attributes in the `/etc/findface-security/config.py` configuration file. Do the following:

   1. In the `FFSECURITY` section, set `'ENABLE_CARS': True`.

```
sudo vi /etc/findface-security/config.py

FFSECURITY = {
    ...

    # optional objects to detect
    'ENABLE_CARS': True,
    ...
```

2. In the same section, specify the car attributes you want to display for the car recognition events.

```
# available features are: description, license_plate
'CAR_EVENTS_FEATURES': ['description', 'license_plate'],
```

3. To improve the quality of individual car recognition, we highly recommend you enable the additional attribute analysis. In this case, the system compares not only the feature vectors of two cars but also their attributes. A conclusion about the cars' match is only made if both the feature vectors and attributes of the cars coincide.

   You can use the following attributes for additional analysis:

   - `color`: car color,
   - `body`: body style,
   - `make`: make,
   - `model`: model.

   To enable the additional attribute analysis, set `True` in the `FFSECURITY` -> `EXTRA_CAR_MATCHING` section for the attributes that you want to compare.

```
FFSECURITY = {
    # use additional features for extra confidence when matching cars by emben
    'EXTRA_CAR_MATCHING': {
        'color': {'enabled': False, 'min_confidence': 0},
        'body': {'enabled': False, 'min_confidence': 0},
        'make': {'enabled': False, 'min_confidence': 0},
        'model': {'enabled': False, 'min_confidence': 0},
    },
```

Important: For the attribute analysis to function, the `description` model must be enabled in the `/etc/findface-extraction-api.ini` and `/etc/findface-security/config.py` configuration files (see above).

Note: Enabling the additional attribute analysis reduces the number of false positives. However, the system might miss out on some real matches as well.

Warning: Do not change the default values of `min_confidence` without consulting with our technical experts (support@ntechlab.com).

4. Restart the `findface-security` service.

```
sudo systemctl restart findface-security.service
```

Enable Body and Body Attribute Recognition

FindFace Multi allows you to recognize individual human bodies and body attributes (currently only clothing color).

To enable recognition of human bodies and their attributes, do the following:

1. Specify neural network models for body object and body attribute recognition in the `/etc/findface-extraction-api.ini` configuration file. Do the following:

   Important: Be sure to choose the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models.

   1. Open the `/etc/findface-extraction-api.ini` configuration file.

      ```
      sudo vi /etc/findface-extraction-api.ini
      ```

   2. Specify the body detector model in the `detectors -> models` section by pasting the following code:

      #### GPU

      ```
      detectors:

        ...
        models:
          ...
          glenn:
            aliases:
            - body
            - silhouette
            model: pedet/glenn_005.gpu.fnk
            options:
              min_object_size: 32
              resolutions: [256x256, 384x384, 512x512, 768x768, 1024x1024, 1536x1536,␣
      ↪2048x2048]

        ...
      ```

      #### CPU

      ```
      detectors:

        ...
        models:
          ...
          glenn:
            aliases:
      ```

(continues on next page)

(continued from previous page)

```
      - body
      - silhouette
      model: pedet/glenn_005.cpu.fnk
      options:
        min_object_size: 32
        resolutions: [256x256, 384x384, 512x512, 768x768, 1024x1024, 1536x1536,␣
↪2048x2048]

  ...
```

3. Specify the extraction models in the `extractors -> models` section, subject to the extractors you want to enable:

GPU

```
extractors:
   ...
   models:
    body_color: pedattr/pedattr.color.v1.gpu.fnk
    body_emben: pedrec/andariel.gpu.fnk
    body_quality: pedattr/pedattr.quality.v0.gpu.fnk
```

CPU

```
extractors:
   ...
   models:
    body_color: pedattr/pedattr.color.v1.cpu.fnk
    body_emben: pedrec/andariel.cpu.fnk
    body_quality: pedattr/pedattr.quality.v0.cpu.fnk
```

The following extractors are available:

Recognition type	Configure as follows
clothing color	`body_color: pedattr/pedattr.color.v1.gpu.fnk`
	`body_color: pedattr/pedattr.color.v1.cpu.fnk`
individual body object	`body_emben: pedrec/andariel.gpu.fnk`
	`body_emben: pedrec/andariel.cpu.fnk`
body quality	`body_quality: pedattr/pedattr.quality.v0.gpu.fnk`
	`body_quality: pedattr/pedattr.quality.v0.cpu.fnk`

Tip: To leave a model disabled, pass the empty value `''` to the relevant parameter. Do not remove the parameter itself. Otherwise, the system will be searching for the default model.

```
extractors:
   ...
   models:
```

(continues on next page)

(continued from previous page)

```
    body_color: ''
    body_emben: ''
    body_quality: ''
```

4. Make sure that the `normalizers` section contains a model for the `cropbbox` normalizer, as shown in the example below. This normalizer is required for the extractors specified in the previous step.

GPU

```
normalizers:
  ...

  models:
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
```

CPU

```
normalizers:
  ...

  models:
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
```

5. Make sure that the `objects -> body` section contains the `quality_attribute:  body_quality`:

GPU

```
objects:
  ...
  body:
    base_normalizer: facenorm/cropbbox.v2.gpu.fnk
    quality_attribute: body_quality
  ...
```

CPU

```
objects:
  ...
  body:
    base_normalizer: facenorm/cropbbox.v2.cpu.fnk
    quality_attribute: body_quality
```

6. After completing the configuration file modification, make sure it looks similar to the `following example`.

7. Restart findface-extraction-api.

```
sudo systemctl restart findface-extraction-api
```

2. Modify the /etc/findface-video-worker-gpu.ini (/etc/findface-video-worker-cpu.ini) configuration file. In the body section, specify the neural network models by analogy with the example below. Restart findface-video-worker-gpu (findface-video-worker-cpu).

GPU

```
sudo vi /etc/findface-video-worker-gpu.ini

#------------------------------
[body]
#------------------------------
## detector param
## type:number env:CFG_BODY_MIN_SIZE longopt:--body-min-size
min_size = 60

## path to body detector
## type:string env:CFG_BODY_DETECTOR longopt:--body-detector
detector = /usr/share/findface-data/models/pedet/glenny_005_fast.gpu.fnk

## path to normalizer (usually crop2x)
## type:string env:CFG_BODY_NORM longopt:--body-norm
norm = /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk

## path to body quality extractor
## type:string env:CFG_BODY_QUALITY longopt:--body-quality
quality = /usr/share/findface-data/models/pedattr/pedattr.quality.v0.gpu.fnk

## path to body quality normalizer
## type:string env:CFG_BODY_NORM_QUALITY longopt:--body-norm-quality
norm_quality = /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
```

```
sudo systemctl restart findface-video-worker-gpu.service
```

CPU

```
sudo vi /etc/findface-video-worker-cpu.ini

#------------------------------
[body]
#------------------------------
## detector param
## type:number env:CFG_BODY_MIN_SIZE longopt:--body-min-size
min_size = 60

## path to body detector
## type:string env:CFG_BODY_DETECTOR longopt:--body-detector
detector = /usr/share/findface-data/models/pedet/glenny_005_fast.cpu.fnk
```

(continues on next page)

(continued from previous page)

```
## path to normalizer (usually crop2x)
## type:string env:CFG_BODY_NORM longopt:--body-norm
norm = /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk

## path to body quality extractor
## type:string env:CFG_BODY_QUALITY longopt:--body-quality
quality = /usr/share/findface-data/models/pedattr/pedattr.quality.v0.cpu.fnk

## path to body quality normalizer
## type:string env:CFG_BODY_NORM_QUALITY longopt:--body-norm-quality
norm_quality = /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
```

```
sudo systemctl restart findface-video-worker-cpu.service
```

3. Open the `/etc/findface-video-manager.conf` configuration file and make sure it contains the `body` section in `detectors` that looks similar to the example below.

   Tip: As a reference value for the `filter_min_quality` parameter, you can take the `MINIMUM_BODY_QUALITY` parameter value from the `/etc/findface-security/config.py` configuration file.

```
sudo vi /etc/findface-video-manager.conf

detectors:
    ...
    body:
      filter_min_quality: 0.6
      filter_min_size: 1
      filter_max_size: 8192
      roi: ''
      fullframe_crop_rot: false
      fullframe_use_png: false
      jpeg_quality: 95
      overall_only: true
      realtime_post_first_immediately: false
      realtime_post_interval: 1
      realtime_post_every_interval: false
      track_interpolate_bboxes: true
      track_miss_interval: 1
      track_overlap_threshold: 0.25
      track_max_duration_frames: 0
      track_send_history: false
      post_best_track_frame: true
      post_best_track_normalize: true
      post_first_track_frame: false
      post_last_track_frame: false
```

4. Enable the recognition of bodies and body attributes in the `/etc/findface-security/config.py` configuration file. Do the following:

   1. In the `FFSECURITY` section, set `'ENABLE_BODIES': True`.

```
sudo vi /etc/findface-security/config.py

FFSECURITY = {
    ...

    # optional objects to detect
    'ENABLE_BODIES': True,
    ...
```

2. In the same section, specify the body attributes (color) that you want to display for the body recognition events.

```
# available features are: color
'BODY_EVENTS_FEATURES': ['color'],
```

3. Restart the `findface-security` service.

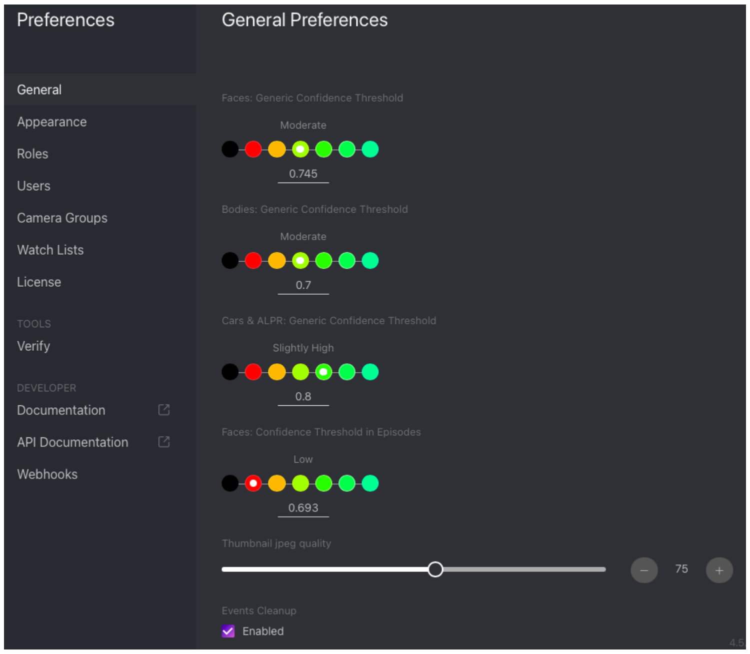
```
sudo systemctl restart findface-security.service
```

General Preferences

The FindFace Multi general preferences determine your system functioning and resource consumption. Here they are:

- generic confidence thresholds for face, body, car recognition (subject to the enabled objects)
- confidence threshold for face recognition episodes
- thumbnail JPEG quality
- schedule for automatic events/episodes cleanup

To configure the general preferences, navigate to the *Preferences* tab and click *General*. After you are finished with adjustments, click *Update*. Find the detailed explanation of each setting below.



Faces: Generic Confidence Threshold
Moderate
0.745

Bodies: Generic Confidence Threshold
Moderate
0.7

Cars & ALPR: Generic Confidence Threshold
Slightly High
0.8

Faces: Confidence Threshold in Episodes
Low
0.693

Thumbnail jpeg quality
75

Events Cleanup
✓ Enabled

In this section:

Generic Confidence Threshold

FindFace Multi verifies that a detected face and some face from the dossiers belong to the same person (i.e., the faces match), based on the pre-defined similarity threshold. The default threshold is set to the optimum value. If necessary, you can change it.

Note: The higher is the threshold, the less are chances that a wrong person will be positively verified, however, some valid photos may also fail verification.

The same principle applies to the body and car recognition, should you have enabled these objects. Based on the pre-defined threshold, FindFace verifies that a detected body and a body from the dossiers belong to the same person. Likewise, a detected car matches a car from the dossiers.

Tip: You can configure the confidence thresholds individually for each *camera group* and *watch list*.

Important: The default generic confidence threshold is optimal for the majority of recognition cases. We do not recommend changing it on your own. Be sure to consult with our technical experts by support@ntechlab.com.

Confidence Threshold for Episodes (Only Faces)

Note: In the current implementation, FindFace Multi supports only episodes of face recognition events. Car and body episodes are expected in future versions.

To construct an *episode*, the system searches the feature vector database for recent events with similar faces with a pre-defined similarity threshold. The default threshold is set to the optimum value. If necessary, you can change this value. Be sure to consult with our technical experts prior (support@ntechlab.com).

Thumbnail JPEG Quality

Subject to JPEG quality, thumbnails may take up a significant amount of disc volume. Use the *General* tab to configure the parameter.

Automatic Event And Episode Cleanup

Use the same tab to schedule automatically purging old events and related episodes from the database. For example, you can purge matched and unmatched events/episodes on different schedules and purge only full frames. You can also separately purge events with faces, cars, and bodies.

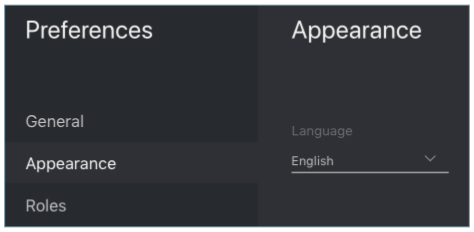
Web Interface Language

By default, only English and Russian languages are available in the web interface. Other languages are provided by request (support@ntechlab.com) and added to the system via the `/etc/findface-security/config.py` configuration file.

In this section:

Switch Language in Web Interface

You can change the system language on the *Preferences* -> *Appearance* tab.



Language
English

Add Custom Language

To add a custom language to the system, do the following:

1. Download the localization file provided by our experts into the `/usr/share/ffsecurity-ui/ui-static/` directory on the FindFace Multi principal server.
2. Open the `/etc/findface-security/config.py` configuration file.

   ```
   sudo vi /etc/findface-security/config.py
   ```

3. Into the `FFSECURITY_UI_CONFIG` section, insert the `languages` section and fill it by analogy with the example below.

```
FFSECURITY_UI_CONFIG = {
...
  "languages": {
    "items": [
      {
        "name": "es",
        "label": "Español",
        "url": "/ui-static/es_i18n_ffsec.po"
      },
    ]
  },
```

4. Restart the `findface-security` service.

```
sudo systemctl restart findface-security.service
```

The new language will be automatically applied to the system and from now on be available on the *Preferences* -> *Appearance* tab. You can at any time switch between it and other available languages.

2.2.2 User Management and System Security

Important: Although FindFace Multi provides tools to ensure its protection from unauthorized access, they are not replacing a properly configured firewall. Be sure to use a firewall to heighten the FindFace Multi network protection.

User Management

In this chapter:

Predefined Roles

FindFace Multi provides the following predefined roles:

- Administrator has rights to *manage cameras*, events, FindFace Multi users, the *dossier database*, and full access to all other functions.

  Important: Whatever the role, the first administrator (Super Administrator) cannot be deprived of its rights.

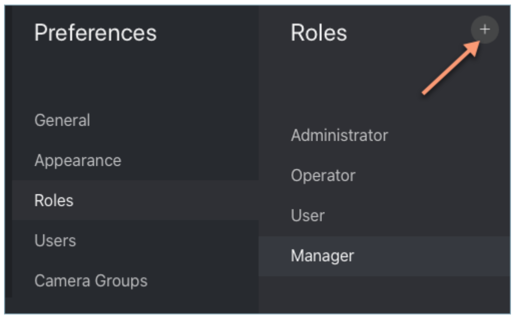
- Operator can create dossiers manually, receive and acknowledge events, and search for objects on the event list. The other data is available read-only. The *batch dossier creation* is unavailable.
- User has a right to receive and acknowledge events, and to search for objects on the event list. The other data is available read-only.

You can change the predefined roles privileges, as well as create various custom roles.

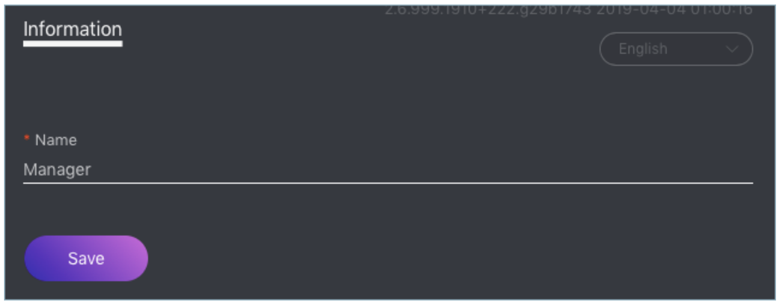
Create Custom Role

To create a custom role, do the following:

1. Navigate to the *Preferences* tab. Click *Roles*.
2. Click +.



3. On the *Information* tab, specify the role name.



* Name
Manager

4. Click *Save*. You will see additional tabs appear next to the *Information* tab. You can use these tabs to assign the role privileges for specific watch lists (the *Watch Lists* tab) and camera groups (*Camera Groups*), as well as for entire system functions and entities (*Permissions*).

   Note: For example, if you set `None` for a certain camera group on the *Camera Groups* tab, users with this role won't be able to work with **this** very group of cameras. Setting `None` for `cameragroup` on the *Permissions* tab will prevent users from viewing and working with **all** camera groups.

   Note: The right for an event consists of the rights for a corresponding camera and watch list. To see unmatched events, you only need the rights for a camera.

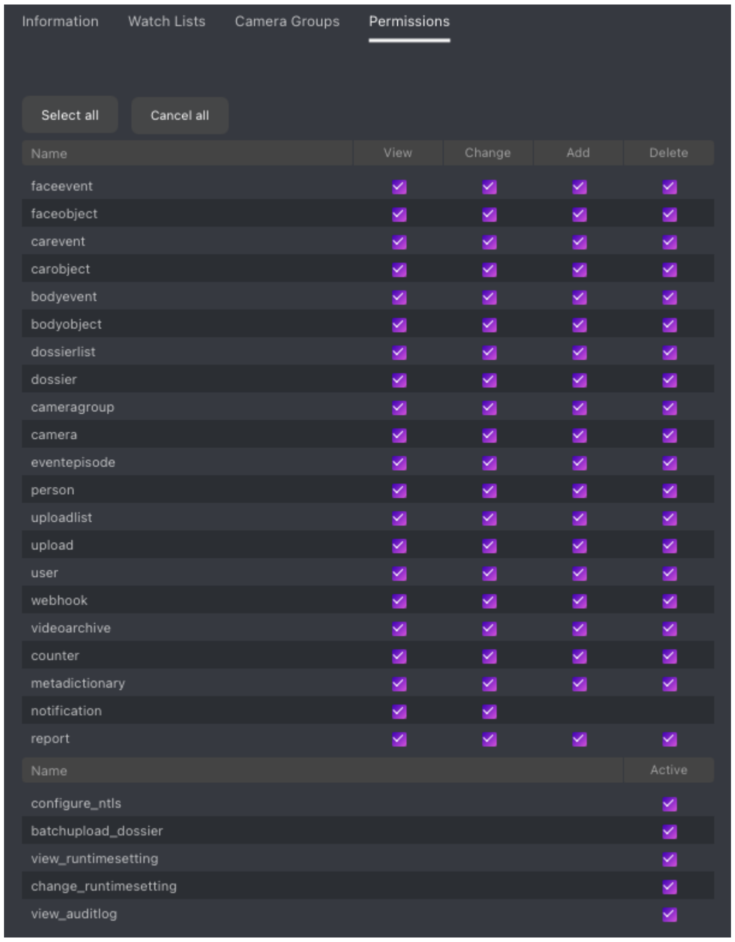
   The full list of the FindFace Multi entities is as follows:

   - `faceevent`: face recognition *event*

- `faceobject`: face photo in a *dossier*
- `carevent`: car recognition event
- `carobject`: car photo in a dossier
- `bodyevent`: body recognition event
- `bodyobject`: full-length photo in a dossier
- `dossierlist`: *watch list*
- `dossier`: *dossier*
- `cameragroup`: *camera group*
- `camera`: *camera*
- `eventepisode`: *episodes*
- `person`: *person gallery*
- `uploadlist`: list of photos in *batch upload*
- `upload`: item (photo) in batch photo upload
- `user`: *user*
- `webhook`: *webhook*
- `videoarchive`: *object identification in offline video*
- `counter`: *counters picking statistics on faces and bodies*
- `report`: *report*

You can also enable and disable rights for the following functionality:

- `configure_ntls`: configuration of the `findface-ntls` *license server*
- `batchupload_dossier`: *batch photo upload*
- `view_runtimesetting`: viewing the FindFace Multi *general preferences*
- `change_runtimesetting`: changing the FindFace Multi general preferences
- `view_auditlog`: viewing and working with the *audit logs*.



Name	View	Change	Add	Delete
faceevent	✓	✓	✓	✓
faceobject	✓	✓	✓	✓
carevent	✓	✓	✓	✓
carobject	✓	✓	✓	✓
bodyevent	✓	✓	✓	✓
bodyobject	✓	✓	✓	✓
dossierlist	✓	✓	✓	✓
dossier	✓	✓	✓	✓
cameragroup	✓	✓	✓	✓
camera	✓	✓	✓	✓
eventepisode	✓	✓	✓	✓
person	✓	✓	✓	✓
uploadlist	✓	✓	✓	✓
upload	✓	✓	✓	✓
user	✓	✓	✓	✓
webhook	✓	✓	✓	✓
videoarchive	✓	✓	✓	✓
counter	✓	✓	✓	✓
metadictionary	✓	✓	✓	✓
notification	✓	✓		
report	✓	✓	✓	✓

Name	Active
configure_ntls	✓
batchupload_dossier	✓
view_runtimesetting	✓
change_runtimesetting	✓
view_auditlog	✓

Primary and Additional User Privileges

You assign privileges to a user by using roles:

- *Primary role*: main user role, mandatory for assignment. You can assign only one primary role to a user.
- *Role*: additional user role, optional for assignment. You can assign several roles to one user. The rights associated with the additional roles will be added to the primary privileges.

All users belonging to a particular primary role automatically get access to camera groups (and cameras within the group) and watch lists (and dossiers assigned to the watchlist) created by a user with the same primary role, subject to the privileges defined by their additional role(s).

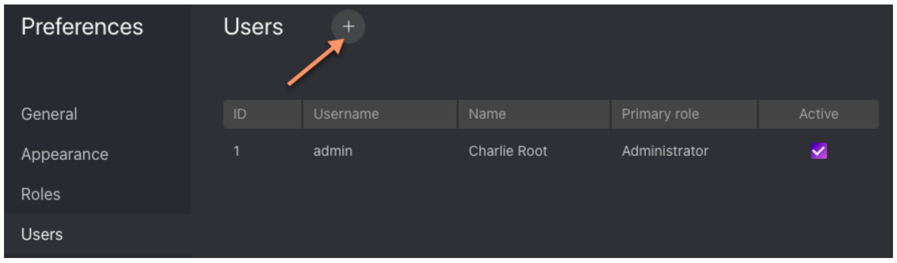
See also:

Create User

Create User

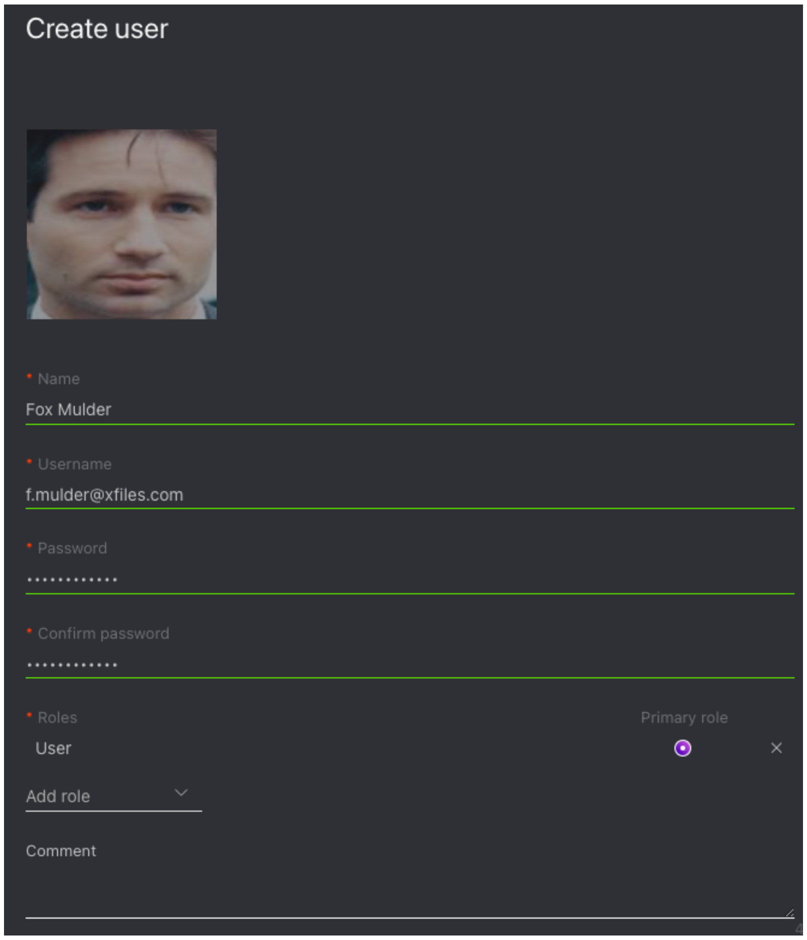
To create a user, do the following:

1. Navigate to the *Preferences* tab. Click *Users*.
2. Click +.



ID	Username	Name	Primary role	Active
1	admin	Charlie Root	Administrator	✓

3. Specify such user data as name, login and password. If necessary, add a comment. Attach the user's photo.



* Name	Fox Mulder
* Username	f.mulder@xfiles.com
* Password	••••••••••••
* Confirm password	••••••••••••
* Roles	User
Primary role	●

Important: A face in the photo must be of high quality, i.e. close to a frontal position. Distance between pupils: 60 px. Supported formats: WEBP, JPG, BMP, PNG. Photos that do not meet the requirements will be rejected with a detailed error description.

Tip: The photo can be used for *biometric authentication*.

4. From the *Roles* drop-down menu, select one or several user roles. Set one of them as the *Primary role*.
5. Check *Active*.
6. Click *Create*.

Deactivate or Delete User

In order to deactivate a user, uncheck *Active* on the user list (*Preferences* -> *Users*).

To delete a user from FindFace Multi, click on the user login on the list. Click *Delete*.

Enable Administrator Privileges for System Plugins

The FindFace Multi package incorporates an extensive set of system plugins that provide the following functionality:

- *partner integrations*,
- management of *distributed dossier database*,
- log-in through a crypto certificate (contact your manager for details).

Note: You have to manually enable the system plugins via the `/etc/findface-security/config.py` configuration file.

By default, the Administrator role is granted no privileges for any of the plugins. To assign relevant privileges to Administrator, do the following:

1. Enable a system plugin in the `/etc/findface-security/config.py` configuration file, following the step-by-step instructions provided by our team.
2. Re-migrate the main database architecture from FindFace Multi to **`PostgreSQL`**.

```
sudo findface-security migrate
```

3. Re-create user groups in the main database.

```
sudo findface-security create_groups
```

4. Restart the `findface-security` service.

```
sudo systemctl restart findface-security.service
```

Authentication and Session Monitoring

In this section:

Authentication Types

FindFace Multi provides the following authentication types:

- `password`: standard login/password authentication. Enabled by default.
- `face`: authentication is possible only by the user's face.
- `face_or_password`: authentication is possible using either a face or login/password.
- `face_and_password`: two-factor authentication. After a face is successfully recognized, the user must enter their credentials.

Important: For all the authentication types based on face recognition, you need the following configuration:

- *standalone liveness service* (`findface-liveness-api`)
- *HTTPS*

Important: Before using face recognition for authentication, you need to *attach photos* to users' profiles and equip their workplaces with webcams.

Note: You can enable a work session monitoring for the authentication types `face` and `face_or_password`. In this case, the system will be periodically renewing the session after verifying that the face of a person at the workplace matches the user's face that has logged in (see *Configure Authentication and Session Renewal* for details).

Tip: FindFace Multi also provides a certificate-based authentication that is configured independently. Contact our support team for details (support@ntechlab.com).

Configure Authentication and Session Renewal

To configure authentication and session monitoring, do the following:

1. Open the `/etc/findface-security/config.py` configuration file. Find the `FFSECURITY` and `FFSECURITY_AUTH_CONFIG` sections.

```
sudo vi /etc/findface-security/config.py

FFSECURITY = {
    # auth config
    # available options: face, password, face_and_password, face_or_password
    'AUTH_TYPE': 'face_or_password',
    # 180 days by default
    'MAXIMUM_SESSION_LENGTH': 15552000,
    ...
    }

...
# - FindFace Security authorization configuration dictionary -

FFSECURITY_AUTH_CONFIG = {
    'FACE_AUTH_CONFIDENCE': 0.740,  # FAR = 2.5E-09  # model: [kiwi_320]
    # 3 settings below are for front-end only
    # session renew works only with face or face_or_password authorization type
    'NEED_SESSION_RENEW': False,
    'RENEW_SESSION_INTERVAL': 0,
    'MAXIMUM_RENEW_ATTEMPTS': 2,
}
```

2. In the `FFSECURITY` section, set the following authentication parameters:
   - `AUTH_TYPE`: authentication type. Available options: `face`, `password`, `face_and_password`, `face_or_password`.
   - `MAXIMUM_SESSION_LENGTH`: the maximum session length, in seconds. After a session expires, the user will be automatically logged out unless the session is renewed.
3. In the `FFSECURITY_AUTH_CONFIG` section, set the following authentication and session monitoring parameters:
   - `FACE_AUTH_CONFIDENCE`: after a face in the webcam video is detected as alive, the system checks this face against the database of user photos with this confidence threshold.
   - `NEED_SESSION_RENEW`: if `True`, a session can be renewed and prolonged by the time equal to `MAXIMUM_SESSION_LENGTH`, after verifying that the face of a person at the workplace matches the user's face that has logged in.
   - `RENEW_SESSION_INTERVAL`: period in seconds before the expected time of the session expiry, during which the system will attempt to renew the session by enabling the webcam and verifying the user's face.
   - `MAXIMUM_RENEW_ATTEMPTS`: the number of user verification attempts. The attempts occur in a row during the renewal interval.

Note: A verification attempt takes about 3 seconds to complete.

Tip: We recommend you to set up the monitoring parameters so that `MAXIMUM_RENEW_ATTEMPTS` multiplied by the attempt duration is less than `RENEW_SESSION_INTERVAL`. Otherwise, the system will extend the renewal interval x2, x3, and so on, subject to the number of attempts.

4. Restart `findface-security`.

```
sudo systemctl restart findface-security.service
```

Log out All Users

To automatically log out all users, execute the following command on the FindFace Multi principal server console:

```
sudo findface-security logout_all_users
```

Tip: This command comes in handy when switching to a different authentication type.

Enable Data Encryption

To ensure data security, we recommend you enabling SSL encryption. Do the following:

1. Under the nginx configuration directory, create a directory that will be used to hold all of the SSL data:

   ```
   sudo mkdir /etc/nginx/ssl
   ```

2. Create the SSL key and certificate files:

   ```
   sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/my-
   ↪example-domain.com.key -out /etc/nginx/ssl/my-example-domain.com.crt
   ```

   You will be asked a few questions about your server in order to embed the information correctly in the certificate. Fill out the prompts appropriately. The most important line is the one that requests the `Common Name`. You need to enter the domain name or public IP address that you want to be associated with your server. Both of the files you created (`my-example-domain.com.key` and `my-example-domain.com.crt`) will be placed in the `/etc/nginx/ssl` directory.

