FindFace

Release 2.0

NtechLab

Mar 29, 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 type and color.
- Vehicles, with recognition of such vehicle 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 records that can accommodate aggregated information about a person and a car/vehicle.
- Fast and robust real-time video monitoring against a record index.
- Multi-object identification and analytics: faces, bodies (silhouettes), vehicles.
- 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, fine-tuning a camera for each object type, video recording (if enabled).
- Multi-object verification: faces, bodies, vehicles.
- AI recognition of gender, age, emotions, glasses, beard, face mask, and other face attributes.
- AI face liveness detector.
- AI recognition of an exact person and vehicle.
- AI recognition of clothing type and color.
- AI recognition of special vehicles, a vehicle's make, model, body style, color, and license plate number.
- Database search for faces, bodies, vehicles.
- Possibility of counting faces, bodies, and vehicles on connected cameras and measuring distance between bodies. Single- and multi-camera counting support.
- Video surveillance. Video Recorder.
- Possibility of monitoring the presence of people in given areas, using rules and monitoring schedules.
- Automatic clustering of objects of the same origin (face/body images belonging to the same person, images of the same vehicle), enriched with the end-to-end integration with the record index.

Deployment 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.
- Network or on-premise licensing.
- CPU- and GPU-based acceleration for your choice.

System security

- Advanced user management.
- Authentication based on a password, certificate, and face recognition for guaranteed system protection.
- Record security.
- Comprehensive, friendly, searchable audit logs.
- Backup and recovery utilities.
- Possibility of monitoring user sessions and blocking devices without deactivating user accounts.

Ethical data usage

- Full compliance with personal data protection laws (GDPR and similar).

Make the most of your system

- Social interaction analysis.
- Audience analysis.
- Detailed reports on face recognition events, episodes, search events, clusters, counters, cameras, records, audit logs, and audience.
- Face liveness detector as a standalone service.

Useful little things

- Quick record index creation.
- Complete record customization.
- Deduplication support for events and records.
- Extended set of search filters.
- Scheduled database cleanup.

Integration

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

CHAPTER

ONE

WHAT'S NEW IN FINDFACE MULTI 2.0

Enhanced Algorithms, UI, UX:

- New design: a brand-new design for fast and comfortable work is here.

  A completely reinvented UI provides smooth work with large amount of data: view object detailed information in sidebar, easily switch between opened tabs. Start exploring new interface with *Web Interface Basics*.

- Superior neural networks: Liveness technology that passed iBeta Level 2 testing.

  This FindFace Multi version presents a brand-new neural network model for mobile phones `liveness.goodwin` that passed iBeta Level 2 test. The model is aimed at silicon/latex masks and paper cylindrical face texture wraps detection. Also, FindFace Multi 2.0 includes a huge amount of brand-new neural network models or updated versions of previous neural network models with enhanced characteristics.

  See *Configure FindFace Multi Neural Networks*

Technical Changes:

- Technical architecture: microservices instead of monolith.

  See *Architecture*

- Docker-based distribution: FindFace Multi 2.0 runs as a set of docker containers described in a docker-compose file.

  See:

  - *Docker Platform*
  - *Useful Docker Commands*

- Vertical scaling support: more events can now be processed due to the better use of server capacity.
- Horizontal scaling support: process more events for less time by deploying new servers.

  Both vertical scaling and horizontal scaling support guarantee increased performance.

- Support of *S3-compatible storage*: an S3 storage provides reliable and long-term storage of an unlimited number of files and data. It gives a way to avoid the limitations of the file system when it comes to large amounts of data.
- CentOS support.

New Features:

- *External VMS*.
- PTZ camera control: the ability to rotate PTZ cameras via ONVIF is added to the video player control panel.
- *VMS smart cleaner*: configure video cleanup on a regular basis.
- Advanced *webhooks*.
- *Active Directory integration*.
- *Episode and event feed and filtering in Video Wall*.
- *Episodes on the timeline in Video Player*.
- Recognition of new object attributes.

  See:

  - *Enable Face and Face Attribute Recognition*
  - *Enable Vehicle and Vehicle Attribute Recognition*
  - *Special Vehicle Recognition*
  - *Enable Body and Body Attribute Recognition*
- New approach to attribute licensing: object attributes that were previously unlicensed or licensed as a group of several object attributes at once are licensed separately now.

Renamed:

- *Interaction Analysis* → *Interaction Tracking*; *Cameras*, *Videos* → *Video Sources*; *Episodes*, *Events* → *Episodes & Events*; *Analytics* → *Audience analysis*; *Preferences* → *Settings*; *Cards* → *Record Index*.
- `car` → `vehicle`

Fresh Neural Networks:

- **Object detection**
  - Face detection: a fresh face detection model `face.jasmine_fast.003` with improved characteristics.
  - Vehicle detection: brand-new vehicle detection neural network models: `car.gustav_accurate.004`, used in `findface-extraction-api` and `car.jasmine_fast.005`, used in `findface-video-worker`.
  - Body detection: brand-new body detection neural network models: `body.gustav_normal.015`, used in `findface-extraction-api` and `body.jasmine_fast.018`, used in `findface-video-worker`.
  - 3-in-1 object detector: a new detector `headbodyface.alpha000_normal.001` that returns three sub-objects belonging to a person: body/head/face.
- **Object image normalization**
  - Face image normalization: new face image normalization models `facenorm.multicrop_full_center_size400` and `facenorm.multicrop_full_crop2x_size400` that use a smarter normalization algorithm and a new version `bee.v3` with improved characteristics.
  - Vehicle license plate image normalization: a new and much faster network model `briacon.v0` and new versions `anaferon.v5` and `anaferon.v7` with improved characteristics for better vehicle license plate image normalization.

- **Object recognition**
  - Face recognition: fresh neural networks `lime.v2` and `mango_320` that ensure fast face recognition.
  - Body recognition: a brand-new model `pedrec.clio` with increased accuracy.
- **Object attribute recognition**
  - Head position recognition: a fresh neural network `headpose.v2` to recognize head tilt/turn.
  - Liveness recognition: a brand-new model for mobile phones `liveness.goodwin` that passed iBeta Level 2 test. The model is aimed at silicon/latex masks and paper cylindrical face texture wraps detection. A brand-new neural network model `liveness.web.v0` for webcam liveness spoofing attack detection. A new version `liveness.pacs.v2` with improved characteristics.
  - Face quality recognition: a new version `quality_fast.v1` that predicts face quality to get the best shot and filter trash.
  - Vehicle category recognition: a brand-new neural network `carattr.categories.v0` that determines whether a vehicle belongs to any of these categories: motorcycle (including moped and scooter) or quad bike, car, truck, car with a trailer, truck with a trailer, bus, or articulated bus.
  - Vehicle weight and body size recognition: a brand-new neural network `carattr.weight_types7.v0` that predicts a vehicle type according to its weight and body size.
  - Recognition of vehicle orientation: a brand-new neural network `carattr.orientation.v0` to recognize vehicle view: front, rear, or side.
  - Special vehicle recognition: a new version `carattr.special_types11.v1` that supports recognition of route transport, carsharing, gas service, military, and road service vehicles in addition to police cars, ambulance, rescue service, and taxi.
  - Vehicle license plate recognition: a new version `carattr.license_plate.v7` that supports license plates of Czech Republic, Pakistan, Thailand in addition to Serbia, Lithuania, Latvia, Moldova, Estonia, Finland, Azerbaijan, Tajikistan, Turkmenistan, Mexico, Argentina, inherited from `carattr.license_plate.v6` and `carattr.license_plate.v5`. Recognition of license plates of the UAE, Russia, Kazakhstan, Georgia, Saudi Arabia, Brazil, India, Uzbekistan, Vietnam, Belarus, Ukraine, Armenia, Kyrgyzstan, introduced in earlier versions of the product, is also supported, which gives a total of 27 supported countries.
  - License plate image quality: a new and much faster version `carattr.license_plate_quality.v1`, which is used to get the best shot of a vehicle's license plate in a sequence of shots.
  - Silhouette-based age and gender recognition: a fresh neural network `pedattr.age_gender.v0` to predict human's age and gender by silhouette.
  - Bag presence recognition: a brand-new neural network `pedattr.bags.v0` to recognize whether a person has a bag, a backpack, or a suitcase.
  - PPE recognition: a fresh neural network `pedattr.protective.v1` to recognize the presence or absence of personal protective equipment and its color.

Functionality changes compared to 1.2:

- The Areas section is deprecated, but its functionality is still available through API. In the future the Areas functionality will be included into Counters. Consider using a previous version of FindFace Multi if you actively employ Areas functionality.
- Counter Chart for the last hour/day/week is removed from the counter settings interface: the chart functionality was not suitable enough for data analysis. Analysis of data, received from Counters, is available through report unloading or post-processing of data from API.
- Card Relations are no longer supported in the user interface, but the functionality still exists in API: being experimental, this feature failed to cover potential customers' needs. It will probably be developed in the future.

CHAPTER

TWO

SYSTEM ADMINISTRATOR'S GUIDE

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

2.1 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:

2.1.1 Recognition Objects and Recognition Process

FindFace Multi can recognize the following objects and their attributes:

- human faces
- human bodies (silhouettes)
- vehicles

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.

2.1.2 Docker Platform

FindFace Multi is deployed in Docker, a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Each FindFace Multi service runs in a Docker container.

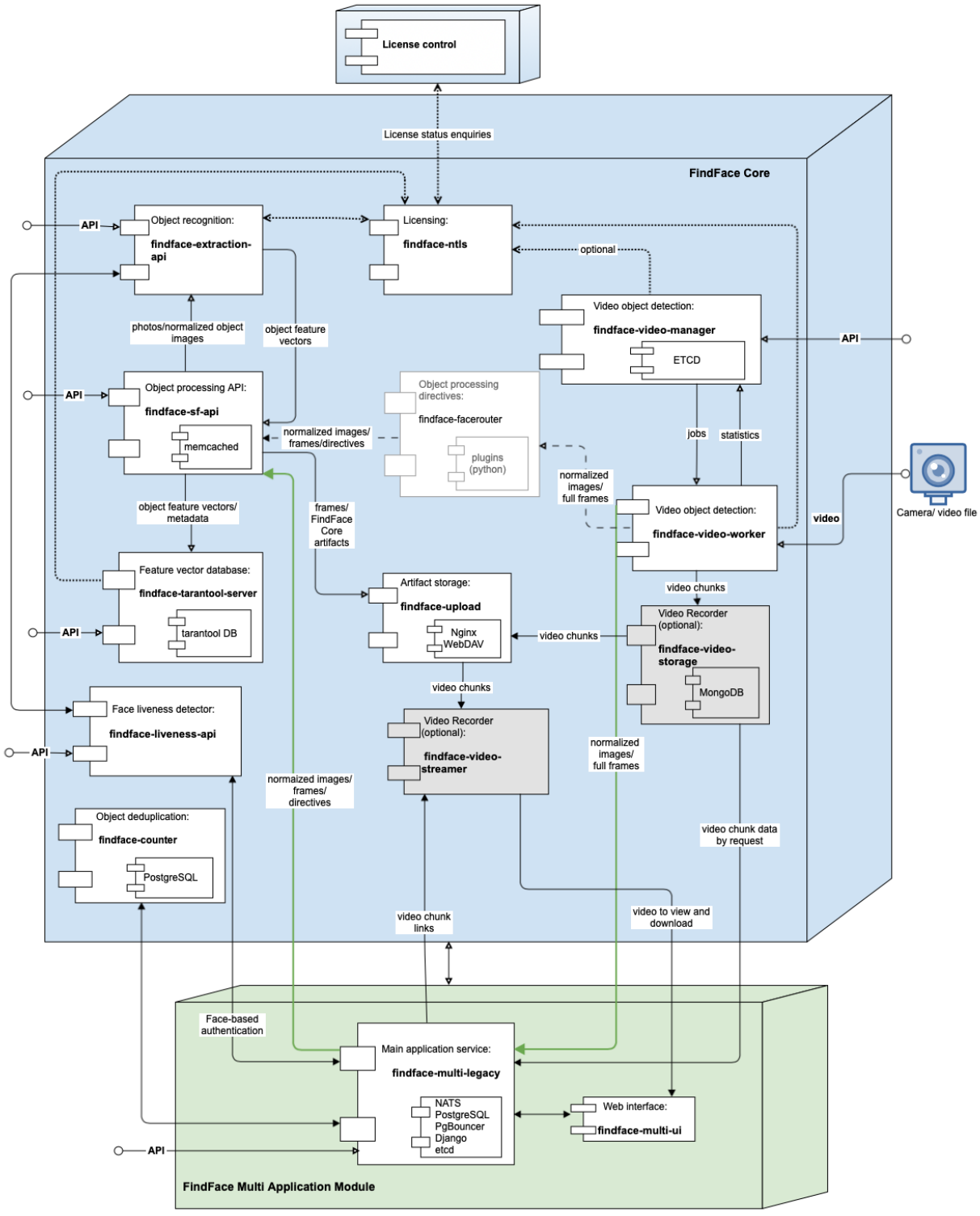
2.1.3 Architectural Elements

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.

The FindFace Core internal architecture allows for embedding Video Recorder, an additional functionality that records, stores, and plays back video data from cameras.

Architecture scheme



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. If Video Recorder is installed, `findface-upload` also stores video data from cameras.	
findface-facerouter	18820	Service used to define processing directives for detected objects. In FindFace Multi, its functions are performed by `findface-multi-legacy` (see *FindFace Multi Application Module*).	
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. If *Video Recorder* is enabled, `findface-video-worker` sends video over to `findface-video-storage`. Provides *face liveness detection* if enabled. CPU- or GPU-acceleration.	
findface-ntls	443 (TCP), 3133, 3185	License server that interfaces with the NtechLab Global License Server, a USB dongle, or hardware fingerprint to verify the *license* of your FindFace Multi instance.	
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 chunk and returns the best quality face, and decimal liveness result averaged across the taken frames. The service is also involved in the course of user *authentication* by face.	
Tarantool	Shard ports (default 330xx, 81xx)	Third-party software that implements the feature vector database that stores extracted object feature vectors and identification events. The system data, records, 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 that implements the system web interfaces.	nginx
memcached	11211	Third-party software that implements a distributed memory caching system. Used by `findface-sf-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-multi-legacy	Configurable	Service 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, object clusterization, and counters.	NtechLab own deployment
findface-multi-pause	n/a	Internal services that assist `findface-multi-legacy`. The `findface-multi-audit` service is created for the future. It will become full-fledged in the upcoming versions. As for now, it partly duplicates the `findface-multi-legacy` functionality. Use `findface-multi-legacy` to work with it. The `findface-multi-identity-provider` is the service for authentication, user management, and management of role-based model of accesses.	
findface-multi-audit	8012, 8013, 8014		
findface-multi-identity-provider	8022, 8023, 8024		
findface-onvif-discovery	n/a		
findface-multi-ui	Configurable	Main web interface used to interact with FindFace Multi. Based on the Django framework. Allows you to work with object recognition events, search for objects, manage cameras, users, records, watch lists, collect real-time statistics, and many more.	
NATS	4222	Third-party software that implements a message broker in `findface-multi-legacy`.	NATS
etcd	2379	Third-party software that implements locks in the `findface-multi-legacy` service e.g. locks in license checker, reports, video processing, object clusterization, etc.	etcd
Pgbouncer	5439	Third-party software, a lightweight connection pooler for PostgreSQL. Optional, used to increase the database performance under high load.	PgBouncer
PostgreSQL	5432	Third-party software that implements the main system database. This database stores records of persons and vehicles and data for internal use, including user accounts and camera settings. The object feature vectors and object identification events are stored in Tarantool (part of the FindFace Core).	PostgreSQL

Video Recorder

Video Recorder is an optional part of the FindFace Core. It includes the following services:

Component	Ports in use	Description	Vendor
findface-video-storage	8611	Service that implements video chunk management. It takes video chunks from the `findface-video-worker` service, puts them into the storage (`findface-upload`), and writes their meta-information and whereabouts to the Video Recorder database (MongoDB). By request from `findface-multi-legacy`, it issues info about existing video chunks in the form of Websocket-links to the corresponding streams. These links are further used by `findface-video-streamer-cpu` to deliver the video to a user for viewing and downloading.	NtechLab own deployment
findface-video-streamer	9000	After receiving a `findface-multi-ui` request, this service uses Websocket to extract requested video chunks from `findface-video-storage` and `findface-video-worker` (only the last chunk if it's not yet recorded to the storage). Then it merges the video chunks into a one-piece video and delivers it to a user for viewing and downloading.	
MongoDB	27017	Third-party software that implements the Video Recorder database. The database stores meta-information of the video chunks, including their location. The video chunks themselves are stored in the `findface-upload` service.	MongoDB

See also:

- *FindFace Multi Data Storages*

2.1.4 Single- and Multi-Host Deployment

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

- Deploy FindFace Multi on a principal host and distribute additional `findface-video-worker` instances across remote hosts.

  See *FindFace Video Worker Deployment on Remote Hosts*.

- Distribute the FindFace Multi services across multiple hosts. If necessary, set up load balancing. Contact our experts by support@ntechlab.com for guidance.

  See *Guide to Typical Multi-Host Deployment*.