3. Configure nginx to use SSL. Open the nginx configuration file `/etc/nginx/sites-available/ffsecurity-nginx.conf`. Apply the following modifications to the file:

   1. Add the new `server {...}` section that contains the URL replacement rule:

      ```
      server {
              listen 80;
              server_name my-example-domain.com www.my-example-domain.com;
              rewrite ^(.*) https://my-example-domain.com$1 permanent;
              access_log off;
      }
      ```

   2. Comment out the following lines in the existing `server {...}` section:

      ```
      # listen 80 default_server;
      # listen [::]:80 default_server;
      ```

3. Add the following lines, including the paths to the certificate and the key, to the existing `server {...}` section:

```
listen 443 ssl;

ssl_certificate     /etc/nginx/ssl/my-example-domain.com.crt;
ssl_certificate_key /etc/nginx/ssl/my-example-domain.com.key;
```

4. In the generic nginx configuration file `/etc/nginx/nginx.conf`, find the `SSL Settings` section and append the following lines:

```
ssl_session_cache   shared:SSL:10m;
ssl_session_timeout 1h;
```

The example of the configuration file `/etc/nginx/sites-available/ffsecurity-nginx.conf` with correctly configured SSL settings is shown below:

```
upstream ffsecurity {
 server 127.0.0.1:8002;
}

upstream ffsecurity-ws {
  server 127.0.0.1:8003;
}

map $http_upgrade $ffsec_upstream {
  default "http://ffsecurity-ws";
  "" "http://ffsecurity";
}

server {
  listen 80;
  server_name my-example-domain.com www.my-example-domain.com;
  rewrite ^(.*) https://my-example-domain.com$1 permanent;
  access_log off;
}

server {
  # listen 80 default_server;
  # listen [::]:80 default_server;
  listen 443 ssl;
  ssl_certificate /etc/nginx/ssl/my-example-domain.com.crt;
  ssl_certificate_key /etc/nginx/ssl/my-example-domain.com.key;

  root /var/lib/findface-security;

  autoindex off;

  server_name _;

  location = / {

    alias /usr/share/findface-security-ui/;
    try_files /index.html =404;
```

(continues on next page)

(continued from previous page)

```
    }
  location /static/ {

  }
  location /uploads/ {
    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Allow-Methods' 'GET';
    add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-
↪Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range,
↪Authorization';
    add_header 'Access-Control-Expose-Headers' 'Content-Length,
↪Content-Range';
    add_header 'Access-Control-Max-Age' 2592000;
  }
  location /ui-static/ {
    alias /usr/share/findface-security-ui/ui-static/;
  }
  location /doc/ {
    alias /opt/findface-security/doc/;
  }
  location ~ /videos/(?<video_id>[0-9]+)/upload/(.*)$ {
    if ($request_method = 'OPTIONS') {
      add_header 'Content-Type' 'text/plain; charset=utf-8';
      add_header 'Content-Length' 0;
      return 204;
    }
    set $auth_request_uri "http://ffsecurity/videos/$video_id/auth-
↪upload/";
    auth_request /video-upload-auth/;

    alias "/var/lib/findface-security/uploads/videos/$video_id.bin";
    client_max_body_size 15g;

    dav_access user:rw group:rw all:rw;
    dav_methods PUT;

    create_full_put_path on;
    autoindex off;
    autoindex_exact_size off;
    autoindex_localtime on;
    charset utf-8;

    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Allow-Methods' 'PUT, OPTIONS';
    add_header 'Access-Control-Allow-Headers' 'authorization';
  }
  location = /video-upload-auth/ {
    internal;
    client_max_body_size 15g;
    proxy_set_header Content-Length "";
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $remote_addr;
```

(continues on next page)

(continued from previous page)

```
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_pass_request_body off;
    proxy_pass $auth_request_uri;
  }

  location / {
    client_max_body_size 300m;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_pass $ffsec_upstream;
    proxy_read_timeout 5m;

    location ~ ^/(cameras|videos)/([0-9]+)/stream/?$ {
      proxy_set_header Host $http_host;
      proxy_set_header X-Forwarded-For $remote_addr;
      proxy_set_header X-Forwarded-Proto $scheme;
      proxy_pass http://ffsecurity;
    }

    location ~ ^/streams/(.*)$ {
      internal;
      proxy_pass $1;
    }
  }
}
```

4. Restart nginx.

```
sudo systemctl restart nginx.service
```

5. Edit the `/etc/findface-security/config.py` configuration file. In the `EXTERNAL_ADDRESS` and `ROUTER_URL` parameters, substitute the `http://` prefix with `https://`.

```
sudo vi /etc/findface-security/config.py

...
EXTERNAL_ADDRESS="https://my-example-domain.com"
...
ROUTER_URL="https://IP_address"
```

6. If there are running `findface-video-worker` services in the system, you need to either recreate cameras in the web interface, or change the `router_url` parameter in relevant video processing jobs, substituting the `http://` prefix with `https://`. This can be done with the following command:

```
curl -s localhost:18810/jobs | jq -r '.[]["id"]' | xargs -I {} curl -X PATCH -d '{
↪"router_url": "https://my-example-domain.com/video-detector/frame"}' http://
↪localhost:18810/job/{}
```

Enable Dossier Security

If the dossier security is disabled, the dossier photos and attachments will be available by direct link regardless of the user rights. To increase dossier security, configure FindFace Multi to run all media requests through the DJANGO application for ACL checks.

Important: Enable the dossier media security only if you need it, as this setting has a severe negative impact on the system performance.

To enable dossier security, do the following:

1. Open the `/etc/findface-security/config.py` configuration file.

   ```
   sudo vi /etc/findface-security/config.py
   ```

2. Uncomment `OVERPROTECT_MEDIA` and set it `True`.

   ```
   ...

   'OVERPROTECT_MEDIA': False,
   ```

3. Open the nginx configuration file `/etc/nginx/sites-available/ffsecurity-nginx.conf`. Uncomment `internal` in the `location /uploads` section.

   ```
   location /uploads/ {
      internal; # Uncomment if you intend to enable OVERPROTECT_MEDIA
      ...
   }
   ```

4. Restart `findface-security` and nginx.

   ```
   sudo systemctl restart findface-security.service
   sudo systemctl restart nginx.service
   ```

5. After the new security policy is applied, logged-in users must re-authenticate. To make the users do so, execute the logout-all command:

   ```
   sudo findface-security logout_all_users
   ```

Disable ACL

You can turn off FindFace Multi ACL if you do not need it, as the constant permission checks consume a significant amount of system resources.

Do the following:

1. Open the `/etc/findface-security/config.py` configuration file.

   ```
   sudo vi /etc/findface-security/config.py
   ```

2. Set `ENABLE_ACL = False`.

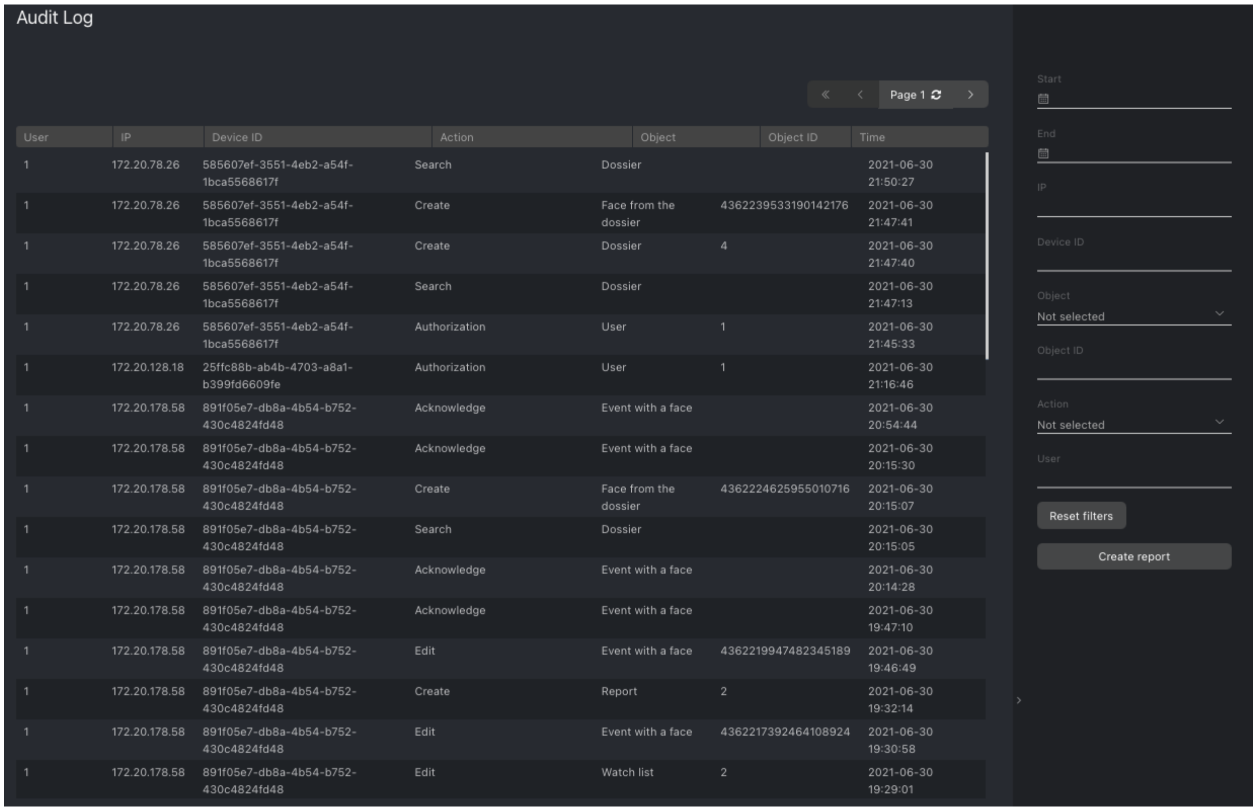
```
...

ENABLE_ACL = False
```

3. Restart `findface-security`.

```
sudo systemctl restart findface-security.service
```

Audit Logs

The FindFace Multi comprehensive and searchable audit logs are an excellent complementary tool for user management that provides you with a thorough audit of the user actions and strengthens your system protection. You can access this functionality on the *Audit Logs* tab.



User	IP	Device ID	Action	Object	Object ID	Time
1	172.20.78.26	585607ef-3551-4eb2-a54f-1bca5568617f	Search	Dossier		2021-06-30 21:50:27
1	172.20.78.26	585607ef-3551-4eb2-a54f-1bca5568617f	Create	Face from the dossier	4362239533190142176	2021-06-30 21:47:41
1	172.20.78.26	585607ef-3551-4eb2-a54f-1bca5568617f	Create	Dossier	4	2021-06-30 21:47:40
1	172.20.78.26	585607ef-3551-4eb2-a54f-1bca5568617f	Search	Dossier		2021-06-30 21:47:13
1	172.20.78.26	585607ef-3551-4eb2-a54f-1bca5568617f	Authorization	User	1	2021-06-30 21:45:33
1	172.20.128.18	25ffc88b-ab4b-4703-a8a1-b399fd6609fe	Authorization	User	1	2021-06-30 21:16:46
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Acknowledge	Event with a face		2021-06-30 20:54:44
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Acknowledge	Event with a face		2021-06-30 20:15:30
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Create	Face from the dossier	4362224625955010716	2021-06-30 20:15:07
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Search	Dossier		2021-06-30 20:15:05
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Acknowledge	Event with a face		2021-06-30 20:14:28
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Acknowledge	Event with a face		2021-06-30 19:47:10
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Edit	Event with a face	4362219947482345189	2021-06-30 19:46:49
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Create	Report	2	2021-06-30 19:32:14
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Edit	Event with a face	4362217392464108924	2021-06-30 19:30:58
1	172.20.178.58	891f05e7-db8a-4b54-b752-430c4824fd48	Edit	Watch list	2	2021-06-30 19:29:01

Each record provides the following data:

- id of the user who performed the action
- IP address where the request came from
- device id: the unique identifier of the client device
- action type such as authorization, search, object modification, restart, and so on
- object type to which the action applies, for example, a dossier or a camera
- object identifier
- details, subject to the action type
- timestamp

Use the filter panel to the right to set up the search conditions.

2.2.3 Camera Management

To configure video-based object monitoring, add cameras to FindFace Multi, grouping them subject to their location.

Note: Privileges to create camera groups and cameras are managed in user's permissions (see *User Management*).

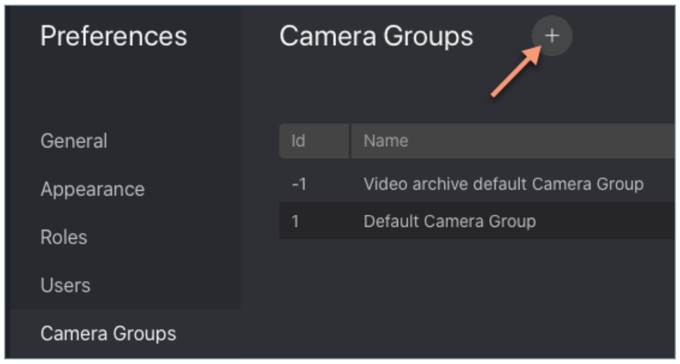
In this chapter:

Create Camera Group

Tip: A default preconfigured camera group is available in the system.

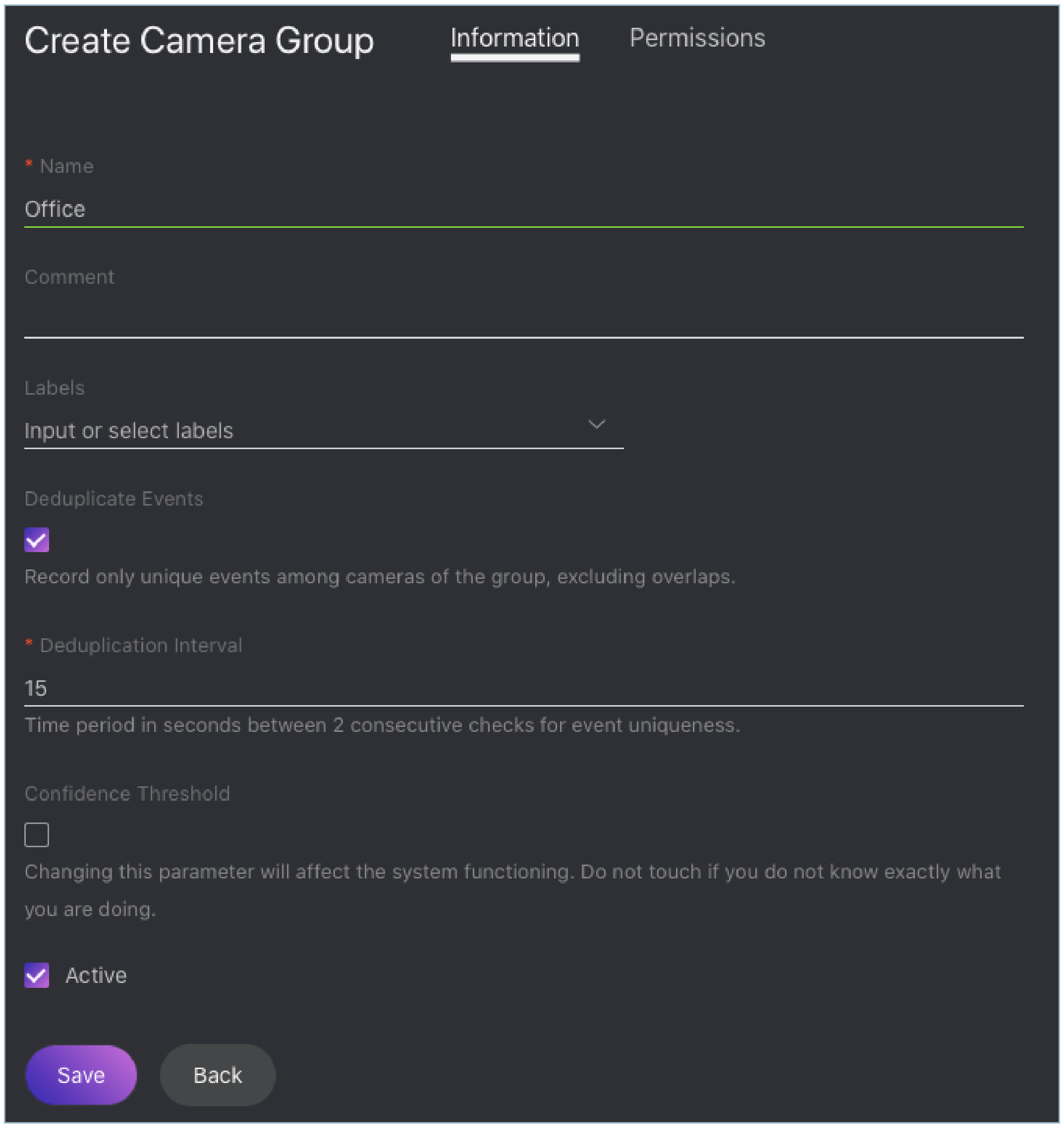
To create a group of cameras, do the following:

1. Navigate to the *Preferences* tab. Click *Camera Groups*.
2. Click +.



Id	Name
-1	Video archive default Camera Group
1	Default Camera Group

3. On the *Information* tab, specify the group name. Add a comment if needed.



Create Camera Group

Information | Permissions

* Name: Office

Comment:

Labels: Input or select labels

Deduplicate Events: ✓

Record only unique events among cameras of the group, excluding overlaps.

* Deduplication Interval: 15

Time period in seconds between 2 consecutive checks for event uniqueness.

Confidence Threshold: ☐

Changing this parameter will affect the system functioning. Do not touch if you do not know exactly what you are doing.

✓ Active

4. If you want to allocate a certain `findface-video-worker` instance to process video streams from the group, create or select one or several allocation labels.

   Note: To complete the allocation, list the labels in the `/etc/findface-video-worker-gpu.ini` (`/etc/findface-video-worker-cpu.ini`) configuration file. See *Allocate findface-video-worker to Camera Group* for details.

5. If you want to deduplicate events from cameras that belong to the same group, i. e. exclude coinciding events, check *Deduplicate Events* and specify the deduplication interval (interval between 2 consecutive checks for event uniqueness).

   Warning: Use deduplication with extreme caution. If cameras within a group observe different scenes, some objects may be skipped. See *Deduplicate Events* for details.

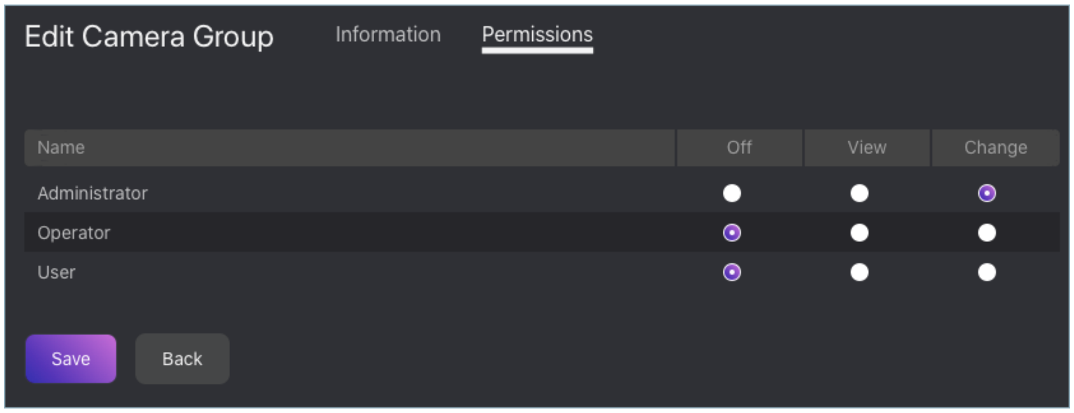
6. By default, video from all camera groups is processed using the *generic confidence threshold*. To set an individual

threshold for the camera group, check *Confidence Threshold* and specify the threshold value.

Important: The default generic confidence threshold is optimal for the majority of recognition cases. We do not recommend changing it on your own. Be sure to consult with our technical experts by support@ntechlab.com.

---

7. Check *Active*.
8. Click *Save*.
9. On the *Permissions* tab, assign privileges on the camera group, specifying which user roles are allowed to change/view the camera group settings.



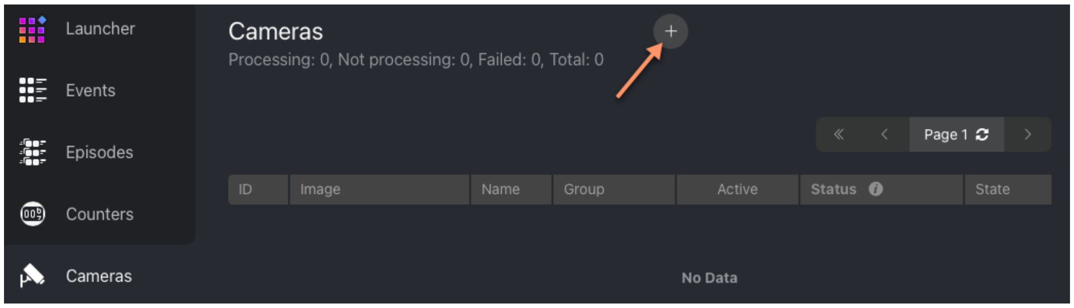
Name	Off	View	Change
Administrator	○	○	●
Operator	●	○	○
User	●	○	○

10. Click *Save*.

Add Camera

To add a camera, do the following:

1. Navigate to the *Cameras* tab.
2. Click +.



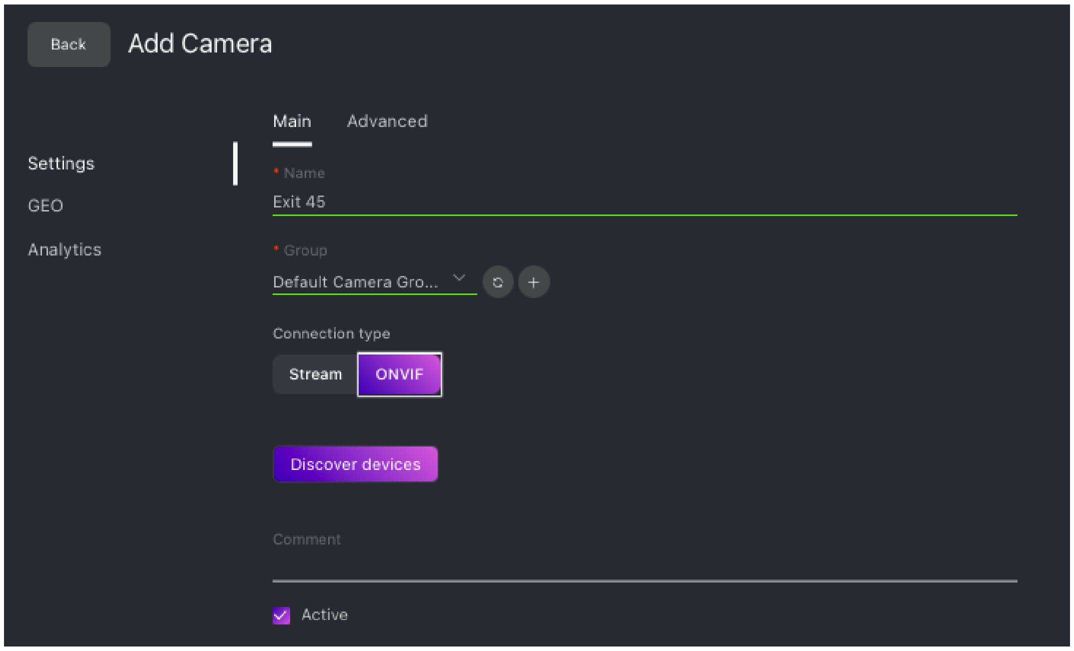
Cameras

Processing: 0, Not processing: 0, Failed: 0, Total: 0

ID	Image	Name	Group	Active	Status	State

No Data

3. On the *Settings* -> *Main* tab, enter the general camera information:



* Name
Exit 45

* Group
Default Camera Gro...

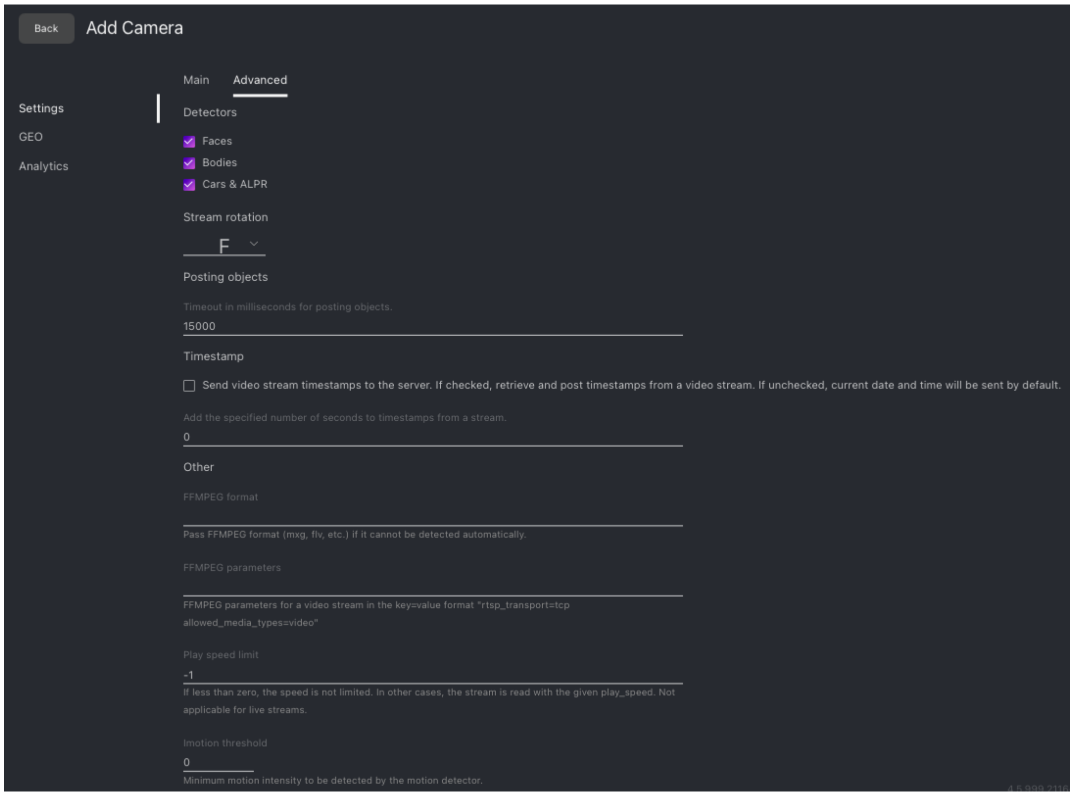
Connection type
Stream [✓ ONVIF]

Comment

✓ Active

- Specify the camera name.
- Add the camera to a camera group.
- Specify the camera URL (*Stream*). If the camera is *ONVIF*, select it from the list of detected devices to automatically load available settings and streams.
- If necessary, add a comment.
- Check *Active*.

4. On the *Settings* -> *Advanced* tab, fine-tune the camera:



Detectors

- [✓] Faces
- [✓] Bodies
- [✓] Cars & ALPR

Stream rotation

F

Posting objects

Timeout in milliseconds for posting objects.
15000

Timestamp

- [ ] Send video stream timestamps to the server. If checked, retrieve and post timestamps from a video stream. If unchecked, current date and time will be sent by default.

Add the specified number of seconds to timestamps from a stream.
0

Other

FFMPEG format

Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.

FFMPEG parameters

FFMPEG parameters for a video stream in the key=value format "rtsp_transport=tcp allowed_media_types=video"

Play speed limit
-1
If less than zero, the speed is not limited. In other cases, the stream is read with the given play_speed. Not applicable for live streams.

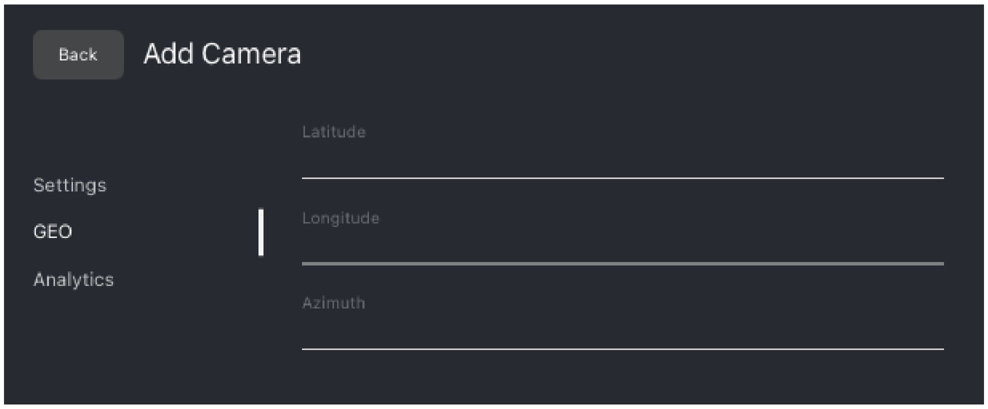
Imotion threshold
0
Minimum motion intensity to be detected by the motion detector.

- Check detectors that you want to enable on this camera: faces, bodies, cars & ALPR.
- If needed, change the video orientation.

Important: Be aware that the `findface-security` server rotates the video using post-processing tools. It can negatively affect performance. Rotate the video via the camera functionality wherever possible.

- *Timeout in ms*: Specify the timeout in milliseconds for posting detected objects.
- *Retrieve timestamps from stream*: Check to retrieve and post timestamps from the video stream. Uncheck the option to post the current date and time.
- *Add to timestamps*: Add the specified number of seconds to timestamps from the stream.
- *FFMPEG format*: Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.
- *FFMPEG parameters*: FFMPEG options for the video stream in the key-value format, for example, [“rtsp_transpotr=tcp”, “ss=00:20:00”].
- *Imotion threshold*: Minimum motion intensity to be detected by the motion detector.
- *Verify SSL*: Check to enable verification of the server SSL certificate when the object tracker posts objects to the server over https. Uncheck the option if you use a self-signed certificate.

5. (Optional) On the *GEO* tab, specify the camera geographical location.



Back Add Camera

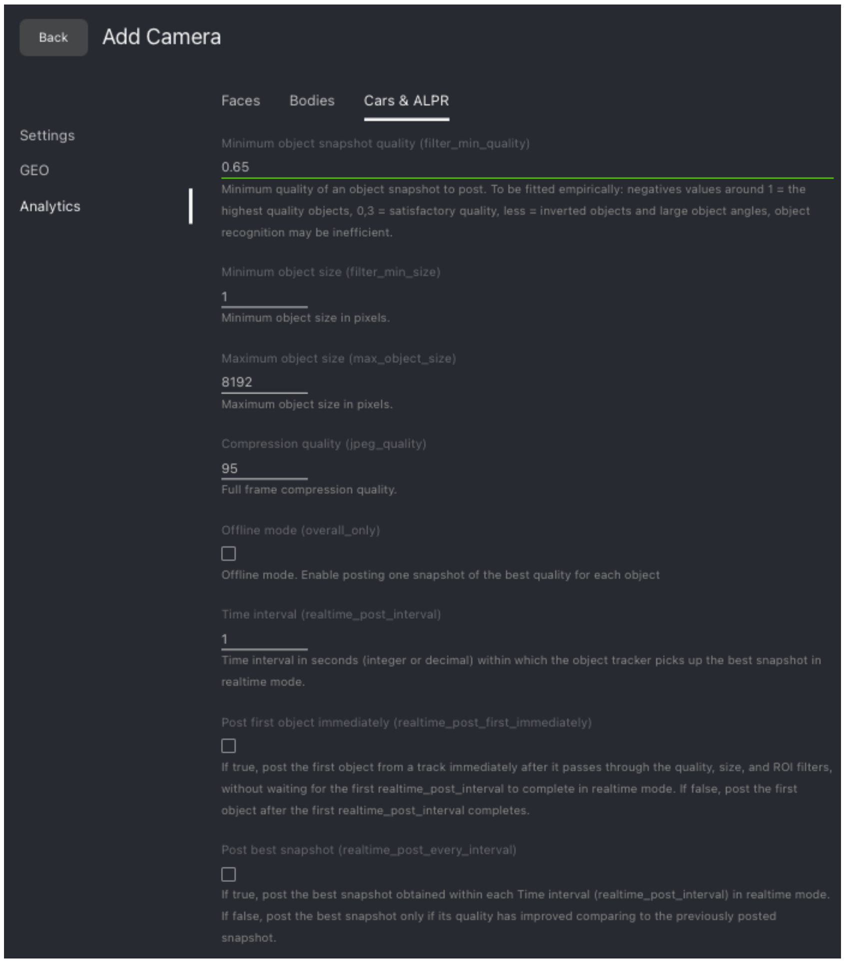
Settings
GEO
Analytics

Latitude

Longitude

Azimuth

6. On the *Analytics* tab, specify settings for each object type detector.



Back Add Camera

Settings
GEO
Analytics

Faces Bodies Cars & ALPR

Minimum object snapshot quality (filter_min_quality)
0.65
Minimum quality of an object snapshot to post. To be fitted empirically: negatives values around 1 = the highest quality objects, 0,3 = satisfactory quality, less = inverted objects and large object angles, object recognition may be inefficient.

Minimum object size (filter_min_size)
1
Minimum object size in pixels.

Maximum object size (max_object_size)
8192
Maximum object size in pixels.

Compression quality (jpeg_quality)
95
Full frame compression quality.

Offline mode (overall_only)
☐
Offline mode. Enable posting one snapshot of the best quality for each object

Time interval (realtime_post_interval)
1
Time interval in seconds (integer or decimal) within which the object tracker picks up the best snapshot in realtime mode.

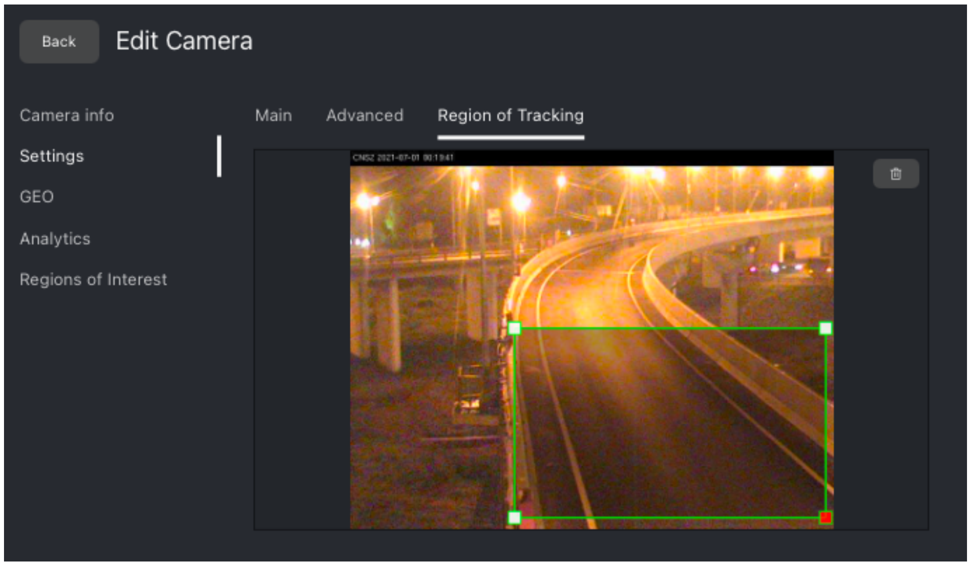
Post first object immediately (realtime_post_first_immediately)
☐
If true, post the first object from a track immediately after it passes through the quality, size, and ROI filters, without waiting for the first realtime_post_interval to complete in realtime mode. If false, post the first object after the first realtime_post_interval completes.

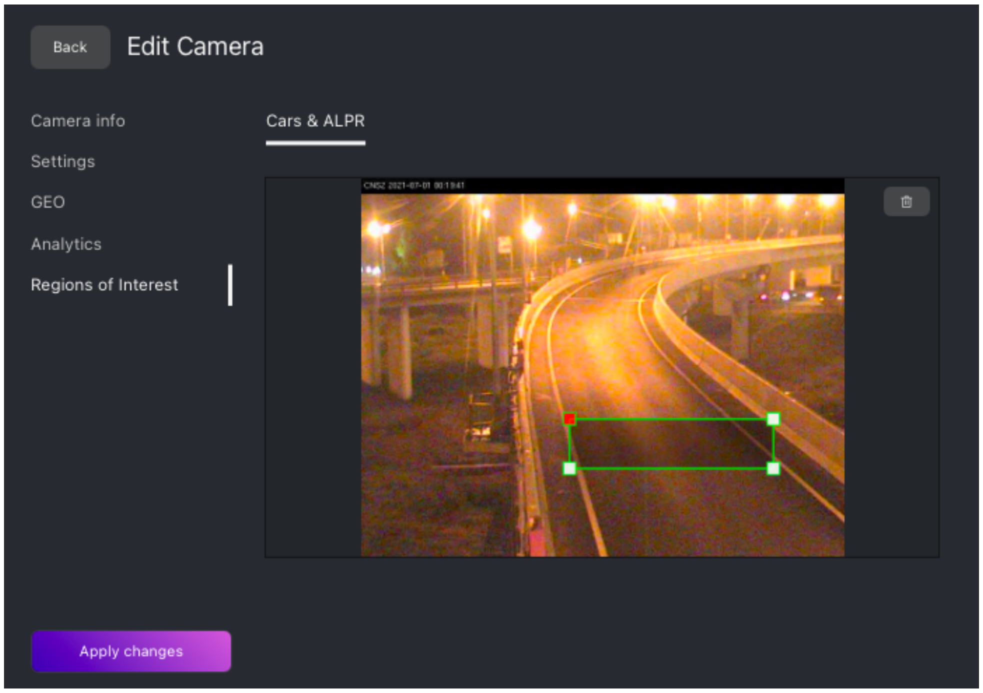
Post best snapshot (realtime_post_every_interval)
☐
If true, post the best snapshot obtained within each Time interval (realtime_post_interval) in realtime mode. If false, post the best snapshot only if its quality has improved comparing to the previously posted snapshot.

- *Minimum object snapshot quality*: Minimum quality of an object snapshot to post. Do not change the default value without consulting with our technical experts (support@ntechlab.com).
- *Minimum object size*: Minimum object size in pixels to post.
- *Maximum object size*: Maximum object size in pixels to post.
- *Compression quality*: Full frame compression quality.
- *Offline mode*: Offline mode. Enable posting one snapshot of the best quality per entire track for each object.

   - *Time interval*: Time interval in seconds (integer or decimal) within which the object tracker picks up the best snapshot in the real-time mode.
   - *Post first object immediately*: Check to post the first object snapshot from a track immediately after it passes through the quality, size, and ROI filters, without waiting for the first `Time interval` to complete. The way the subsequent snapshots are posted will depend on the `Post best snapshot` value. Uncheck the option to post the first object snapshot only after the first `Time interval` completes.
   - *Post best snapshot*: Check to post the best snapshot obtained within each `Time interval` in the real-time mode, regardless of its quality. Uncheck the option to post the best snapshot only if its quality has improved compared to the previously posted snapshot.

7. Click *Apply changes*.
8. Specify the region of tracking within the camera field (*General* -> *Region of Tracking*) and detection zones (*Regions of Interest*) for each object type detector if necessary. Click *Apply changes*.

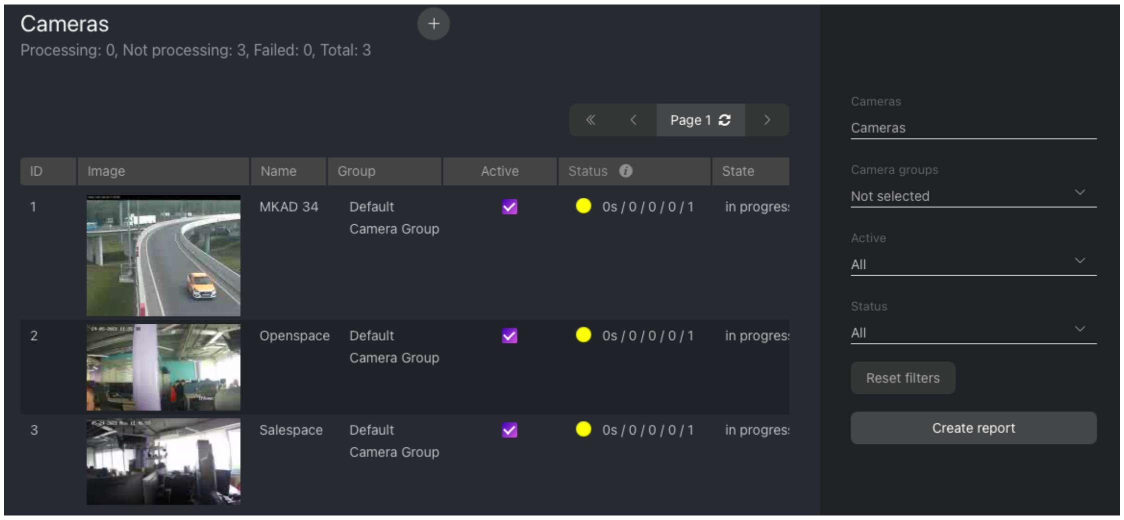




Note: Each created camera is associated with a so-called job, a video processing task that contains configuration settings and stream data and is assigned to `findface-video-worker`. This task can be restarted (see *Monitor Camera Operation*).

Monitor Camera Operation

To monitor the operation of cameras, navigate to the *Cameras* tab.



Cameras

Processing: 0, Not processing: 3, Failed: 0, Total: 3

ID	Image	Name	Group	Active	Status	State
1		MKAD 34	Default Camera Group	✓	● 0s / 0 / 0 / 0 / 1	in progres
2		Openspace	Default Camera Group	✓	● 0s / 0 / 0 / 0 / 1	in progres
3		Salespace	Default Camera Group	✓	● 0s / 0 / 0 / 0 / 1	in progres

Cameras: Cameras

Camera groups: Not selected

Active: All

Status: All

Camera statuses:

- Green: the video stream is being processed without errors.
- Yellow: the video stream is being processed for less than 30 seconds, or one or more errors occurred when posting an object.
- Red: the video stream cannot be processed.
- Grey: camera disabled.

Tip: You can configure the yellow and red statuses based on the portion of dropped frames and failed object postings. To do so, modify the following parameters in the `/etc/findface-security/config.py` configuration file:

```
sudo vi /etc/findface-security/config.py

FFSECURITY = {
    ...
    # max camera frames_dropped percent
    'MAX_CAMERA_DROPPED_FRAMES': {'yellow': 0.1, 'red': 0.3},
    # max camera objects_failed percent
    'MAX_CAMERA_FAILED_FACES': {'yellow': 0.1, 'red': 0.3},
    ...
}
```

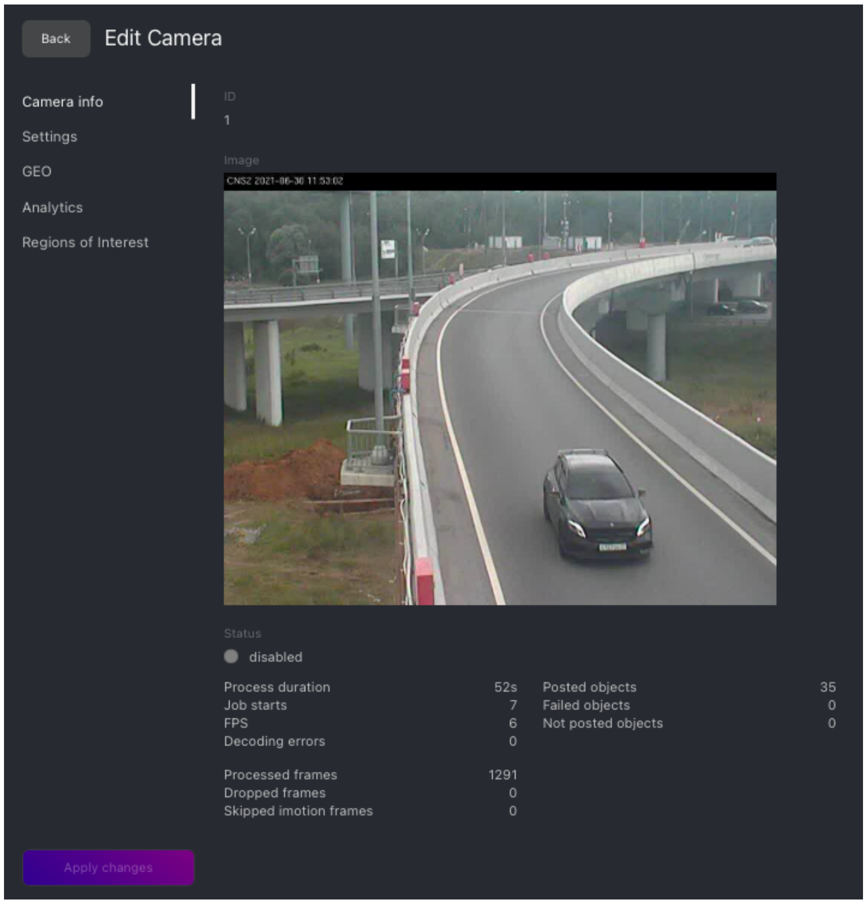
Each created camera is associated with a so called job, a video processing task that contains configuration settings and stream data and is assigned to `findface-video-worker`. This task can be restarted.

To restart a job, click in the *Action* column. In this case, the number of errors will be reset to `0`.

With a large number of cameras in the system, use the following filters:

- *Camera groups*,
- *Active*,
- *Status*.

For each camera, you will be provided with complete statistics such as current session duration, the number of successfully posted objects, the number of objects processed with errors after the last job restart, the number of frame drops, and other data. To consult these data, click the camera and go to the *Camera Info* tab.



```
Process duration          52s   Posted objects        35
Job starts                  7   Failed objects         0
FPS                         6   Not posted objects     0
Decoding errors             0

Processed frames         1291
Dropped frames              0
Skipped imotion frames      0
```

See also:

- *Allocate findface-video-worker to Camera Group*
- *Deduplicate Events*
- *Video object detection: findface-video-manager and findface-video-worker*

2.2.4 Configure Object Monitoring. Dossier Database

This chapter is all about configuring object monitoring and creating the dossier database.

FindFace Multi provides video monitoring of the following objects:

- human face
- human body (silhouette)
- car

Object monitoring is implemented using a set of default and custom watch lists, e.g., wanted, VIP, etc., and a database of dossiers. You can create as many custom watch lists as necessary. Each dossier contains one or several object photos and is allocated to one or several watch lists. To put an object on monitoring, you need to make a relevant watch list active.

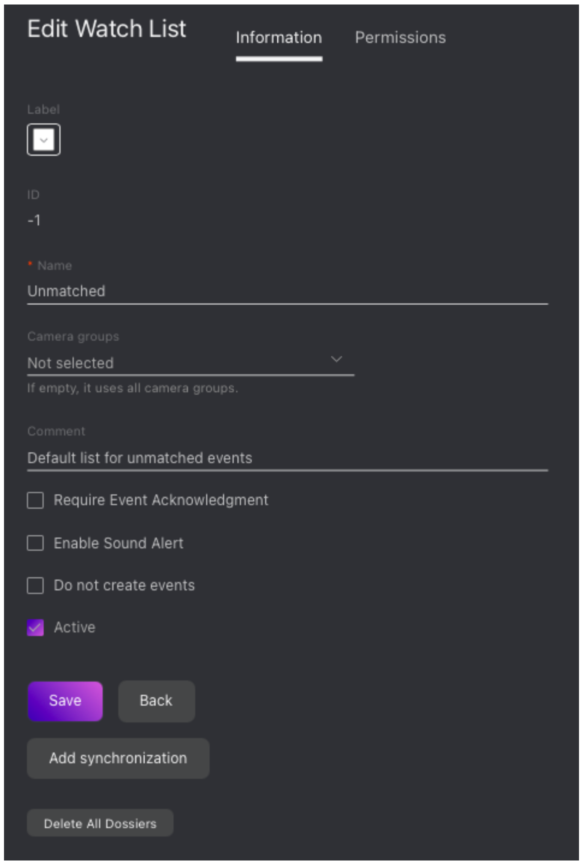
You can save mixed data and objects of different kinds to the same dossier as long as these objects are connected. For example, a dossier can contain a face and full-length photo, car photo, and license plate number, all linked to a wanted perpetrator. In this case, the system will be looking for any of these objects in the camera field.

Tip: To create dossiers in bulk, use the *batch photo upload* functionality.

In this section:

Monitoring Unmatched Objects

FindFace Multi features a special pre-configured watch list used for monitoring only unmatched objects (objects that do not match any dossier). This watch list cannot be removed from the system. To edit its settings, navigate to the *Preferences* tab. Click *Watch Lists* and then click *Unmatched* in the table.


Edit Watch List
Information
Permissions

Label

ID
-1

* Name
Unmatched

Camera groups
Not selected
If empty, it uses all camera groups.

Comment
Default list for unmatched events

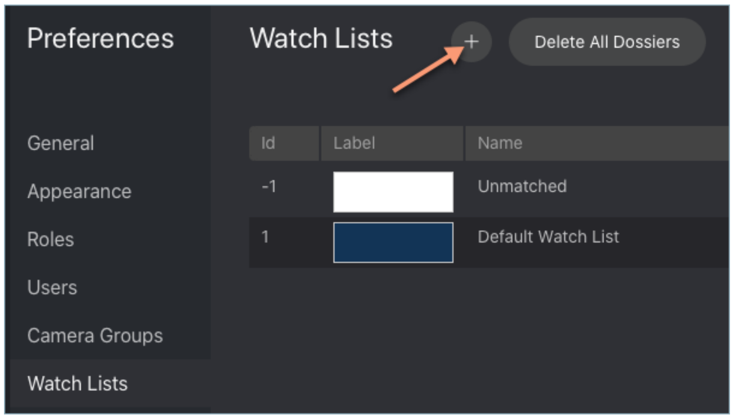
- [ ] Require Event Acknowledgment
- [ ] Enable Sound Alert
- [ ] Do not create events
- [x] Active

Note: To view only unmatched objects in the event list, select *Only without matches* in the *Matches* filter on the *Events* tab.

Create Watch List

You can create a custom watch list. Do the following:

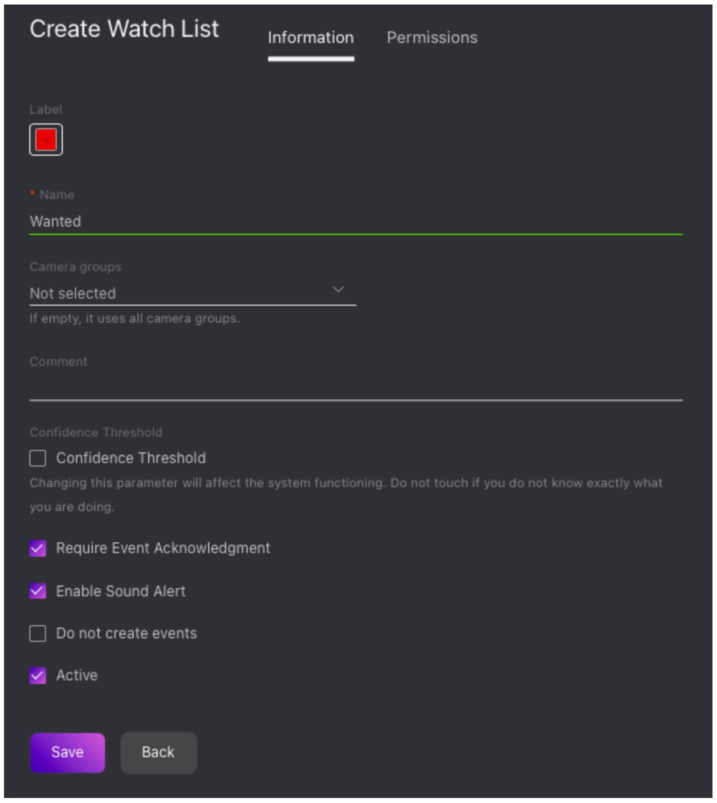
1. Navigate to the *Preferences* tab. Click *Watch Lists*.
2. Click +.



Watch Lists

Id	Label	Name
-1	⬜	Unmatched
1	🟦	Default Watch List

3. From the *Label* palette, select a color which will be shown in event notifications for this list. Keep in mind that the right color makes for a quicker response of the person on duty.



Create Watch List

Information | Permissions

Label

🟥

* Name

Wanted

Camera groups

Not selected

If empty, it uses all camera groups.

Comment

Confidence Threshold

☐ Confidence Threshold

Changing this parameter will affect the system functioning. Do not touch if you do not know exactly what you are doing.

☑ Require Event Acknowledgment

☑ Enable Sound Alert

☐ Do not create events

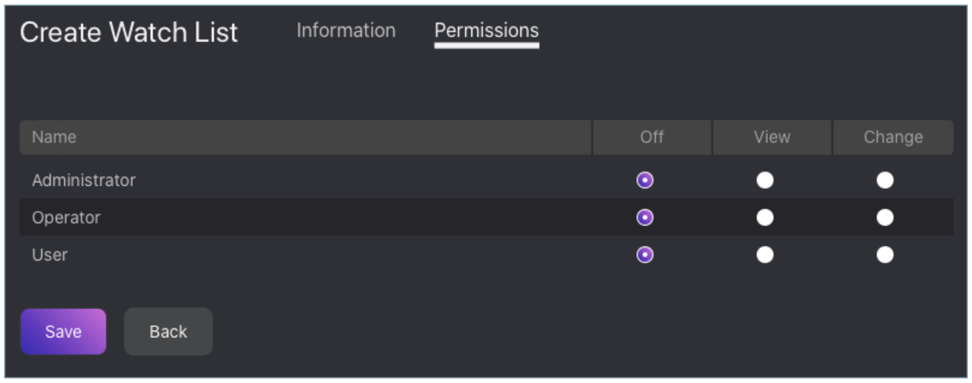
☑ Active

Save | Back

4. Specify the watch list name. Add a comment if needed.
5. Select a camera group(s) that will be used to monitor the watch list. If no groups specified, the watch list will be monitored by all active cameras in the system.
6. Check *Require acknowledgment* if it is mandatory that events associated with the list be manually acknowledged.
7. Check *Enable sound alert* to turn on sound notifications for the list if needed.
8. By default, all watch lists in the system are applied the *generic confidence threshold*. To set an individual threshold for the watch list, check *Confidence Threshold* and specify the threshold value.

Important: The default generic confidence threshold is optimal for the majority of recognition cases. We do not recommend changing it on your own. Be sure to consult with our technical experts by support@ntechlab.com.

9. Check *Active*.
10. Click *Save*.
11. On the *Permissions* tab, assign privileges on the watch list, specifying which user roles are allowed to change/view the watch list settings.



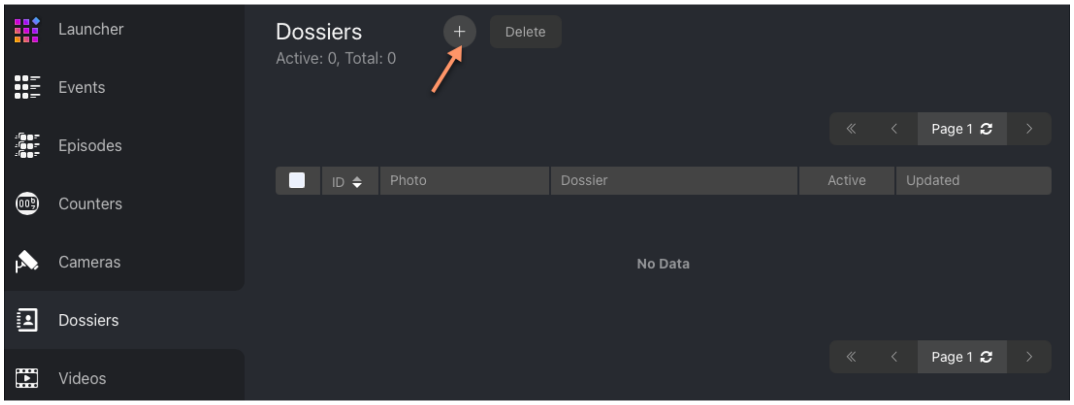
Name	Off	View	Change
Administrator	●	○	○
Operator	●	○	○
User	●	○	○

12. Click *Save*.

Create Dossier Manually

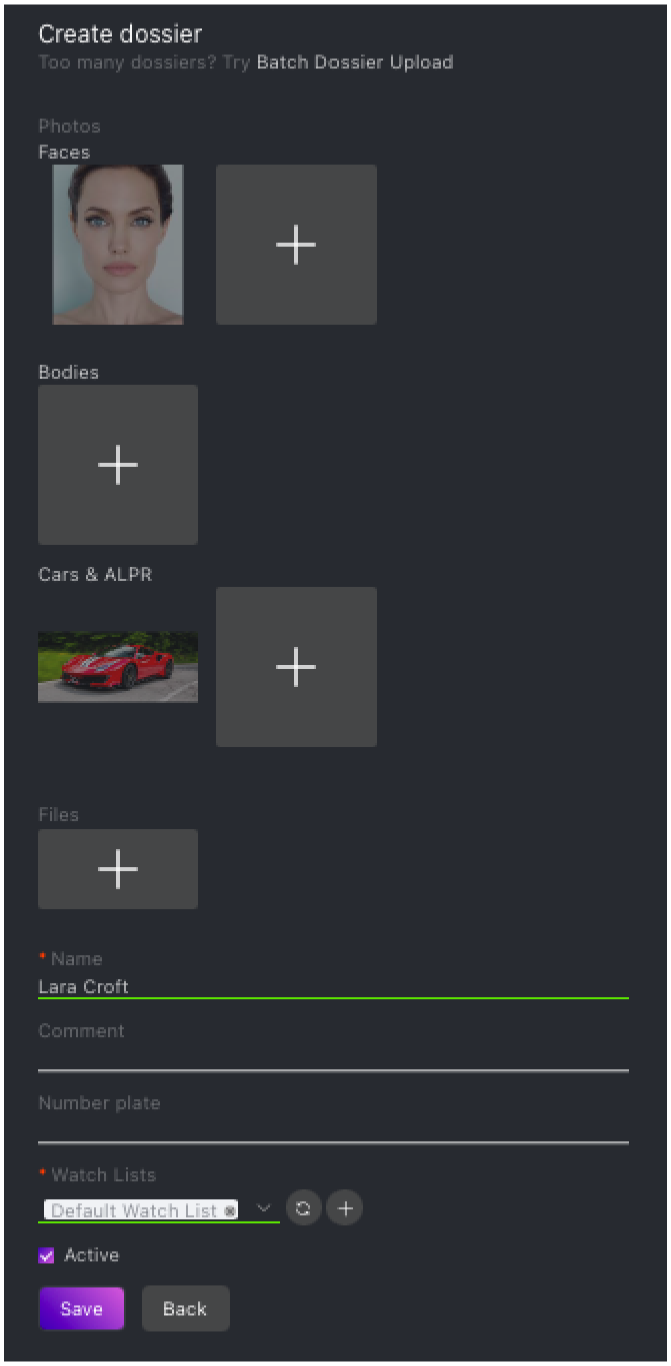
To create a dossier manually, do the following:

1. Navigate to the *Dossiers* tab.
2. Click +.



3. Attach photos of at least one of the following objects: face, body, car. Supported formats: WEBP, JPG, BMP, PNG.

Important: A face or body in the photos must be close to a frontal position. Distance between pupils: 60 px. Photos that do not meet the requirements will be rejected with a detailed error description.



4. Attach related files.
5. Specify the person’s name. If necessary, add a comment.
6. Specify the car’s license plate number if applicable.
7. From the *Watch lists* drop-down menu, select a watch list (or several lists, one by one) for the dossier.
8. Check *Active*. If a dossier is inactive, it is excluded from the real time monitoring.
9. Click *Save*. If a similar dossier already exists in the database, you will be able to merge it with the new dossier, create the new dossier anyway, or cancel creation.

Batch Photo Upload

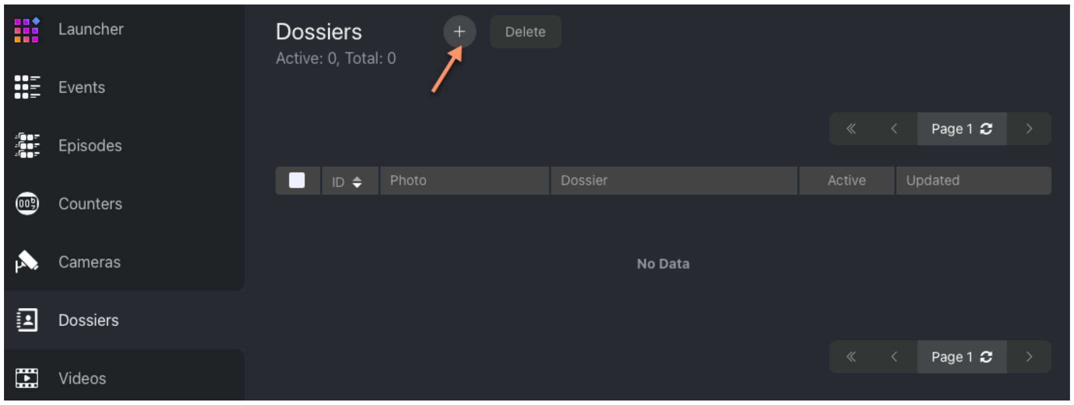
To create dossiers in bulk, use the batch photo upload. Do the following:

Tip: If you need to upload a large number of photos (more than 10,000), use *Console Bulk Photo Upload*.

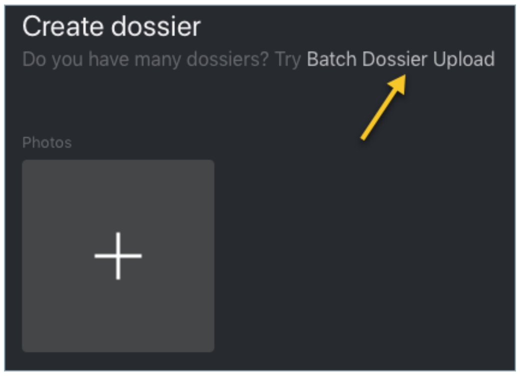
Important: Supported formats: WEBP, JPG, BMP, PNG.

Important: A face and body in the photos must be close to a frontal position. Distance between pupils: 60 px. Photos that do not meet the requirements will be rejected with a detailed error description.

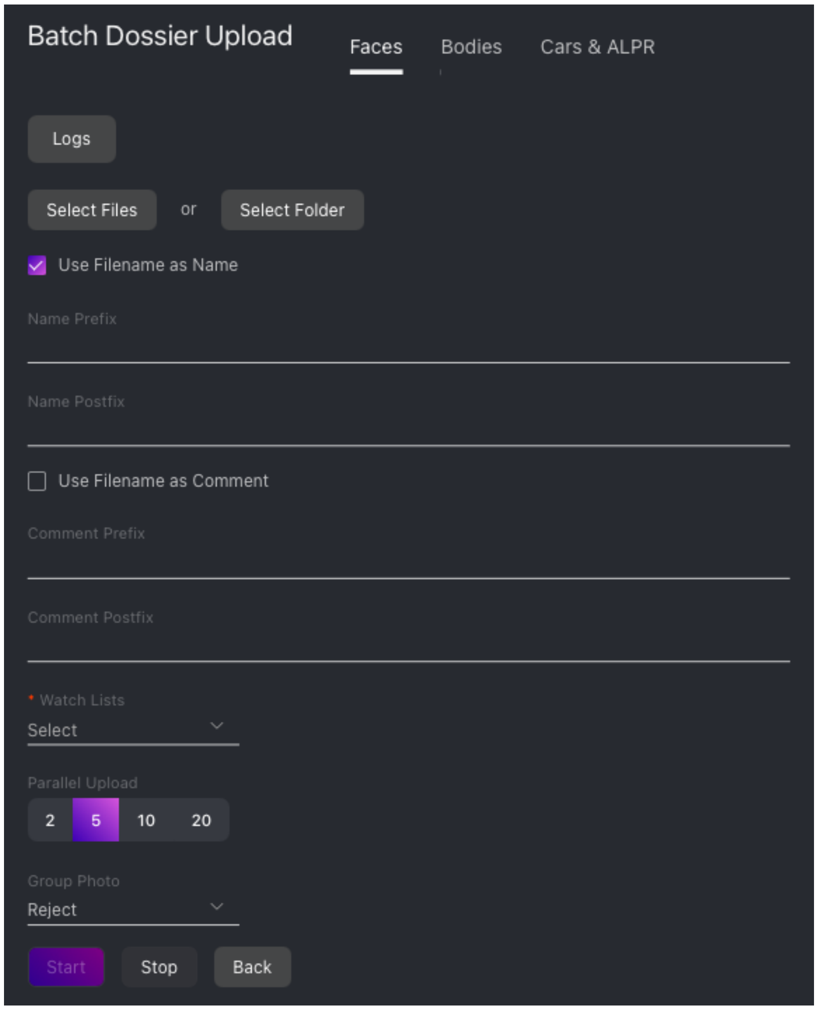
1. Navigate to the *Dossiers* tab.
2. Click +.



3. Click *Batch Dossier Upload*.



4. Specify the type of objects to detect in the photos.
5. Select multiple image files, or a directory.

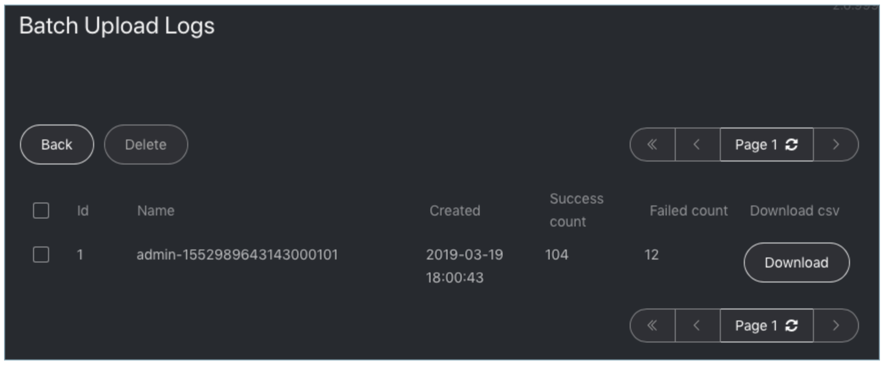


6. You can use image file names as a basis for names and/or comments in dossiers to be created. Select the necessary option(s). Then configure the automatic name/comment generation rule by appending a custom prefix and/or postfix to the file name.

Tip: To avoid merging the 3 words into one, use underscore or another symbol in the prefix and postfix.

7. From the *Watch lists* drop-down menu, select a classification list for the dossiers.
8. Use the *Parallel Upload* option to specify the number of photo upload streams. The more streams you use, the faster it takes to complete the upload, however it requires more resources as well.
9. From the *Group Photo* drop-down menu, select the system behavior upon detecting several objects in a photo: reject the photo, upload the biggest object, or upload all objects.
10. Click *Start* to launch the photo upload.

Important: To view the batch photo upload log, click *Logs*. You can then download the log in the `.csv` format if needed.

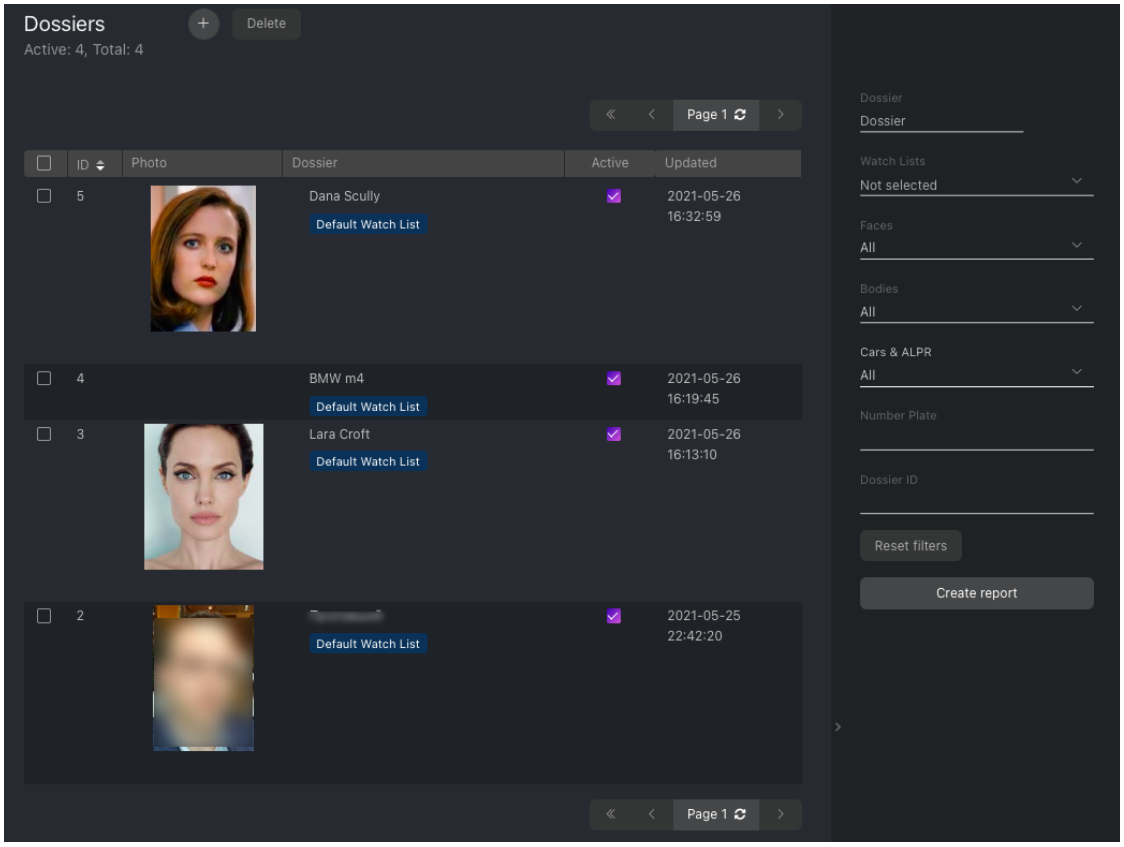


Batch Upload Logs

Id	Name	Created	Success count	Failed count	Download csv
1	admin-1552989643143000101	2019-03-19 18:00:43	104	12	Download

Filter Dossiers

You can find all dossiers created in FindFace Multi on the *Dossiers* tab. Use filters to sort them out.