2.1.5 CPU- and GPU-acceleration

The `findface-extraction-api` and `findface-video-worker` services can be either CPU- or GPU-based. You will have an opportunity to choose the acceleration type you need during the installation.

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`.

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

2.2 Requirements

In this chapter:

2.2.1 System Requirements for Basic Configuration

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

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

Important: If the video resolution 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 require 2 cores HT > 2.5 GHz. The characteristics also depend on the number of video files in process. A single video file `720p@25FPS` requires 2 cores >2.5 GHz. AVX2 support.	
GPU (optional)	NVIDIA GeForce® GTX 1060 6 GB	NVIDIA GeForce® GTX 1080Ti+ with 11+ GB RAM
	Supported devices: NVIDIA, Pascal architecture and above. *Note: NVIDIA GeForce RTX 40 Series graphics cards are currently not supported.*	
RAM	10 Gb	16+ Gb
	The own needs of FindFace require 8 Gb. The RAM consumption also depends on the number of video files in process. A single video file `720p@25FPS` requires 2 GB RAM.	
HDD (SSD for best performance)	16 Gb	16+ Gb
	The own needs of the operating system and FindFace 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 from 18 to 22, x64 only, RHEL / CentOS 7, Debian 11.	

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.

2.2.2 Required Administrator Skills

A FindFace Multi administrator must know and understand OS, on which the product instance is deployed, at the level of an advanced user.

2.2.3 Video File Formats

FindFace Multi supports a wide variety of file formats depending on the acceleration type, CPU or GPU.

Both CPU- and GPU-accelerated instances support all FFmpeg codecs. In addition to that, the following codecs are supported:

- *CPU-based acceleration*: FLV (both as a codec and as a container), H263, H264, H265, MJPEG, VP8, VP9, MPEG1VIDEO, MPEG2VIDEO, MSMPEG4v2, MSMPEG4v3.
- *GPU-based acceleration*: MJPEG, H264, H265, VP9, and others, depending on the list of codecs supported by the used video card. Apart from that, for instances with `video-worker-gpu`, you can expand the number of supported codecs by enabling video decoding on the CPU, which is not available by default.

To enable video decoding on the CPU for GPU-based acceleration, do the following:

1. Open the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` file.

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml
```

2. Set `cpu: true` in the `video_decoder` section.

```
...
video_decoder:
  cpu: true
...
```

3. Restart `findface-multi-findface-video-worker-1` container.

```
sudo docker container restart findface-multi-findface-video-worker-1
```

2.3 Deploy and Remove FindFace Multi

Docker platform

FindFace Multi 2.0 is Docker-based. You must install and start a set of Docker products before proceeding with the FindFace Multi deployment. For your convenience, this chapter contains the *Ubuntu Server Preparation* section covering the intricacies of installing Docker on Ubuntu. For other platforms, please refer to the Docker documentation.

NVIDIA driver and NVIDIA Container Runtime (GPU only)

If you intend to deploy FindFace Multi with GPU-acceleration, you need to install the NVIDIA driver and NVIDIA Container Runtime. Again, you will find the relevant information in the *Ubuntu Server Preparation* section.

Deployment options

After you are finished with the server preparation, you are all set for the FindFace Multi deployment. You are provided with the following options here:

1. Automatically install FindFace Multi standalone in one run. Being the simplest, this installation type is excellent to kick-start your experience with FindFace Multi. We recommend choosing it if you are a first-timer. See *FindFace Multi Standalone Automated Deployment* for guidance.
2. Automatically install FindFace Multi and configure it to interact with additional remote `findface-video-worker` instances. This installation type is for those who are deploying FindFace Multi in a multi-host environment. It requires a bit of technical expertise and knowledge of the product architecture. To automatically install only FindFace Video Worker (`findface-video-worker`) on a remote host, refer to the *FindFace Video Worker Deployment on Remote Hosts* section.
3. Fully customized installation. It requires fundamental understanding of the product architecture. See *Fully Customized Installation*.

Note: If you select the installation type #3, keep in mind to *install necessary neural network models* along with the `findface-extraction-api` component.

Installer questions and automatic deployment from the installation file

Before starting the actual installation, the installer asks you a few questions and performs several automated checks to ensure that the host meets the system requirements. After filling out each prompt, press `Enter`.

To install the same configuration of FindFace Multi on a different host, use the automatic deployment from the installation file. In this case, you won't have to answer the installation questions again. The exact path to the installation file is displayed right after the last question, before the installer starts off active progress:

```
[I 2023-02-09 11:13:37,187 main:142] Your answers were saved to /tmp/findface-installer-
↪p01n9sn3.json
```

Important: Be sure to remove fields `*.config`, `exp_ip`, and `int_ip` before installing FindFace Multi on a host with a different IP address.

To launch the automatic deployment from the `/tmp/<findface-installer-*>.json` file, execute:

```
sudo ./<findface-*>.run -f /tmp/<findface-installer-*>.json
```

Post-development procedures and how-to's

Browse through the *Post-deployment Procedures and How-to's* section to learn how to set the time zone, license your instance, and configure logging. This section will also provide you with few basic commands that will help you to kick-start your work with FindFace Multi containers, in case you are a newbie with Docker.

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

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.

Instance removal

To remove your instance, you must run a set of commands. See the *Remove FindFace Multi Instance* section.

2.3.1 Ubuntu Server Preparation

To prepare a server on Ubuntu for the FindFace Multi deployment, follow the instructions below minding the sequence.

Note: For other platforms, please refer to the following resources:

- NVIDIA drivers
- Docker Engine
- Docker Compose
- NVIDIA Container Toolkit

In this section:

GPU: Install NVIDIA Drivers

The first step of the server preparation is the NVIDIA driver installation. It's applicable only for the GPU configuration. Go straight to the *Docker installation* if your configuration is CPU-accelerated.

With the GPU-accelerated FindFace Multi, you'll need the NVIDIA driver version `530` or above. Add the NVIDIA repository and install an applicable driver from it.

> **Warning:** We do not recommend using a `.run` installer from NVIDIA Driver Downloads instead, as its drivers may conflict with the drivers installed by packages.

To install the `530` driver from a repository, do the following:

1. Install the repository signature key:

```
arch=$(uname -m); version=$(. /etc/os-release; echo $ID$VERSION_ID | sed -r 's/\.//g
↪'); sudo bash -c \
"sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/cuda/
↪repos/$ID$version/$arch/3bf863cc.pub \
&& apt update"
```

2. Install `aptitude`:

```
sudo apt-get install aptitude
```

3. Install `nvidia-driver-530`:

```
sudo aptitude install nvidia-driver-530
```

4. Reboot the system:

```
sudo reboot
```

Install Docker Products

Docker products must be installed on both CPU and GPU servers. Do the following:

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS.

```
sudo apt-get update

sudo apt-get install \
  ca-certificates \
  curl \
  gnupg \
  lsb-release
```

> **Tip:** You may receive the following errors when running `sudo apt-get install \`:

```
E: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporarily␣
↪unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is␣
↪another process using it?
```

There are two ways to resolve them:

1. Forcibly terminate all the `apt-get` processes currently running in the system.

```
sudo killall apt apt-get
```

2. If the previous command didn't help, run the set of commands below. It's fine if some of the to-be-deleted directories do not exist — just keep going.

```
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock
sudo rm /var/lib/dpkg/lock-frontend
sudo dpkg --configure -a
```

---

2. Add the Docker's official GPG key (GNU Privacy Guard key) to the host.

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /
↪etc/apt/keyrings/docker.gpg
```

3. Set up the Docker repository.

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]␣
↪https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/
↪null
```

4. Update the `apt` package index one more time.

```
sudo apt-get update
```

Tip: If you have received a GPG error when running this command, try granting read permission for the Docker public key file before updating the package index.

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
sudo apt-get update
```

---

5. Install `24.*` versions of Docker products.

```
sudo apt-get install docker-ce=5:24* docker-ce-cli=5:24* containerd.io docker-
↪buildx-plugin
```

6. Check whether the Docker installation was a success. The following command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

```
sudo docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
```

(continues on next page)

(continued from previous page)

```
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

7. Install `docker-compose`.

```
sudo curl -SL https://github.com/docker/compose/releases/download/v2.15.1/docker-
↪compose-linux-x86_64 -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

8. Perform the Docker Engine post-installation procedures to ease your future work with Docker and *FindFace Multi containers*. Once you can manage Docker as a non-root user, you don't have to apply `sudo` in the commands related to Docker.

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

9. Configure the Docker network.

```
sudo su
BIP=10.$((RANDOM % 256)).$((RANDOM % 256)).1
cat > /etc/docker/daemon.json <<EOF
{
    "bip": "$BIP/24",
    "fixed-cidr": "$BIP/24"
}
EOF
```

GPU: Install NVIDIA Container Runtime

To deploy containerized GPU-accelerated FindFace Multi, you need NVIDIA Container Runtime. We recommend installing NVIDIA Container Toolkit that includes this runtime. Do the following:

1. Specify the repository and install NVIDIA Container Toolkit from it by executing the following commands.

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
      && curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg -
↪-dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
      && curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/
↪libnvidia-container.list | \
            sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-
↪toolkit-keyring.gpg] https://#g' | \
            sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
sudo apt-get update
sudo apt-get install -y nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

2. Configure the Docker network. Configure usage of the NVIDIA Container Runtime installed along with NVIDIA Container Toolkit.

```
sudo su
BIP=10.$((RANDOM % 256)).$((RANDOM % 256))
cat > /etc/docker/daemon.json <<EOF
{
   "default-address-pools":
        [
                {"base":"$BIP.0/16","size":24}
        ],
    "bip": "$BIP.1/24",
    "fixed-cidr": "$BIP.0/24",
    "runtimes": {
         "nvidia": {
             "path": "nvidia-container-runtime",
             "runtimeArgs": []
         }
    },
    "default-runtime": "nvidia"
}
EOF
```

3. Restart Docker.

```
systemctl restart docker
```

Now you are all set to install FindFace Multi. Refer to the following sections:

- *FindFace Multi Standalone Automated Deployment*
- *FindFace Video Worker Deployment on Remote Hosts*
- *Fully Customized Installation*

2.3.2 CentOS 7 Server Preparation

To prepare a server on CentOS 7 for the FindFace Multi deployment, follow the instructions below minding the sequence.

Note: For other platforms, please refer to the following resources:

- NVIDIA drivers
- Docker Engine
- Docker Compose
- NVIDIA Container Toolkit

In this section:

Install Updates

1. Run updates and reboot the server.

```
sudo yum update
sudo reboot
```

2. Install `fuse` by running the following command.

```
sudo yum -y install fuse
```

GPU: Install NVIDIA Drivers

The first step of the server preparation is the NVIDIA driver installation. It's applicable only for the GPU configuration. Go straight to the *Docker installation* if your configuration is CPU-accelerated.

With the GPU-accelerated FindFace Multi, you'll need the NVIDIA driver version `530` or above. Download a relevant `.run` installer from NVIDIA Driver Downloads.

Since you use a `.run` installer from NVIDIA, the following dependencies must be installed:

```
sudo yum install kernel-devel gcc kernel-headers
```

Install Docker Products

Docker products must be installed on both CPU and GPU servers. Do the following:

1. Install the `yum-utils` package (which provides the `yum-config-manager` utility) and set up the repository.

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-
↪ce.repo
```

2. Install `24.*` versions of Docker products.

```
sudo yum install docker-ce-3:24* docker-ce-cli-1:24* docker-ce-rootless-extras-24*␣
↪containerd.io docker-buildx-plugin docker-compose-plugin
```

3. Start and enable Docker.

```
sudo systemctl start docker
sudo systemctl enable docker
```

4. Check whether the Docker installation was a success. The following command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

```
sudo docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

5. Install `docker-compose`.

```
sudo curl -SL https://github.com/docker/compose/releases/download/v2.15.1/docker-
↪compose-linux-x86_64 -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

6. Perform the Docker Engine post-installation procedures to ease your future work with Docker and *FindFace Multi containers*. Once you can manage Docker as a non-root user, you don't have to apply `sudo` in the commands related to Docker.

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

7. Configure the Docker network and `devicemapper` usage.

```
sudo su
BIP=10.$((RANDOM % 256)).$((RANDOM % 256)).1
cat > /etc/docker/daemon.json <<EOF
{
    "bip": "$BIP/24",
    "fixed-cidr": "$BIP/24",
    "storage-driver": "devicemapper"
}
EOF
```

GPU: Install NVIDIA Container Runtime

To deploy containerized GPU-accelerated FindFace Multi, you need NVIDIA Container Runtime. We recommend installing NVIDIA Container Toolkit that includes this runtime. Do the following:

1. Specify the repository and install NVIDIA Container Toolkit from it by executing the following commands.

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
   && curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/
↪libnvidia-container.repo | sudo tee /etc/yum.repos.d/nvidia-container-toolkit.repo
sudo yum clean expire-cache
sudo yum install -y nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

2. Configure the Docker network, `devicemapper` usage, and usage of the NVIDIA Container Runtime installed along with NVIDIA Container Toolkit.

```
sudo su
BIP=10.$((RANDOM % 256)).$((RANDOM % 256))
cat > /etc/docker/daemon.json <<EOF
{
   "default-address-pools":
        [
                {"base":"$BIP.0/16","size":24}
        ],
    "bip": "$BIP.1/24",
    "fixed-cidr": "$BIP.0/24",
```

(continues on next page)

(continued from previous page)

```
    "runtimes": {
         "nvidia": {
             "path": "nvidia-container-runtime",
             "runtimeArgs": []
         }
    },
    "default-runtime": "nvidia",
    "storage-driver": "devicemapper"
}
EOF
```

3. Restart Docker.

```
systemctl restart docker
```

Now you are all set to install FindFace Multi. Refer to the following sections:

- *FindFace Multi Standalone Automated Deployment*
- *FindFace Video Worker Deployment on Remote Hosts*
- *Fully Customized Installation*

2.3.3 Debian 11 Server Preparation

To prepare a server on Debian 11 for the FindFace Multi deployment, follow the instructions below minding the sequence.

Note: For other platforms, please refer to the following resources:

- NVIDIA drivers
- Docker Engine
- Docker Compose
- NVIDIA Container Toolkit

In this section:

Install FUSE

1. Install FUSE (Filesystem in Userspace) by running the following command.

```
sudo apt install fuse -y
```

GPU: Install NVIDIA Drivers

The first step of the server preparation is the NVIDIA driver installation. It's applicable only for the GPU configuration. Go straight to the *Docker installation* if your configuration is CPU-accelerated.

With the GPU-accelerated FindFace Multi, you'll need the NVIDIA driver version `530` or above. Download a relevant `.run` installer from NVIDIA Driver Downloads.

Since you use a `.run` installer from NVIDIA, the following dependencies must be installed:

```
sudo apt install linux-headers-$(uname -r)
sudo apt install build-essential
```

Install Docker Products

Docker products must be installed on both CPU and GPU servers. Do the following:

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS.

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
```

2. Add the Docker's official GPG key (GNU Privacy Guard key) to the host.

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /
↪etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

3. Set up the Docker repository.

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]␣
↪https://download.docker.com/linux/debian \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

4. Update the `apt` package index one more time.

```
sudo apt-get update
```

5. Install `24.*` versions of Docker products.

```
sudo apt-get install docker-ce=5:24* docker-ce-cli=5:24* containerd.io docker-
↪buildx-plugin
```

6. Check whether the Docker installation was a success.

```
sudo docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

7. Install `docker-compose`.

```
sudo curl -SL https://github.com/docker/compose/releases/download/v2.15.1/docker-
↪compose-linux-x86_64 -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

8. Perform the Docker Engine post-installation procedures to ease your future work with Docker and *FindFace Multi containers*. Once you can manage Docker as a non-root user, you don't have to apply `sudo` in the commands related to Docker.

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

9. Configure the Docker network.

```
sudo su
BIP=10.$((RANDOM % 256)).$((RANDOM % 256)).1
cat > /etc/docker/daemon.json <<EOF
{
    "bip": "$BIP/24",
    "fixed-cidr": "$BIP/24"
}
EOF
```

GPU: Install NVIDIA Container Runtime

To deploy containerized GPU-accelerated FindFace Multi, you need NVIDIA Container Runtime. We recommend installing NVIDIA Container Toolkit that includes this runtime. Do the following:

1. Specify the repository and install NVIDIA Container Toolkit from it by executing the following commands.

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
      && curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg -
↪-dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
      && curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/
↪libnvidia-container.list | \
            sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-
↪toolkit-keyring.gpg] https://#g' | \
            sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
sudo apt-get update
sudo apt-get install -y nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

2. Switch to the superuser account.

```
sudo su
```

3. Configure the Docker network. Configure usage of the NVIDIA Container Runtime installed along with NVIDIA Container Toolkit.

```
BIP=10.$((RANDOM % 256)).$((RANDOM % 256))
cat > /etc/docker/daemon.json <<EOF
{
   "default-address-pools":
        [
                {"base":"$BIP.0/16","size":24}
        ],
    "bip": "$BIP.1/24",
    "fixed-cidr": "$BIP.0/24",
    "runtimes": {
         "nvidia": {
             "path": "nvidia-container-runtime",
             "runtimeArgs": []
         }
    },
    "default-runtime": "nvidia"
}
EOF
```

4. Restart Docker.

```
systemctl restart docker
```

Now you are all set to install FindFace Multi. Refer to the following sections:

- *FindFace Multi Standalone Automated Deployment*
- *FindFace Video Worker Deployment on Remote Hosts*
- *Fully Customized Installation*

2.3.4 FindFace Multi Standalone Automated Deployment

To deploy FindFace Multi as a standalone server in one run, follow the instructions below. Being the simplest, this installation type is excellent to start off your work with FindFace Multi. Be sure to meet the *system requirements* and *prepare the server* first.