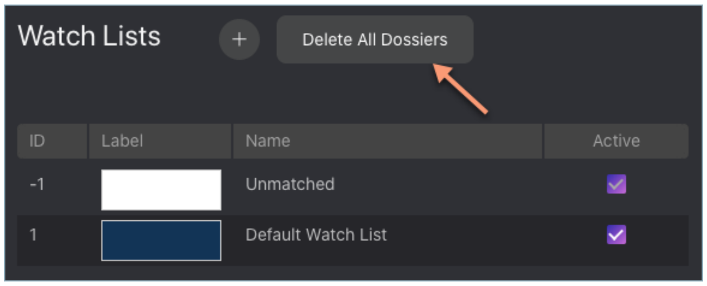
Dossiers
Active: 4, Total: 4

ID	Photo	Dossier	Active	Updated
5		Dana Scully Default Watch List	✓	2021-05-26 16:32:59
4		BMW m4 Default Watch List	✓	2021-05-26 16:19:45
3		Lara Croft Default Watch List	✓	2021-05-26 16:13:10
2		Default Watch List	✓	2021-05-25 22:42:20

Dossier: Dossier
Watch Lists: Not selected
Faces: All
Bodies: All
Cars & ALPR: All
Number Plate:
Dossier ID:

Purge Dossier Database

You can purge the entire dossier database in one click. To do so, navigate to the *Preferences* tab. Click *Watch Lists*. Click *Delete All Dossiers*.


Watch Lists

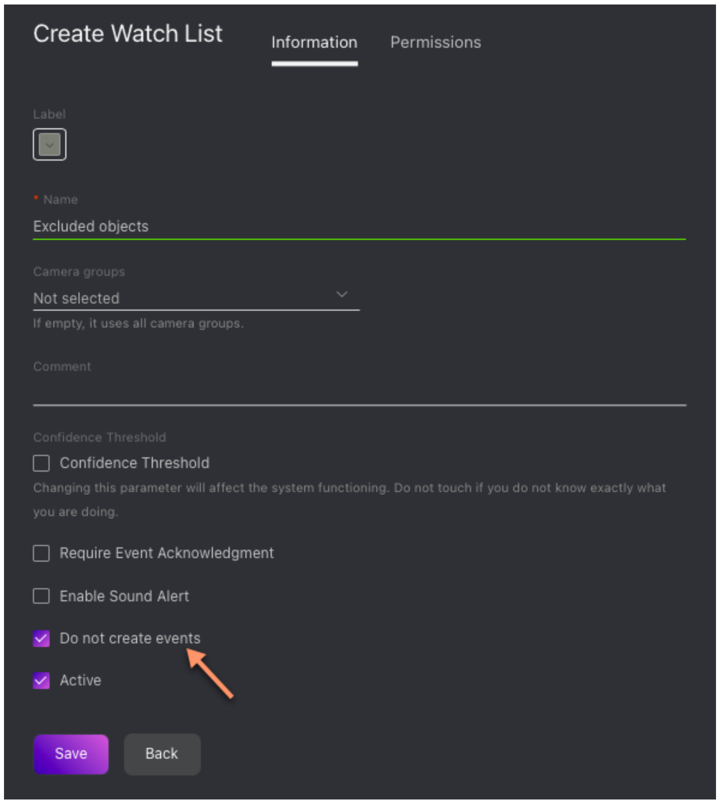
ID	Label	Name	Active
-1	⬜	Unmatched	✓
1	🟦	Default Watch List	✓

Disable Event Creation for Specific Objects

Sometimes, it is necessary to omit certain objects during monitoring. One of the most common cases is the objects in advertisement media located in the camera field. Being detected continuously by your system, they can easily overflow the event feed and the database.

To prevent this from happening, do the following:

1. *Create a watch list* that will store the objects excluded from detection. In its settings, check *Do not create events*.


Create Watch List

Label

* Name
Excluded objects

Camera groups
Not selected
If empty, it uses all camera groups.

Comment

Confidence Threshold
- [ ] Confidence Threshold

Changing this parameter will affect the system functioning. Do not touch if you do not know exactly what you are doing.

- [ ] Require Event Acknowledgment
- [ ] Enable Sound Alert
- [x] Do not create events
- [x] Active

2. For each excluded object, *create a dossier* and add it to the watch list.

2.2.5 Face and Body Counters

Important: To be able to count human bodies (silhouettes), you first have to enable *body detection*.

FindFace Multi allows you to count faces and bodies on connected cameras. This functionality can apply to a wide range of situations, such as people counting in queues and waiting areas, monitoring public gatherings, crowding prevention, and more.

You can count objects on each camera independently or collectively on all cameras of one or several camera groups.

The counting method is based on time slices, which means that the system counts faces and bodies in static screenshots taken with a given `count interval`. The counter shows how the number of faces and bodies changes over time.

In this section:

Configure Counters

To configure counters, open the `/etc/findface-security/config.py` configuration file and modify the following parameters:

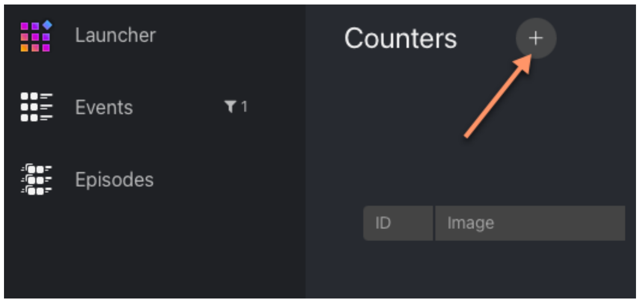
- `COUNTERS_SAVE_FULLFRAME` determines saving options of full frames in counters: `always`, `detect` - only save if faces or bodies have been detected, `never`.
- `COUNTERS_FULLFRAME_JPEG_QUALITY`: JPEG quality of full frames,
- `COUNTERS_THUMBNAIL_JPEG_QUALITY`: JPEG quality of thumbnails.

```
sudo vi /etc/findface-security/config.py

# counters full frame saving options:
# `always` - save always
# `detect` - save only if faces or bodies have been detected
# `never` - never save full frames
'COUNTERS_SAVE_FULLFRAME': 'always',
'COUNTERS_FULLFRAME_JPEG_QUALITY': 75,
'COUNTERS_THUMBNAIL_JPEG_QUALITY': 75,
...
```

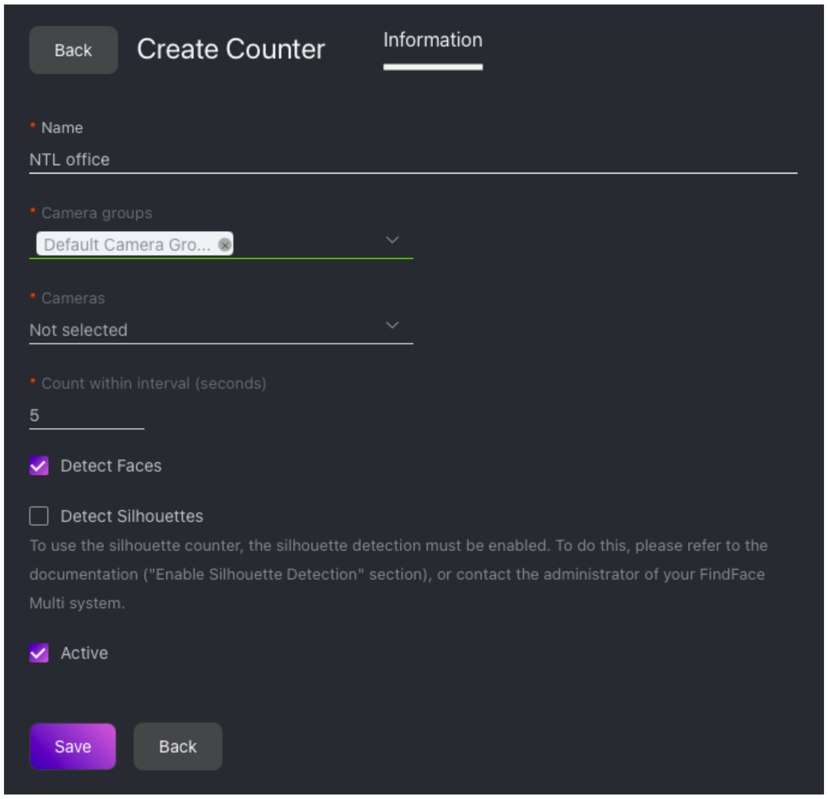
Create Counter

To set up a counter, do the following:

1. Navigate to the *Counters* tab.
2. Click +.



3. Specify the counter name.



* Name
NTL office

* Camera groups
Default Camera Gro...

* Cameras
Not selected

* Count within interval (seconds)
5

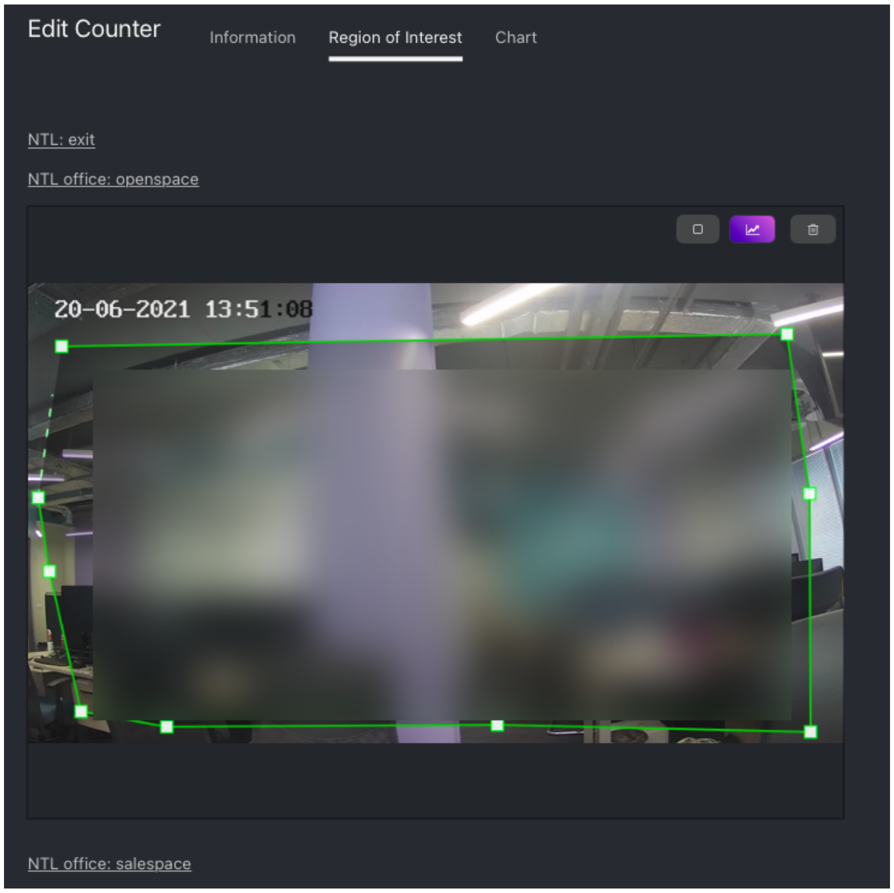
✓ Detect Faces

☐ Detect Silhouettes
To use the silhouette counter, the silhouette detection must be enabled. To do this, please refer to the documentation ("Enable Silhouette Detection" section), or contact the administrator of your FindFace Multi system.

✓ Active

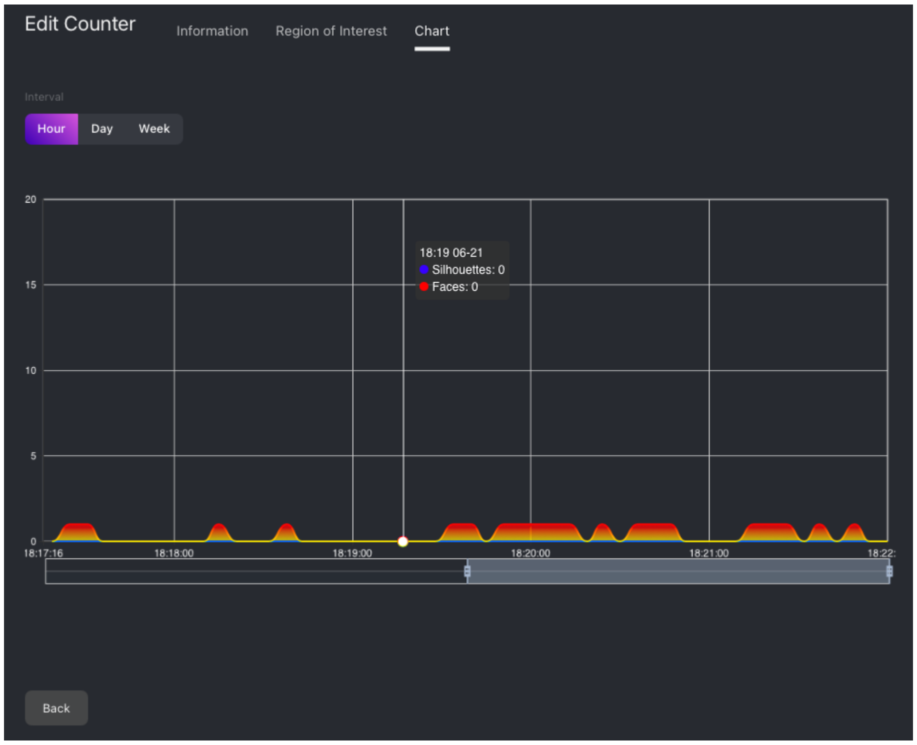
4. Select one or several camera groups for counting. By default, counting will apply to all cameras within the groups. If necessary, you can disable counting on specific cameras.
5. Specify the interval between two consecutive screenshots used for counting.
6. Check *Detect Faces* to count faces.
7. Check *Detect Silhouettes* to count bodies. Body detection has to be *enabled*.
8. Make sure that the counter is *Active*.
9. Click *Save*. You will see two new tabs appear.

10. (Optional) Navigate to the *Region of Interest* tab to specify the face/body tracking region within the camera(s) field.



Counter Chart

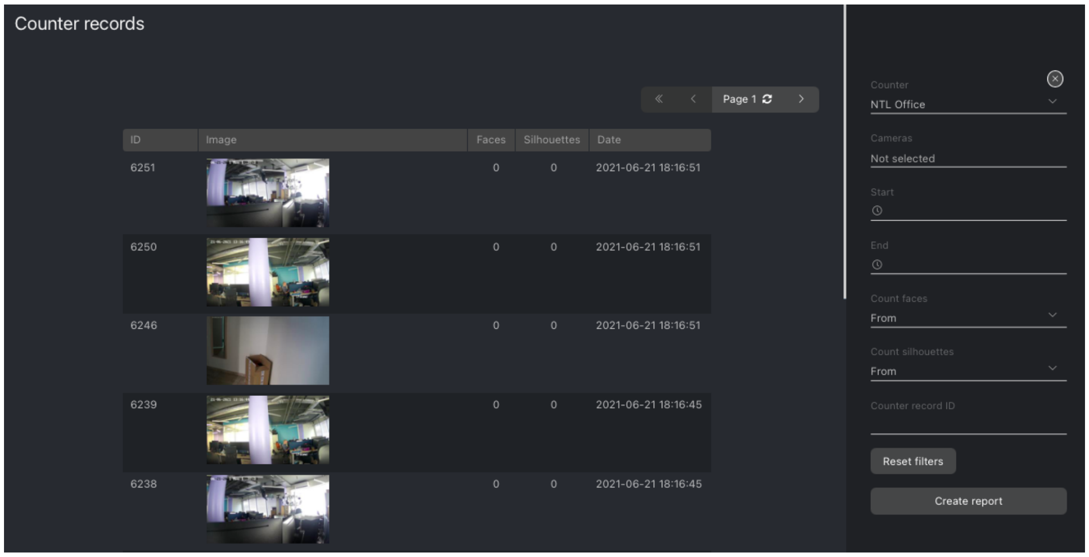
To see the counter chart for the last hour, 24 hours, or week, navigate to the *Chart* tab in the counter settings.



Work with Counter Records

Static screenshots taken by a counter, with the number of faces and bodies in them, are saved as counter records.

To see the counter records, navigate to the *Counters* tab. Click on any column for the counter, except `ID` (leads to the counter settings).



ID	Image	Faces	Silhouettes	Date
6251		0	0	2021-06-21 18:16:51
6250		0	0	2021-06-21 18:16:51
6246		0	0	2021-06-21 18:16:51
6239		0	0	2021-06-21 18:16:45
6238		0	0	2021-06-21 18:16:45

To work with counter records, use the following filters:

- Counter
- Cameras
- Camera groups

- Time period
- Number of faces in record
- Number of bodies in record
- Record id

Set Webhook for Counter

To take it up a notch, *configure a webhook* for counter records with a specific number of faces and bodies.

See also:

- *Enable Body and Body Attribute Recognition*
- *Webhooks*

2.2.6 Events and Episodes of Object Recognition

To monitor the real-time object identification in live videos, use the *Events* and *Episodes* tabs. Besides monitoring, both tabs allow you to access the history of identification events.

Important: In the current version, the Episodes functionality supports only faces.

Tip: Search for objects through the event database and dossier database on the *Search* tab.

Tip: To perform the object identification in archived videos, see *Object Identification in Offline Videos*.

Work with Events

This section is about the *Events* tab.

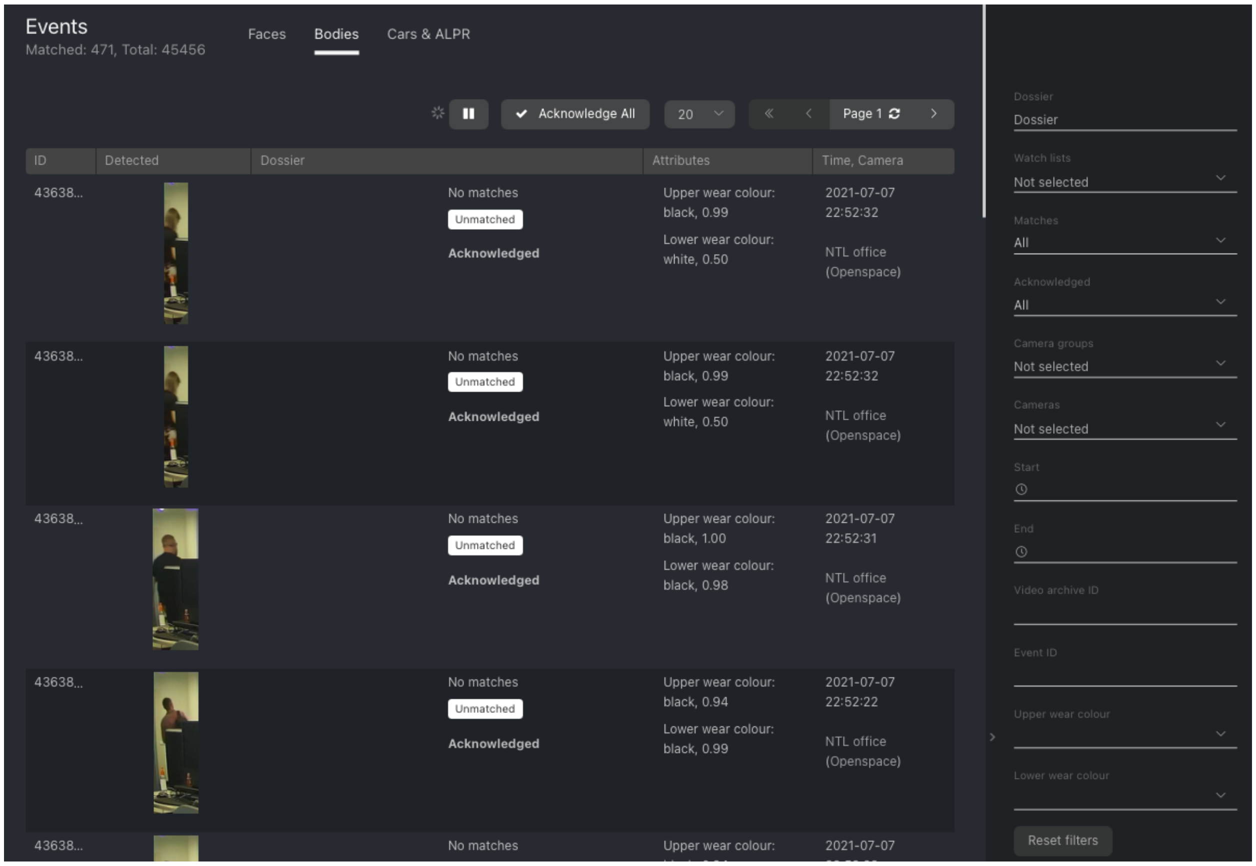
Tip: Take your security up a notch with *episodes*.

Important: You can *enable sound notifications* for events related to specific watch lists. In some browsers, the tab with events has to remain in focus to get a sound played. To put a tab in focus, open it, and click anywhere on the page.

In this chapter:

View Identification Events

Once an object is detected, you will see a notification on one of the event lists: *Faces*, *Bodies*, or *Cars & ALPR*, subject to the object type.



Events
Matched: 471, Total: 45456

ID	Detected	Dossier	Attributes	Time, Camera
43638...		No matches Unmatched Acknowledged	Upper wear colour: black, 0.99 Lower wear colour: white, 0.50	2021-07-07 22:52:32 NTL office (Openspace)
43638...		No matches Unmatched Acknowledged	Upper wear colour: black, 0.99 Lower wear colour: white, 0.50	2021-07-07 22:52:32 NTL office (Openspace)
43638...		No matches Unmatched Acknowledged	Upper wear colour: black, 1.00 Lower wear colour: black, 0.98	2021-07-07 22:52:31 NTL office (Openspace)
43638...		No matches Unmatched Acknowledged	Upper wear colour: black, 0.94 Lower wear colour: black, 0.99	2021-07-07 22:52:22 NTL office (Openspace)
43638...		No matches	Upper wear colour:	2021-07-07

Dossier: Dossier
Watch lists: Not selected
Matches: All
Acknowledged: All
Camera groups: Not selected
Cameras: Not selected
Start
End
Video archive ID
Event ID
Upper wear colour
Lower wear colour

A notification can feature different pieces of information, depending on whether a detected object has a match in the database:

- Match not found: the normalized object image, detection date and time, camera group name.
- Match found: the normalized object image, reference object photo from the related dossier, person name, the similarity between the matched objects, comment from the dossier, watch list, detection date and time, camera group.

Note: You can configure the system in such a way that you will get notifications only for the objects with a match.

Important: In order to pause the notifications thread, click ⏸ above the list of events.

When working with events, the following filters may come in handy:

Note: Some filters from the list below may be hidden, subject to enabled recognition features.

- *Dossier*: display events only for a selected dossier.
- *Watch lists*: display events only for a selected dossier category (watch list).
- *Matches*: display events only with/without matches, or all events.
- *Acknowledged*: display only acknowledged/unacknowledged events, or all events.
- *Cameras*: display only events from a selected camera.
- *Camera groups*: display only events from a selected group of cameras.
- *Start*, *End*: display only events that occurred within a certain time period.
- *Video Archive ID*: display events from the video archive with a given ID.
- *Event ID*: display an event with a given ID.

Specific filters for faces

- *Episode ID*: display events from the episode with a given ID.
- *Age*: display events with people of a given age.
- *Gender*: display events with people of a given gender.
- *Emotions*: display events with given emotions.
- *Glasses*: filter events by the fact of wearing glasses.
- *Beard*: filter events by the fact of having a beard.
- *Liveness*: filter events by face liveness.
- *Face mask*: filter events by the fact of wearing a face mask.

Specific filters for bodies

- *Upper wear color*: display only events with a person wearing a top of a given color.
- *Lower wear color*: display only events with a person wearing a bottom of a given color.

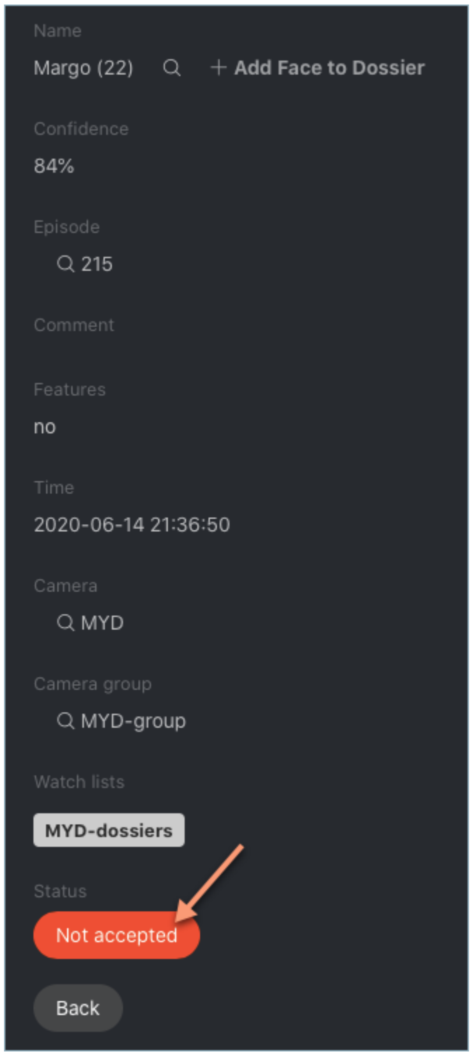
Specific filters for cars

- *Car body style*: display only events with cars of a given body style.
- *Car color*: display only events with cars of a given color.
- *Country of the license plate*: display only events with cars registered in a given country.
- *Number plate*: display an event with a given plate number.
- *Region of the license plate*: display only events with cars registered in a given region.
- *Make*: filter car events by car make.
- *Model*: filter car events by car model.

Event Ticket. Acknowledging Event

In order to navigate to an event ticket from the list of events, click on the recognition result in a notification (*No matches* or the name of a matching person).

An event ticket contains the same data as a relevant *notification*. It also allows for acknowledging the event. To do so, click *Not accepted* to change the event acknowledgment status. Click *Save*.



Name
Margo (22)
Confidence
84%
Episode
215
Comment
Features
no
Time
2020-06-14 21:36:50
Camera
MYD
Camera group
MYD-group
Watch lists
MYD-dossiers
Status
Not accepted

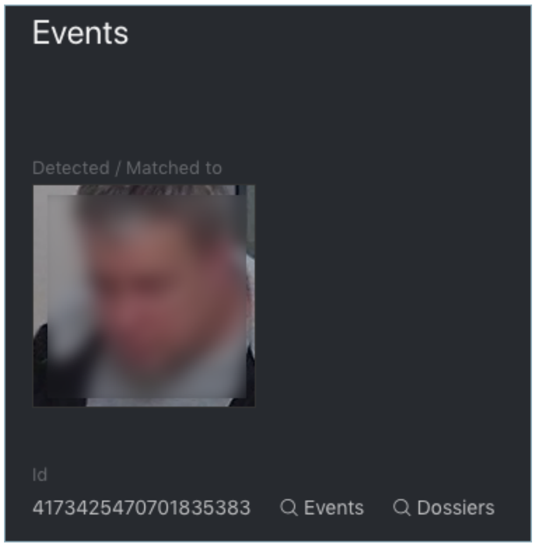
Tip: If a detected object has a match in the dossiers, you can navigate into a relevant one by clicking on the person's name in the event ticket.

Tip: In order to acknowledge all the events, click ✓ above the list of events.

Note: Event acknowledgment can be automated for selected watch lists.

Event Ticket. Object Search

FindFace Multi allows you to search detected objects through the list of events and dossier database. To navigate from an event ticket to the search tab, click *Events* or *Dossiers* respectively.



See also:

- search.

Organize Events with Episodes

This section is about the *Episodes* tab.

See also:

- *Work with Events*

An episode is a set of identification events that feature faces of the same person, detected within a specific period of time. As events on the *Events* tab show up in an arbitrary order, a large number of miscellaneous events can make the work difficult and unproductive. With the episodes, the system uses AI to organize incoming events based on the faces similarity and detection time. This allows for easy processing of diverse events, even in large numbers.

In this chapter:

About Episodes

An episode is a set of identification events that feature faces of the same person, detected within a certain period of time.

There are two types of episodes:

- LIVE: an episode is currently active, with more events to be possibly added.
- Closed: an episode is closed, no events can be added.

Episode Settings

To configure the episodes, use the `/etc/findface-security/config.py` configuration file. You need the following parameters into the `FFSECURITY` section:

- `EPISODE_SEARCH_INTERVAL`: The period of time preceding an event, within which the system searches the feature vector database for events with similar faces. If no such an event is found, the system creates a new episode. Otherwise, it picks up the most relevant event from a LIVE episode after sorting out the 100 most recent similar faces.

  Note: The threshold similarity in episodes differs from that for face verification. See *General Preferences*.

- `EPISODE_MAX_DURATION`: The maximum episode duration in seconds. After this time, an episode automatically closes.
- `EPISODE_EVENT_TIMEOUT`: The maximum time in seconds since the last event has been added to an episode. After this time, an episode automatically closes.
- `EPISODE_KEEP_ONLY_BEST_EVENT`: When closing an episode, delete all events in it, except the one with the best face. Use this option to save disk space.

```
sudo vi /etc/findface-security/config.py

...

FFSECURITY = {
    ...
    'EPISODE_KEEP_ONLY_BEST_EVENT': True,
```

(continues on next page)

(continued from previous page)

```
      'EPISODE_SEARCH_INTERVAL': 60,
      'EPISODE_MAX_DURATION': 300,
      'EPISODE_EVENT_TIMEOUT': 30,
      ...
}

...
```

Grant Rights for Episodes

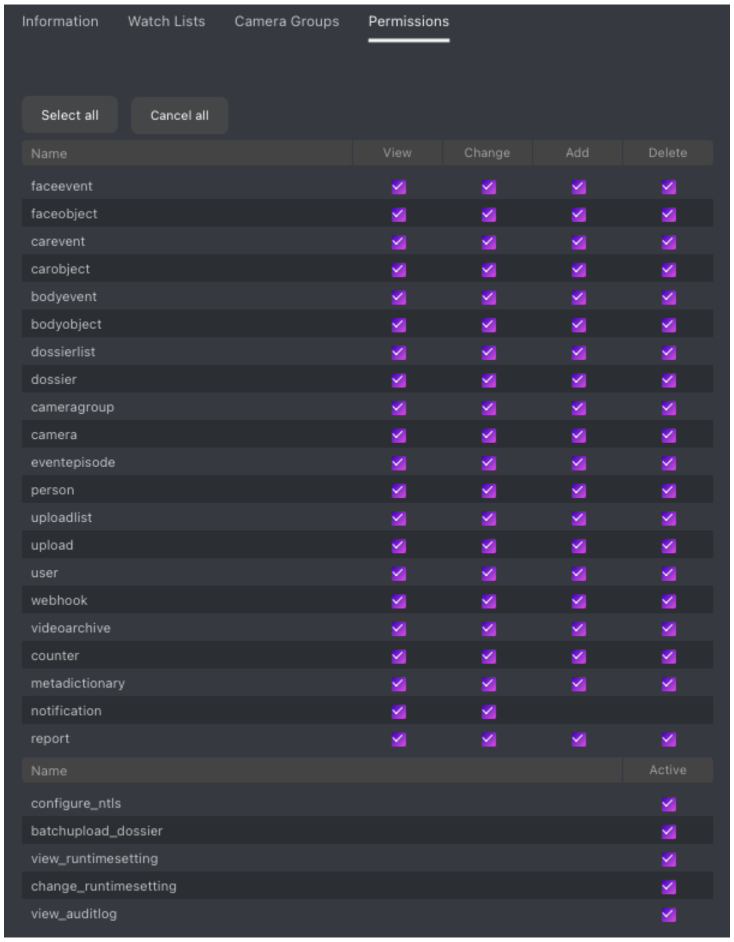
A user receives a notification of a new episode if they have rights for the first event. Viewing new events in the episode also requires proper rights.

The right for an event consists of the rights for a corresponding camera and watch list.

Note: To see unmatched events, you only need the rights for a camera.

To manage rights of a role for the entire `Episode` entity, open permissions for this role and adjust the `eventepisode` permission.

Tip: See *User Management*.

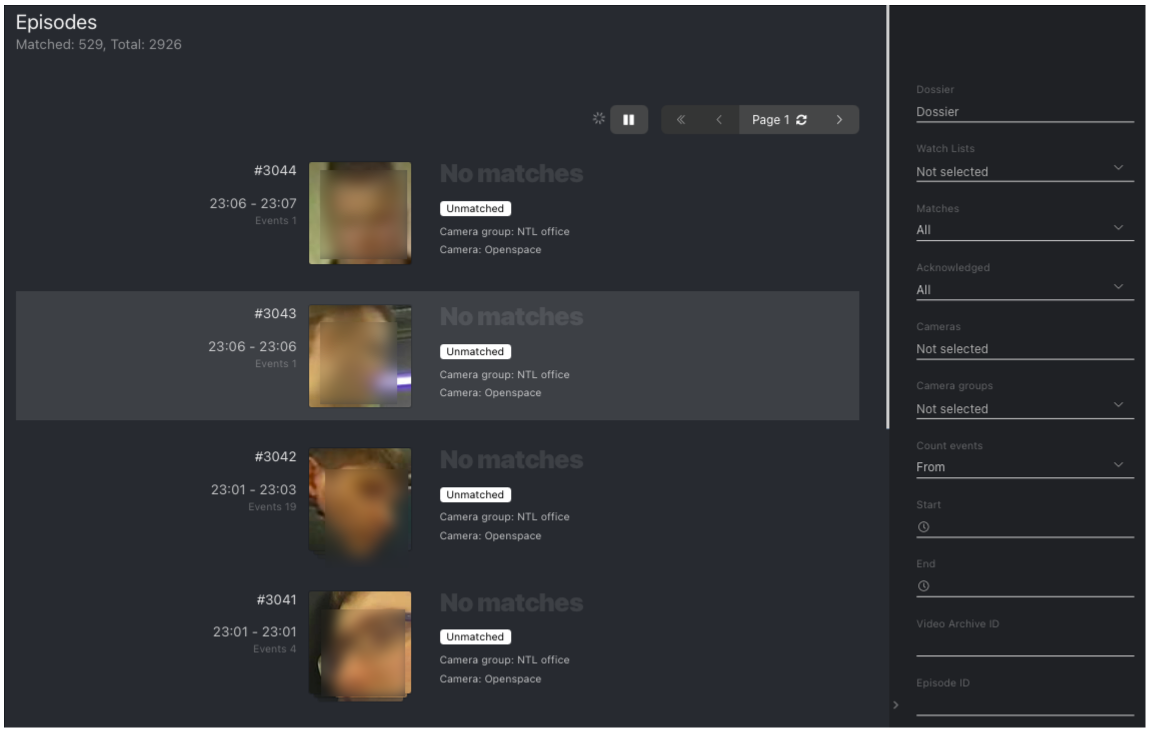


Name	View	Change	Add	Delete
faceevent	✓	✓	✓	✓
faceobject	✓	✓	✓	✓
carevent	✓	✓	✓	✓
carobject	✓	✓	✓	✓
bodyevent	✓	✓	✓	✓
bodyobject	✓	✓	✓	✓
dossierlist	✓	✓	✓	✓
dossier	✓	✓	✓	✓
cameragroup	✓	✓	✓	✓
camera	✓	✓	✓	✓
eventepisode	✓	✓	✓	✓
person	✓	✓	✓	✓
uploadlist	✓	✓	✓	✓
upload	✓	✓	✓	✓
user	✓	✓	✓	✓
webhook	✓	✓	✓	✓
videoarchive	✓	✓	✓	✓
counter	✓	✓	✓	✓
metadictionary	✓	✓	✓	✓
notification	✓	✓		
report	✓	✓	✓	✓

Name	Active
configure_ntls	✓
batchupload_dossier	✓
view_runtimesetting	✓
change_runtimesetting	✓
view_auditlog	✓

View Episodes

You can find the list of episodes with filters and statistics on the *Episodes* tab. Once a face is detected, it is either added to an existing LIVE episode, or used as a starting point of a new episode. Each episode is assigned an identifier which can be later used to filter events and episodes.

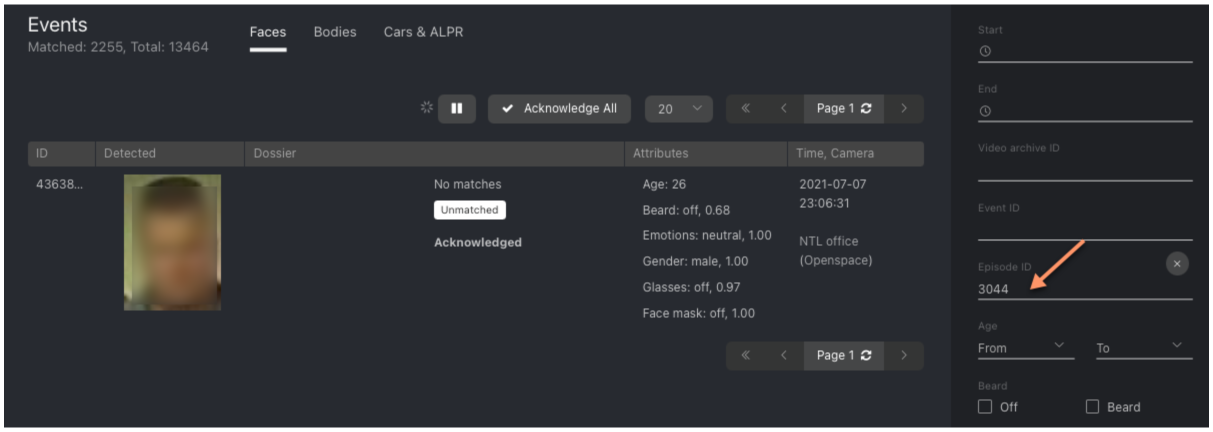


When working with episodes, the following default filters may come in handy:

- *Dossier*: display episodes only for a selected dossier.
- *Watch lists*: display episodes only for a selected dossier category (watch list).
- *Matches*: display episodes only with/without matches, or all episodes.
- *Acknowledged*: display only acknowledged/unacknowledged episodes, or all episodes.
- *Cameras*: display only episodes from a selected camera.
- *Camera groups*: display only episodes from a selected group of cameras.
- *Start*, *End*: display only episodes that occurred within a certain time period.
- *Count from*: display only episodes with a given number of events.
- *Video Archive ID*: display episodes related to the video archive with a given ID.
- *Episode ID*: display an episode with a given ID.

You can also filter episodes by face liveness and face features (if applicable).

To view the events added to an episode, click it on the list. You will be redirected to the *Events* tab with the corresponding episode ID set in the *Episode* filter:



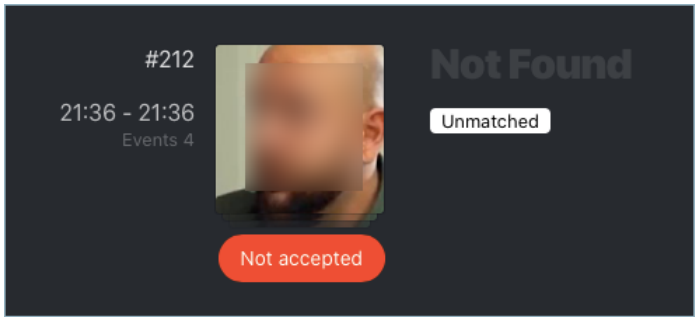
43638... No matches Unmatched Acknowledged Age: 26 Beard: off, 0.68 Emotions: neutral, 1.00 Gender: male, 1.00 Glasses: off, 0.97 Face mask: off, 1.00 2021-07-07 23:06:31 NTL office (Openspace)

Episode ID 3044

Work with the *Events* tab as described in *Work with Events*.

Event and Episode Acknowledging

To acknowledge an entire episode, click *Not accepted* for this episode on the list. As a result, all events in the episode will be automatically acknowledged, including those that are yet-to-appear (in the case of a LIVE episode).



An episode is also automatically acknowledged after acknowledging all its events one by one.

Filter Events by Episode ID

To display events by episode ID, either use the *id* filter on the *Episodes* tab or the *Episode ID* filter on the *Events* tab.

2.2.7 Object Identification in Offline Videos

Besides real-time object identification, FindFace Multi allows for offline video processing. This functionality has a wide range of possible applications, among which the most common case is object detection and recognition in archived videos.

In this chapter:

Configure Offline Video Processing

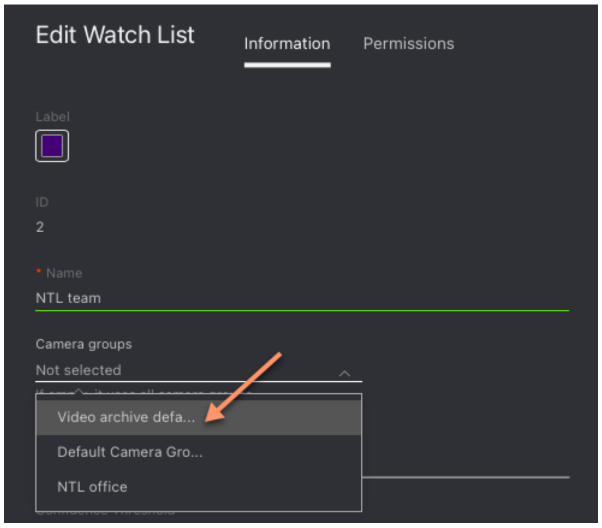
By default, video files are processed in a queued mode to prevent event drops due to resource overconsumption. You can modify the default number of simultaneously processed video files. To do so, open the `/etc/findface-security/config.py` configuration file and change the `MAX_VIDEO_ARCHIVE_JOBS` parameter. Please contact our experts prior (support@ntechlab.com) to make sure your resources are enough.

```
sudo vi /etc/findface-security/config.py

...

FFSECURITY = {
...
    # maximum concurrent video manager jobs for video archives processing
    'MAX_VIDEO_ARCHIVE_JOBS': 3,
    ...
}
...
```

Process Video File

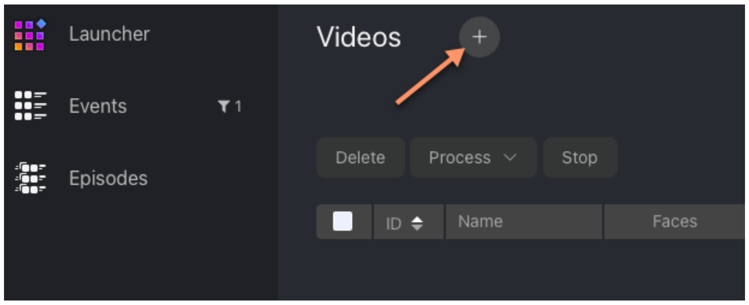
Important: Before the video processing, the following prep work is required:

1. Designate a camera group to which the system will attribute the object recognition events from the video. The `Video archive default camera group` is perfect for this task. You can also create a new camera group with basic settings specifically for this video file.
2. Assign the designated camera group to all *watch lists* you want to monitor when processing the video.

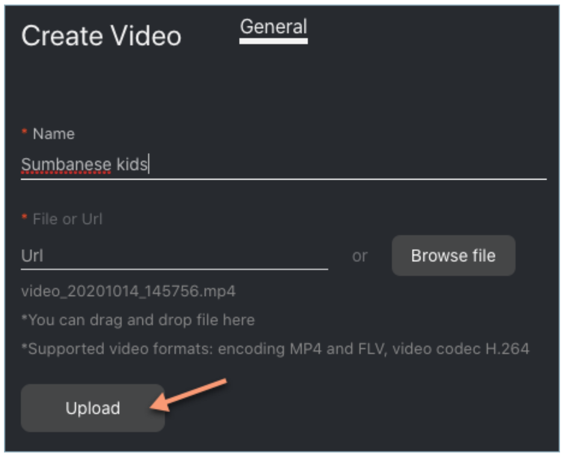


To identify objects in an offline video, do the following:

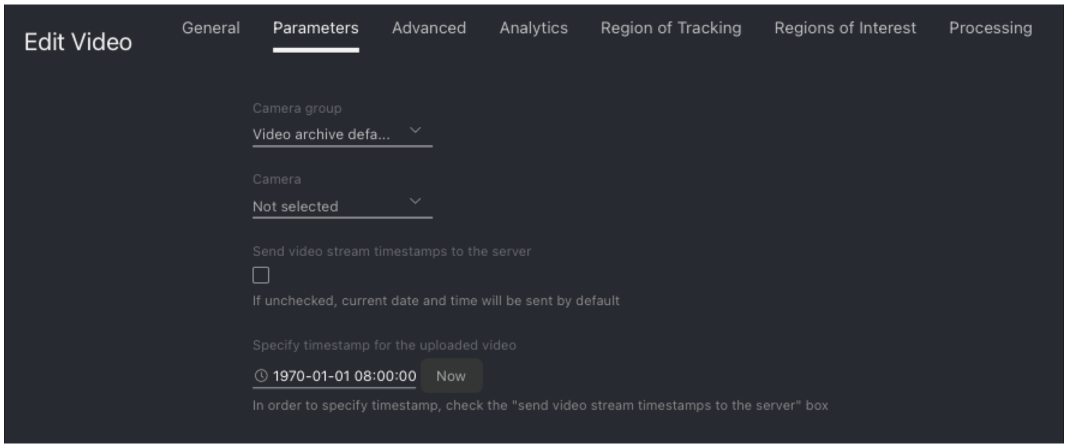
1. Create a video in FindFace Multi by uploading it from a file or online storage/cloud. To do so, navigate to the *Videos* tab.
2. Click +.



3. Specify the video name.


Create Video
General
* Name
Sumbanese kids
* File or Url
Url
or
Browse file
video_20201014_145756.mp4
*You can drag and drop file here
*Supported video formats: encoding MP4 and FLV, video codec H.264
Upload

4. Specify a URL in online storage, or select a file.
5. Click *Upload*.
6. After the video is uploaded, navigate to the *Parameters* tab. Specify parameters of video processing:


Edit Video
General | Parameters | Advanced | Analytics | Region of Tracking | Regions of Interest | Processing

Camera group
Video archive defa...

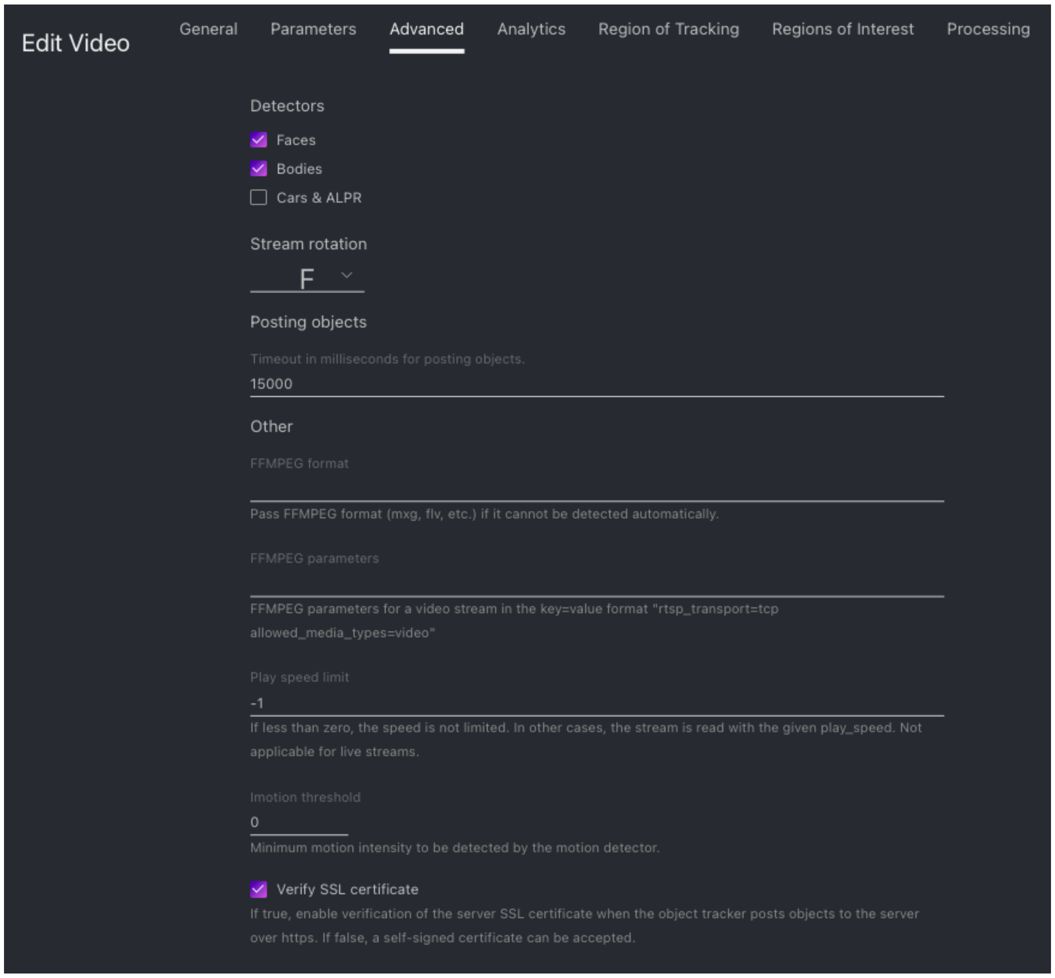
Camera
Not selected

Send video stream timestamps to the server
☐
If unchecked, current date and time will be sent by default

Specify timestamp for the uploaded video
1970-01-01 08:00:00 Now
In order to specify timestamp, check the "send video stream timestamps to the server" box

- Indicate the camera group you have designated prior. (Optional) Select a camera within that camera group to tag the object recognition events from this video more precisely.
- Configure the timestamps for object recognition events.

7. On the *Advanced* tab, fine-tune the video processing:



Detectors

✓ Faces

✓ Bodies

☐ Cars & ALPR

Stream rotation

F

Posting objects

Timeout in milliseconds for posting objects.

15000

Other

FFMPEG format

Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.

FFMPEG parameters

FFMPEG parameters for a video stream in the key=value format "rtsp_transport=tcp allowed_media_types=video"

Play speed limit

-1

If less than zero, the speed is not limited. In other cases, the stream is read with the given play_speed. Not applicable for live streams.

Imotion threshold

0

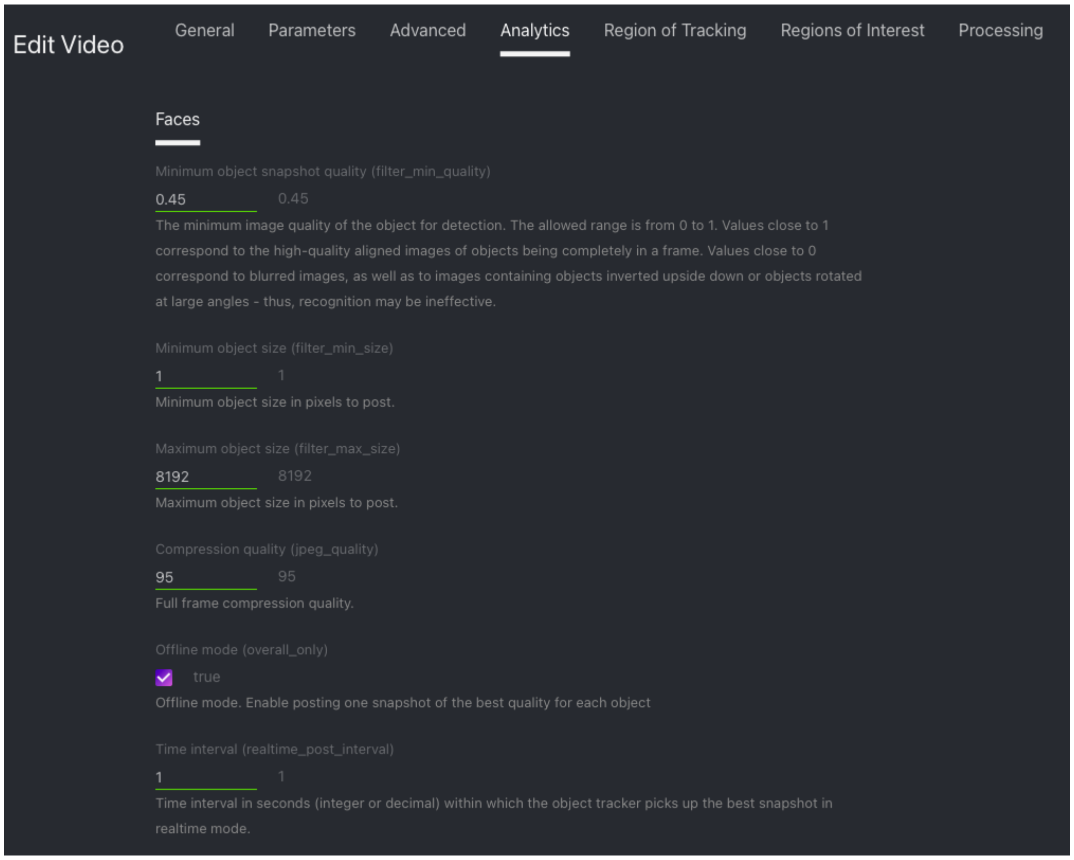
Minimum motion intensity to be detected by the motion detector.

✓ Verify SSL certificate

If true, enable verification of the server SSL certificate when the object tracker posts objects to the server over https. If false, a self-signed certificate can be accepted.

- Check detectors that you want to enable for this video: faces, bodies, cars & ALPR.
- If needed, change the video orientation.
- *Timeout in ms*: Specify the timeout in milliseconds for posting detected objects.
- *FFMPEG format*: Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.
- *FFMPEG parameters*: FFMPEG options for the video in the key-value format, for example, [“rtsp_transpotr=tcp”, “ss=00:20:00”].
- *Play speed limit*: If less than zero, the speed is not limited. In other cases, the stream is read with the given `play_speed`.
- *Imotion threshold*: Minimum motion intensity to be detected by the motion detector.
- *Verify SSL*: Check to enable verification of the server SSL certificate when the object tracker posts objects to the server over https. Uncheck the option if you use a self-signed certificate.

8. On the *Analytics* tab, specify settings for each object type detector.



Minimum object snapshot quality (filter_min_quality)
0.45 0.45
The minimum image quality of the object for detection. The allowed range is from 0 to 1. Values close to 1 correspond to the high-quality aligned images of objects being completely in a frame. Values close to 0 correspond to blurred images, as well as to images containing objects inverted upside down or objects rotated at large angles - thus, recognition may be ineffective.

Minimum object size (filter_min_size)
1 1
Minimum object size in pixels to post.

Maximum object size (filter_max_size)
8192 8192
Maximum object size in pixels to post.

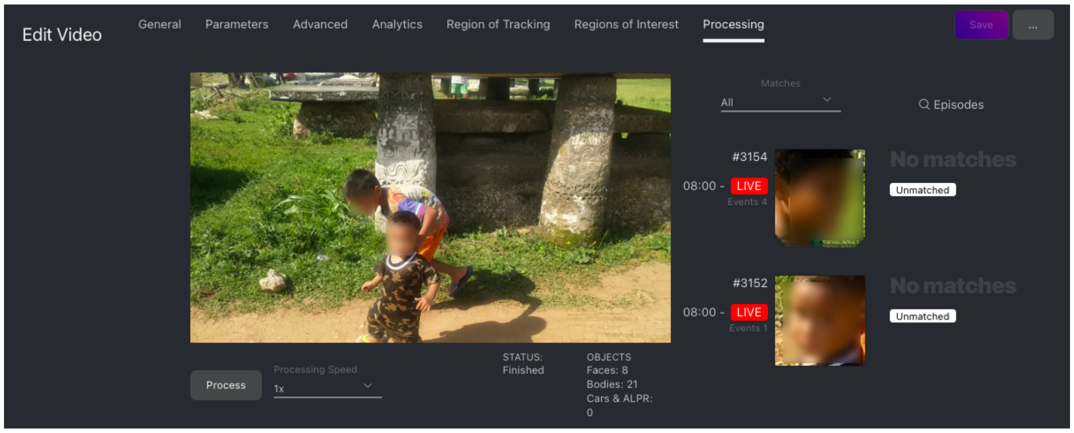
Compression quality (jpeg_quality)
95 95
Full frame compression quality.

Offline mode (overall_only)
✓ true
Offline mode. Enable posting one snapshot of the best quality for each object

Time interval (realtime_post_interval)
1 1
Time interval in seconds (integer or decimal) within which the object tracker picks up the best snapshot in realtime mode.

- *Minimum object snapshot quality*: Minimum quality of an object snapshot to post. Do not change the default value without consulting with our technical experts (support@ntechlab.com).
- *Minimum object size*: Minimum object size in pixels to post.
- *Maximum object size*: Maximum object size in pixels to post.
- *Compression quality*: Full frame compression quality.
- *Offline mode*: Offline mode. Enable posting one snapshot of the best quality per entire track for each object in addition to the sequence of snapshots taken per track in the real-time mode (enabled by default).
- *Time interval*: Time interval in seconds (integer or decimal) within which the object tracker picks up the best snapshot in the real-time mode.
- *Post first object immediately*: Check to post the first object from a track immediately after it passes through the quality, size, and ROI filters, without waiting for the first `Time interval` to complete. The way the subsequent snapshots are posted will depend on the `Post best snapshot` value. Uncheck the option only to post the first object after the first `Time interval` completes.
- *Post best snapshot*: Check to post the best snapshot obtained within each `Time interval` in the real-time mode, regardless of its quality. Uncheck the option to post the best snapshot only if its quality has improved compared to the previously posted snapshot.

9. (Optional) On the *Region of Tracking* and *Regions of Interest* tabs, specify the region of tracking within the camera field and detection zones for each object type detector.
10. Navigate to the *Processing* tab. Click *Process* to start object identification.



STATUS: Finished

OBJECTS
Faces: 8
Bodies: 21
Cars & ALPR: 0

You can view object identification events right here, as well as on the *Events* and *Episodes* tabs by filtering the list of events by the camera group/camera associated with the video.

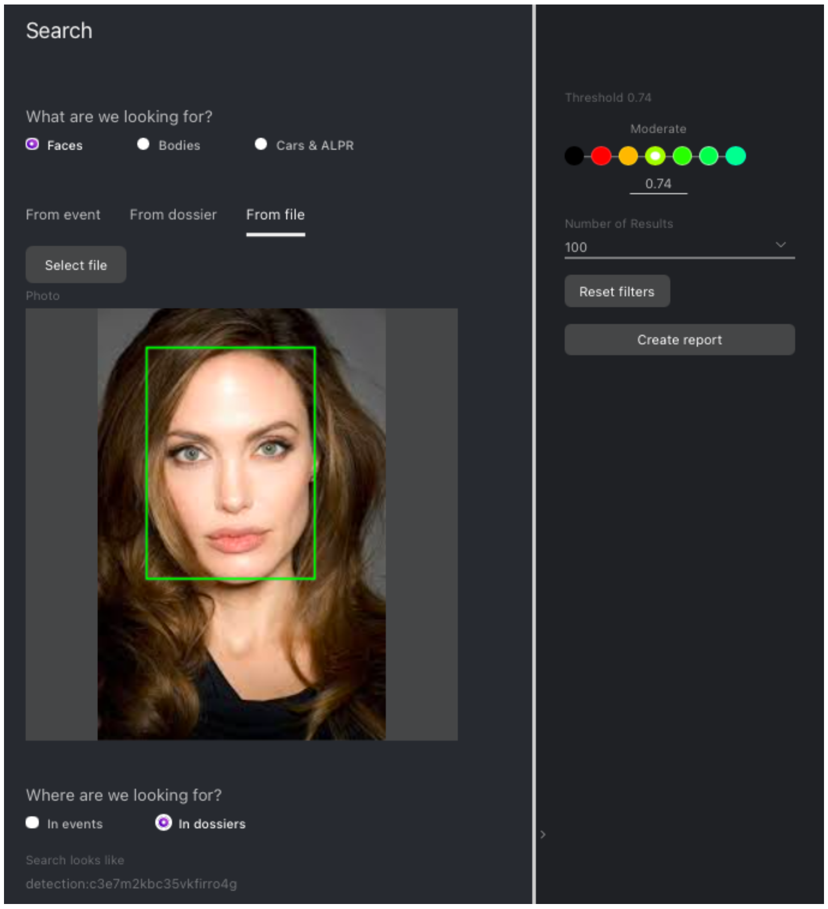
2.2.8 Search Objects in Databases

FindFace Multi allows you to search for objects in the following databases:

- Database of detected objects (the *Events* tab).
- Dossier database (the *Dossiers*). Contains object reference images.

To find an object in a database, do the following:

1. Navigate to the *Search* tab.



What are we looking for? [✓ Faces] Bodies Cars & ALPR

From event From dossier [✓ From file]

Where are we looking for? In events [✓ In dossiers]

Search looks like
detection:c3e7m2kbc35vkfirro4g

Threshold 0.74

0.74

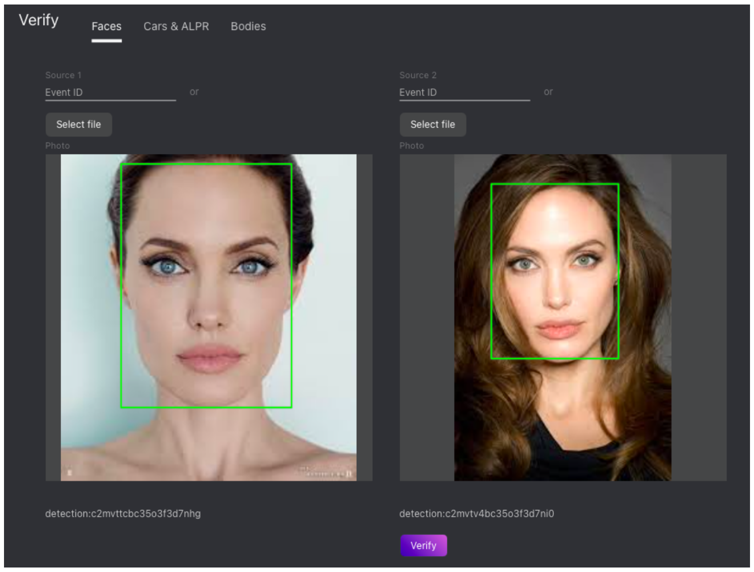
Number of Results
100

2. Choose the type of objects to search for: *Faces*, *Cars & ALPR*, *Bodies*.
3. Specify the database to search in: *Events*, *Dossiers*.
4. Specify the object to search for in one of the following ways:
   - By event ID with the object.
   - By dossier ID with the object. Should the dossier contain multiple photos, select some of them to use in the search.
   - By uploading a photo. It will be displayed in the *Photo* area. If there are multiple objects in the image, select the one of your interest.
5. By default, the system searches for objects using the pre-defined identification threshold, different for objects of different types. If necessary, set your value using the *Threshold* filter.
6. Specify the maximum number of search results.
7. Click *Search*. You will see the search results appear below. For each object found, the matching confidence level is provided.

2.2.9 Compare Two Objects

FindFace Multi allows you to compare two objects and verify that they match. Do the following:

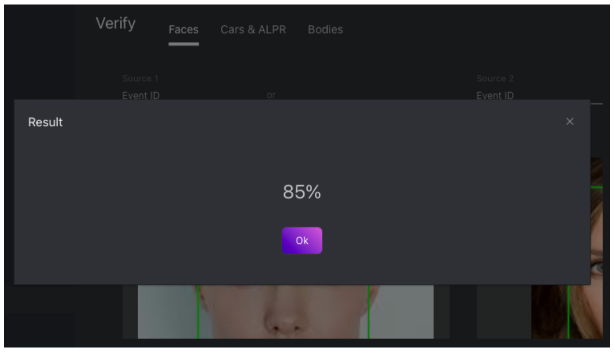
1. Navigate to the *Preferences* tab. Click *Verify*. Choose the proper tab, depending on the type of objects you are going to compare: *Faces*, *Cars & ALPR*, *Bodies*.



2. Specify the IDs of events that feature the objects you want to compare, and/or upload photos with the objects.

Tip: You can find event IDs on the *Events* tab.

3. Click *Verify*. You will see the probability that the objects match.



2.2.10 Person Recognition and People-Related Analytics

FindFace Multi is an ideal tool to gather people-related analytics. Enable person recognition first and then make the most of it with our analytical features.

Person Recognition

FindFace Multi allows for automatic person recognition. The system on-the-fly recognizes faces belonging to the same person and clusters them, building a person gallery. You can work with the person gallery on the *Persons* tab.

Note: In the person clusterization is enabled, the system databases will hold the `person event` entity linked to all *episodes* that feature a person's face.

Important: By default, person clusterization is disabled. *Enable and configure it* via the `/etc/findface-security/config.py` configuration file.

In this section:

Clusterization Methods

FindFace Multi uses the following methods to cluster faces belonging to the same person:

- Dynamic clusterization. The clusterization takes place on-the-fly after an episode is closed. The result of dynamic clusterization is shown in real-time on the *Persons* tab.

  Note: The technical details are as follows. Not every episode is qualified: the number of events in it must be equal or greater than `PERSON_EVENT_MIN_EPISODE_EVENTS` (set up via the `/etc/findface-security/config.py` configuration file). If an episode meets this requirement, the system selects the best quality event and performs the following operations:

  – Creates a new entity `PersonEvent` in the main system database **`PostgreSQL`**. The entity contains the event metadata, a link to the parent episode, face feature vector, and thumbnail.

  – Searches for a similar face centroid in the `person_events` gallery of the **`Tarantool`** feature vector database. A face centroid is a virtual feature vector averaged across all person's faces that have been detected so far. If a similar centroid is found, the system updates it using the new event. Otherwise, it creates a new centroid.

- Scheduled clusterization. We recommend scheduling it on late night hours as it takes up a lot of CPU resources and time.

  Note: The schedule is defined in the RRULE format as `PERSONS_CLUSTERIZATION_SCHEDULE` in the `/etc/findface-security/config.py` configuration file. The rest of the technical implementation resembles the dynamic method. However, the face centroid quality is better in the scheduled method as centroids are averaged across a larger array of accumulated feature vectors.

Important: The scheduled clusterization completely overwrites the person gallery, including ids.

Enable and Configure Person Clusterization

By default, person clusterization is disabled. To enable it, open the `/etc/findface-security/config.py` configuration file and modify the `SERVICES` section as such:

```
sudo vi /etc/findface-security/config.py

...
SERVICES = {
    "ffsecurity": {
        ...
        "persons": True,
    }
...
```

You will see the *Persons* tab appear in the FindFace Multi web interface.