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.

Do the following:

1. Download the installer file `findface-*.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.

   Note: Be sure to specify the actual file name instead of `findface-*`.

   ```
   chmod +x findface-*.run
   ```

4. Execute the `.run` file.

   ```
   sudo ./findface-*.run
   ```

   The installer will ask you a few questions and perform several automated checks to ensure that the host meets the system requirements. After filling out each prompt, press `Enter`. The questions and answers are the following:

   1. Q: `Which product should be installed?`

      A: 1

      ```
      1. [multi   ]  FindFace Multi
      2. [server  ]  FindFace Server
      3. [video-worker]  FindFace Video Worker

      (default: multi)
      product> 1
      ```

   2. Q: `Please choose installation type:`

      A: 1

      ```
      -  1 [stand-alone ]  Single Server
      -  2 [multi-worker]  Single Server, Multiple video workers
      -  3 [images      ]  Don't install anything, just load the images
      -  4 [custom      ]  Fully customized installation

      (default: stand-alone)
      type> 1
      ```

   3. Q: `Directory to install into:`

      A: Specify the FindFace Multi installation directory. By default, it's `/opt/findface-multi`. Press `Enter` to confirm it. Otherwise, specify the directory of your choice and press `Enter`.

```
Directory to install into:
(default: /opt/findface-multi)
dest_dir>
```

4. Q: `Do you want to install Video Recorder?(y/n)`

   A: `y` or `n`, subject to your needs.

```
Do you want to install Video Recorder?(y/n)
install_video_recorder> n
```

5. Q: `Found X interface(s). Which one should we announce as our external address?`

   A: Choose the interface that you are going to use as the instance IP address.

```
Found 1 interface(s). Which one should we announce as our external address?

-  1 [lo      ]  127.0.0.1
-  2 [ens3    ]  192.168.112.254

(default: 192.168.112.254)
ext_ip.advertised> 2
```

6. Q: `Which variant of Video Worker should be installed?`

   A: Specify the `findface-video-worker` package type, CPU or GPU.

```
Which variant of Video Worker should be installed?

-  1 [cpu]  CPU-based implementation, slower but doesn't require GPU
-  2 [gpu]  CUDA-based implementation of video detector, requires NVIDIA GPU

(default: cpu)
findface-video-worker.variant> 1
```

7. Q: `Which variant of Extraction API should be installed?`

   A: Specify the `findface-extraction-api` package type, CPU or GPU.

```
Which variant of Extraction API should be installed?

-  1 [cpu]  CPU-only implementation, slower but doesn't require GPU
-  2 [gpu]  CUDA-based implementation, faster, requires NVIDIA GPU (supports␣
↪both CPU and GPU models)

(default: cpu)
findface-extraction-api.variant> 1
```

8. Q: `Do you want to configure detectors and features right now?(y/n)`

   A: We recommend you to install and configure the object detection and object attribute recognition functionality straightaway during the installation. Answer `y` to initiate the process. You can skip it, though, by answering `n` and perform the necessary steps later, following the instructions in the *Enable Face and Face Attribute Recognition*, *Enable Vehicle and Vehicle Attribute Recognition*, and *Enable Body and Body Attribute Recognition* sections.

```
Do you want to configure detectors and features right now?(y/n)
configure> y
```

9. Q: `Please select detectors to install:`

   A: This question appears if you have requested installation and configuration of the object detection and object attribute recognition functions. In the multiple choice form, the face detector is checked by default. Enter a relevant number to select an unpicked detector and vice versa. For example, to add the body and vehicle detector to your configuration , enter `2 3`. Enter `done` to proceed.

```
Please select detectors to install:

-  1 [v]  Face
-  2 [ ]  Body
-  3 [ ]  Car

Enter keyword to select matching choices or -keyword to clear selection.
Enter "done" to save your selection and proceed to another step.
detectors>2 3
-  1 [v]  Face
-  2 [v]  Body
-  3 [v]  Car

Enter keyword to select matching choices or -keyword to clear selection.
Enter "done" to save your selection and proceed to another step.
detectors> done
```

10. Q: `Enable liveness and attempt to continue installation?(y/n)`

    A: This question appears if you have requested installation and configuration of the object detection and object attribute recognition functions. To install the embedded face liveness detector, enter `y`. Enter `n` otherwise.

```
Enable liveness and attempt to continue installation?(y/n)
enable_liveness> y
```

11. Q: `Please select face features to install:`

    A: This question appears if you have requested installation and configuration of the face detection and face attribute recognition functions. By default, all face attributes are subject to installation. Answer `done` to confirm it. If some attribute is not necessary, you can enter the keyword (number) related to it. For example, enter `7` to exclude the head pose recognition. Then enter `done`.

    Questions on body and vehicle attribute recognition will be analogous, with multiple choice forms depending on the requested detector type.

```
Please select face features to install:

-  1 [v]  Age
-  2 [v]  Gender
-  3 [v]  Emotions
-  4 [v]  Beard
-  5 [v]  Glasses
-  6 [v]  Medicine masks
-  7 [v]  Headpose
```

(continues on next page)

(continued from previous page)

```
Enter keyword to select matching choices or -keyword to clear selection.
Enter "done" to save your selection and proceed to another step.
face_features> done
```

Questions on body and vehicle attribute recognition will be analogous, with multiple choice forms depending on the requested detector type.

12. Q: `Please set findface-multi admin password`

    A: Set the Super Administrator (`superuser`) password.

```
Please set findface-multi admin password
findface-multi-admin-password> admin
```

The installer will pull the FindFace Multi images from the Ntechlab registry and start the following services in Docker containers:

Service	Container	Configuration
findface-multi-pause	findface-multi-findface-multi-pause-1	Started
nats-jetstream	findface-multi-nats-jetstream-1	Started
mongodb	findface-multi-mongodb-1	Installed with Video Recorder. Started
findface-ntls	findface-multi-findface-ntls-1	Started
nats	findface-multi-nats-1	Started
postgresql	findface-multi-postgresql-1	Started
memcached	findface-multi-memcached-1	Started
findface-upload	findface-multi-findface-upload-1	Started
etcd	findface-multi-etcd-1	Started
findface-sf-api	findface-multi-findface-sf-api-1	Started
findface-extraction-api	findface-multi-findface-extraction-api-1	Started (CPU/GPU-acceleration)
findface-tarantool-server-shard-*	findface-multi-findface-tarantool-server-shard-*-1	Started. The number of 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, if the first obtained value is greater.
findface-video-manager	findface-multi-findface-video-manager-1	Started
findface-counter	findface-multi-findface-counter-1	Started

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                              #
################################################################################
- all configuration and data is stored in /opt/findface-multi
- upload your license to http://192.168.0.90/#/license/
- user interface: http://192.168.0.90/
superuser:      admin
documentation:  http://192.168.0.90/doc/
Installation logfile: /tmp/installer_run_2570703278.log
```

5. Perform the *post-deployment procedures*.

Tip: To install the same configuration of FindFace Multi on a different host, use the automatic deployment from the installation file. In this case, you won't have to answer the installation questions again. The exact path to the installation file is displayed right after the last question, before the installer starts active progress:

```
[I 2023-02-09 11:13:37,187 main:142] Your answers were saved to /tmp/findface-installer-
↪p01n9sn3.json
```

Be sure to remove fields `*.config`, `exp_ip`, and `int_ip` before installing FindFace Multi on a host with a different IP address.

To launch the automatic deployment from the `/tmp/<findface-installer-*>.json` file, execute:

```
sudo ./<findface-*>.run -f /tmp/<findface-installer-*>.json
```

2.3.5 FindFace Video Worker Deployment on Remote Hosts

You can install `findface-video-worker` instances apart from the FindFace Multi principal server when creating a multi-host environment.

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 `/opt/findface-multi/configs/findface-ntls/findface-ntls.yaml` configuration file on the server with `findface-ntls` and set `listen: 0.0.0.0:3133`. Restart the `findface-multi-findface-ntls-1` container.

```
sudo vi /opt/findface-multi/configs/findface-ntls/findface-ntls.yaml

listen: 0.0.0.0:3133
```

```
sudo docker container restart findface-multi-findface-ntls-1
```

2. Allow accessing the `findface-video-manager` service from any IP address. To do so, open the `/opt/findface-multi/configs/findface-video-manager/findface-video-manager.yaml` con-

figuration 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-multi-findface-video-manager-1` container.

```
sudo vi /opt/findface-multi/configs/findface-video-manager/findface-video-manager.
↪yaml

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

```
sudo docker container restart findface-multi-findface-video-manager-1
```

3. On the FindFace Multi server, open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.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 /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.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 meet the *system requirements* and *prepare the server* first.

1. Download the installer file `findface-*.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.

Note: Be sure to specify the actual file name instead of `findface-*`.

```
chmod +x findface-*.run
```

4. Execute the `.run` file.

```
sudo ./findface-*.run
```

The installer will ask you a few questions and perform several automated checks to ensure that the host meets the system requirements. After filling out each prompt, press `Enter`. The questions and answers are the following:

1. Q: `Which product should be installed?`

   A: 3

```
1. [multi   ]  FindFace Multi
2. [server  ]  FindFace Server
3. [video-worker]  FindFace Video Worker

(default: multi)
product> 3
```

2. Q: `Which variant of Video Worker should be installed?`

   A: Specify the `findface-video-worker` package type, CPU or GPU.

```
Which variant of Video Worker should be installed?

-  1 [cpu]  CPU-based implementation, slower but doesn't require GPU
-  2 [gpu]  CUDA-based implementation of video detector, requires NVIDIA GPU
```

3. Q: `Found X interface(s). Which one should we announce as our inter-service communication address?`

   A: Choose the default interface `2` to connect to the FindFace Multi server. We don't recommend you to use localhost.

```
Found 1 interface(s). Which one should we announce as our external address?

-  1 [lo      ]  127.0.0.1
-  2 [ens3    ]  192.168.112.254

(default: 192.168.112.254)
inter_ip.advertised> 2
```

4. Q: `Please enter FF.Multi or FF.Server IP address:`

   A: Specify the FindFace Multi server IP address.

```
Please enter FF.Multi or FF.Server IP address:
server_addr> 192.168.112.25
```

After that, the installation process will automatically begin.

Important: If you chose to install `findface-ntls` and/or `findface-video-manager` on different hosts than that with `findface-multi-legacy`, specify their IP addresses in the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file after the installation.

```
/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml
```

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 install a `findface-video-worker` instance on a different host, use the automatic deployment from the installation file. In this case, you won't have to answer the installation questions again. The exact path to the installation file is displayed right after the last question before the installer starts active progress:

```
[I 2023-02-09 11:13:37,187 main:142] Your answers were saved to /tmp/findface-installer-
↪p01n9sn3.json
```

To launch the automatic deployment from the `/tmp/<findface-installer-*>.json` file, execute:

```
sudo ./<findface-*>.run -f /tmp/<findface-installer-*>.json
```

2.3.6 Deploy Video Recorder Step-by-Step

There are the following ways to deploy Video Recorder:

- Automatically during the FindFace Multi deployment from a console installer. See *FindFace Multi Standalone Automated Deployment* for details.
- Step-by-step. Recommended in a multi-host environment.

This section will guide you through the Video Recorder step-by-step deployment. Follow the instructions below minding the sequence.

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

In this section:

Install APT Repository

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

1. Download the installer file `findface-*.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-*.run
   ```

4. Execute the `.run` file.

   ```
   sudo ./findface-*.run
   ```

The installer will ask you a few questions and perform several automated checks to ensure that the host meets the system requirements. After filling out each prompt, press `Enter`. The questions and answers are the following:

1. Q: `Which product should be installed?`

   A: 1

```
Which product should be installed?

  1. [multi   ]  FindFace Multi
  2. [server  ]  FindFace Server
  3. [video-worker]  FindFace Video Worker

(default: multi)
product> 1
```

2. Q: `Please choose installation type:`

   A: 3

```
-  1 [stand-alone ]  Single Server
-  2 [multi-worker]  Single Server, Multiple video workers
-  3 [images      ]  Don't configure or start anything, just load␣
↪the images and copy the models
-  4 [custom      ]  Fully customized installation

(default: stand-alone)
type> 3
```

3. Q: `Directory to install into:`

   A: Specify the FindFace Multi installation directory. By default, it's `/opt/findface-multi`. Press `Enter` to confirm it. Otherwise, specify the directory of your choice and press `Enter`.

```
Directory to install into:
(default: /opt/findface-multi)
dest_dir>
```

4. Q: `Select models to install. Note that you will need to accordingly edit extraction-api and tntapi configuration files. At least one of recognition models has to be enabled.`

   A: By default, all neural network models are subject to installation. You can leave it as is by entering `done`, or select specific models. To do so, deselect all those on the list by entering `-*` in the command line, then select the required models by entering their sequence numbers (keyword): for example, `1 3 4`. Enter `done` to save your selection and proceed to another step. If FindFace Multi is deployed earlier, the models do not need to be installed.

```
Select models to install.
Note that you will need to accordingly edit extraction-api and␣
↪tntapi configuration files.
At least one of recognition models has to be enabled.

-  1 [ ] ./models/carattr/carattr.categories.v0.cpu.fnk
```

(continues on next page)

(continued from previous page)

```
...
- 91 [ ] ./models/pedrec/pedrec.clio.gpu.fnk

Enter keyword to select matching choices or -keyword to clear␣
↪selection.
Enter "done" to save your selection and proceed to another step.
findface-data.models> done
```

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

Install Main Components

To install the Video Recorder components, do the following:

1. Open the `/opt/findface-multi/docker-compose.yaml` configuration file. Add the `findface-video-storage` and `findface-video-streamer` services.

```
sudo vi /opt/findface-multi/docker-compose.yaml


...
findface-video-storage:
  command: [--config=/etc/findface-video-storage.conf]
  depends_on: [mongodb]
  image: docker.int.ntl/ntech/universe/video-storage:ffserver-8.221216
  network_mode: service:pause
  restart: always
  volumes: ['./configs/findface-video-storage/findface-video-storage.yaml:/etc/
↪findface-video-storage.conf:ro']
findface-video-streamer:
  command: [--config=/etc/findface-video-streamer-cpu.ini]
  depends_on: [findface-ntls, mongodb]
  image: docker.int.ntl/ntech/universe/video-streamer-cpu:ffserver-8.221216
  network_mode: service:pause
  restart: always
  volumes: ['./configs/findface-video-streamer/findface-video-streamer.yaml:/etc/
↪findface-video-streamer-cpu.ini:ro',
    './cache/findface-video-streamer:/var/cache/findface/video-streamer']
```

2. Create directories.

```
sudo mkdir -p /opt/findface-multi/configs/findface-video-storage/ /opt/findface-
↪multi/configs/findface-video-streamer/
```

3. Create a configuration file `/opt/findface-multi/configs/findface-video-storage/findface-video-storage.yaml` and add the following:

```
sudo vi /opt/findface-multi/configs/findface-video-storage/findface-video-storage.
↪yaml

listen: :18611
debug: false
external-address: http://ip_address:18611/
```

(continues on next page)

(continued from previous page)

```
streamer:
  endpoints:
  - 127.0.0.1:9000
chunk-storage:
  type: webdav
  webdav:
    timeouts:
      connect: 5s
      response_header: 30s
      overall: 35s
      idle_connection: 10s
    max-idle-conns-per-host: 20
    keepalive: 24h0m0s
    trace: false
    upload-url: http://127.0.0.1:3333/uploads/video_storage
  s3:
    endpoint: ''
    bucket-name: ''
    access-key: ''
    secret-access-key: ''
    secure: true
    region: ''
    public-url: ''
    operation-timeout: 30
  localfs:
    directory: ''
meta-storage:
  mongo-uri: mongodb://127.0.0.1
  database: video-storage
  timings:
    connect: 3s
```

Note: <ip_address> in the section `external-address` must be substituted into your ip address for FindFace Multi.

4. Create a configuration file `/opt/findface-multi/configs/findface-video-streamer/findface-video-streamer.yaml` and add the following:

```
sudo vi /opt/findface-multi/configs/findface-video-streamer/findface-video-streamer.
↪yaml

streamer:
  port: 9000
  max_backpressure: 33554432
  io_buffer_size: 524288
video_storage:
  url: http://127.0.0.1:18611
  timeout: 6
cache:
  dir: /var/cache/findface/video-streamer
```

5. Start FindFace Multi containers:

```
cd /opt/findface-multi/
docker-compose up -d
```

Video Recorder is now successfully deployed. To configure Video Recorder after the deployment, follow *these instructions*.

2.3.7 Fully Customized Installation

The FindFace Multi developer-friendly installer provides you with a few installation options, including the fully customized installation. This option is mostly used when deploying FindFace Multi in a highly distributed environment and requires a high level of knowledge and experience.

To initiate the fully customized installation, do the following:

1. Download the installer file `findface-*.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.

Note: Be sure to specify the actual file name instead of `findface-*`.