In the same configuration file, you can modify the following parameters:

- `PERSON_EVENT_MIN_QUALITY`: minimum quality of faces used in person clusterization.
- `PERSON_EVENT_MIN_EPISODE_EVENTS`: minimum number of events in episodes used in person clusterization.
- `PERSONS_CONFIDENCE_THRESHOLD`: confidence threshold to match a face to a person.

  Warning: Consult with our experts by support@ntechlab.com before changing this parameter.

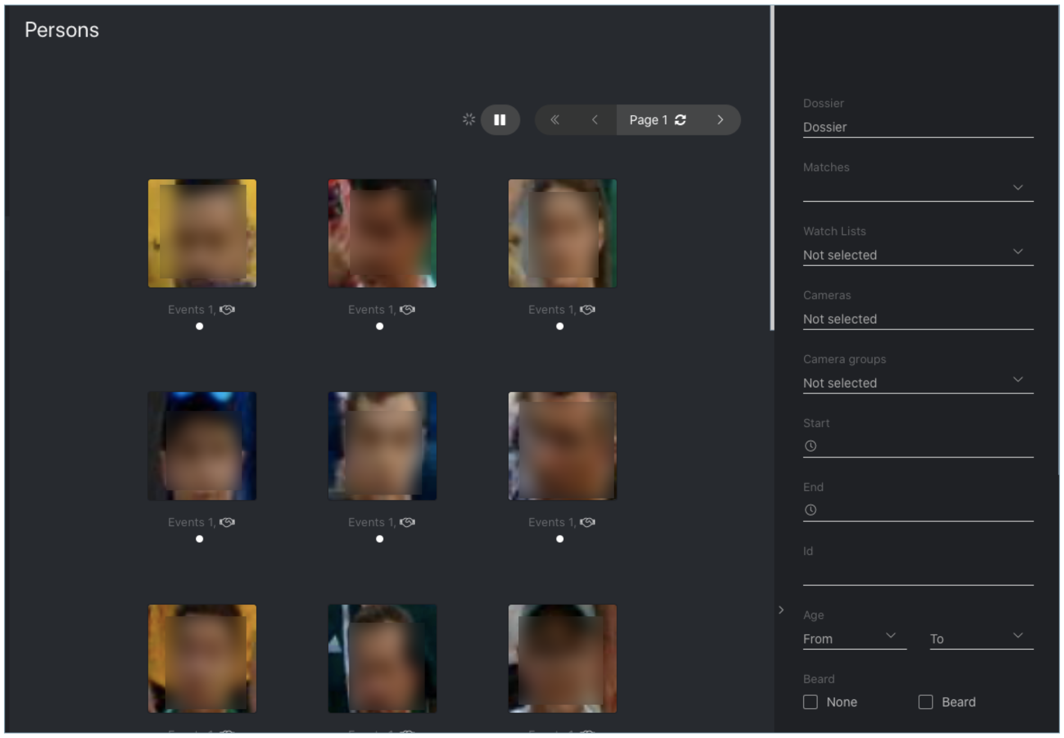
- `PERSONS_CLUSTERIZATION_SCHEDULE`: recurrence rule (RRULE) for scheduling person clusterization.

  Tip: See the RRULE calculator here.

```
# -- Persons configuration --
# rrule (recurrence rule) for scheduling persons clusterization
# WARNING: all scheduling works with UTC time and NOT aware of any timezone
'PERSONS_CLUSTERIZATION_SCHEDULE': 'RRULE:FREQ=DAILY;INTERVAL=1;WKST=MO;BYHOUR=0;
↪BYMINUTE=0',
# face to person matching confidence threshold
'PERSONS_CONFIDENCE_THRESHOLD': 0.739,
# minimum required face quality for person creation
'PERSON_EVENT_MIN_QUALITY': 0.45,
# minimum required number events in episode for person creation
'PERSON_EVENT_MIN_EPISODE_EVENTS': 1,
```

Work with Person Gallery

To see the person gallery, navigate to the *Persons* tab.



To work with the person gallery, use the following filters:

- Dossier
- Matches
- Cameras
- Camera groups
- Watch lists
- Time period
- Person id
- Face features (if enabled)
- Liveness (if enabled)

See also:

- *Configuration file of findface-security*
- *Webhooks*
- *Social Interaction Analysis*
- *Video Analytics*

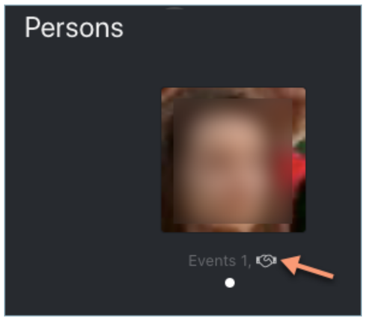
Social Interaction Analysis

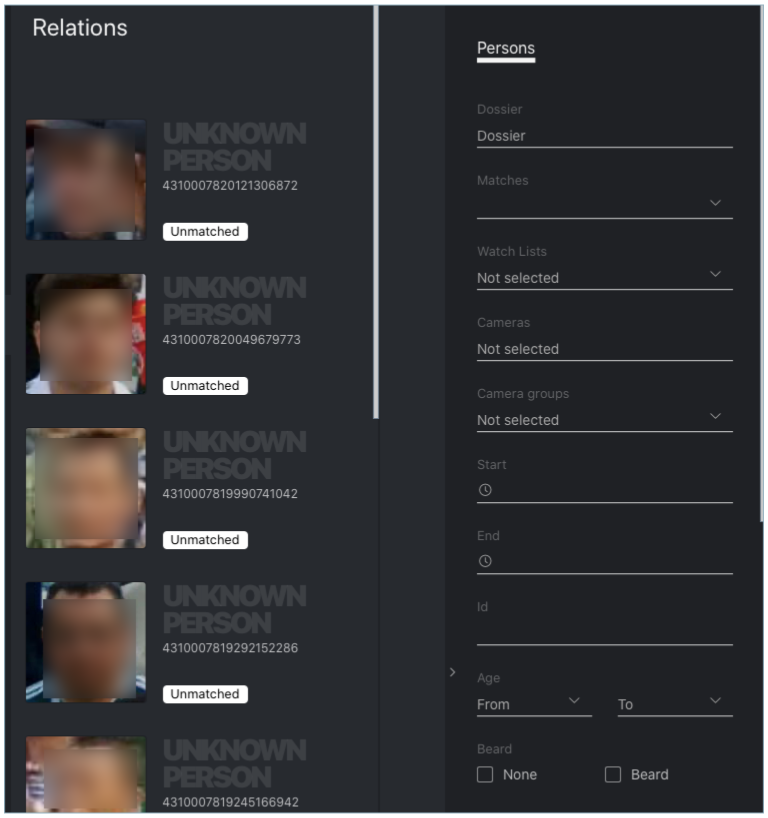
It is possible to see a circle of people with whom a person has previously been in contact. For each person from the first circle, the system determines another circle of connected people, and so on. Overall, social interaction analysis is three-circle deep.

Important: The social interaction analysis is provided only when the *person recognition* is enabled.

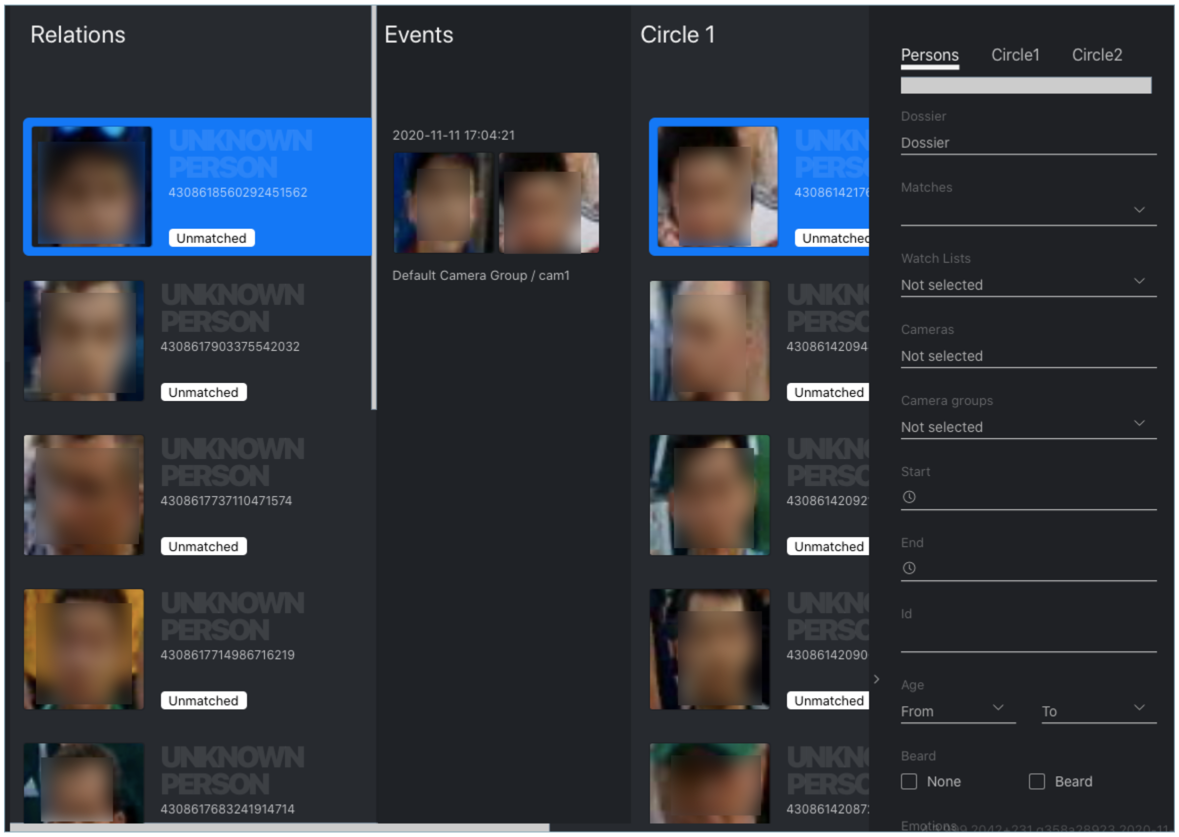
The social interaction analysis is available on the *Relations* tab.

Tip: You can also display the circle of connected people right from the *Persons* tab by clicking on the handshake icon.





On the *Relations* tab, click on a person to display their first circle of relations. Keep on to unveil the entire tree of social interactions.


Relations

UNKNOWN PERSON 4308618560292451562 Unmatched

UNKNOWN PERSON 4308617903375542032 Unmatched

UNKNOWN PERSON 4308617737110471574 Unmatched

UNKNOWN PERSON 4308617714986716219 Unmatched

UNKNOWN PERSON 4308617683241914714

Events

2020-11-11 17:04:21

Default Camera Group / cam1

Circle 1

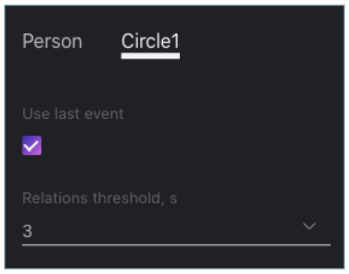
Dossier	Dossier
Matches	
Watch Lists	Not selected
Cameras	Not selected
Camera groups	Not selected
Start	
End	
Id	
Age	From To
Beard	☐ None ☐ Beard

You can apply available filters to every circle.

Tip: For example, you can find older adults or people without a face mask who are directly or indirectly related to a potentially contagious person.

When searching through a circle of relations, apply the following settings:

- *Use the last event*: use the last event of an episode to analyze contacts between individuals. In this case, having found truly associated people is most probable as they simultaneously leave a camera's field of view. If the option is disabled, the system will use the best event of an episode for relations search.
- *Relations threshold*: maximum time in seconds between the appearance of individuals to consider them related.



Use last event ✓

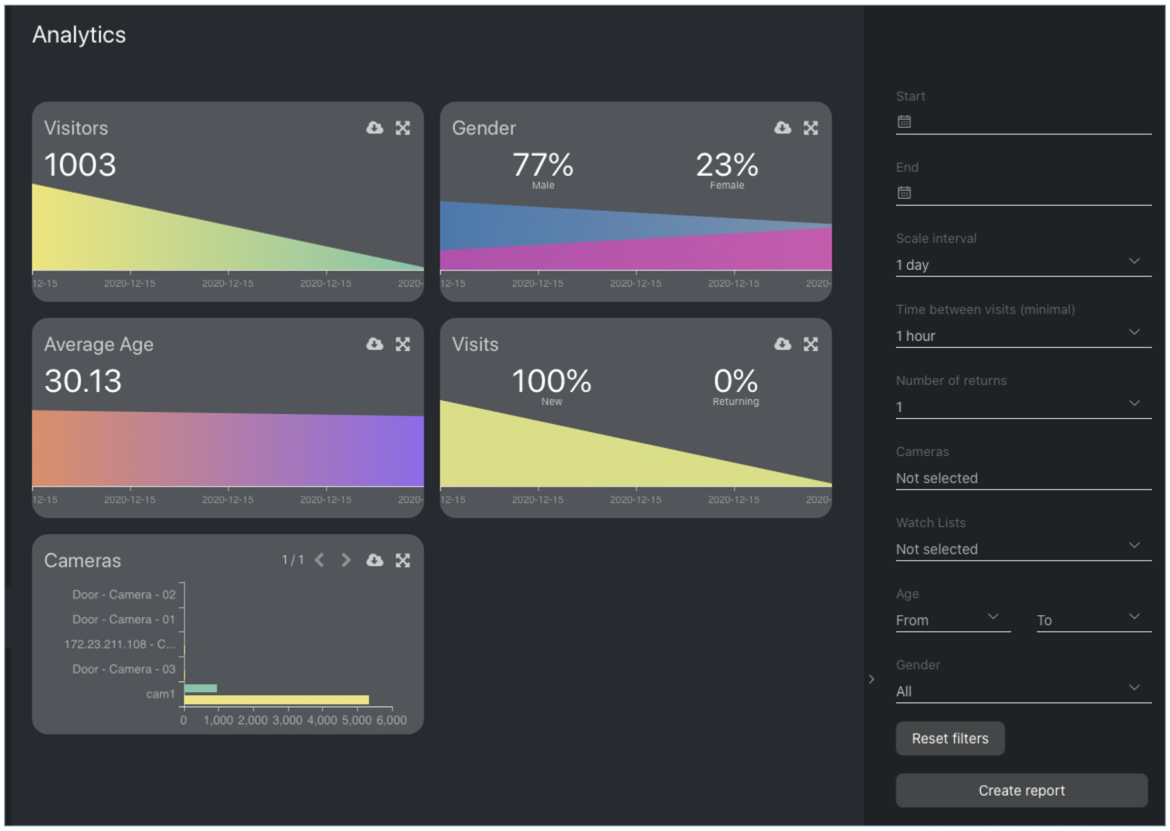
Relations threshold, s: 3

Video Analytics

Video analytics provided by FindFace Multi includes statistics on the number of visitors, their gender, average age, most frequently visited zones (judged by most active cameras), and the character of visits (first-timers or returners). It is a great starter tool to incorporate the know your customer guidelines into your business.

The analytical data charts are available on the *Analytics* tab.

Important: The analytics is built only when the *person recognition* is enabled.



To work with the analytical data, use the following filters:

- Time period
- Scale interval
- Time between visits
- Number of returns
- Cameras
- Watch lists
- Age
- Gender

See also:

- *Person Recognition*

2.2.11 Reports

FindFace Multi allows you to build reports on the following system entities:

- object recognition events
- episodes
- search events
- persons
- cameras
- dossiers
- analytical data
- audit logs

In this chapter:

Configure Saving Images in Reports

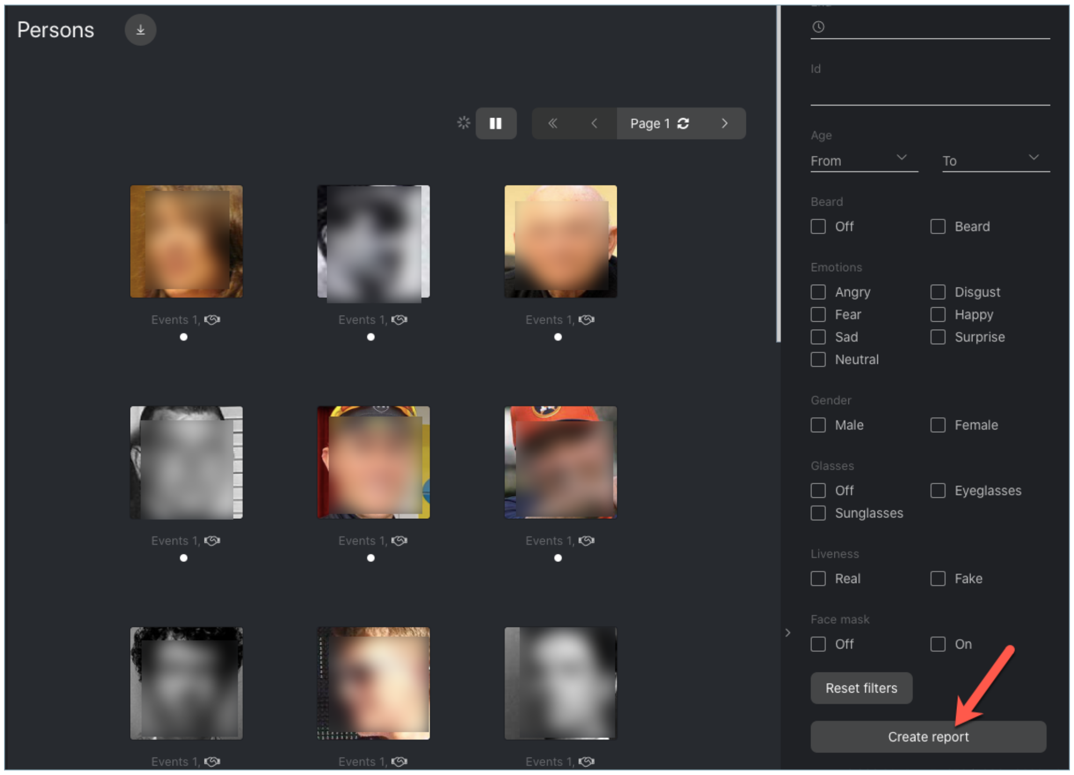
When building a report, you will be able to choose to save the report images as links, thumbnails, or full frames. It is possible to configure the image parameters. To do so, open the `/etc/findface-security/config.py` configuration file and alter the default JPEG quality and the maximum height of thumbnails and full frames, subject to your free disc space.

```
sudo vi /etc/findface-security/config.py

# reports image saving options
'REPORT_THUMBNAIL_JPEG_QUALITY': 75,
'REPORT_THUMBNAIL_MAX_HEIGHT': 100,
'REPORT_FULLFRAME_JPEG_QUALITY': 75,
'REPORT_FULLFRAME_MAX_HEIGHT': 250,
```

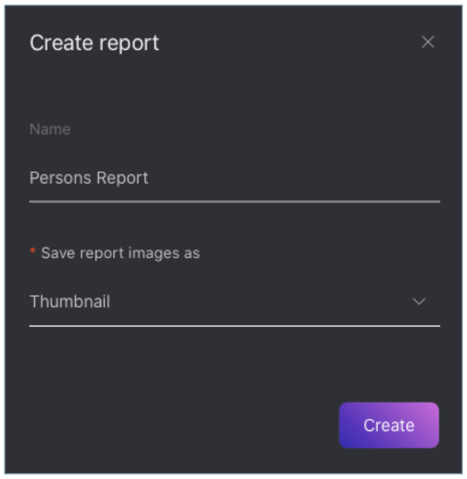
Build Report

To build a report, do the following:

1. Navigate to the tab associated with the required entity: *Events*, *Episodes*, *Search*, *Persons*, *Cameras*, *Dossiers*, *Analytics*, *Audit Logs*.
2. Set the filters for the report.
3. Click *Create Report*.



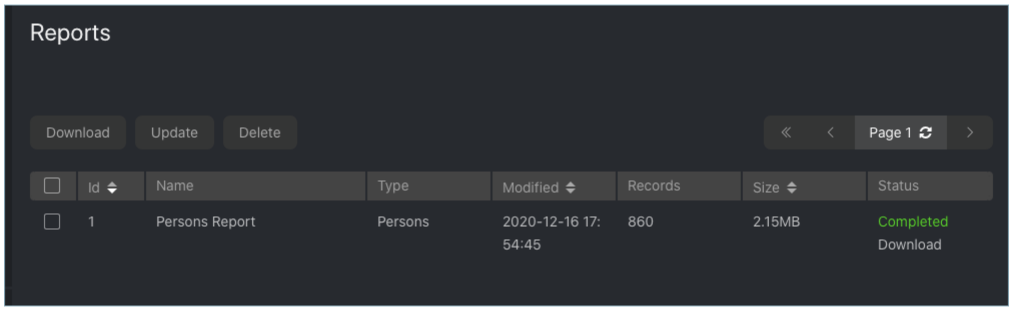
4. Specify the report name. Choose whether to save the report images as links, thumbnails, or full frames. Click *Create*.



Name: Persons Report

* Save report images as: Thumbnail

5. The report will be available for download on the *Reports* tab.


Reports

	Id	Name	Type	Modified	Records	Size	Status
☐	1	Persons Report	Persons	2020-12-16 17:54:45	860	2.15MB	Completed Download

2.2.12 Video Wall

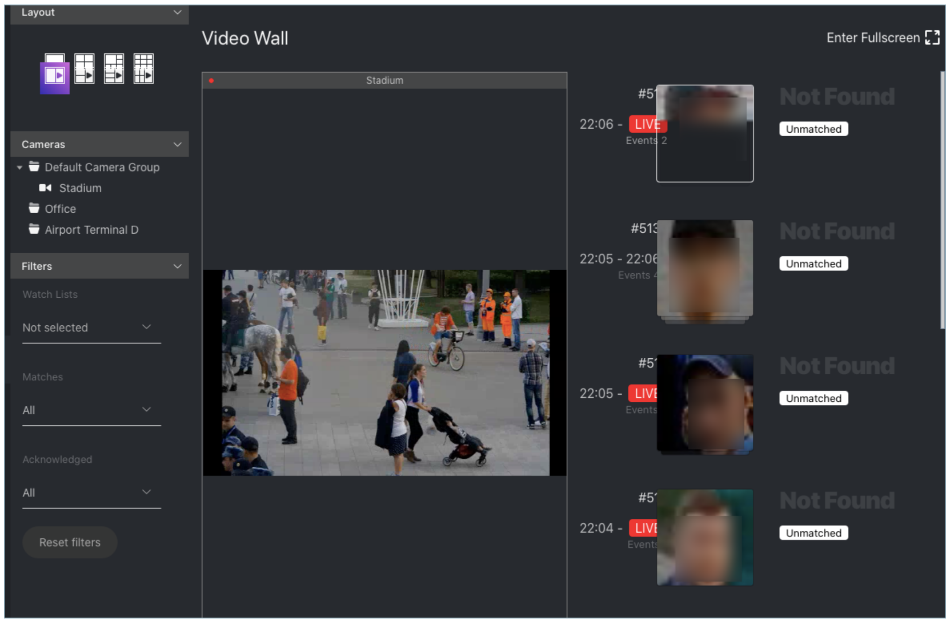
FindFace Multi allows for basic video surveillance. Use the Video Wall to display the video image from cameras and video files.

The Video Wall offers two modes, 4 predefined layouts in each:

- video streaming,
- video streaming with face detection and episode feed.

To display video on the Video Wall, do the following:

1. Navigate to the *Video Wall* tab.
2. Select a Video Wall mode and camera layout.



Watch Lists: Not selected

Matches: All

Acknowledged: All

3. Drag-n-drop cameras of your choice to the Video Wall.

You can work with the episode feed on the Video Wall in the *same manner* as with the *Episodes* tab, including the following basic filters:

- *Watch Lists*
- *Matches*.

- *Acknowledged*.

2.3 Advanced Functionality

2.3.1 Allocate `findface-video-worker` to Camera Group

In a distributed architecture, it is often necessary that video streams from a group of cameras be processed *in situ*, without being redistributed across remote `findface-video-worker` instances by the principal server.

Note: Among typical use cases are hotel chains, chain stores, several security checkpoints in the same building, etc.

In this case, allocate the local `findface-video-worker` to the camera group.

Do the following:

1. Navigate to the *Preferences* tab. Click *Camera Groups*.
2. Open the camera group settings.
3. In the *Labels*, create or select one or several allocation labels. Save changes.
4. Open the `/etc/findface-video-worker-cpu.ini` (`/etc/findface-video-worker-gpu.ini`) configuration file and specify the allocation labels in the following format: `label_name=true` (label `terminal_1` in the example below).

   ```
   sudo vi /etc/findface-video-worker-cpu.ini
   sudo vi /etc/findface-video-worker-gpu.ini

   labels = terminal_1=true
   ```

5. Restart `findface-video-worker`.

   ```
   sudo systemctl restart findface-video-worker-cpu.service
   sudo systemctl restart findface-video-worker-gpu.service
   ```

Note: If a camera is assigned an allocation label, its video stream can be processed by a `findface-video-worker` instance with the same label, as well as by all unlabeled `findface-video-worker` instances.

Warning: If a labeled camera is processed by an unlabeled `findface-video-worker` instance and a free similar-labeled instance appears, the camera won't automatically switch to the latter. To switch the camera, restart the similar-labeled `findface-video-worker` instance.

2.3.2 Distributed Dossier Database

In a distributed architecture, it is often necessary to have the dossier database distributed among several hosts.

In the current implementation, the dossier database is available for editing only on the principal server known as `master`. It is in sync with several additional FindFace Multi instances that serve as `slaves`. On the slaves, the dossier database is available only for reading and monitoring.

Important: You will be able to delete dossiers on the slaves if the master is unavailable.

Important: If a watch list on the future slave already contains dossiers, synchronization will be canceled. Make sure that the watch list is empty.

Warning: Neural networks on the master and slaves must be identical.

In this section:

Configure Master/Slave Synchronization

To configure master/slave synchronization, do the following:

1. On the master, open the `/etc/findface-security/config.py` configuration file. Come up with a synchronization token and specify it in the `SYNC_TOKEN` parameter (be sure to uncomment it prior).

```
sudo vi /etc/findface-security/config.py

...

# ========== DossierLists sync ===========
...
# token must be identical on master and slave
# use pwgen -s 64 1
SYNC_TOKEN = 'ABC_123456789'
...
```

2. On the slave(s), uncomment the `SYNC_TOKEN` parameter in the `/etc/findface-security/config.py` configuration file and paste the created synchronization token into it. The tokens on the master and slaves must be identical.

The master/slave sync is now set and will be enabled once you configure a watch list replication from the master to slave(s).

Replicate Watch List from Master to Slaves

To replicate a watch list from the master to slave instances, send a POST request to the slave with the following parameters in the body:

- `remote_dossier_list`: id of the original watch list on the master
- `remote_url`: master URL
- `slave_dossier_list`: id of the watch list on the slave, which is to be a replica of the original watch list

```
POST /sync/dossier-lists/
{remote_dossier_list: 1,
remote_url: "http://172.17.46.14",
slave_dossier_list: 3}
```

Set Synchronization Time

To schedule dossier synchronization, do the following:

1. Open the `/etc/findface-security/config.py` configuration file on the master.

   ```
   sudo vi /etc/findface-security/config.py
   ```

2. Uncomment and set the following parameters:

   - `SYNC_SCHEDULE`: recurrence rule (RRULE) that defines the sync schedule.

     Tip: See the RRULE calculator here.

   - `SYNC_AT_STARTUP`: if `True`, synchronization occurs on the FindFace Multi startup and restart.
   - `SYNC_AT_CREATION`: if `True`, synchronization immediately occurs after you set up synchronization for a watch list.

   ```
   ...

   # ========== DossierLists sync ===========
   ...

   # rrule that defines sync schedule
   SYNC_SCHEDULE = 'RRULE:FREQ=DAILY;WKST=MO;BYHOUR=11;BYMINUTE=0'
   # if True synchronization will occur on FindFace Security startup and restart
   SYNC_AT_STARTUP = True
   # if True synchronization will occur immediately after creating synchronization for␣
   ↪dossier list
   SYNC_AT_CREATION = True
   ```

3. Uncomment the mentioned above parameters on each slave. The parameter values can be arbitrary.

Cancel Watch List Replication and Synchronization

To cancel a watch list replication and synchronization, send the following API request to the slave with the {id} of the watch list on the slave:

```
DELETE /sync/dossier-lists/{id}/
```

Duplicate Functionality to Web Interface

By default, you can enable and disable watch list replication only via API. To make the functionality available in the web interface as well, do the following:

1. Open the /etc/findface-security/config.py configuration file on the master.

   ```
   sudo vi /etc/findface-security/config.py
   ```

2. Enable the ffsecurity_sync plugin by uncommenting the line INSTALLED_APPS.append('ffsecurity_sync') into the plugins section:

   ```
   ...

   # ========== DossierLists sync ===========
   INSTALLED_APPS.append('ffsecurity_sync')
   ...
   ```

3. Do the same on each slave.

4. On each host, migrate the main database architecture from FindFace Multi to **PostgreSQL**. Re-create user groups in the main database. Restart the findface-security service.

   ```
   sudo findface-security migrate
   sudo findface-security create_groups
   sudo systemctl restart findface-security.service
   ```

2.3.3 Dossier Custom Tabs, Fields, and Filters

It is often necessary that a dossier feature additional tabs and fields in the web interface.

See also:

To create dossier custom fields in your Tarantool-based feature vector database, see *Dossier Face Custom Metadata in Tarantool*.

To add custom tabs and fields to a dossier, do the following:

1. Prepare the list of custom tabs and fields you want to add to a dossier.

2. Open the /etc/findface-security/config.py configuration file.

   ```
   sudo vi /etc/findface-security/config.py
   ```

3. Into the FFSECURITY section, uncomment the CUSTOM_FIELDS section and modify the exemplary content, considering the following:

   - 'items': the list of fields in a dossier. Describe each field with the following parameters:

     – 'name': field's internal name, string.

  - 'label': field's label in the web interface, string.
  - 'display': display format (form or list), string or array.
  - 'tab': tab that features the field. If not specified, the field appears on the main dossier page (that with a photograph).
  - 'editable': field's editability, boolean.
  - 'type': field data type, string. Possible values:
    * list: requires items, additional parameter for lists (see below), expects objects {id, name} in dictionaries;
    * valuelist: expects elements of primitive types.
    * objectlist: allows for creating arrays of objects of required types.
    * datetime: primitive data type displayed as a datetime list.
    * date: primitive data type displayed as a date picker.
    * boolean: primitive data type displayed as a checkbox.
    * string: primitive data type string.
  - additional parameters for lists (type=list, type=valuelist):
    * multiple: possibility of selecting several items in the list, boolean.
    * items: dictionary used as a data source for the list.
    * allow_create: possibility of adding new items to the list.
    * custom_id: custom field for id (type=list).
  - additional parameters for object lists (type=objectlist).
    * object: objects used as a data source for the object list.
    * simple: indicator that the field expects data of a primitive type instead of objects, for example, expects strings with phone numbers.
- 'filters': the list of search filters associated with the custom fields. Parameters:
  - 'name': filter's internal name,
  - 'label': filter's label in the web interface,
  - 'field': associated field in the format [field name].
- 'tabs': the list of tabs in a dossier. The first listed tab corresponds to the main dossier page.

```
FFSECURITY = {

...

    # Edit CUSTOM_FIELDS section to customize dossier content.
    # Below is an example for integration FindFace Security with Sigur.
    'CUSTOM_FIELDS': {
        'dossier_meta': {
            'items': [
                {
                    'name': 'personid',
                    'default': '',
```

(continues on next page)

(continued from previous page)

```
                    'label': 'PersonID',
                    'display': ['list', 'form'],
                    'description': 'Sigur person ID'
                },
                {
                    'name': 'firstname',
                    'default': '',
                    'label': 'First Name',
                    'display': ['list', 'form'],
                    'description': 'Sigur first name'
                },
                {
                    'name': 'lastname',
                    'default': '',
                    'label': 'Last Name',
                    'display': ['list', 'form'],
                    'description': 'Sigur last name'
                },
                {
                    'name': 'version',
                    'default': '',
                    'label': 'Version',
                    'display': ['list', 'form'],
                    'description': 'Sigur photo version'
                }
            ],
            'filters': [
                {
                    'name': 'personid',
                    'label': 'Sigur person ID filter',
                    'field': 'personid'
                }
            ]
        }
    },
```

4. Restart the `findface-security` service.

```
sudo systemctl restart findface-security.service
```

5. You will see the custom content appear in the web interface.

2.3.4 Dossier Face Custom Metadata in Tarantool

It is often necessary to assign additional metadata to the dossier faces in your Tarantool-based feature vector database.

See also:

To create dossier custom tabs, fields, and filters in the web interface, see *Dossier Custom Tabs, Fields, and Filters*.

To set the face custom metadata, do the following:

1. Prepare the list of custom meta fields you want to assign to a dossier face in Tarantool.
2. Open the `/etc/findface-security/config.py` configuration file.

```
sudo vi /etc/findface-security/config.py
```

3. Into the FFSECURITY section, uncomment the CUSTOM_FIELDS -> face_object section and modify the exemplary content, considering the following:
   - field_name: field name;
   - type: data type;
   - default: field default value. If a default value exceeds '1e14 – 1', use a string data type to specify it, for example, "123123.." instead of 123123...

```
FFSECURITY = {

...

'CUSTOM_FIELDS': {
  ...
        'face_object': {
            'items': [
                {
                    "field_name": "tag_name_1",
                    "type": "string",
                    "default": "change_me"
                },
                {
                    "field_name": "tag_name_2",
                    "type": "uint",
                    "default": 123
                },
                {
                    "field_name": "tag_name_3",
                    "type": "bool",
                    "default": True
                },
            ]
        }
},
```

4. *Add the new meta fields* to the Tarantool database structure.
5. You can work with the new meta fields through *HTTP API* using the objects/faces/ methods.