```
chmod +x findface-*.run
```

4. Execute the `.run` file.

```
sudo ./findface-*.run
```

The installer will ask you a few questions and perform several automated checks to ensure that the host meets the system requirements. After filling out each prompt, press `Enter`. The questions and answers are the following:

1. Q: `Which product should be installed?`

   A: 1

```
Which product should be installed?

  1. [multi   ]  FindFace Multi
  2. [server  ]  FindFace Server
  3. [video-worker]  FindFace Video Worker

(default: multi)
product> 1
```

2. Q: `Please choose installation type:`

   A: 4

```
Please choose installation type:

-  1 [stand-alone ]  Single Server
-  2 [multi-worker]  Single Server, Multiple video workers
-  3 [images      ]  Don't configure or start anything, just load the images␣
```

(continues on next page)

(continued from previous page)

```
↪and copy the models
-  4 [custom      ]  Fully customized installation

(default: stand-alone)
type> 4
```

3. Q: `Directory to install into:`

   A: Specify the FindFace Multi installation directory. The default suggestion is `/opt/findface-multi`. Press `Enter` to confirm it. Otherwise, specify the directory of your choice and press `Enter`.

```
Directory to install into:
(default: /opt/findface-multi)
dest_dir>
```

4. Q: `Please enter path to docker-compose binary:`

   A: Specify the actual path to the `docker-compose` binary. The default suggestion is `/usr/local/bin/docker-compose` and it is the path you get when you install `docker-compose` as *prescribed by this document*. Press `Enter` to confirm it. Otherwise, specify another path and press `Enter`.

```
Please enter path to docker-compose binary
(default: /usr/local/bin/docker-compose)
docker_compose>
```

5. Q: `Found X interface(s). Which one should we announce as our external address?`

   A: Choose the interface that you are going to use as the instance IP address.

```
Found 1 interface(s). Which one should we announce as our external address?

-  1 [lo      ]  127.0.0.1
-  2 [ens3    ]  192.168.112.254

(default: 192.168.112.254)
ext_ip.advertised> 2
```

6. Q: `Found X interface(s). Which one should we announce as our inter-service communication address?`

   A: Choose the interface for inter-service communication.

```
Found 1 interface(s). Which one should we announce as our inter-service␣
↪communication address?

-  1 [lo      ]  127.0.0.1
-  2 [ens3    ]  192.168.112.254

(default: 192.168.112.254)
inter_ip.advertised> 2
```

7. Q: `Please select FindFace Multi components to install:`

   A: Choose FindFace components to install. By default, all components are subject to installation. You can leave it as is by entering `done`, or select specific components. 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.

> **Warning:** We do not recommend excluding the `pause` component from installation, as this will leave the remaining components without a network namespace to attach to.
>
> If you exclude the `pause` component intentionally, make sure to edit the `/opt/findface-multi/docker-compose.yaml` file and provide a host name for each service in the `network_mode` parameter.

```
Please select FindFace Multi components to install:

-  1 [v]  findface-data        - Face recognition models
...
...

Enter keyword to select matching choices or -keyword to clear selection.
Enter "done" to save your selection and proceed to another step.
components> done
```

8. Specific questions related to the selected components: acceleration type, the required number of component instances, neural network models, etc. If you are experiencing a difficulty answering them, try to find an answer in this documentation, or submit your question to support@ntechlab.com.

9. Q: `Please set findface-multi admin password`

   A: Set the Super Administrator (`superuser`) password.

```
Please set findface-multi admin password
findface-multi-admin-password> admin
```

The installer will pull the FindFace Multi images from the Ntechlab registry and start the associated services in Docker containers.

5. Perform the *post-deployment procedures*.

Tip: To install the same configuration of FindFace Multi on a different host, use the automatic deployment from the installation file. In this case, you won't have to answer the installation questions again. The exact path to the installation file is displayed right after the last question, before the installer starts active progress:

```
[I 2023-02-09 11:13:37,187 main:142] Your answers were saved to /tmp/findface-installer-
↪p01n9sn3.json
```

Be sure to remove fields `*.config`, `exp_ip`, and `int_ip` before installing FindFace Multi on a host with a different IP address.

To launch the automatic deployment from the `/tmp/<findface-installer-*>.json` file, execute:

```
sudo ./<findface-*>.run -f /tmp/<findface-installer-*>.json
```

2.3.8 Guide to Typical Multi-Host Deployment

This section is all about deploying FindFace Multi in a multi-host environment.

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

Important: This section doesn't cover the Video Recorder deployment. You can find a step-by-step instruction on this subject *here*.

The reasons for deploying FindFace Multi in a multi-host environment 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`, `pause`, 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. This procedure requires a high level of expertise and some extra coding. Please do not hesitate to contact our experts for help (support@ntechlab.com).

To deploy FindFace Multi in a multi-host 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 (don't forget to *prepare a server* prior the FindFace Multi deployment):
   - 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                              #
################################################################################
- all configuration and data is stored in /opt/findface-multi
- upload your license to http://172.20.77.17/#/license/
- user interface: http://172.20.77.17/
  superuser:      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 `superuser` 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 `/opt/findface-multi/configs/findface-ntls/findface-ntls.yaml` configuration file and set `listen:  0.0.0.0:3133`. Restart the `findface-multi-findface-ntls-1` container.

```
sudo vi /opt/findface-multi/configs/findface-ntls/findface-ntls.yaml

listen: 0.0.0.0:3133
```

```
sudo docker container restart findface-multi-findface-ntls-1
```

4. Allow accessing the `findface-video-manager` service from any IP address. To do so, open the `/opt/findface-multi/configs/findface-video-manager/findface-video-manager.yaml` con-

figuration file and set `listen: 0.0.0.0:18810` and `rpc:listen: 0.0.0.0:18811`. Restart the `findface-multi-findface-video-manager-1` container.

```
sudo vi /opt/findface-multi/configs/findface-video-manager/findface-video-manager.
↪yaml

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

```
sudo docker container restart findface-multi-findface-video-manager-1
```

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-*>.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 `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file after the installation.

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-
↪worker.yaml
```

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.

```
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`, `findface-data`, and `pause`. To make a selection, first, deselect all the listed components by entering `-*` in the command line, then select `findface-extraction-api`, `findface-data`, and `pause` by entering their sequence number (keyword). Enter `done` to save your selection and proceed to another step.

  Note: The `pause` component keeps information about other components' network namespaces. It's essential that you install it.

- Type of `findface-extraction-api` acceleration: CPU or GPU.
- Modification of the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` 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, vehicle and vehicle attributes, and body and body attributes. 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 Networks Summary*, *Enable Face and Face Attribute Recognition*, *Enable Vehicle and Vehicle Attribute Recognition*, *Enable Body and Body Attribute Recognition* for details.

- To move the principal `findface-extraction-api` instance to another host, in the `/opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml` configuration file specify the IP address of the extraction server host and set `listen: 0.0.0.0:18411`.

```
listen: 0.0.0.0:18411
extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  keepalive: 24h0m0s
  trace: false
  extraction-api: http://172.20.77.19:18666
```

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-*>.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 running `findface-sf-api` service 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*.

   ---

2. On the designated server with the installed `findface-sf-api` instance create a `nginx` folder that contains the `extapi.conf` file in the `/opt/findface-multi/configs/` directory. Make sure that the `extapi.conf` file includes the information like in the example below. 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;
   }
   ```

3. Define the Nginx service in the `docker-compose.yaml` file. To do that, add the container with Nginx image to the `docker-compose.yaml` file:

   ```
   sudo vi /opt/findface-multi/docker-compose.yaml

   nginx:
     image: nginx:latest
     ports:
       - 18667:18667
   ```

(continues on next page)

(continued from previous page)

```
  volumes:
    - ./configs/nginx/extapi.conf:/etc/nginx/conf.d/default.conf:ro
```

4. In the `findface-sf-api` configuration file, specify the distributor address:

```
sudo vi /opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml

listen: 0.0.0.0:18411
...
extraction-api: http://172.168.1.9:18667
```

5. Restart the containers.

```
cd /opt/findface-multi/
sudo docker-compose down
sudo docker-compose up -d
```

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

```
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.
↪yaml

listen: 172.168.1.10:18666
```

7. Restart the `findface-multi-findface-extraction-api-1` container on the principal server and each additional extraction server.

```
sudo docker container restart findface-multi-findface-extraction-api-1
```

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*.

Deploy Additional Database Servers

The `findface-tarantool-server` component connects the Tarantool-based feature vector database and the `findface-sf-api` component, transferring search results from the database to `findface-sf-api` for further processing.

To increase search speed, you can allocate several additional servers to the feature vector database and create multiple `findface-tarantool-server` shards on each additional server. The concurrent functioning of multiple shards will lead to a remarkable increase in performance, as each shard can handle up to approximately 10,000,000 feature vectors.

To deploy additional database servers, do the following:

1. Install the `findface-tarantool-server` component on the first designated server. The `pause` component should already be installed on the server. If not, install it along with the `findface-tarantool-server` component. Answer the installer questions as follows:
   - Product to install: `FindFace Multi`.
   - Installation type: `Fully customized installation`.

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

After that, the installation process will automatically begin.

As a result of the installation, the `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 if the first obtained value is greater.

2. Use the created `/tmp/<findface-installer-*>.json` file to install `findface-tarantool-server` on other servers without answering the questions again. To do so, execute:

```
sudo ./<findface-*>.run -f /tmp/<findface-installer-*>.json
```

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

```
sudo cat /opt/findface-multi/configs/findface-tarantool-server/*shard-* | grep -E ".
↪start|(listen =)"
```

You will get the following result:

```
    listen = '127.0.0.1:32001',
FindFace.start("127.0.0.1", 8101, {
    listen = '127.0.0.1:32002',
FindFace.start("127.0.0.1", 8102, {
    listen = '127.0.0.1:32003',
FindFace.start("127.0.0.1", 8103, {
    listen = '127.0.0.1:32004',
FindFace.start("127.0.0.1", 8104, {
```

The port numbers are `8101`, `8102`, etc.

4. Allow each shard access the `findface-ntls` license server. To do that, modify the configuration file for each shard like in the example below. For each shard, make sure to set `0.0.0.0` in the `FindFace.start` section.

```
sudo vi /opt/findface-multi/configs/findface-tarantool-server/shard-00*

-- host:port to bind, HTTP API
FindFace = require("FindFace")
FindFace.start("0.0.0.0", 8101, {
    license_ntls_server="172.23.218.110:3133",
    replication = replication_master,
    spaces = spaces
})
```

5. Open the `/opt/findface-multi/configs/findface-ntls/findface-ntls.yaml` configuration file and set `listen: 0.0.0.0:3133`. Restart the `findface-multi-findface-ntls-1` container.

```
sudo vi /opt/findface-multi/configs/findface-ntls/findface-ntls.yaml

listen: 0.0.0.0:3133
license_dir: /ntech/license
```

(continues on next page)

(continued from previous page)

```
proxy: ''
ui: 0.0.0.0:3185
```

```
sudo docker container restart findface-multi-findface-ntls-1
```

6. Modify the `/opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml` configuration file. Set `listen: 0.0.0.0:18411` and specify shards. Restart the `findface-multi-findface-sf-api-1` container.

```
sudo vi /opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml

listen: 0.0.0.0:18411
extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  keepalive: 24h0m0s
  trace: false
  extraction-api: http://127.0.0.1:18666
storage-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  keepalive: 24h0m0s
  trace: false
  shards:
  - master: http://172.20.77.19:8101/v2/
    slave: ''
  - master: http://172.20.77.19:8102/v2/
    slave: ''
```

```
sudo docker container restart findface-multi-findface-sf-api-1
```

7. To apply migrations, restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

Configure Network

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

1. Open the `/opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml` configuration file:

```
sudo vi /opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml
```

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 object 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: ''
```

Restart the `findface-multi-findface-sf-api-1` container.

```
sudo docker container restart findface-multi-findface-sf-api-1
```

2. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

Specify the following parameters:

Parameter	Description
SERVICE_EXTERNAL_ADDRESS	External 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 use 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 /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py


...
# SERVICE_EXTERNAL_ADDRESS is prioritized for FFSecurity webhooks and Genetec␣
↪plugin.
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',
    ...
}
```

Restart the findface-multi-findface-multi-legacy-1 container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

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
```

(continues on next page)

(continued from previous page)

```
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 *Logging* for the step-by-step instructions.

2.3.9 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 `findface-*` file.

   Note: Be sure to specify the actual file name instead of `findface-*`.

   ```
   sudo ./findface-*.run
   ```

2. Product to install: `FindFace Multi`
3. Select the installation type: `Fully customized installation`.
4. Select a FindFace 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 the next step.
5. 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 `/opt/findface-multi/models/`. See *Neural Networks Summary*.

2.3.10 Post-deployment Procedures and How-to's

After you are finished with the FindFace Multi deployment, perform the procedures below.

In this section:

Specify Time Zone

The time zone on the FindFace Multi server determines the time in reports, logs, and names of such FindFace Multi artifacts as detection full frames and thumbnails, etc.

The time zone is specified in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file, either in the `Region/Country/City` or `Etc/GMT+H` format. The best way to do so is to copy/paste your time zone from this table on Wikipedia.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

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

Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

License Instance

FindFace Multi provides several licensing options. Whichever option you choose, you upload the FindFace Multi license file via the main web interface `http://<Host_IP_address>/#/license`. To access the web interface, use the `superuser` credentials.

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.

Refer to the *Licensing* section to learn about the licensing options available.

Configure Logging

By default, the FindFace Multi processes are logged to Docker container logs, which can be accessed via the `docker logs` and `docker service logs` commands. In addition, Docker uses the json-file logging driver, which caches container logs in JSON. You can configure Docker to use another logging driver, choosing from the multiple logging mechanisms available. See *Logging* to learn how to do it.

Useful Docker Commands

In order to efficiently and easily administer FindFace Multi, you must have extensive knowledge and skills with Docker. If you're new to Docker, get started with the commands below. Then explore the Docker documentation for additional skills.

- View all Docker containers, including the stopped ones:

  ```
  docker ps -a
  ```

  To get a more compact and understandable output, execute:

  ```
  docker ps -a --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.State}}"
  ```

  To extend the previous output, execute:

```
docker ps --format='{{json .}}' | jq
```

- Restart the Docker service:

```
sudo systemctl restart docker
```

- View a container log if the `journald` logging driver is *enabled*:

```
journalctl CONTAINER_NAME=findface-multi-findface-multi-legacy-1 -f
```

- Stop a Docker container:

```
sudo docker container stop <container_name>/<container_id>
```

  Stop all Docker containers:

```
sudo docker container stop $(sudo docker ps -a -q)
```

- Start a Docker container:

```
sudo docker container start <container_name>/<container_id>
```

  Start all Docker containers:

```
sudo docker container start $(sudo docker ps -a -q)
```

- The FindFace Multi `docker-compose.yaml` file can be viewed as such:

```
cat /opt/findface-multi/docker-compose.yaml
```

- FindFace Multi configuration files can be found here:

```
cd /opt/findface-multi/configs/
```

  Once you made changes to a configuration file, restart a relevant container by executing:

```
sudo docker container restart <container_name>/<container_id>
```

- Enter a running Docker container to execute a command in it:

```
sudo docker container exec -it <container_name> /bin/bash
```

- Stop and remove all FindFace Multi containers:

```
cd /opt/findface-multi
sudo docker-compose down
```

- Build, recreate, and start FindFace Multi containers:

```
cd /opt/findface-multi
sudo docker-compose up -d
```

2.3.11 Remove FindFace Multi Instance

FindFace Multi can be removed either by using the commands or by running the script.

Important: Make sure to *back up* your instance before deleting it if you plan to *restore* FindFace Multi and its data later on.

To remove your FindFace Multi instance, run the following commands, minding the sequence:

```
cd /opt/findface-multi
sudo docker-compose down
sudo docker system prune -a
cd ~
sudo rm -rf /opt/findface-multi
```

An alternative way to remove the instance is to run the following script:

```
sudo chmod +x /opt/findface-multi/uninstall.sh
cd /opt/findface-multi/
./uninstall.sh /opt/findface-multi/
```

2.4 Administration and Basic Configuration

2.4.1 Licensing

In this chapter:

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, vehicle).

   Note: The feature vectors are extracted from objects detected in the video, from images in the record index, and user photos, and when building so-called cluster centroids.

   The licensing scheme is the following:

   - Events: 1 event of video object detection = 1 object in a license.
   - Record Index: 1 photo in a record = 1 object in a license.
   - Clusters: 1 person = 1 object in a license.

    - Users: 1 photo of a user = 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` service.
4. Face attribute recognition: gender/age/emotions/glasses/beard/face mask/etc.
5. Body attribute recognition: clothing color/type/etc.
6. Vehicle attribute recognition: make/model/color/body style/etc.
7. License plate recognition.
8. Face liveness detection.
9. Video recording.
10. Integrations with partners.
11. Integration with external VMS.

You can choose between the following licensing methods:

- 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.

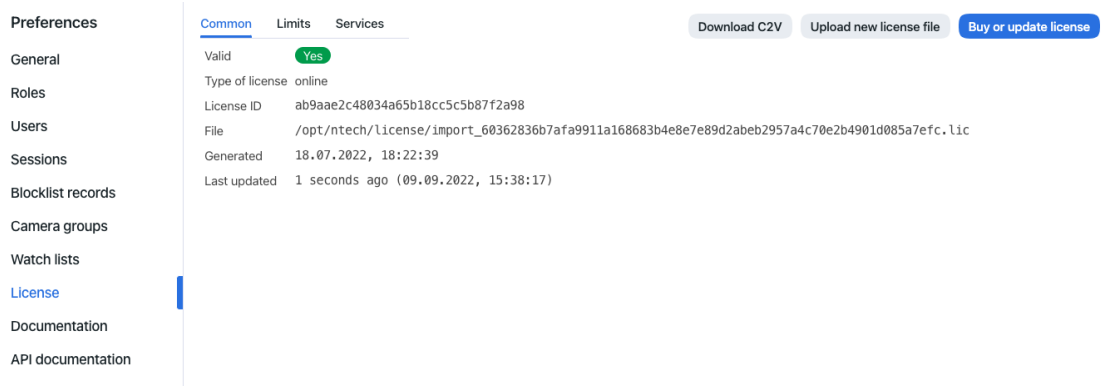
Note: It is possible to prolongate the off-grid period for up to 2 days. Inform your manager if you need that.

- The offline licensing via a USB dongle requires a USB port on the physical server with the `findface-ntls` service (license server in the *FindFace core*).
- The offline licensing via hardware fingerprint requires Sentinel drivers installed on the physical server with the `findface-ntls` service.

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.

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 *Settings -> License*.



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

Offline Licensing via USB dongle

To implement the licensing via a USB dongle, do the following:

1. Inform your manager that you intend to apply this licensing method and request your USB dongle and a license file.
2. Open the `/opt/findface-multi/docker-compose.yaml` configuration file.

```
sudo vi /opt/findface-multi/docker-compose.yaml
```

3. Add the line `privileged: true`. Mount the `/dev` directory into the `findface-multi-findface-ntls-1` container by listing it in the volumes of the `findface-ntls` section. As a result, the entire section will look as follows:

```
findface-ntls:
    command: [--config=/etc/findface-ntls.cfg]
    image: docker.int.ntl/ntech/universe/ntls:ffserver-8.221216
    network_mode: service:pause
    privileged: true
    restart: always
    user: root
    volumes: ['./configs/findface-ntls/findface-ntls.yaml:/etc/findface-ntls.cfg:ro
↪',
      './data/findface-ntls:/ntech/license', '/dev:/dev']
```

4. Create a new **udev** rule.
    1. Download the `95-grdnt.rules` file (e.g., into the `/home/username/tmp/` directory).
    2. Copy the `95-grdnt.rules` file into the `/etc/udev/rules.d/` directory.

```
sudo cp /home/username/tmp/95-grdnt.rules /etc/udev/rules.d/
```

5. Rebuild all FindFace Multi containers.

```
cd /opt/findface-multi

sudo docker-compose down

sudo docker-compose up -d
```

6. Insert the USB dongle into a USB port.
7. Upload the license file on the *License* tab.

Offline Licensing via Hardware Fingerprint

Note: Sentinel is a type of offline licenses that, unlike guardant licenses, do not require any physical media for its work.

Glossary:

- Sentinel is a software protection and licensing system by Thales. It allows you to implement offline licensing without access to a global server.
- The C2V file is a file, containing data about a hardware fingerprint of the client's machine, for binding the license only to this machine. This file is generated by the sentinel library. The C2V file is generated on the client's machine where the license key will be installed later.

To implement the fingerprint licensing, do the following:

1. Inform your manager that you intend to apply this licensing method and request your unique license id. The manager will also supply you with the `findface-sentinel-lib_*.deb` package necessary for the FindFace Multi and Sentinel integration.
2. Install the Sentinel drivers on the physical server with the `findface-ntls` component.

   Do the following:

   1. Download Sentinel drivers from the official website.
   2. Unzip the downloaded archive and browse to it.

      ```
      tar -xvzf Sentinel_LDK_Linux_Runtime_Installer_script.tar.gz
      cd Sentinel_LDK_Linux_Runtime_Installer_script/
      ```

   3. There is another archive `aksusbd-8.31.1.tar.gz` inside the archive. Unzip it and browse to the resulting directory.

      ```
      tar -xvzf aksusbd-8.31.1.tar.gz
      cd aksusbd-8.31.1/
      ```

   4. Run the installation command.

      ```
      sudo ./dinst
      ```

   5. Run and check the statuses of the Sentinel services.

      ```
      sudo systemctl start aksusbd.service hasplmd.service
      sudo systemctl status aksusbd.service hasplmd.service
      ```

3. Mount the `/var/hasplm` and `/etc/hasplm` directories into the `findface-multi-findface-ntls-1` container. To do so, open the `/opt/findface-multi/docker-compose.yaml` configuration file and list them in the volumes of the `findface-ntls` section.

```
sudo vi /opt/findface-multi/docker-compose.yaml

findface-ntls:
  ...
  volumes: ['./configs/findface-ntls/findface-ntls.yaml:/etc/findface-ntls.cfg:ro',
↪'./data/findface-ntls:/ntech/license', '/var/hasplm:/var/hasplm', '/etc/hasplm:/
↪etc/hasplm']
```

4. Rebuild all FindFace Multi containers.

```
cd /opt/findface-multi

sudo docker-compose down

sudo docker-compose up -d
```

5. Put the `findface-sentinel-lib_*.deb` package received from your manager into some directory on the same host. Install the package.

```
sudo dpkg -i /path/to/findface-sentinel-lib_*.deb
```

6. In the FindFace Multi web interface, navigate to *Settings* -> *License*. Take a hardware fingerprint (C2V file) by clicking the *Download C2V for activation* button.

Tip: If you prefer working with the console, you can send the following API request to `findface-ntls` instead:

```
curl <findface-ntls-server-ip>:3185/c2v  >my_pc.c2v
```

---

7. Send the License ID and the C2V file to your manager and receive your license file in return.
8. Upload the license file on the *License* tab.

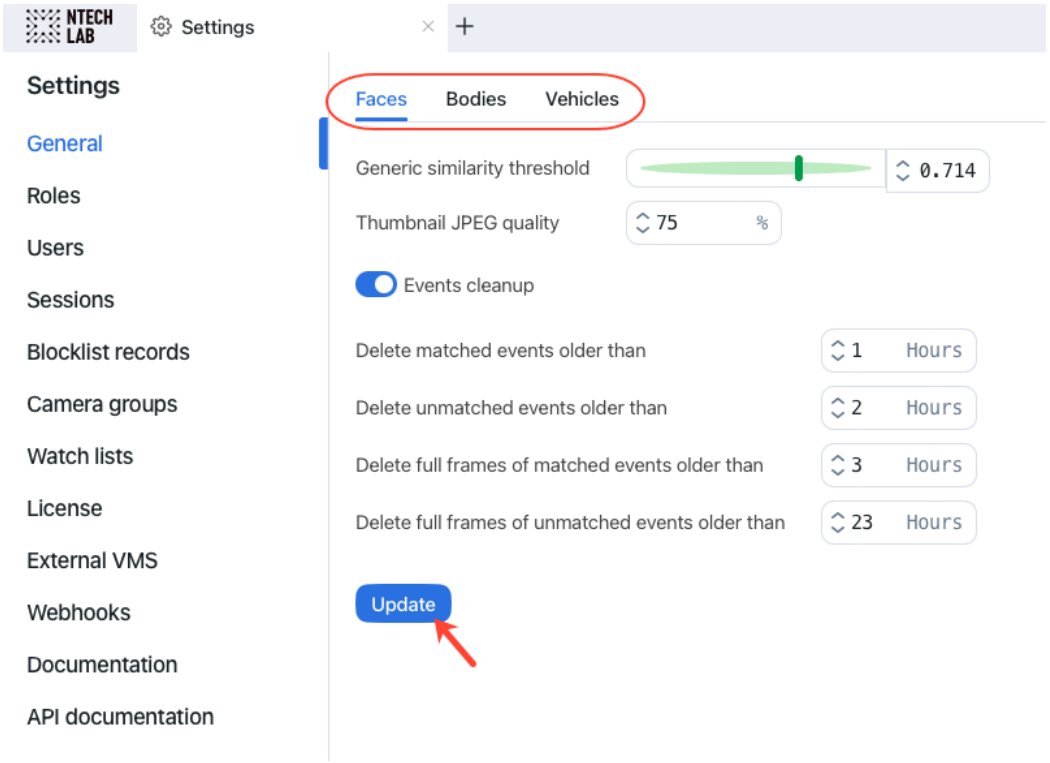
2.4.2 General Settings

The FindFace Multi general settings determine your system functioning and resource consumption. Here they are:

- generic confidence threshold
- thumbnail JPEG quality
- schedule for automatic events/episodes cleanup

The general settings for faces, bodies, and vehicles are provided separately, depending on the enabled recognition objects.

To configure the general settings, navigate *Settings* -> *General*. After you are finished with adjustments, click *Update*. Find the detailed explanation of each general setting below.



In this section:

Generic Confidence Threshold

FindFace Multi verifies that objects match (i.e. two faces belong to the same person), based on the pre-defined confidence 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 object will be positively verified, however, some valid photos may also fail verification.

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 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.

2.4.3 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.

Role and User Management

In this chapter:

Predefined Roles

FindFace Multi provides the following predefined roles:

- Administrator is granted full access to the FindFace Multi functionality, integrative and administrative tools.

  Important: Whatever the role, the first administrator (Super Administrator) cannot be deprived of its rights.

- Operator is granted full access to the FindFace Multi functionality.

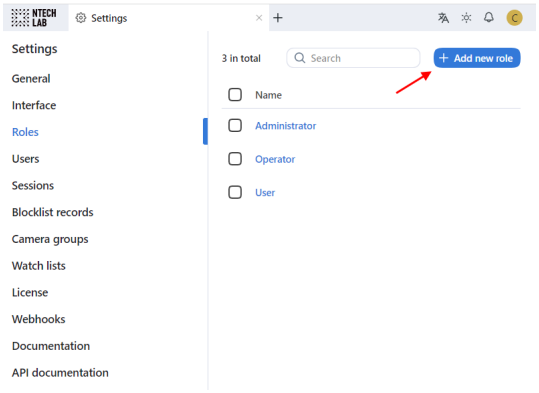
- User is granted rights to modify their profile and work with events and episodes. The other functions are available read-only.

You can change the predefined roles privileges, as well as create various custom roles.

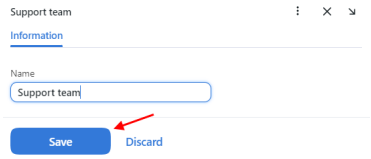
Create Custom Role

To create a custom role, do the following:

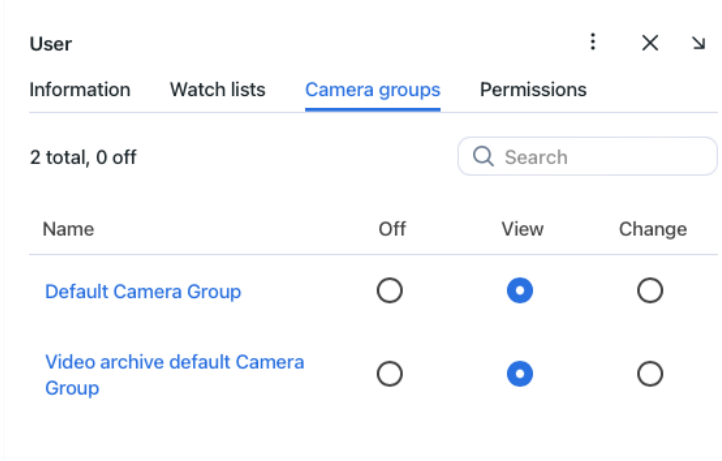
1. Navigate *Settings* -> *Roles*.
2. Click + *Add new role*.



3. On the *Information* tab, specify the role name. Save the role.



4. After saving the role, you will see the following tabs appear next to the *Information* tab:



- *Watch Lists*: role privileges for specific watch lists

- *Camera Groups*: role privileges for specific camera groups
- *Permissions*: role privileges for entire system functions and entities

Set role privileges, subject to your needs. Note that there is a distinction between role privileges for a specific watch list/camera group and a system entity with the name `watchlist/cameragroup`. For example, if you set `Off` 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. Unchecking all checkboxes for the `cameragroup` entity on the *Permissions* tab will prevent users from viewing and working with **all** camera groups.

The full list of the FindFace Multi entities which are used in the current version is as follows:

- `all_own_sessions`: all *sessions* of the current user on different devices

---

Note: If relevant permissions for this entity are set, users will be able to view (`view`) and close (`delete`) all their sessions on different devices. Otherwise, users will be only allowed to view and close their session on the current device. Working with sessions takes place on the *Sessions* tab (*Settings*).

- `bodycluster`: *cluster of bodies*
- `bodyevent`: body recognition *event*
- `bodyobject`: full-length photo in a *record*
- `camera`: *camera*
- `cameragroup`: *camera group*
- `carcard`: *vehicle record*
- `carcluster`: *cluster of vehicles*
- `carepisode`: *vehicle-related episode*
- `carevent`: vehicle recognition *event*
- `carobject`: vehicle photo in a record
- `counter`: *counters*
- `deviceblacklistrecord`: *blocklist*
- `facecluster`: *cluster of faces*
- `faceevent`: face recognition *event*
- `faceobject`: face photo in a *record*
- `humancard`: *record of an individual*
- `humanepisode`: *person-related episode*
- `report`: *report*
- `upload`: item (photo) in batch photo upload
- `uploadlist`: list of photos in batch upload
- `user`: *user*
- `videoarchive`: *object identification in video files*
- `watchlist`: *watch list*
- `webhook`: *webhook*

You can also enable and disable rights for the following functionality:

- `batchupload_cards`: *bulk record upload*
- `change_runtimesetting`: changing the FindFace Multi general settings
- `view_auditlog`: viewing and working with the *audit logs*.
- `configure_ntls`: configuration of the `findface-ntls` *license server*
- `view_runtimesetting`: viewing the FindFace Multi *general settings*

5. Save the changes.

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 video archives within the group) and watch lists (and records in the watch list) created by a user with the same primary role, subject to the privileges defined by their additional role(s).

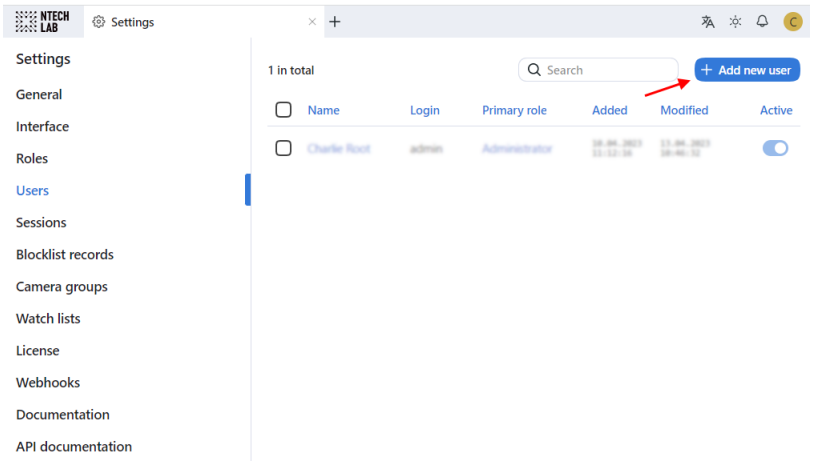
See also:

Create User Account Manually

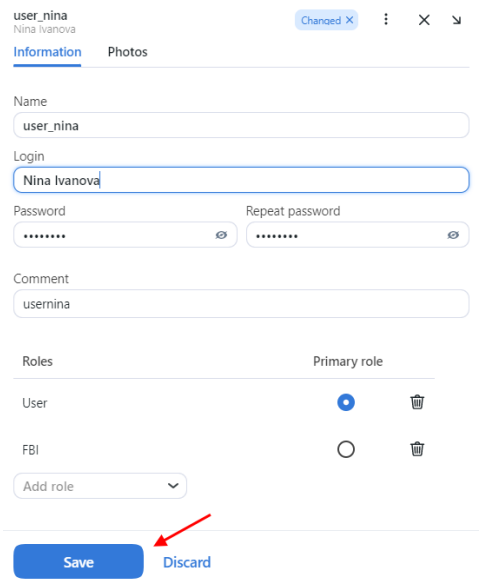
Create User Account Manually

To create a user account manually, do the following:

1. Navigate *Settings* -> *Users*.
2. Click + *Add new user*.



3. On the *Information* tab, specify user data such as name, login, and password. If necessary, add a comment.
4. From the *Roles* drop-down menu, select one or several user roles. Set one of them as the *Primary role*.
5. On the *Photos* tab, attach the user's photo.
6. Save the user account.



Integrate with Active Directory for Auto User Creation

If there are many users in FindFace Multi, it can be inconvenient to create their accounts one by one. One of the ways to facilitate the user creation is to harness the FindFace Multi integration with Active Directory. To do so, follow the step-by-step instructions below, minding the sequence.

Generate Keytab File

Log-in into the Active Directory server and do the following:

1. Create a new user account in the Active Directory domain to use as a service account.

   Do the following:

   1. Open Active Directory. Click *Start*, point to *Administrative Tools*, and then click *Active Directory Users and Computers*.
   2. Click the domain name and then expand the contents. Right-click *Users*, point to *New*, and then click *User*. You will see a user creation form.
   3. Fill-in the fields in the form at your discretion. On the second tab, check the *Password never expires* checkbox.
   4. Click *Next*. Review the information that you provided, and if everything is correct, click *Finish*.
   5. Right-click the just created user account, and then navigate *Properties -> Member Of -> Add*.
   6. In the *Select Groups* dialog box, add the *Domain Administrators* and *Domain Users* groups to the list, and then click *OK*.
   7. Click *OK* to finish.

2. Register a Service Principal Name (SPN) for the service account that you created. To do so, open PowerShell as administrator and execute the following command, specifying your actual `SERVICE USER NAME` and domain. In the example below, the domain name is `testntl.local`.