2.3.5 Console Bulk Photo Upload

To bulk-upload photos to the dossier database, you can use the **findface-security-uploader** utility from the FindFace Multi package (in addition to the web interface upload functionality). Use this utility when you need to upload a large number of photos (more than 10,000).

Warning: In the current version, the **findface-security-uploader** utility does not support cars and bodies, only faces.

Tip: To view the **findface-security-uploader** help, execute:

```
findface-security-uploader --help
```

Do the following:

1. Write the list of photos and metastrings to a CSV or TSV file.

   Important: The file used as a metadata source must have the following format: `path to photo | metastring`.

   To prepare a TSV file, use either a `script` or the `find` command.

   Note: Both the script and the command in the examples below create the `images.tsv` file. Each image in the list will be associated with a metastring coinciding with the image file name in the format `path to photo | metastring`.

   To build a TSV file listing photos from a specified directory (`/home/user/25_celeb/` in the example below), run the following command:

   ```
   python3 tsv_builder.py /home/user/25_celeb/
   ```

   The `find` usage example:

   ```
   find photos/ -type f -iname '*g' | while read x; do y="${x%.*}"; printf "%s\t%s\n" "
   ↪$x" "${y##*/}"; done
   ```

2. Create a job file out of a CSV or TSV file by using `add`. As a result, a file `enroll-job.db` will be created and saved in a current directory.

   ```
   findface-security-uploader add images.tsv
   ```

   The `add` options:

   - `--format`: input file format, `tsv` by default,
   - `--delimiter`: field delimiter, by default `"\t"` for TSV, and `","` for CSV.

   Note: A job file represents a sqlite database which can be opened on the **sqlite3** console.

3. Process the job file by using `run`.

   ```
   findface-security-uploader run --dossier-lists 2 --api http://127.0.0.1:80 --user␣
   ↪admin --password password
   ```

   The important `run` options:

   - `--parallel`: the number of photo upload threads, 10 by default. The more threads you use, the faster the bulk upload is completed, however it requires more resources too.
   - `--all-faces`: upload all faces from a photo if it features several faces.
   - `--api`: `findface-security` API URL, `http://127.0.0.1:80/` by default. Mandatory option.

- `--user`: login. Mandatory option.
- `--password`: password. Mandatory option.
- `--dossier-lists`: comma-separated list of the watch lists id's. Mandatory option.
- `--failed`: should an error occur during the job file processing, correct the mistake and try again with this option.
- `--inactive`: mark new dossiers as inactive.
- `--noface`: include images without detection.

2.3.6 Deduplicate Events

In this section:

Consider enabling Deduplication to exclude coinciding object recognition events within one camera group.

Enable Deduplication

To enable event deduplication, do the following:

1. Enable the offline video detection mode for each camera in the group. See *Add Camera* for details.
2. Navigate to the *Preferences* tab. Click *Camera Groups*.
3. Open the camera group settings.
4. Check *Deduplicate Events* and specify the deduplication interval in seconds.

How It Works

The deduplication algorithm works as follows. In the offline mode, the server receives one best object snapshot per tracking session on a camera.

Note: A tracking session continues until an object disappears from the camera field.

If there are several tracking sessions on a camera(s) of a camera group within the specified deduplication interval, FindFace Multi handles the received snapshots in the following way:

- If there is a match with a dossier within the preceding deduplication interval, FindFace Multi drops a newly acquired snapshot. Otherwise, it saves the snapshot to the database.
- For unmatched objects, FindFace Multi considers both the similarity between objects and snapshot quality when performing deduplication. As a result, FindFace Multi drops all snapshots within the deduplication interval unless a new object snapshot is of higher quality. Thus, it guarantees the system deduplicates events without skipping high-quality objects, which are essential for further video analytics.

2.3.7 Liveness Detection as Standalone Service

See also:

Real-time Face Liveness Detection

Besides the *integrated* anti-spoofing system that distinguishes a live face from a face image, FindFace Multi provides an API-based face liveness detection service `findface-liveness-api`.

The `findface-liveness-api` service takes a specific number of frames from a video fragment and returns the best quality face, and decimal liveness result averaged across the taken frames. If configured, the service can also return full-frame and normalized face images and save the detection result in the `findface-sf-api` cache, returning `detection_id`.

FindFace Multi uses `findface-liveness-api` for face-based *authentication*. If needed, you can install and use this service standalone.

In this section:

Install and Configure `findface-liveness-api`

The `findface-liveness-api` service is automatically installed with FindFace Multi.

To install the service standalone, install the FindFace Multi *APT repository* and execute the following commands:

```
sudo apt update
sudo apt install findface-liveness-api
```

You can configure the `findface-liveness-api` parameters in the `/etc/findface-liveness-api.ini` configuration file:

```
sudo vi /etc/findface-liveness-api.ini

listen: :18301
liveness-threshold: 0.95
fullframe-jpeg-quality: 75
max-decoded-frames: 30
min-selected-frames: 10
mf-selector: reject
extraction-api:
  request-batch-size: 16
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  extraction-api: http://127.0.0.1:18666
sf-api:
  timeouts:
```

(continues on next page)

(continued from previous page)

```
    connect: 5s
    response_header: 30s
     overall: 35s
    idle_connection: 10s
  sf-api: http://127.0.0.1:18411
limits:
  video-size: 10485760
  video-length-sec: 60
  video-fps: 30
  video-width-px: 1920
  video-height-px: 1080
```

Parameter	Description
`fullframe-jpeg-quality`	JPEG quality of full frames in the `photo` field.
`max-decoded-frames`	Finish decoding after reaching the specified number of frames.
`min-selected-frames`	The minimum number of final frames successfully passed through decoding and liveness extraction. Must be equal or less than `max-decoded-frames`.
`mf-selector`	Service behavior upon having multiple faces in the video frame: `reject` - reject this frame, `biggest` - use the biggest face for liveness detection.
`extraction-api` -> `request-batch-size`	Batch size for liveness extraction.
`limits` -> `video-size`	Maximum video size, bytes.
`limits` -> `video-length-sec`	Maximum video length, seconds.
`limits` -> `video-fps`	Maximum video FPS.
`limits` -> `video-width-px`	Maximum video width, pixels.
`limits` -> `video-height-px`	Maximum video height, pixels.

To start the `findface-liveness-api` service and enable its autostart, execute:

```
sudo systemctl enable findface-liveness-api.service && sudo systemctl start findface-
↪liveness-api.service
```

Liveness API Usage

To interact with the `findface-liveness-api` service, use HTTP API requests. In the example below, the POST request is sent with the following optional parameters:

- `return_detection` (`default=False`): save the best face in the `findface-sf-api` cache and return its `detection_id`.
- `return_normalized` (`default=False`): return the face normalized image in the `normalized` field.
- `return_photo` (`default=False`): return the full frame in the `photo` field.

Example

Request

```
curl -i -X POST \
  'http://127.0.0.1:18301/v1/video-liveness?return_detection=true&return_normalized=true&
↪return_photo=true' \
  -H 'Content-Type: video/mp4' \
  --data-binary '@/home/my_video.mp4'
```

Response

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: LA:WSP2NcHc
Date: Mon, 07 Sep 2020 15:30:05 GMT
Transfer-Encoding: chunked
{
        "alive": true,
        "average_liveness": 0.9706386,
        "best_face": {
                "liveness": 0.97768883,
                "quality": 0.89638597,
                "bbox": {
                        "left": 0,
                        "top": 578,
                        "right": 307,
                        "bottom": 1154
                },
                "detection_id": "btb53vbp688s1njt3bv0",
                "photo": "/9j/2wCEAAgGBgcGBQgHBwcJ...",
                "normalized": "iVBORw0KGgoAAAANSUhEU...",
                "frame_no": 1,
                "frame_ts": 0.033275817
        }
}
```

2.3.8 Multiple Video Cards Usage

Should you have several video cards installed on a physical server, you can create additional `findface-extraction-api-gpu` or `findface-video-worker-gpu` instances and distribute them across the video cards, one instance per card.

In this section:

Distribute `findface-extraction-api-gpu` Instances Across Several Video Cards

To distribute `findface-extraction-api-gpu` instances across several video cards, do the following:

1. Stop the initial `findface-extraction-api-gpu` service.

```
sudo service findface-extraction-api stop
```

2. Create several copies of the `/etc/findface-extraction-api.ini` configuration file, subject to how many video cards you are going to use for feature vector extraction. Append the appropriate GPU device numbers to the new configuration files names as shown in the example below (GPU devices #0 and #6).

```
/etc/findface-extraction-api@0.ini
/etc/findface-extraction-api@6.ini
```

3. Open the new configuration files. Specify the GPU device numbers and adjust the listening ports.

```
sudo vi /etc/findface-extraction-api@0.ini

listen: 127.0.0.1:18666
...

gpu_device: 0
...
```

```
sudo vi /etc/findface-extraction-api@6.ini

listen: 127.0.0.1:18667
...

gpu_device: 6
...
```

4. Start the new services.

```
sudo service findface-extraction-api@0 start
sudo service findface-extraction-api@6 start
```

Allocate `findface-video-worker-gpu` to Additional Video Card

To create an additional `findface-video-worker-gpu` instance and allocate it to a different video card, do the following:

1. Display the status of the `findface-video-worker-gpu` primary service by executing:

   ```
   sudo systemctl status findface-video-worker-gpu.service
   ```

2. Find the full path to the service in the following line:

   ```
   Loaded: loaded (/usr/lib/systemd/system/findface-video-worker-gpu.service); enabled;
   ↪ vendor preset: enabled
   ```

   It is `findface-video-worker-gpu.service` in our example (name may vary). Create a copy of the service under a new name.

   ```
   sudo cp /usr/lib/systemd/system/findface-video-worker-gpu.service /usr/lib/systemd/
   ↪system/findface-video-worker-gpu2.service`
   ```

3. In the same manner, create a copy of the primary service configuration file under a new name.

   ```
   sudo cp /etc/findface-video-worker-gpu.ini /etc/findface-video-worker-gpu2.ini
   ```

4. Open the just created configuration file and actualize the GPU device number to use. Modify the streamer port number by the following formula: `18999 (port number for GPU #0) - GPU device number`, i.e. for the GPU #1, `port = 18998`, for the GPU #2, `port = 18997`, and so on.

   ```
   sudo vi /etc/findface-video-worker-gpu2.ini

   ## cuda device number
   device_number = 1

   ...

   #------------------------------
   [streamer]
   #------------------------------
   ## streamer/shots webserver port, 0=disabled
   ## type:number env:CFG_STREAMER_PORT longopt:--streamer-port
   port = 18999
   ...
   ```

5. Open the new service and specify the just created configuration file.

   ```
   sudo vi /usr/lib/systemd/system/findface-video-worker-gpu2.service

   ExecStart=/usr/bin/findface-video-worker-gpu --config /etc/findface-video-worker-
   ↪gpu2.ini
   ```

6. Reload the `systemd` daemon to apply the changes.

   ```
   sudo systemctl daemon-reload
   ```

7. Enable the new service autostart.

```
sudo systemctl enable findface-video-worker-gpu2.service

Created symlink from /etc/systemd/system/multi-user.target.wants/findface-video-
↪worker-gpu2.service to /usr/lib/systemd/system/findface-video-worker-gpu2.service
```

8. Launch the new service.

```
sudo systemctl start findface-video-worker-gpu2.service
```

9. Check the both `findface-video-worker-gpu` services status.

```
sudo systemctl status findface-video-worker-* | grep -i 'Active:' -B 3

 findface-video-worker-gpu2.service - findface-video-worker-gpu daemon
  Loaded: loaded (/usr/lib/systemd/system/findface-video-worker-gpu2.service;␣
↪enabled; vendor preset: enabled)
  Active: active (running) since Thu 2019-07-18 10:32:02 MSK; 1min 11s ago

...

 findface-video-worker-gpu.service - findface-video-worker-gpu daemon
  Loaded: loaded (/usr/lib/systemd/system/findface-video-worker-gpu.service;␣
↪enabled; vendor preset: enabled)
  Active: active (running) since Mon 2019-07-15 15:18:33 MSK; 2 days ago
```

2.3.9 Direct API Requests to Tarantool

You can use HTTP API to extract object data (faces, bodies, cars) directly from the Tarantool Database.

Note: In the current implementation, Tarantool operates objects as faces. For example, to add an object, send `POST /:ver/faces/add/:name`.

In this section:

General Information

API requests to Tarantool are to be sent to `http://<tarantool_host_ip:port>`.

Tip: The port for API requests can be found in the `FindFace.start` section of the Tarantool configuration file `/etc/tarantool/instances.available/*.lua`:

```
cat /etc/tarantool/instances.available/*.lua

##8101:
FindFace.start("127.0.0.1", 8101)
```

Note: In the case of the standalone deployment, you can access Tarantool by default only locally (127.0.0.1). If you want to access Tarantool remotely, *alter* the Tarantool configuration file (`/etc/tarantool/instances.available/*.lua`).

API requests to Tarantool may contain the following parameters in path segments:

- `:ver`: API version (v2 at the moment).
- `:name`: gallery name.

By default, there are the following galleries in the Tarantool database:

- `ffsec_body_events`: feature vectors extracted from bodies detected in the video.
- `ffsec_body_objects`: feature vectors extracted from bodies in dossier photos.
- `ffsec_car_events`: feature vectors extracted from cars detected in the video.
- `ffsec_car_objects`: feature vectors extracted from cars in dossier photos.
- `ffsec_face_events`: feature vectors extracted from faces detected in the video.
- `ffsec_face_objects`: feature vectors extracted from faces in dossier photos.
- `ffsec_user_face`: feature vectors extracted from the FindFace Multi users' photos, used for face-based authentication.
- `ffsec_persons`: centroids of persons (virtual feature vectors averaged across all person's faces) and metadata.

Tip: To list gallery names on a shard, type in the following command in the address bar of your browser:

```
http://<tarantool_host_ip:shard_port>/stat/list/1/99
```

The same command on the console is as such:

```
curl <tarantool_host_ip:shard_port>/stat/list/1/99 \| jq
```

You can also list gallery names by using a direct request to Tarantool:

```
echo 'box.space.galleries:select()' | tarantoolctl connect <tarantool_host_ip:shard_port>
```

Note that if there is a large number of shards in the system, chances are that a randomly taken shard does not contain all the existing galleries. In this case, just list galleries on several shards.

Add Object

```
POST /:ver/faces/add/:name
```

Parameters in body:

JSON-encoded array of objects with the following fields:

- "id": object id in the gallery, uint64_t,
- "facen": raw feature vector, base64,
- "meta": object metadata, dictionary.

Returns:

- HTTP 200 and empty body on success.
- HTTP 404 if a gallery with the given name doesn't exist.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8101/v2/faces/add/testgal' --data '
[
  {
    "id": 9223372036854776000,
    "facen": "qgI3vZRv/z...NpO9MdHavW1WuT0=",
    "meta": {
"cam_id": "223900",
"person_name": "Mary Ostin",

    }
  }
]
```

Response

```
HTTP/1.1 200 Ok
Content-length: 1234
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

Remove Object

```
POST /v2/faces/delete/:name
```

Parameters in body:

JSON-encoded array of object ids to be removed

Returns:

- HTTP 200 and empty body on success.
- HTTP 404 if an object with the given id is not found in the gallery.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D -  -s 'http://localhost:8101/v2/faces/delete/testgal' --data '[1, 4, 922, 3]'
```

Response

```
HTTP/1.1 200 Ok
Content-length: 111
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

Object Search

```
POST /v2/faces/search/:name
```

Parameters in body:

JSON-encoded search request with the following fields:

- `limit`: maximum number of objects in the response.
- `sort`: sorting order. Pass one of the following values: `id`: increasing order by id, `-id`: decreasing order by id, `-score`: decreasing order by object similarity (only if you search for objects with similar feature vectors).
- `filter` (filters):
  - `facen`: (optional) search for objects with similar feature vectors. Pass a dictionary with the following fields: `data`: raw feature vector, base64; `score`: range of similarity between objects [threshold similarity; 1], where 1 is 100% match.
  - `id` and `meta/<meta_key>`: search by object id and metastring content. To set this filter, use the following operators:

  * `range`: range of values, only for numbers.
  * `set`: id or metastring must contain at least one value from a given set, for numbers and strings.
  * `subset`: id or metastring must include all values from a given subset, for numbers and strings.
  * `like`: by analogy with `like` in SQL requests: only 'aa%', '%aa', and '%aa%' are supported. Only for strings and set[string]. In the case of set[string], the filter will return result if at least one value meets the filter condition.
  * `ilike`: by analogy with `like` but case-insensitive, only for strings and set[string].

Returns:

- JSON-encoded array with objects on success. The value in the `X-search-stat` header indicates whether the fast index was used for the search: `with_index` or `without_index`.

  Note: Fast index is not used in API v2.

- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8101/v2/testgal/search' --data '
{
        "limit": 2,
    "sort": {
                "score": -1
        },
        "filter": {
                "facen": {
                        "data": "qgI3vZRv/z0BQTk9rcirOyZrNpO9MdHavW1WuT0=",
                        "score": [0.75, 1]
                },
                "id": {
                        "range": [9223372036854000000, 9223372036854999000]
                },
                "meta": {
                        "person_id": {
                                "range": [444, 999]
                        },
                        "cam_id": {
                                "set": ["12767", "8632", "23989"]
                        }
                }
        }
}'
```

Response

```
HTTP/1.1 200 Ok
Content-length: 1234
X-search-stat: without_index
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{
  "results": [
    {
      "facen": " qgI3vZRv/z0BQTk9rcirOyZrNpO9MdHavW1WuT0=",
      "meta": {
        "timestamp": 0,
        "photo_hash": "",
        "person_id": 777,
        "cam_id": "8632"
      },
      "score": 0.9964,
      "id": 9223372036854776000
    }
  ]
}
```

Edit Object Metadata and Feature Vector

```
POST /v2/faces/update/:name
```

Parameters in body:

JSON-encoded array with objects with the following fields:

- `"id"`: object id, uint64_t.
- `"facen"`: (optional) new feature vector, base64. If omitted or passed as `null`, the relevant field in the database won't be updated.
- `"meta"`: dictionary with metadata to be updated. If some metastring is omitted or passed as `null`, the relevant field in the database won't be updated.

Returns:

- HTTP 200 and dictionary with all object parameters, including not updated, on success.
- HTTP 404 and error description if an object with the given id doesn't exist.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8101/v2/faces/update/sandbox' --data '[{"id":1,"facen
↪":null,"meta":{"m:timestamp":1848}}]'
```

Response

```
HTTP/1.1 200 Ok
Content-length: 151
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"meta":{"m:timestamp":1848,"normalized_id":"1_b9pkrf00mjt6h1vmq1kg.png","m:cam_id":
↪"a9f7a973-f07e-469d-a3bd-41ddd510b26f","feat":"{\"score\":0.123}"}, "id":1, ... }
```

List Galleries

```
POST /v2/galleries/list
```

Returns:

JSON-encoded array with galleries with the following fields: `name`: gallery name, `faces`: number of objects in a gallery.

Example

Request

```
curl -D - -s -X POST http://localhost:8101/v2/galleries/list
```

Response

```
HTTP/1.1 200 Ok
Content-length: 42
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{
  "results": [
    {
      "name": "testgal",
      "faces": 2
    }
  ]
}
```

Get Gallery Info

```
POST /v2/galleries/get/:name
```

Returns:

- HTTP 200 and dictionary with gallery parameters on success.
- HTTP 404 and error description if a gallery with the given name doesn't exist.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s -X POST http://localhost:8101/v2/galleries/get/testgal
```

```
HTTP/1.1 200 Ok
Content-length: 11
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"faces":2}
```

Create Gallery

```
POST /v2/galleries/add/:name
```

Returns:

- HTTP 200 and empty body on success.
- with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/add/123'
```

Response

```
HTTP/1.1 409 Conflict
Content-length: 57
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"error":{"message":"gallery already exists","code":409}}
```

Remove Gallery

```
POST /v2/galleries/delete/:name
```

Returns:

- HTTP 200 and empty on success.
- with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/delete/123'
```

Response

```
HTTP/1.1 204 No content
Content-length: 0
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

2.3.10 Enable Personal Data Protection

FindFace Multi supports laws related to the processing of personal data of individuals (GDPR and similar).

To apply personal data protection to your system, do the following:

1. Open the `/etc/findface-security/config.py` configuration file.

   ```
   sudo vi /etc/findface-security/config.py
   ```

2. Disable saving unmatched events by setting `'IGNORE_UNMATCHED': True`.

   ```
   ...

   FFSECURITY = {
       ...
   ```

(continues on next page)

(continued from previous page)

```
    # do not save unmatched events (GDPR support)
    'IGNORE_UNMATCHED': False,

    ...

}
```

3. For events with matches, enable blurring all unmatched faces in full frames. To do so, set `'BLUR_UNMATCHED_FACES': True`. Optionally, you can modify the default JPEG quality of those frames.

```
...

FFSECURITY = {
    ...

    # blur all unmatched faces on the full frame of the matched event (GDPR support)
    'BLUR_UNMATCHED_FACES': False,

    # full frame jpeg quality when `BLUR_UNMATCHED_FACES` is enabled
    'BLURRED_FULLFRAME_JPEG_QUALITY': 85,
    ...

}
```

4. Restart `findface-security`.

```
sudo systemctl restart findface-security.service
```

CHAPTER THREE

INTEGRATIONS

This chapter is all about integration with FindFace Multi. Integrate your system through HTTP API and webhooks, or check out our turnkey partner integrations.

3.1 HTTP API

Detailed interactive documentation on the FindFace Multi HTTP API is available after installation at `http://<findface-security-ip:port>/api-docs`. Learn and try it out.

Tip: You can also find it by navigating to *Preferences* -> *API Documentation* in the web interface.

3.2 Webhooks

You can set up FindFace Multi to automatically send notifications about specific events, episodes, and counter records to a given URL. To do so, create and configure a webhook. In this case, when such an event, episode, or counter record occurs, FindFace Multi will send an HTTP request to the URL configured for the webhook.

You can use webhooks for various purposes, for instance, to notify a user about a specific event, invoke required behavior on a target website, and solve security tasks such as automated access control.

In this section:

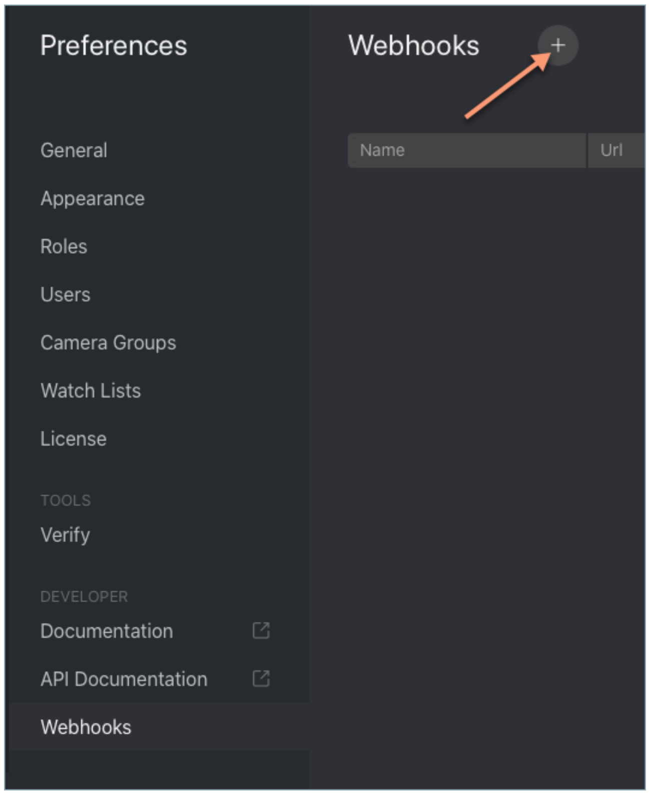
3.2.1 Configure Webhook

Important: You need Administrator privileges to create a webhook.

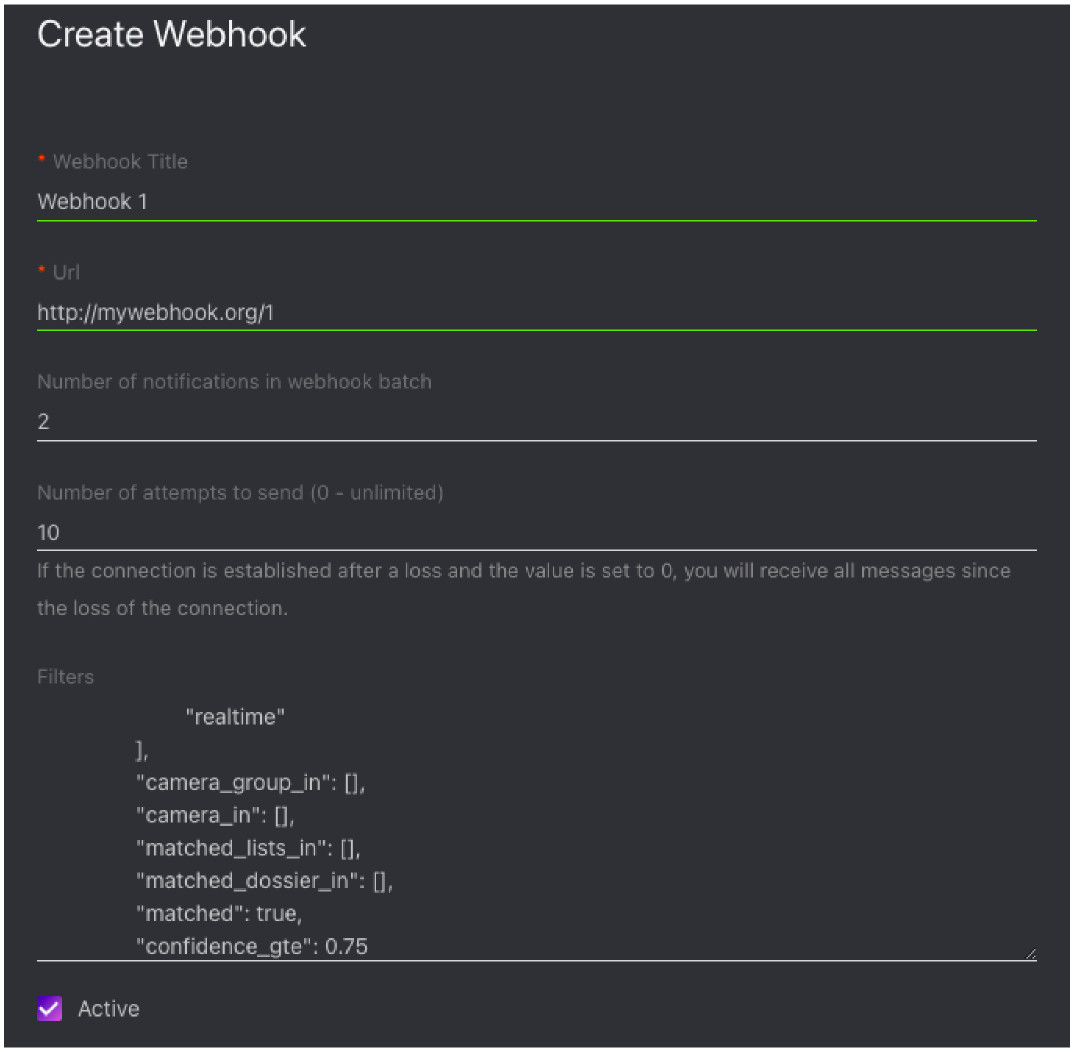
Note: To use the webhooks, make sure that at least one of the following parameters is specified in `/etc/findface-security/config.py`: `SERVICE_EXTERNAL_ADDRESS` or `EXTERNAL_ADDRESS`.

To create and configure a webhook, do the following:

1. Navigate to the *Preferences* tab. Click *Webhooks*.
2. Click +.



3. Specify the webhook title.



Create Webhook

* Webhook Title

Webhook 1

* Url

http://mywebhook.org/1

Number of notifications in webhook batch

2

Number of attempts to send (0 - unlimited)

10

If the connection is established after a loss and the value is set to 0, you will receive all messages since the loss of the connection.

Filters

```
            "realtime"
        ],
        "camera_group_in": [],
        "camera_in": [],
        "matched_lists_in": [],
        "matched_dossier_in": [],
        "matched": true,
        "confidence_gte": 0.75
```

✓ Active

4. Specify URL to automatically send notifications to.
5. You can send notifications in batches. Specify the maximum number of notifications in a webhook batch. The actual number may be less.
6. Specify the maximum number of attempts to send a notification. The interval between attempts increases exponentially with a maximum of 100 seconds.

   Important: To receive all messages since the connection loss, should it happen, set `0`. Set 1 to omit old messages.

7. FindFace Multi will be automatically sending notifications on events, episodes, and counters which match given filters. The following filters are available:

Recognition events (face, body, car):

- allowed_bs_types: *object tracking mode*, possible values: overall, realtime.
- camera_group_in: camera group id, number.
- camera_in: camera id, number.
- matched_lists_in: watch list id, number.
- matched_dossier_in: matched dossier id, number.
- matched: event matched status (true or false), boolean.
- confidence_gte: minimum confidence, number.

Episodes:

- allowed_types: episode status, possible values: an episode opening (episode_open), adding a new event into an episode (episode_event), an episode closing (episode_close).
- camera_group_in: camera group id, number.
- camera_in: camera id, number.
- matched_lists_in: watch list id, number.
- matched: event matched status (true or false), boolean.
- events_count_gte: minimum number of events in an episode, number.
- events_count_lte: maximum number of events in an episode, number.

Counters:

- counter_in: counter id, number
- camera_group_in: camera group id, number.
- camera_in: camera id, number
- faces_gte: minimum number of faces in a counter record, number.
- faces_lte: maximum number of faces in a counter record, number.
- silhouettes_gte: minimum number of silhouettes in a counter record, number.
- silhouettes_lte: maximum number of silhouettes in a counter record, number.

```
{
      "face_events": {
            "allowed_bs_types": [
                  "overall",
                  "realtime"
            ],
            "camera_group_in": [],
            "camera_in": [],
            "matched_lists_in": [],
            "matched_dossier_in": [],
            "matched": true,
            "confidence_gte": 0.75
```

(continues on next page)

(continued from previous page)

```
        },
        "body_events": {
                "allowed_bs_types": [
                        "overall",
                        "realtime"
                ],
                "camera_group_in": [],
                "camera_in": [],
                "matched_lists_in": [],
                "matched_dossier_in": [],
                "matched": true,
                "confidence_gte": 0.75
        },
        "car_events": {
                "allowed_bs_types": [
                        "overall",
                        "realtime"
                ],
                "camera_group_in": [],
                "camera_in": [],
                "matched_lists_in": [],
                "matched_dossier_in": [],
                "matched": true,
                "confidence_gte": 0.75
        },
        "episodes": {
                "allowed_types": [
                        "episode_open",
                        "episode_event",
                        "episode_close"
                ],
                "camera_group_in": [],
                "camera_in": [],
                "matched_lists_in": [],
                "matched": true,
                "events_count_gte": 0,
                "events_count_lte": 999
        },
        "counters": {
                "counter_in": [],
                "camera_group_in": [],
                "camera_in": [],
                "faces_gte": 1,
                "faces_lte": 100,
                "silhouettes_gte": 1,
                "silhouettes_lte": 100
        }
}
```

Important: Use only filters which match your search needs. To turn off a filter, remove it from a webhook. Do not leave a filter empty (`[]`) as in this case the result of filtration will be empty as well.

Note: To get all notifications, pass only curly braces without any enclosed filters:

```
{}
```

Tip: Example #1. Get notifications about all events:

```
{ "events": {} }
```

Example #2. Get notifications of the opening of matched episodes:

```
{ "episodes": { "allowed_types": ["episode_open"], "matched": true }}
```

Note: You can specify several values for filters with square braces. In this case, the webhook will be triggered once one of the values from this filter has been matched. In the example below, you will get an event from the camera group 1 or 3 if a matched dossier is 12 or 25.