```
setspn -A HTTP/<SERVICE USER NAME>.testntl.local@TESTNTL.LOCAL <SERVICE USER NAME>
```

3. In the same PowerShell window, generate a Keytab file by executing the command below with your actual SERVICE USER NAME, domain, and desirable KEYTAB FILE NAME.

```
ktpass.exe -princ HTTP/<SERVICE USER NAME>.testntl.local@TESTNTL.LOCAL -mapuser
↪<SERVICE USER NAME> -crypto ALL -ptype KRB5_NT_PRINCIPAL -pass * -out c:\<KEYTAB␣
↪FILE NAME>.keytab
```

   To check the result, navigate to the root directory of the C drive. You will see a keytab file with the relevant name.

4. Move the keytab file that you created to the FindFace Multi server.

Configure Volumes for Kerberos and Keytab File

To successfully establish a link between FindFace Multi and Active Directory, you need to enable the Kerberos support in the findface-multi-findface-multi-ui-1 container on the FindFace Multi principal server. Do the following:

1. Mount the /opt/findface-multi/configs/kerberos/krb5.conf file and the /opt/findface-multi/configs/keytab directory into the findface-multi-findface-multi-ui-1 container. To do so, open the /opt/findface-multi/docker-compose.yaml configuration file and list them in the volumes of the findface-multi-ui section.

```
sudo vi /opt/findface-multi/docker-compose.yaml

findface-multi-ui:
  ...
  volumes: ['./configs/findface-multi-ui/nginx-site.conf:/etc/nginx/conf.d/default.
↪conf:ro',
    './data/findface-multi-legacy/uploads:/var/lib/findface-security/uploads',
    './configs/kerberos/krb5.conf:/etc/krb5.conf:ro',
    './data/findface-multi-ui/keytab/:/keytab/']
```

2. Create directories for the mounted volumes: /opt/findface-multi/configs/kerberos/ and /opt/findface-multi/data/findface-multi-ui/keytab/. Copy the Kerberos configuration file from the findface-multi-findface-multi-ui-1 container to the /opt/findface-multi/configs/kerberos/ directory.

```
sudo mkdir -p /opt/findface-multi/data/findface-multi-ui/keytab/
sudo mkdir /opt/findface-multi/configs/kerberos/
sudo docker cp findface-multi-findface-multi-ui-1:/etc/krb5.conf /opt/findface-
↪multi/configs/kerberos/
```

3. Open the /opt/findface-multi/configs/kerberos/krb5.conf configuration file. Specify the Active Directory realm in the libdefaults section. It must be equal to the Active Directory domain name, but spelled in upper case (TESTNTL.LOCAL in the example below). Specify the Active Directory domain in the realms section as well by analogy with the example below.

```
sudo vi /opt/findface-multi/configs/kerberos/krb5.conf

[libdefaults]
        default_realm = TESTNTL.LOCAL
...
[realms]
TESTNTL.LOCAL = {
  kdc = testntl.local
```

(continues on next page)

(continued from previous page)

```
  default_domain = testntl.local
}
```

4. Copy the keytab file to the `/opt/findface-multi/data/findface-multi-ui/keytab/` directory.
5. Append the following string to the `/etc/hosts` file: `<Active Directory server IP address> <Active Directory domain name>`.

```
vi /etc/hosts

...
192.168.0.5 testntl.local
```

Configure NGINX on FindFace Multi Server to Support Active Directory

1. Open the `/opt/findface-multi/configs/findface-multi-ui/nginx-site.conf` configuration file. Find the `location /users/me/ad` section and uncomment it. Fill in the section by analogy with the example below, placing your actual variables in the strings with comments (#).

   The variables to specify are the following:

   - `auth_gss_realm`: realm name in Kerberos
   - `auth_gss_keytab`: location of the keytab file.
   - `auth_gss_service_name`: full service user name in Active Directory, including the name of the domain it belongs to

```
sudo vi /opt/findface-multi/configs/findface-multi-ui/nginx-site.conf

location /users/me/ad {
      proxy_pass http://127.0.0.1/auth/ad_login/; # e.g http://127.0.0.1/auth/ad_
↪login/;
      proxy_method POST;

      proxy_set_header   X-Real-IP $remote_addr;
      proxy_set_header   Host $http_host;
      proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header  Authorization $http_authorization;
      proxy_pass_header Authorization;
      proxy_no_cache 1;
      proxy_cache_bypass 1;

      auth_gss on;
      auth_gss_realm TESTNTL.LOCAL; # e.g. TESTNTL.LOCAL;
      auth_gss_keytab /keytab/user.keytab; # e.g. /var/lib/web.keytab
      auth_gss_service_name HTTP/user.testntl.local; # e.g. HTTP/web.testntl.local;
      auth_gss_allow_basic_fallback on;
}
```

Finalize FindFace Multi Configuration

To finalize the FindFace Multi integration with Active Directory, perform the following configuration steps on the FindFace Multi side:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. In the `SERVICES` section, set `"active_directory": True`.

```
SERVICES = {
        ...
        "active_directory": True,
        ...
    }
}
```

3. Fill in the `ACTIVE_DIRECTORY_CONFIG` section as follows:

   - `AUTH_LDAP_SERVER_URI`: `ldap: <Active Directory server IP address>`
   - `AUTH_LDAP_BIND_DN`: the name of the service user that you created in Active Directory
   - `AUTH_LDAP_BIND_PASSWORD`: the service user password
   - `SEARCH_GROUPS`: Active Directory organization units which FindFace Multi will search for user accounts

```
# Specify server credentials
ACTIVE_DIRECTORY_CONFIG = {
    'AUTH_LDAP_SERVER_URI': 'ldap://192.168.0.5',
    # Domain Administrator user
    'AUTH_LDAP_BIND_DN': '<SERVICE USER NAME IN ACTIVE DIRECTORY>',
    # Domain Administrator user password
    'AUTH_LDAP_BIND_PASSWORD': 'SERVICE USER NAME PASSWORD',
    # Specify organization units where users search will be executed.
    # Follow pattern (e.g. OU=DEV,DC=domain,DC=com)
    'SEARCH_GROUPS': 'OU=DEV,DC=testntl,DC=local',
}
```

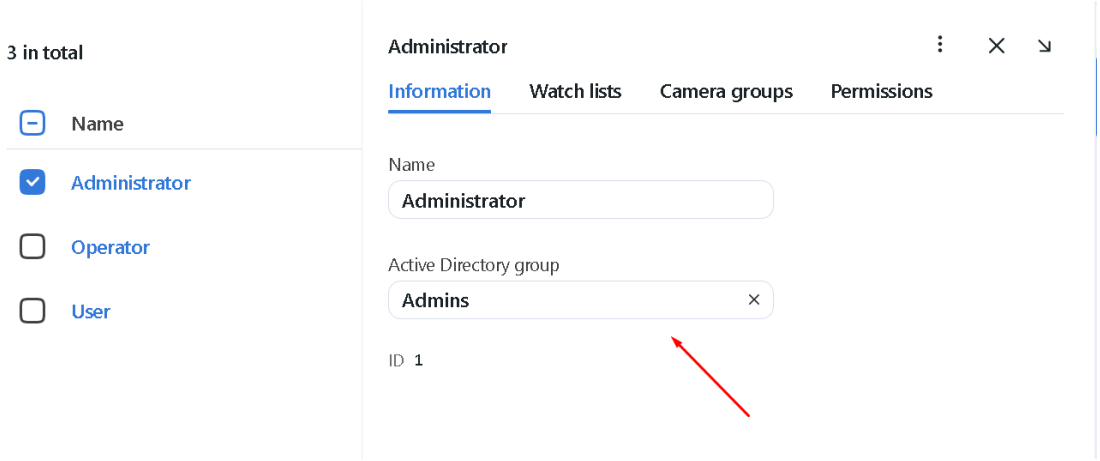
4. Open the `/opt/findface-multi/configs/findface-multi-identity-provider/findface-multi-identity-provider.py` configuration file and repeat the previous steps.

5. Rebuild all FindFace Multi containers.

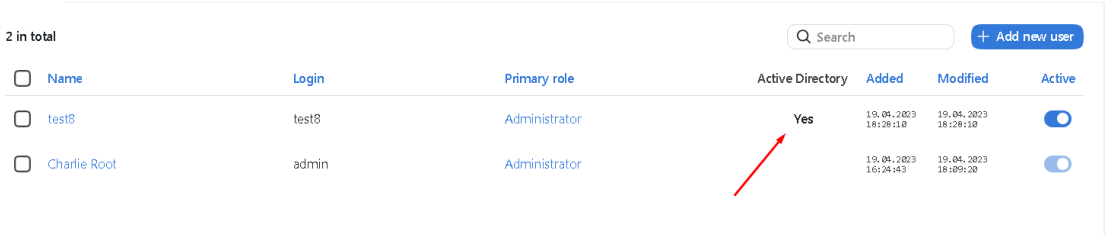
```
cd /opt/findface-multi

sudo docker-compose down

sudo docker-compose up -d
```

Manage FindFace Multi Users via Active Directory

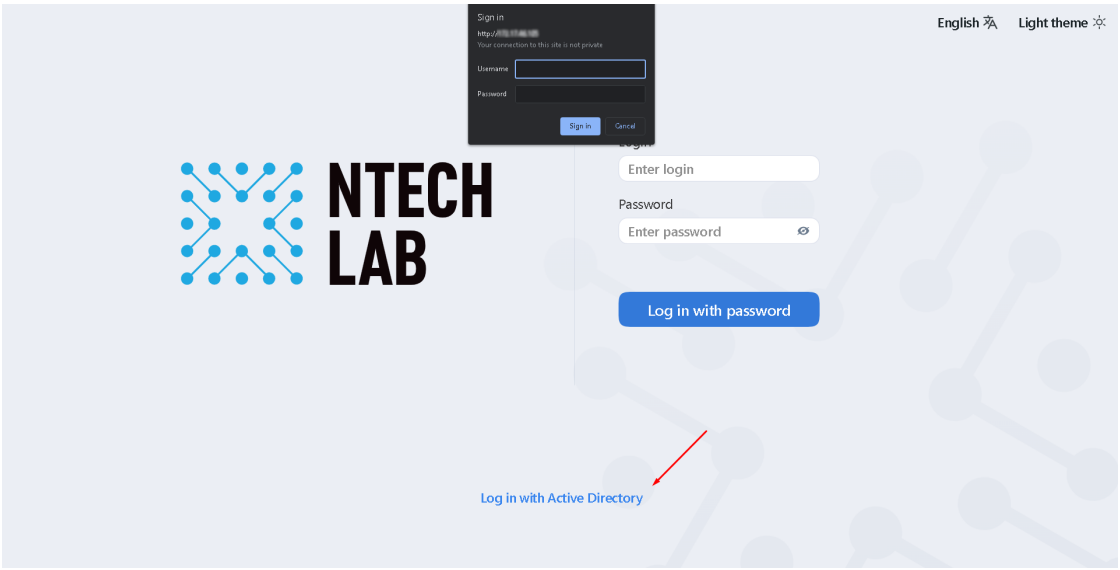
If the FindFace Multi integration with Active Directory is enabled, you will be able to set one of the Active Directory groups for a role you are creating or editing.



Once a user from the selected Active Directory group logs-in into FindFace Multi for the first time, they will be automatically added to the FindFace Multi user list.



To log-in with Active Directory, a user must click the *Log in with Active Directory* button in the authentication window, specify their Active Directory credentials, and click *Sign in*.



Deactivate or Delete Users

In order to deactivate a user, unset *Active* on the user list (*Settings -> Users*).

If you are going to deactivate multiple users, select them on the user list and then click *Deactivate selected*.

To delete users from FindFace Multi, select them on the user list and then click *Delete selected*.

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 `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. Find the `FFSECURITY` and `FFSECURITY_AUTH_CONFIG` sections.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.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 the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

Log out All Users

To automatically log out all users, execute the following command on the FindFace Multi principal server console:

```
sudo docker exec findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py 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 enabling SSL encryption. Do the following:

1. On the host system, create the nginx configuration directory with the subdirectory that will be used to store all the SSL data:

```
sudo mkdir -p /etc/nginx/ssl/
```

2. Create the SSL key and certificate files. When using self-signed certificate, use the following command:

```
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. When using CA-certificate, add the certificate path to **volumes** for the `findface-video-worker` service, add the CA-certificates installation and update the root certificate store in the service container.

    1. Open the `docker-compose.yaml` file:

    ```
    sudo vi /opt/findface-multi/docker-compose.yaml
    ```

    2. Locate the `findface-video-worker` section and adjust it to make sure it looks as follows.

    For CPU:

    ```
    findface-video-worker:
        entrypoint: ["sh", "-c", "apt-get update && DEBIAN_FRONTEND=noninteractive␣
    ↪apt-get install --no-install-recommends --yes ca-certificates && update-ca-
    ↪certificates && exec /tini -- /findface-video-worker-cpu --config=/etc/
    ↪findface-video-worker.yaml"]
    ```

(continues on next page)

(continued from previous page)

```
    depends_on: [findface-video-manager, findface-ntls, mongodb]
    image: docker.int.ntl/ntech/universe/video-worker-cpu:ffserver-8.221216
    logging: {driver: journald}
    network_mode: service:pause
    restart: always
    volumes: ['./configs/findface-video-worker/findface-video-worker.yaml:/etc/
↪findface-video-worker.yaml:ro',
      './models:/usr/share/findface-data/models:ro', './cache/findface-video-
↪worker/models:/var/cache/findface/models_cache',
      './cache/findface-video-worker/recorder:/var/cache/findface/video-worker-
↪recorder',
      '/etc/nginx/ssl/my-example-domain.crt:/usr/local/share/ca-certificates/my-
↪example-domain.crt:ro']
```

For GPU, it will be enough to add the path to the certificate and update the root certificate store:

```
findface-video-worker:
    entrypoint: ["sh", "-c", "update-ca-certificates && exec /tini -- /findface-
↪video-worker-gpu --config=/etc/findface-video-worker.yaml"]
    depends_on: [findface-video-manager, findface-ntls, mongodb]
    environment: [CUDA_VISIBLE_DEVICES=0]
    image: docker.int.ntl/ntech/universe/video-worker-gpu:ffserver-8.221216
    logging: {driver: journald}
    network_mode: service:pause
    restart: always
    runtime: nvidia
    volumes: ['./configs/findface-video-worker/findface-video-worker.yaml:/etc/
↪findface-video-worker.yaml:ro',
      './models:/usr/share/findface-data/models:ro', './cache/findface-video-
↪worker/models:/var/cache/findface/models_cache',
      './cache/findface-video-worker/recorder:/var/cache/findface/video-worker-
↪recorder',
      '/etc/nginx/ssl/my-example-domain.crt:/usr/local/share/ca-certificates/my-
↪example-domain.crt:ro']
```

Important: For CPU version, the configuration requires internet access. If there is no access, please, contact our support team (support@ntechlab.com).

Warning: For CPU version, the startup time will increase by ~15 seconds for the `findface-video-worker` container.

3. Rebuild all FindFace Multi containers.

```
cd /opt/findface-multi/
docker-compose down
docker-compose up -d
```

4. Configure nginx to use SSL. Open the nginx configuration file `/opt/findface-multi/configs/findface-multi-ui/nginx-site.conf`. Apply the following modifications to the file:

1. Add the new `server {...}` section that contains the URL replacement rule. In the `rewrite ^(.*) https://...` line, replace `ip_address_server_ffmulti` with IP address of the server where FindFace Multi is installed.

```
server {
        listen 80;
        server_name my-example-domain.com www.my-example-domain.com;
        rewrite ^(.*) https://ip_address_server_ffmulti$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;
```

The example of the configuration file `/opt/findface-multi/configs/findface-multi-ui/nginx-site.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;
}

upstream ffsecurity-django {
        server 127.0.0.1:8004;
}

upstream audit {
        server 127.0.0.1:8012;
}

upstream identity-provider {
        server 127.0.0.1:8022;
}

map $http_upgrade $ffsec_upstream {
        default "http://ffsecurity-ws";
        "" "http://ffsecurity";
}


server {
```

(continues on next page)

(continued from previous page)

```
        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;
        }
        location /static/ {
        }
        location /uploads/ {
                # internal; # uncomment if you intend to enable OVERPROTECT_
↪MEDIA
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' '*';
                add_header 'Access-Control-Allow-Headers' '*';
                add_header 'Access-Control-Expose-Headers' 'Content-Length,
↪Content-Range';
                add_header 'Access-Control-Max-Age' 2592000;

                location ~ /card/(?<card_type>[a-zA-Z]+)/(?<card_id>[0-9]+)/
↪attachments/(.*)$ {
                        add_header 'Access-Control-Allow-Origin' '*';
                        add_header 'Access-Control-Allow-Methods' '*';
                        add_header 'Access-Control-Allow-Headers' '*';
                        add_header 'Access-Control-Expose-Headers' 'Content-
↪Length,Content-Range';
                        add_header 'Access-Control-Max-Age' 2592000;
                        add_header 'Content-Disposition' 'attachment';
                        add_header 'Content-Security-Policy' 'sandbox';
                }
        }
        location /ui-static/ {
                alias /usr/share/findface-security-ui/ui-static/;
        }
        location /doc/ {
```

(continues on next page)

(continued from previous page)

```
                alias /opt/findface-security/doc/;
        }
        location /api-docs {
                alias /opt/findface-security/rapidoc;
                index index.html;
        }
        location /api-docs/ {
                alias /opt/findface-security/rapidoc/;
                try_files $uri index.html =404;
        }
        location ~ /videos/(?<video_id>[0-9]+)/upload/(.*)$ {
                client_max_body_size 15g;

                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 @django {
                internal;
                client_max_body_size 1g;
                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_read_timeout 5m;
                proxy_pass http://ffsecurity-django;
        }

#       location /v1/video-liveness {
#           add_header Access-Control-Allow-Headers "*" always;
#           add_header Access-Control-Allow-Methods "*" always;
#           add_header Access-Control-Allow-Origin  "*" always;
#
#           if ($request_method = 'OPTIONS') {
#               return 204;
#           }
#
#           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_pass http://127.0.0.1:18301;
#           proxy_read_timeout 5m;
#       }

        location / {
                client_max_body_size 1g;
                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;
```

(continues on next page)

(continued from previous page)

```
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_pass $ffsec_upstream;
                proxy_read_timeout 5m;

                location ~ ^/(cameras|videos|vms|external-vms).*/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 $1$is_args$args;
                }

                location /audit-logs {
                        proxy_pass http://audit;
                }

                location ~ ^/(auth|ad_
↪groups|cproauth|groups|permissions|sessions|users|user-face|device-blacklist-
↪records) {
                        proxy_pass http://identity-provider;
                }
        }
#       location /users/me/ad {
#
#                 proxy_pass <FFmulti_address>/auth/ad_login/; e.g http://127.0.
↪0.1/auth/ad_login/;
#                 proxy_method POST;
#
#                 proxy_set_header   X-Real-IP $remote_addr;
#                 proxy_set_header   Host $http_host;
#                 proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
#                 proxy_set_header  Authorization $http_authorization;
#                 proxy_pass_header Authorization;
#                 proxy_no_cache 1;
#                 proxy_cache_bypass 1;
#
#                 auth_gss on;
#                 auth_gss_realm <REALM>; # e.g. TESTNTL.LOCAL;
#                 auth_gss_keytab <path/to/file.keytab>; # e.g. /var/lib/web.
↪keytab
#                 auth_gss_service_name <service_name>; # e.g. HTTP/web.testntl.
↪local;
#                 auth_gss_allow_basic_fallback on;
#         }

}
```

4. Copy the generic nginx configuration file `nginx.conf` from the

findface-multi-findface-multi-ui-1 container to the /etc/nginx/ directory:

```
sudo docker cp findface-multi-findface-multi-ui-1:/etc/nginx/nginx.conf /etc/
↪nginx/nginx.conf
```

5. In the 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;
```

5. In the /opt/findface-multi/docker-compose.yaml file, mount the SSL-encryption data directory /etc/nginx/ssl/ and the configuration file /etc/nginx/nginx.conf of the host system into the findface-multi-findface-multi-ui-1 container:

   1. Open the docker-compose.yaml file:

   ```
   sudo vi /opt/findface-multi/docker-compose.yaml
   ```

   2. Locate the findface-multi-ui section and adjust it to make sure it looks like this:

   ```
   findface-multi-ui:
     depends_on: [findface-multi-legacy]
     image: docker.int.ntl/ntech/multi/multi/ui:ffmulti-2.0.0
     network_mode: service:pause
     restart: always
     volumes: ['./configs/findface-multi-ui/nginx-site.conf:/etc/nginx/conf.d/
   ↪default.conf:ro',
       './data/findface-multi-legacy/uploads:/var/lib/findface-security/uploads',
       '/etc/nginx/ssl:/etc/nginx/ssl',
       '/etc/nginx/nginx.conf:/etc/nginx/nginx.conf:ro']
   ```

6. Edit the /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py configuration file.

   1. In the ROUTER_URL and IMAGE_CROP_URL parameters, substitute the http:// prefix with https://.

   ```
   sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.
   ↪py

   ...
   'ROUTER_URL': 'https://127.0.0.1',
   'IMAGE_CROP_URL': 'https://127.0.0.1',
   ...
   ```

   2. If you use a CA-certificate, specify in the ROUTER_URL parameter the domain for which the certificate was created:

   ```
   'ROUTER_URL': 'https://my-example-domain.com
   ```

   3. Add https://my-example-domain.com address to the EXTERNAL_ADDRESS parameter:

   ```
   ...
   EXTERNAL_ADDRESS = 'https://my-example-domain.com'
   ...
   ```

7. Open the /etc/hosts file on the server where FindFace Multi is installed and add the following line:

```
sudo vi /etc/hosts

...
127.0.0.1 my-example-domain.com
```

8. In the system where you use a browser to interact with FindFace Multi navigate to the `hosts` file. Add IP address of the server that hosts FindFace Multi instead of the `ip_address_server_ffmulti`. Replace `my-example-domain.com` with your domain address - the same way you did it in the previous steps.

    1. For Linux OS do the following:

    ```
    sudo vi /etc/hosts

    ...
    *ip_address_server_ffmulti* my-example-domain.com
    ```

    2. If you use Windows OS, run `C:\Windows\System32\drivers\etc\hosts` as an administrator. Add the following line to the `hosts` file:

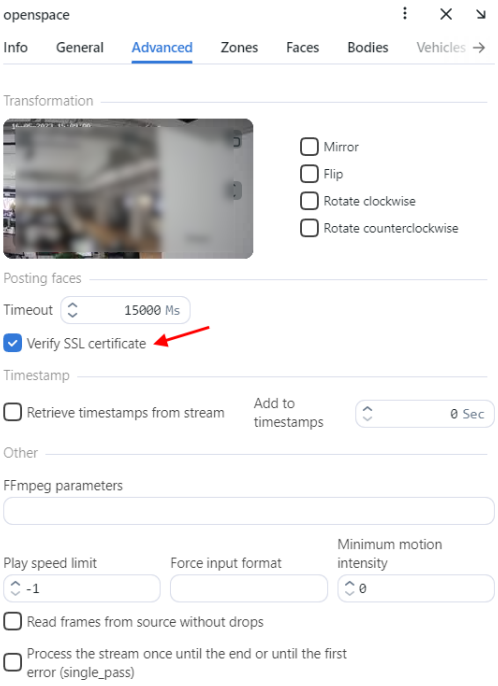
    ```
    *ip_address_server_ffmulti* my-example-domain.com
    ```

9. Restart the containers:

```
cd /opt/findface-multi/
sudo docker-compose down
sudo docker-compose up -d
```

10. If you use self-signed certificate, disable SSL certificate verification for cameras and uploaded video archives:

    1. Navigate to the *Video Sources* -> *Cameras* or *Uploads*.
    2. Click to the camera or uploaded video archive.
    3. On the *Advanced* tab, uncheck *Verify the SSL certificate*.



Warning: You may receive errors with getting screenshots from camera and working with video recorder, when deploying FindFace Multi in a highly distributed environment by *fully customized installation* and the `pause` component is excluded from installation. To resolve this problem, add the following lines to the end of the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` file:

```
USE_X_FORWARDED_HOST = True
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
```

Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker restart findface-multi-findface-multi-legacy-1
```

Enable Record Index Protection

If the record index security is turned off, photos and attachments in records will be available by direct link, no matter what the user's rights are. Configure FindFace Multi to run all media requests through the DJANGO application for ACL checks to increase record index security.

Important: Enable the record media security only if you need it, as this setting severely negatively impacts the system performance.

Important: For the ACL checks to work properly, you must set the view permission for photos of faces, bodies, and vehicles stored in records. To do so, navigate *Settings* → *Roles* → specific role → *Permissions* and set *View* for the `faceobject`, `bodyobject`, and `carobject` entities, subject to the object types enabled in the system. See *Create Custom Role* for details.

See also:

Record Index

To enable record index security, do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. Locate the `OVERPROTECT_MEDIA` parameter and set it `True`.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

...

'OVERPROTECT_MEDIA': True,
```

2. Do the same in the `/opt/findface-multi/configs/findface-multi-identity-provider/findface-multi-identity-provider.py` configuration file: locate the `OVERPROTECT_MEDIA` parameter and set it `True`.

```
sudo vi /opt/findface-multi/configs/findface-multi-identity-provider/findface-multi-
↪identity-provider.py

...

'OVERPROTECT_MEDIA': True,
```

3. Open the nginx configuration file `/opt/findface-multi/configs/findface-multi-ui/nginx-site.conf`. Uncomment `internal` in the `location /uploads` section.

```
sudo vi /opt/findface-multi/configs/findface-multi-ui/nginx-site.conf

location /uploads/ {
   internal; # Uncomment if you intend to enable OVERPROTECT_MEDIA
   ...
}
```

4. Restart the `findface-multi-findface-multi-legacy-1`, `findface-multi-findface-multi-identity-provider-1`, and `findface-multi-findface-multi-ui-1` containers.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
sudo docker container restart findface-multi-findface-multi-identity-provider-1
sudo docker container restart findface-multi-findface-multi-ui-1
```

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 docker container exec -it findface-multi-findface-multi-identity-provider-1 /
↪opt/findface-security/bin/python3 /tigre_prototype/manage.py 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 `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. Set `ENABLE_ACL = False`.

```
...

ENABLE_ACL = False
```

3. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

List of User Sessions. Blocklist

In this chapter:

FindFace Multi allows you to monitor user sessions and learn associated data, such as the connected device UUID, type of user interface, IP address, last ping time, and so on.

If necessary, you can add a device to the blocklist without deactivating the user account. The device block may come in handy in various situations. For example, if you want users to access the system only from their workplaces. Use the blocklist functionality to take your system safety to the next level.

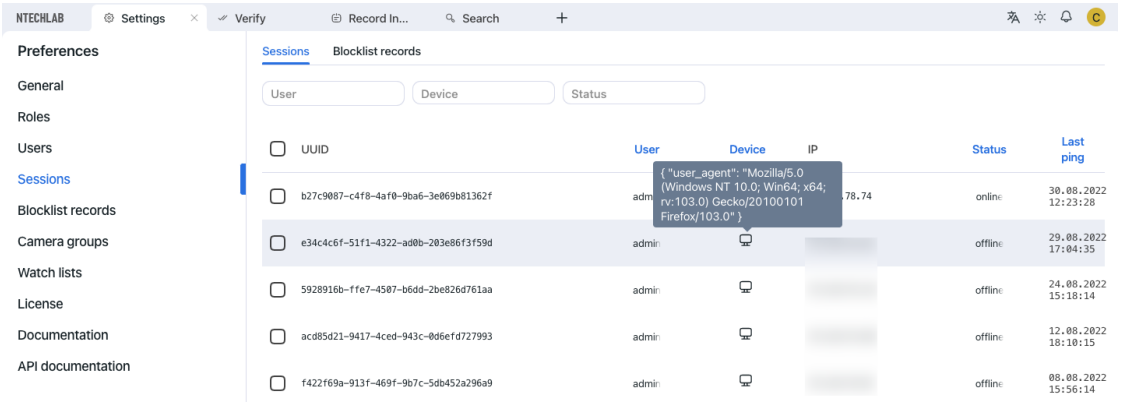
Grant Permissions to Work with Sessions

A user's access to the list of sessions depends on the granted *permissions*:

- Administrator: can view and close sessions of all users
- User with the `all_own_sessions` permissions: can view/close all sessions initiated with their username
- User without the `all_own_sessions` permissions: can only view/close their current session

View User Sessions

To view the list of user sessions, navigate *Settings -> Sessions*.



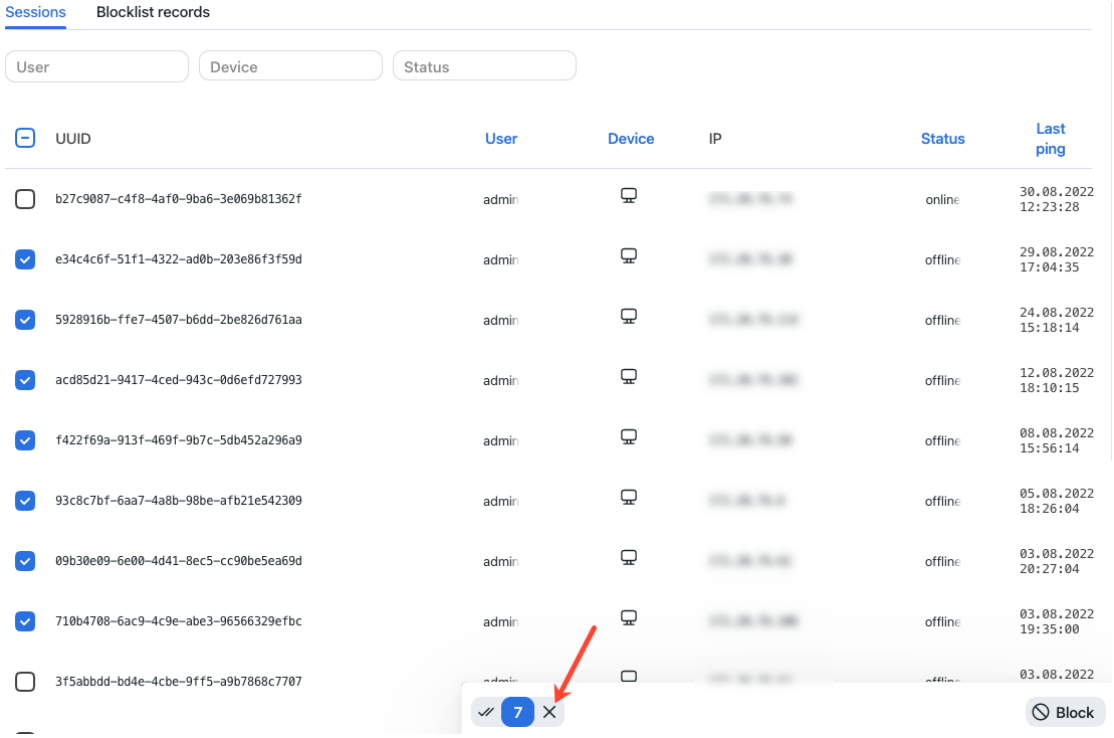
Each session record provides the following data:

- device UUID
- username
- type of the user interface (mobile/web)
- device information
- IP address
- status (online, offline, blocked)

- last ping time

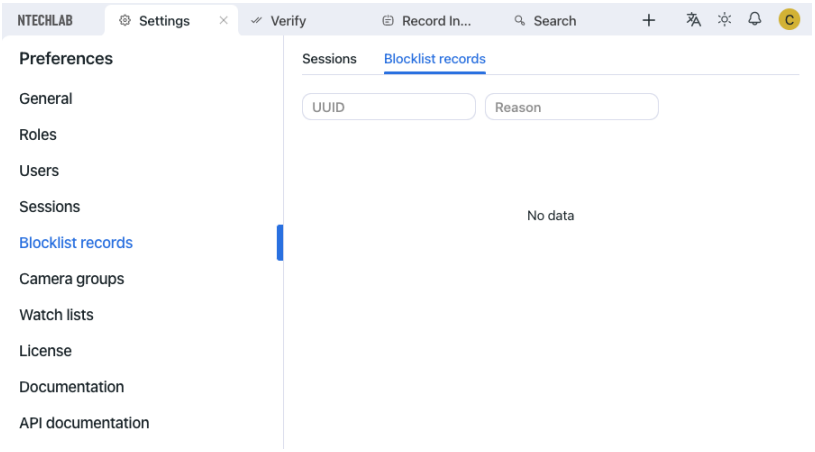
Use the filter panel above the list of sessions to set up the search conditions.

To close a session, select it in the list and click *x*.