```
{
        "events": {
                "camera_group_in": [1, 3],
                "matched_dossier_in": [12, 25],
        },
}
```

8. Check *Active*.
9. Click *Save*.

3.2.2 Webhook in Action

Try out a webhook by capturing event notifications with a simple web server in Python:

```
from pprint import pprint
from aiohttp import web


async def handle(request):
    pprint(await request.json())
    return web.Response(status=200)


app = web.Application()
# for aiohttp v 3.x
# app.add_routes([web.post('/', handle)])

# for aiohttp v 2.x
app.router.add_post('/', handle)

web.run_app(app, port=8888)
```

Important: A webhook catcher that you use must return an `HTTP 200` response after receiving the webhook request from FindFace Multi, like in the example above.

If no filters are configured for a webhook, this web server will be getting notifications about all events, episodes, and counter records that occur in the system. The notifications have the following format:

- `Face event`
- `Body event`
- `Car event`
- `Episode opening`
- `Episode closing`
- `Counter record`

To view a webhook pulling status, execute:

```
sudo journalctl -u findface-security.service | grep 'Webhook'
```

Success:

```
May 30 14:13:43 ffsecurity[12441]: INFO    [Webhook(id=6) worker(type=face_events]
↪<queue:  0> Sent batch(len-1, type-"face_events"): ['4355024961160384430']
May 30 14:13:43 ffsecurity[12441]: INFO    [SC:OQSrsPV9] [Webhooks manager-38bc5]␣
↪Processing message(type="face_events:event_created"). Consumer reception delta: 0.
↪003450
May 30 14:13:43 ffsecurity[12441]: INFO    [Webhook(id=6) worker(type=face_events]
↪<queue:  0> Sent batch(len-1, type-"face_events"): ['4355024961658847580']
May 30 14:13:44 ffsecurity[12441]: INFO    [SC:JtRz2Vuo] [Webhooks manager-38bc5]␣
↪Processing message(type="face_events:event_created"). Consumer reception delta: 0.
↪001263
May 30 14:13:44 ffsecurity[12441]: INFO    [Webhook(id=6) worker(type=face_events]
↪<queue:  0> Sent batch(len-1, type-"face_events"): ['4355024962087522421']
May 30 14:13:44 ffsecurity[12441]: INFO    [SC:9AnzRJwU] [Webhooks manager-38bc5]␣
↪Processing message(type="face_events:event_created"). Consumer reception delta: 0.
↪001691
May 30 14:13:44 ffsecurity[12441]: INFO    [Webhook(id=6) worker(type=face_events]
↪<queue:  0> Sent batch(len-1, type-"face_events"): ['4355024962355957878']
```

Failure:

```
May 30 14:18:49 ffsecurity[12441]: INFO    [SC:sp34rVQR] [Webhooks manager-38bc5]␣
↪Processing message(type="face_events:event_created"). Consumer reception delta: 0.
↪001376
May 30 14:18:49 ffsecurity[12441]: WARNING [Webhook(id=6) worker(type=face_events]
↪<queue:  0> Error sending webhook: Cannot connect to host 127.0.0.1:8888 ssl:None␣
↪[Connection refused]. Attempt 1 out of 10. Next attempt in 0.270 seconds.
May 30 14:18:50 ffsecurity[12441]: WARNING [Webhook(id=6) worker(type=face_events]
↪<queue:  0> Error sending webhook: Cannot connect to host 127.0.0.1:8888 ssl:None␣
```

(continues on next page)

(continued from previous page)

```
↪[Connection refused]. Attempt 2 out of 10. Next attempt in 0.729 seconds.
May 30 14:18:50 ffsecurity[12441]: INFO    [SC:zUhLHNxN] [Webhooks manager-38bc5]␣
↪Processing message(type="face_events:event_created"). Consumer reception delta: 0.
↪001368
May 30 14:18:50 ffsecurity[12441]: INFO    [SC:1Q66tcUS] [Webhooks manager-38bc5]␣
↪Processing message(type="face_events:event_created"). Consumer reception delta: 0.
↪001386
May 30 14:18:50 ffsecurity[12441]: WARNING [Webhook(id=6) worker(type=face_events]
↪<queue:  2> Error sending webhook: Cannot connect to host 127.0.0.1:8888 ssl:None␣
↪[Connection refused]. Attempt 3 out of 10. Next attempt in 1.968 seconds.
May 30 14:18:52 ffsecurity[12441]: WARNING [Webhook(id=6) worker(type=face_events]
↪<queue:  2> Error sending webhook: Cannot connect to host 127.0.0.1:8888 ssl:None␣
↪[Connection refused]. Attempt 4 out of 10. Next attempt in 5.314 seconds.
May 30 14:18:55 ffsecurity[12441]: INFO    [SC:5kl6zGrF] [Webhooks manager-38bc5]␣
↪Processing message(type="face_events:event_created"). Consumer reception delta: 0.
↪001542
May 30 14:18:58 ffsecurity[12441]: WARNING [Webhook(id=6) worker(type=face_events]
↪<queue:  3> Error sending webhook: Cannot connect to host 127.0.0.1:8888 ssl:None␣
↪[Connection refused]. Attempt 5 out of 10. Next attempt in 14.349 seconds.
```

3.2.3 Verbose Webhooks

By default, webhook notifications contain only ids of such entities as dossiers, watch lists, cameras, and camera groups. It is possible to get whole entities in notifications by switching webhooks to the verbose mode.

To do so, open the `/etc/findface-security/config.py` configuration file and set `'VERBOSE_WEBHOOKS': True`:

```
sudo vi /etc/findface-security/config.py

...
FFSECURITY = {
    ...
    # send serialized dossiers, dossier-lists, camera and camera groups in webhooks
    'VERBOSE_WEBHOOKS': True,
    ...
}
...
```

In the verbose mode, the format of webhook notifications is the following:

- `Face event (verbose)`
- `Body event (verbose)`
- `Car event (verbose)`
- `Episode opening (verbose)`
- `Episode closing (verbose)`
- `Counter record (verbose)`

3.3 Partner Integrations

3.3.1 Genetec Security Center

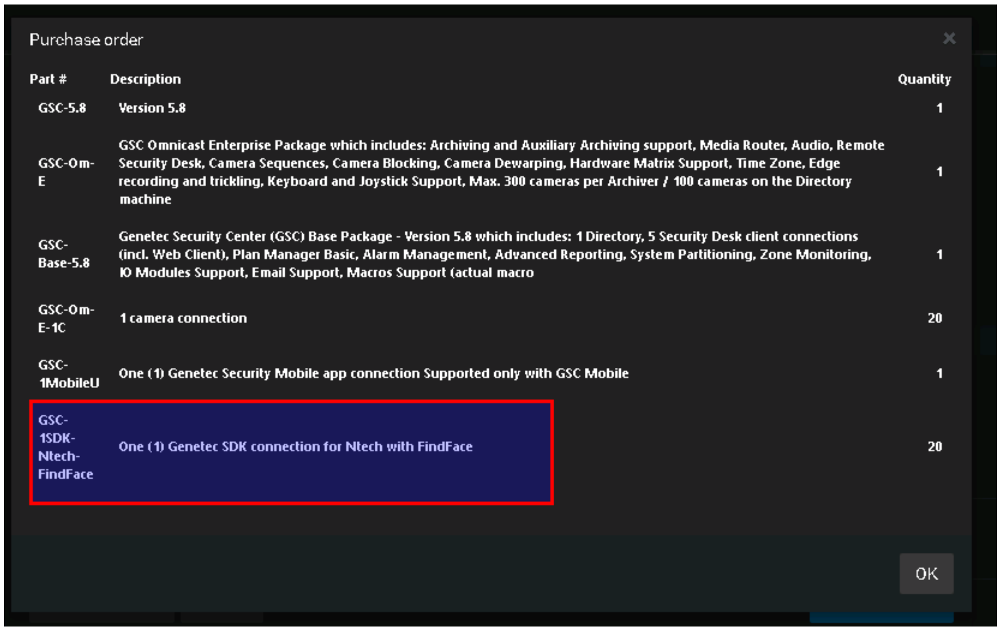
FindFace Multi integration with Genetec Security Center allows you to expand the capabilities of your Genetec-based security system with face recognition functionality.

Configure Integration

Integration with Genetec Security Center is implemented via the `findface-genetec` plugin. By default, the plugin is disabled.

Before getting started with the integration on the FindFace Multi side, deploy the Genetec Web SDK and Media Gateway packages, and create an `Alarm` entity that will be triggered in Genetec Security Center when a face recognition event occurs in FindFace Multi.

Important: For the Genetec-FindFace integration to work, you also need to purchase a proper license from Genetec (license part number `GSC-1SDK-Ntech-FindFace`) and activate it in Genetec Security Center.



Purchase order

Part #	Description	Quantity
GSC-5.8	Version 5.8	1
GSC-Om-E	GSC Omnicast Enterprise Package which includes: Archiving and Auxiliary Archiving support, Media Router, Audio, Remote Security Desk, Camera Sequences, Camera Blocking, Camera Dewarping, Hardware Matrix Support, Time Zone, Edge recording and trickling, Keyboard and Joystick Support, Max. 300 cameras per Archiver / 100 cameras on the Directory machine	1
GSC-Base-5.8	Genetec Security Center (GSC) Base Package - Version 5.8 which includes: 1 Directory, 5 Security Desk client connections (incl. Web Client), Plan Manager Basic, Alarm Management, Advanced Reporting, System Partitioning, Zone Monitoring, IO Modules Support, Email Support, Macros Support (actual macro	1
GSC-Om-E-1C	1 camera connection	20
GSC-1MobileU	One (1) Genetec Security Mobile app connection Supported only with GSC Mobile	1
GSC-1SDK-Ntech-FindFace	One (1) Genetec SDK connection for Ntech with FindFace	20

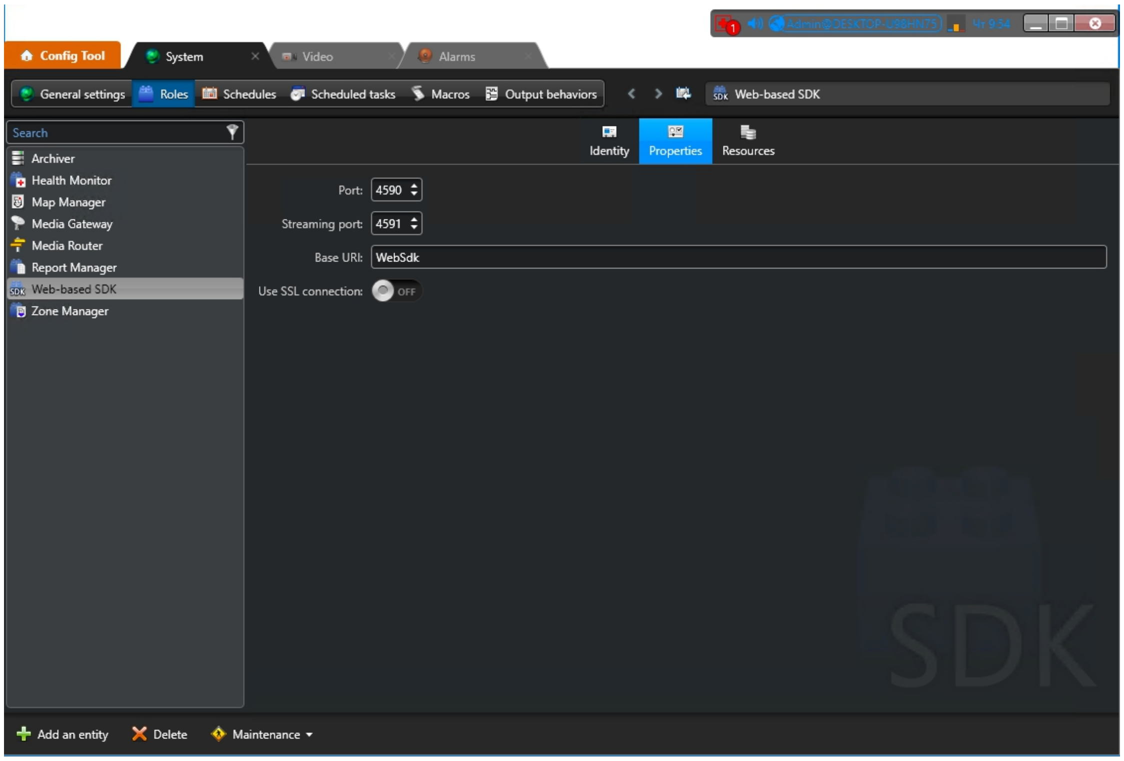
OK

In this chapter:

- *Create Watch Lists and Dossiers in FindFace Multi*

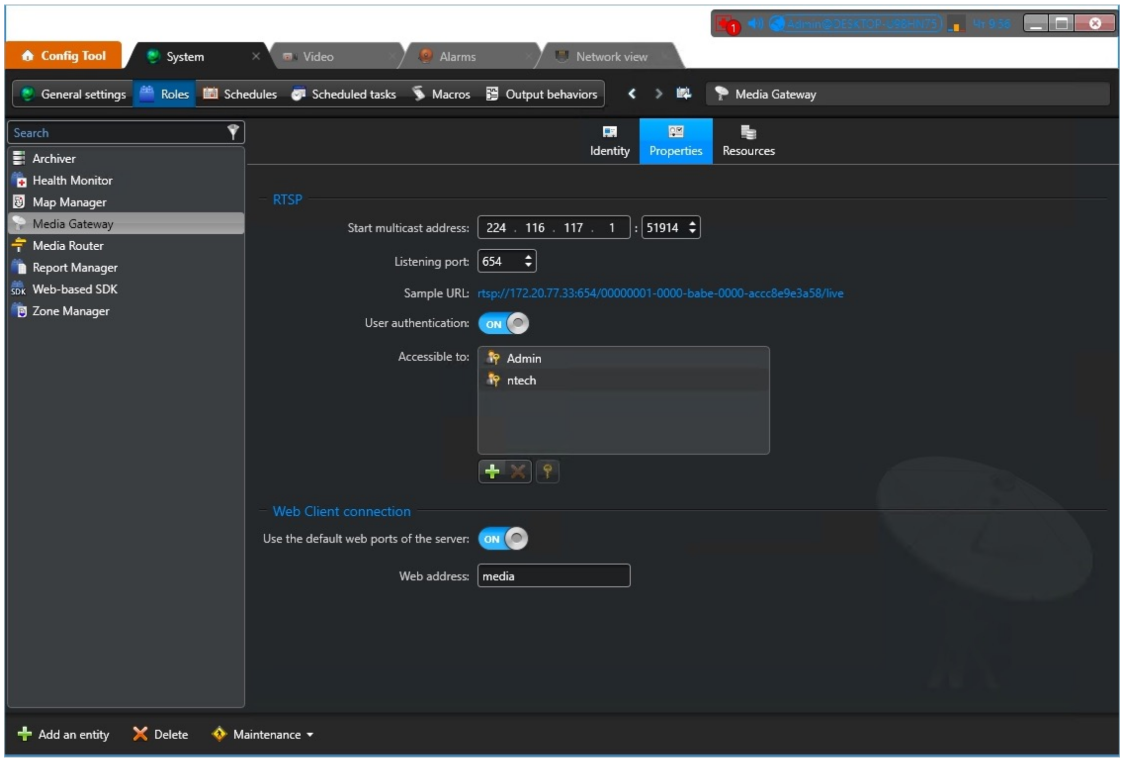
Configure Genetec Web SDK and Media Gateway

To enable and configure Web SDK, use Genetec Config Tool. For details, refer to *Security Center Administrator Guide -> Chapter 52: Role Types -> Web-based SDK configuration tabs*.



Port: 4590
Streaming port: 4591
Base URI: WebSdk
Use SSL connection: OFF

When enabling and configuring Media Getaway in Genetec Config Tool, refer to *Security Center Administrator Guide -> Chapter 24: Video Deployment*.



RTSP

Start multicast address: 224 . 116 . 117 . 1 : 51914

Listening port: 654

Sample URL: rtsp://172.20.77.33:654/00000001-0000-babe-0000-accc8e9e3a58/live

User authentication: ON

Accessible to: Admin, ntech

Web Client connection

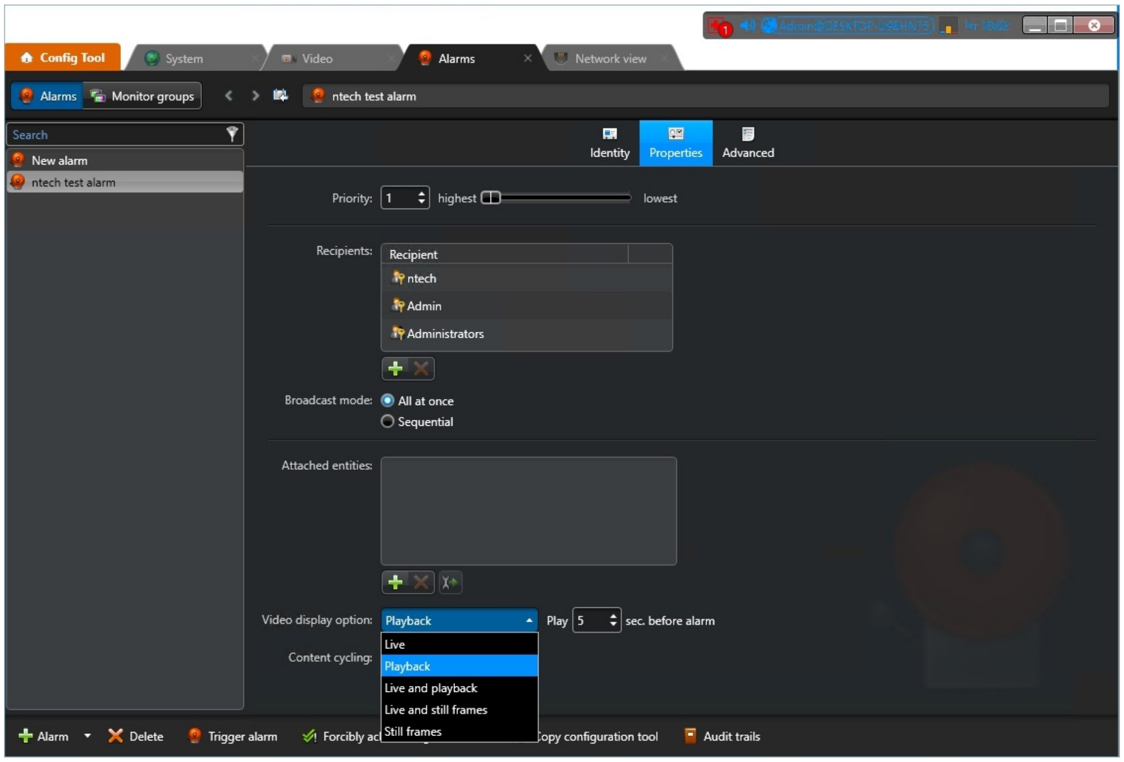
Use the default web ports of the server: ON

Web address: media

Important: Make sure that the firewall is configured so that the ports for the WebSDK and Media Gateway are open.

Create Alarm in Genetec Security Center

Create and configure a new `Alarm` entity in Genetec Config Tool. Refer to *Security Center Administrator Guide -> Chapter 48: Alarms -> Creating Alarms* for details.



Priority: 1 highest lowest

Recipients:

Recipient
ntech
Admin
Administrators

Broadcast mode: [✓ All at once] Sequential

Attached entities:

Video display option: Playback — Play 5 sec. before alarm

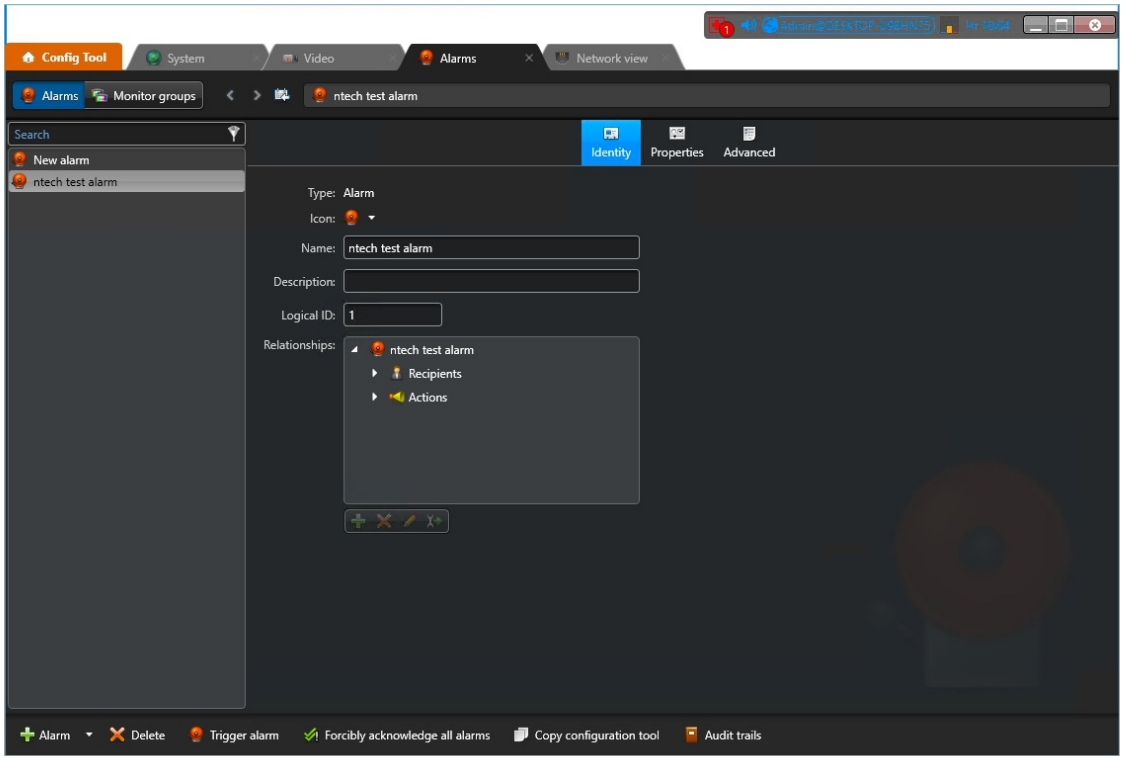
Live
Playback
Live and playback
Live and still frames
Still frames

Content cycling:

Tip: On the *Properties* tab, select the *Video display option* that suits your needs the best. Available options are *Live*, *Playback*, etc.

Tip: To enable alarm procedures and auto rotation of video right within the alarm pop-up window, enable *Content cycling*.

When configuring the integration in FindFace Multi, you will have to enter the alarm logical id that is specified on the *Identity* tab.



Type: Alarm
Icon:
Name: ntech test alarm
Description:
Logical ID: 1
Relationships: ntech test alarm
Recipients
Actions

Enable Genetec Integration in FindFace Multi

To enable the Genetec integration in FindFace Multi, do the following:

1. Enable the `findface-genetec` plugin. To do so, open the `/etc/findface-security/config.py` configuration file and uncomment the `INSTALLED_APPS.append('ffsecurity_genetec')` line. Make sure that at least one of the following parameters is specified: `SERVICE_EXTERNAL_ADDRESS` or `EXTERNAL_ADDRESS`.

```
sudo vi /etc/findface-security/config.py

...
# SERVICE_EXTERNAL_ADDRESS is prioritized for FFSecurity webhooks and Genetec␣
↪plugin.
# EXTERNAL_ADDRESS is used instead if SERVICE_EXTERNAL_ADDRESS is not provided.
# You must provide either SERVICE_EXTERNAL_ADDRESS or EXTERNAL_ADDRESS in order
# to be able to work with FFSecurity webhooks and Genetec plugin.
SERVICE_EXTERNAL_ADDRESS = 'http://127.0.0.1'
# EXTERNAL_ADDRESS is used to access objects created inside FFSecurity via external␣
↪links.
EXTERNAL_ADDRESS = ''

...
# FINDFACE SECURITY PLUGINS
# ============================================================================
# Uncomment lines below to enable plugins. Please consult documentation for
# a plugin specific settings.
...
# =============== Genetec ================
```

(continues on next page)

(continued from previous page)

```
INSTALLED_APPS.append('ffsecurity_genetec')
```

2. Migrate the main database architecture from FindFace Multi to **PostgreSQL** and re-create user groups with *predefined* rights.

```
sudo findface-security migrate
sudo findface-security create_groups
```

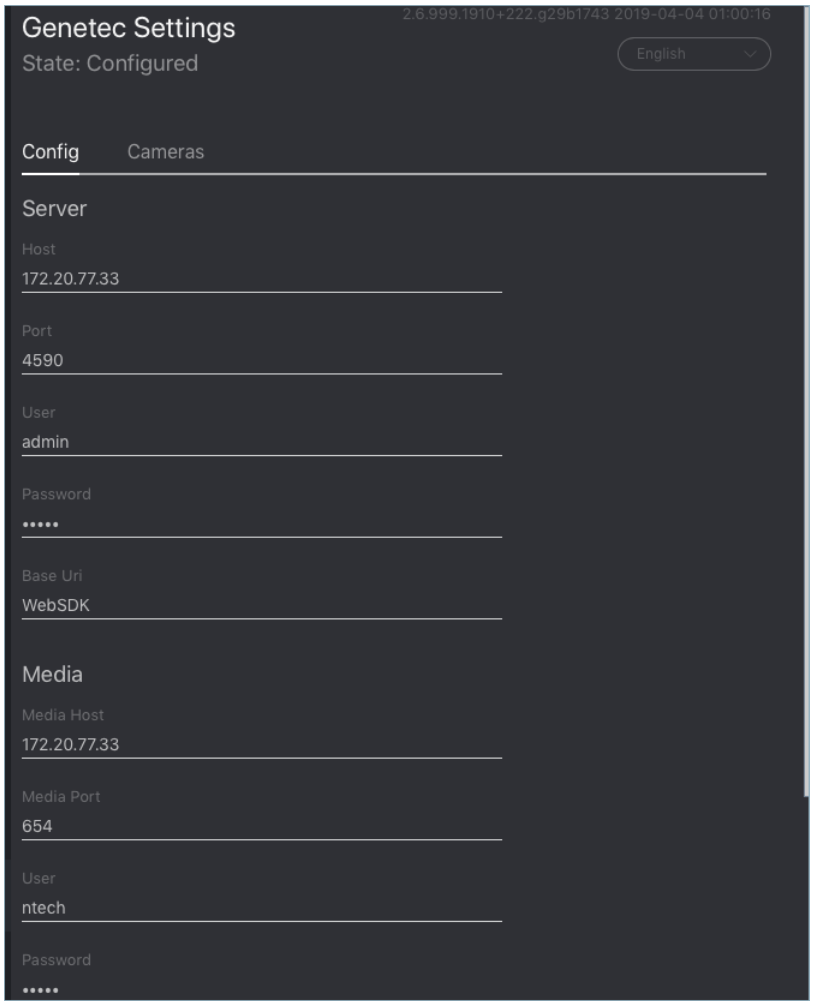
3. Restart `findface-security`.

```
sudo systemctl restart findface-security.service
```

Configure Endpoints in FindFace Multi

To establish connection between FindFace Multi and Genetec Security Center, do the following:

1. Navigate to the *Preferences* tab. Click *Genetec*.



Genetec Settings
State: Configured

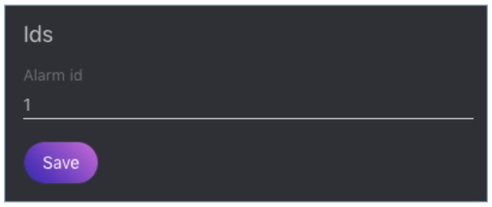
Server
Host
172.20.77.33
Port
4590
User
admin
Password
•••••
Base Uri
WebSDK

Media
Media Host
172.20.77.33
Media Port
654
User
ntech
Password
•••••

2. In the *Server* and *Media* sections, specify *settings* of the Web SDK and Media Gateway endpoints.

Important: The ports for the WebSDK and Media Gateway need to be open.

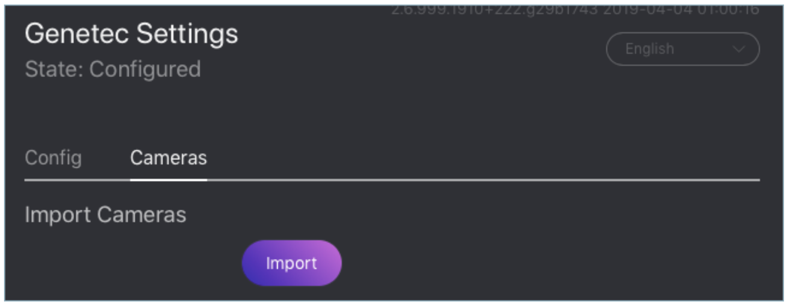
3. In the *Ids* section, specify the *logical id* of the `Alarm` entity that will be triggered in Genetec Security Center when a face recognition event occurs in FindFace Multi.



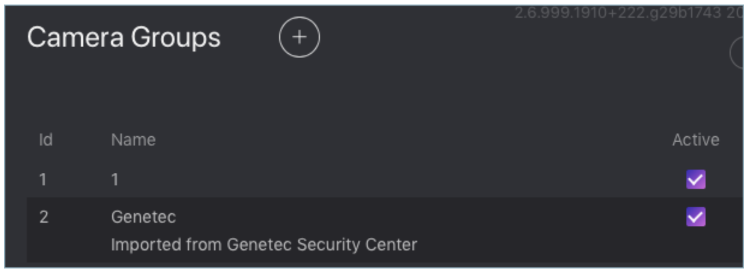
4. Click *Save*. If the connection to Genetec Security Center is successfully established, you will see the *State* change to *Configured*.

Import Cameras from Genetec Security Center

Once the connection to Genetec Security Center is established, import cameras. To do so, click *Cameras* on the *Genetec* tab. Click *Import*.



This action will create a *group of cameras* `Genetec` listing all the cameras from Genetec Security Center.



Id	Name	Active
1	1	✓
2	Genetec Imported from Genetec Security Center	✓

To view this list of cameras, navigate to the *Cameras* tab on the FindFace Multi navigation bar. If you want to exclude a camera from face recognition, simply deactivate it in the list.

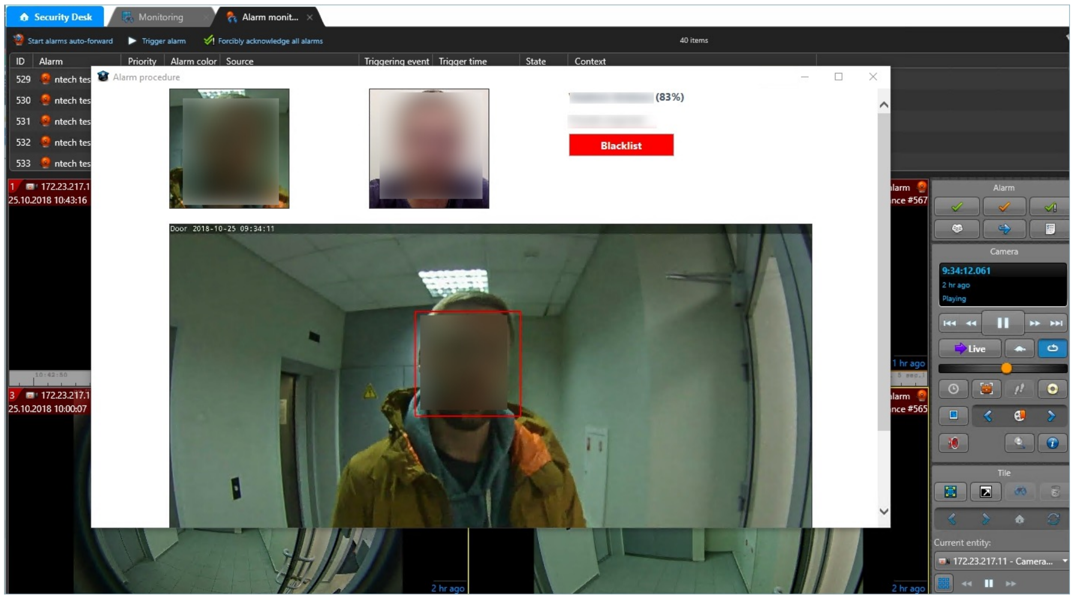
Create Watch Lists and Dossiers in FindFace Multi

After you have configured the endpoints and camera settings, finish the integration by creating a *dossier database*. Notifications about face recognition events will be automatically sent to Genetec Security Center. See *Notifications in Genetec Security Center*.

Notifications in Genetec Security Center

Each face recognition event from a Genetec camera, that has a match with a dossier, triggers a relevant alarm in Genetec Security Center. Every alarm triggered by FindFace Multi is associated with a relevant camera (source of the face recognition event) so you can instantly watch live or playback video within the `Alarm Monitoring` task in Genetec Security Desk. FindFace Multi also utilizes `Alarm Procedures` to provide a user with additional content related to the alarm, such as:

- face detected in video
- matching face from the dossier database
- person's name and comment from the dossier
- matching confidence
- watch list's name
- full frame



After you receive a face recognition alarm, process it as you usually do with other alarms in Genetec Security Center.

3.3.2 Axxon Next

FindFace Multi integration with Axxon Next allows you to detect and identify faces in video streams from an Axxon-based security system.

Important: One FindFace Multi instance supports interaction with only one Axxon Next server.

Integration with Axxon Next is implemented via the `ffsecurity_axxon` plugin.

To configure the FindFace Multi integration with Axxon Next in Ubuntu, do the following:

1. Activate the plugin by uncommenting the `INSTALLED_APPS.append('ffsecurity_axxon')` line in the `/etc/findface-security/config.py` configuration file.

```
sudo vi /etc/findface-security/config.py

...

# ==============================================================================
# FINDFACE SECURITY PLUGINS
# ==============================================================================
# Uncomment lines below to enable plugins. Please consult documentation for
# a plugin specific settings.
# =============== Axxon ================
INSTALLED_APPS.append('ffsecurity_axxon')
```

2. Uncomment the `FFSECURITY->AXXON` section in the configuration file. Fill it out as shown in the example below. In the `api` parameter, specify the IP address of the Axxon Next server that will provide FindFace Multi with Axxon API and HLS-archive streams. In the `rtsp` parameter, specify the common segment of Axxon video stream addresses. `name`, `user`, `password`: the Axxon Next server name and credentials to access it.

```
FFSECURITY['AXXON'] = [
    {
        'name': 'server_name',
        'api': 'http://example.com/',
        'rtsp': 'rtsp://example.com:554/',
        'user': 'user',
        'password': 'password',
    }
]
```

3. (Optional). If facial recognition events are required to contain video from Axxon Next, uncomment the `FFSECURITY_UI_CONFIG['dossier']` section.

```
FFSECURITY_UI_CONFIG['dossier'] = {
   'video': True,
}
```

4. Create representations of Axxon Next cameras in FindFace Multi (see *Camera Management*). A camera representation URL must be specified in the format `axxon:<friendlyNameLong>`, where `friendlyNameLong` is a camera name on the Axxon Next server. Find out this name in the Axxon user interface, or via Axxon API by executing:

```
curl http://user:password@127.0.0.1/video-origins/

{
   "OLOLOE-DEV/DeviceIpint.vhod_1/SourceEndpoint.video:0:0" : {
      "friendlyNameLong" : "vhod_1.Vhod_1",
      "friendlyNameShort" : "Vhod_1",
      "origin" : "OLOLOE-DEV/DeviceIpint.vhod_1/SourceEndpoint.video:0:0",
      "state" : "signal_restored"
   }
}
```

For the camera from the example above, URL must be specified as `axxon:vhod_1.Vhod_1`.

The configuration is now finished. If the integration is properly configured, FindFace Multi will be detecting and identifying faces in Axxon Next video streams, and facial recognition events will be featuring video clips from Axxon Next (upon relevant settings).

PYTHON MODULE INDEX

INDEX