Block Device

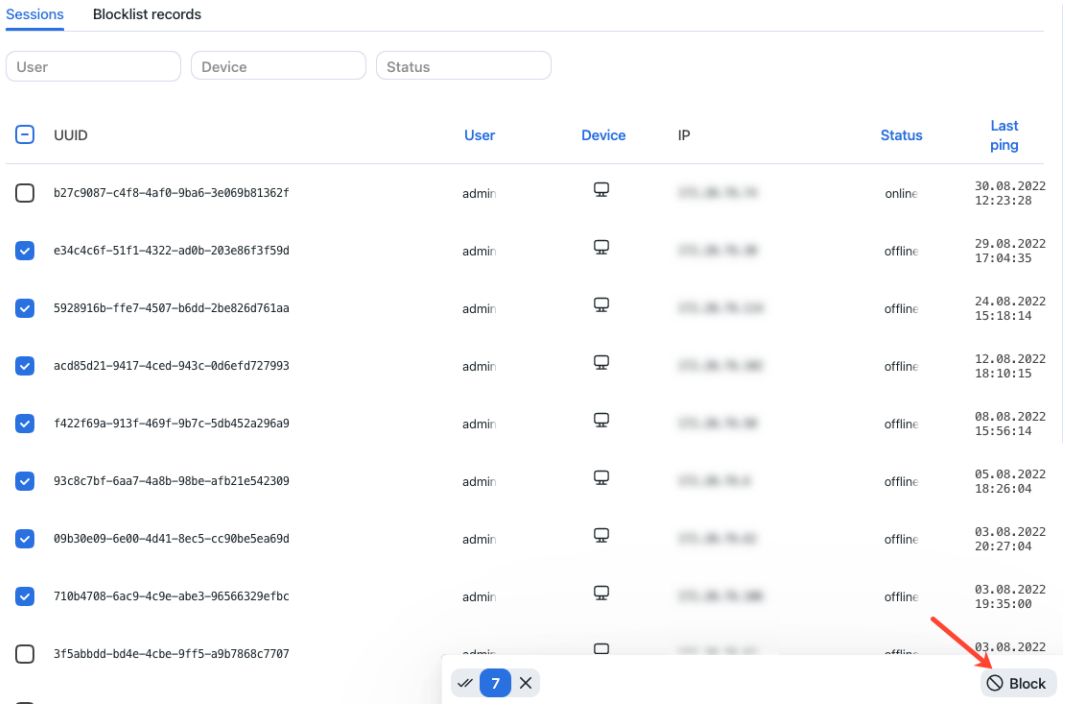
The list of blocked devices is available on the *Blocklist Records* tab.



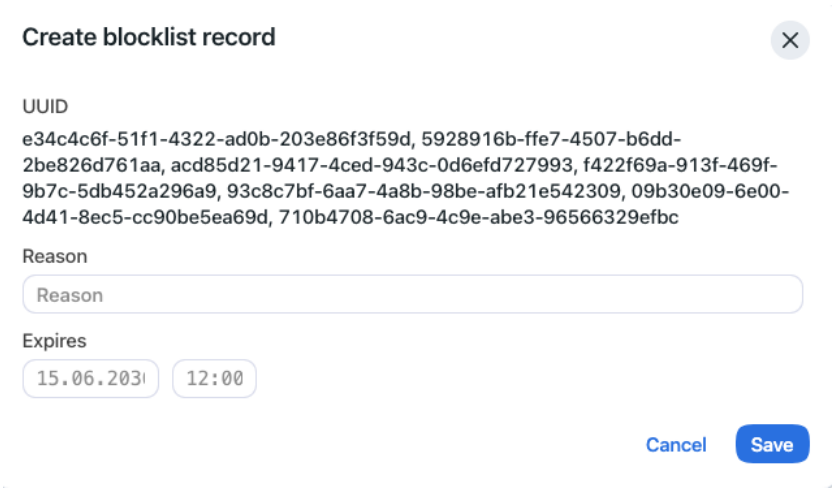
You can add a device to the blocklist on the *Sessions* tab. Blocking a device leads to the user's automatic log-out.

To block a device, do the following:

1. Select the relevant session record(s).
2. Click *Block*.



3. Specify the reason for the device to be blocked (mandatory) and the block expiry date (optional). If no date is specified, the block will be permanent.
4. Click *Save*.



Allowed File Extensions in Records

By default, you can attach a file of any extension to a *record*. It is possible to strengthen your system safety by creating the allowlist of file extensions. It will prevent your users from uploading files of unwanted formats, including those that might contain hidden malicious code, such as `.js`, `.swf`, and such.

To create the allowlist of file extensions, do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. In the `FFSECURITY` section, find the `CARD_ATTACHMENTS_FILENAME_REGEXP` parameter. Set an expression with the allowed file extensions. Any valid Python regular expression will do.

   Examples:

   - `r'.*\.png'`: allows only files with the `.png` extension
   - `r'.*\(png|jpg)'`: allows the `.png` and `.jpg` extensions
   - `r'.*'`: allows all file extensions
   - `None`: allows all file extensions
   - `'XXXXXX'`: uploading files of any extension is prohibited

```
FFSECURITY = {
   ...
   'CARD_ATTACHMENTS_FILENAME_REGEXP': r'.*\.txt',
   ...

   }
```

Tip: Commenting out the `CARD_ATTACHMENTS_FILENAME_REGEXP` parameter also allows all file extensions.

3. Restart the `findface-multi-findface-multi-legacy-1` container.

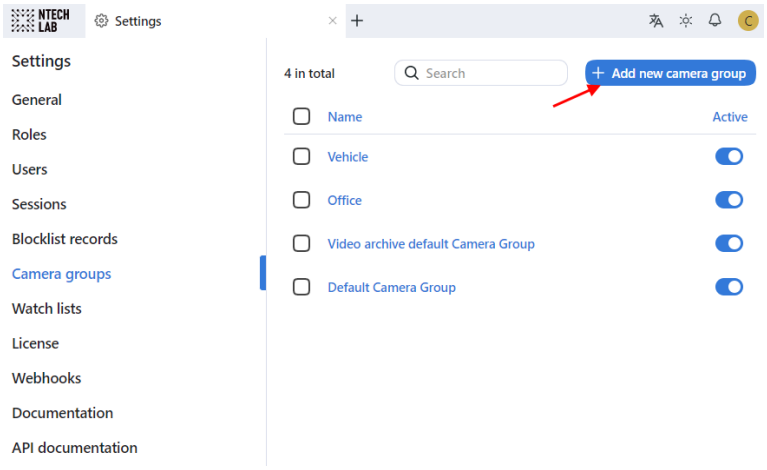
```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

2.4.4 Camera Groups

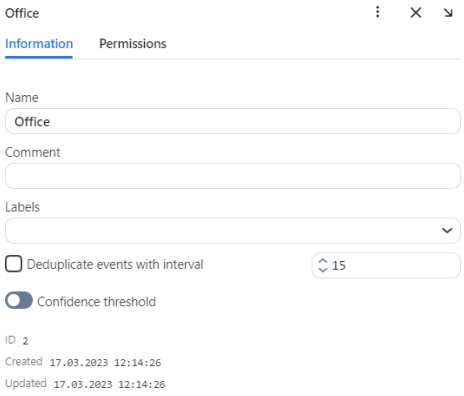
Camera groups are entities that are used for video sources classification. After processing a video, the system will attribute the object recognition events obtained from the video to a designated camera group. It makes the further event handling and search a lot easier.

To create a camera group, do the following:

1. Navigate *Settings* -> *Camera groups*.
2. Click + *Add new camera group*.



3. On the *Information* tab, specify the group name. Add a comment if needed.



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.

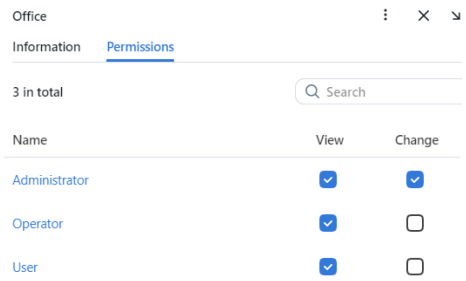
Note: To complete the allocation, list the labels in the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file. See *Allocate findface-video-worker to Camera Group* for details.

---

5. If you want to deduplicate events from cameras that belong to the same group, i. e. exclude coinciding events, check *Deduplicate event with interval* and specify the deduplication interval in seconds (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, enable *Confidence threshold* and specify the threshold value.
7. Save the changes.
8. On the *Permissions* tab, assign privileges on the camera group, specifying which user roles are allowed to change/view the camera group settings.



9. Click *Save*.
10. Activate the camera group.

2.4.5 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 *Settings* 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 `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file and specify the allocation labels in the dictionary format (labels `MyLabel1`, `MyLabel2` in the example below).

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml

labels: {"MyLabel1": "true", "MyLabel2": "true"}
```

5. Restart the `findface-multi-findface-video-worker-1` container.

```
sudo docker container restart findface-multi-findface-video-worker-1
```

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. Restart the similar-labeled camera.

2.4.6 Watch Lists

The appearance of specific individuals and vehicles in the video is monitored with a set of default and custom watch lists.

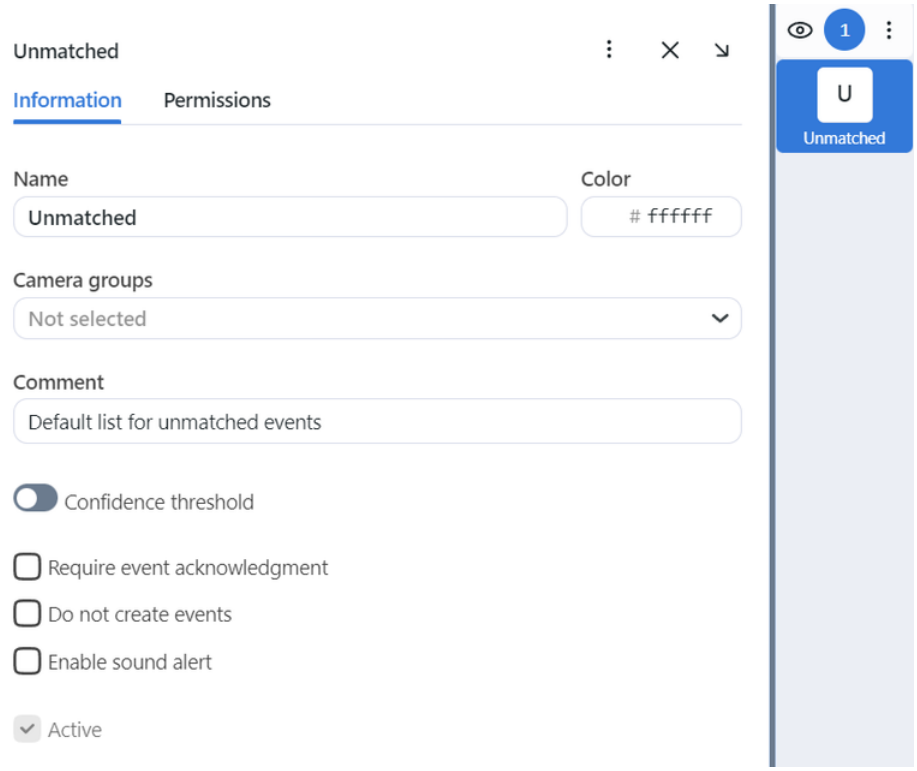
Records of individuals and vehicles are assigned to watch lists. Once a watch list is activated, the system will be looking for each person or a vehicle on it during video processing.

You can create as many custom watch lists as necessary: wanted, suspects, etc. — subject to your needs.

In this section:

Monitoring Unmatched Faces

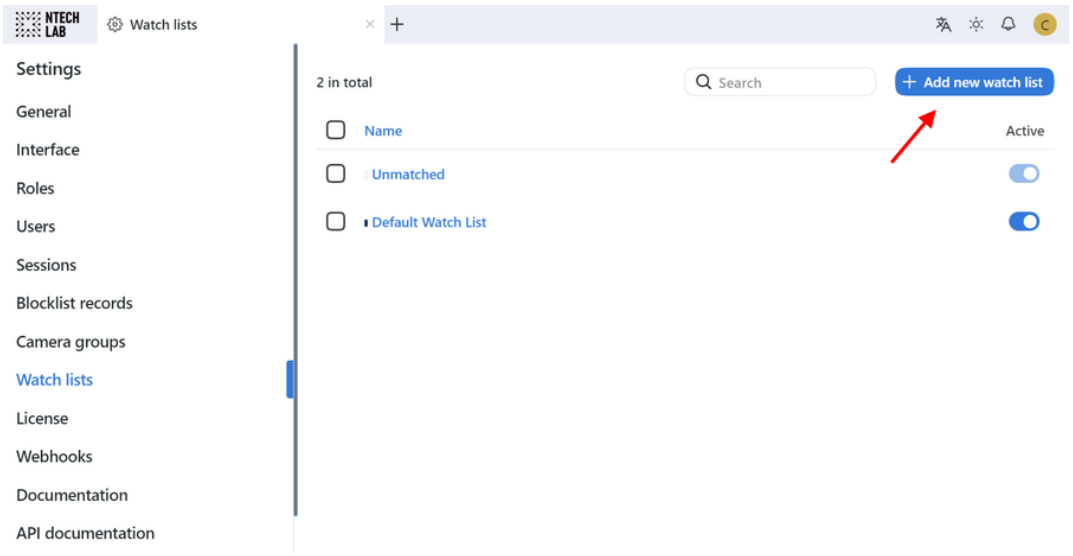
FindFace Multi features a special pre-configured watch list used for monitoring only unmatched events. This watch list cannot be removed from the system. To edit its settings, navigate to the *Settings* tab. Click *Watch Lists* and then click *Unmatched*.



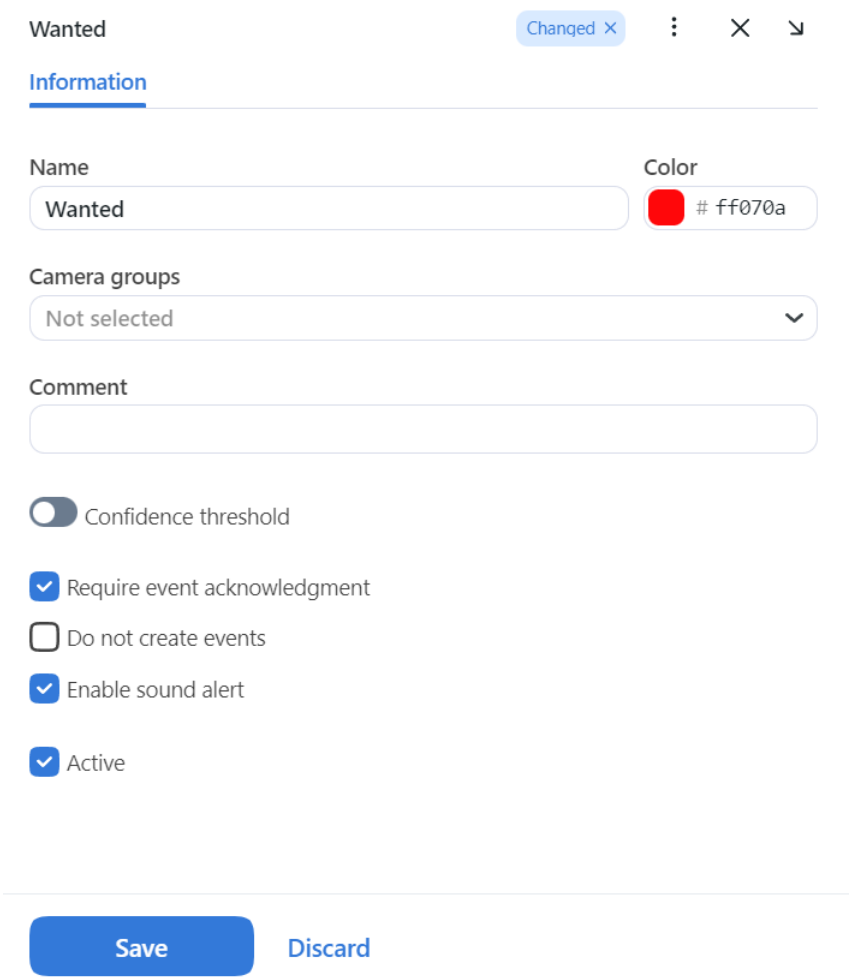
Create Watch List

To create a custom watch list, do the following:

1. Navigate *Settings* -> *Watch Lists*.
2. Click + *Add new watch list*.



3. On the *Information* tab, specify the watch list name.
4. From the *Color* palette, select a color which will be shown in notifications for this list.

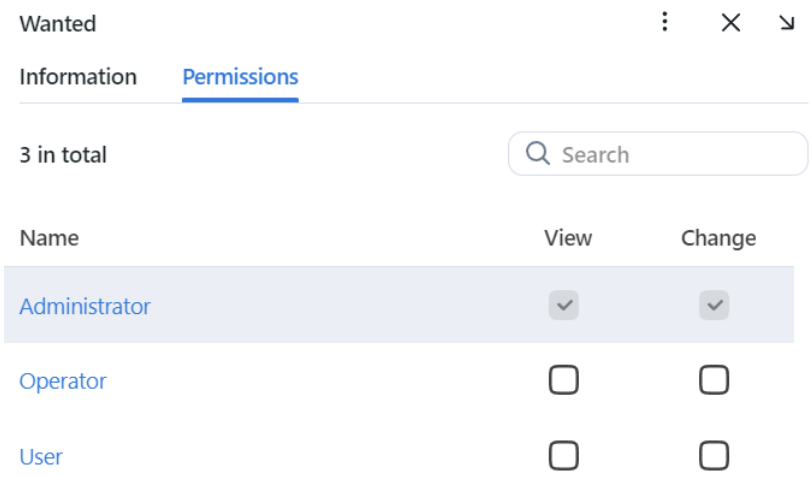


5. Describe the watch list in a comment if needed.
6. By default, all watch lists in the system are applied the *generic confidence threshold*. To set an individual threshold for the watch list, check *Confidence threshold* and specify the threshold value.

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 prior (support@ntechlab.com).

---

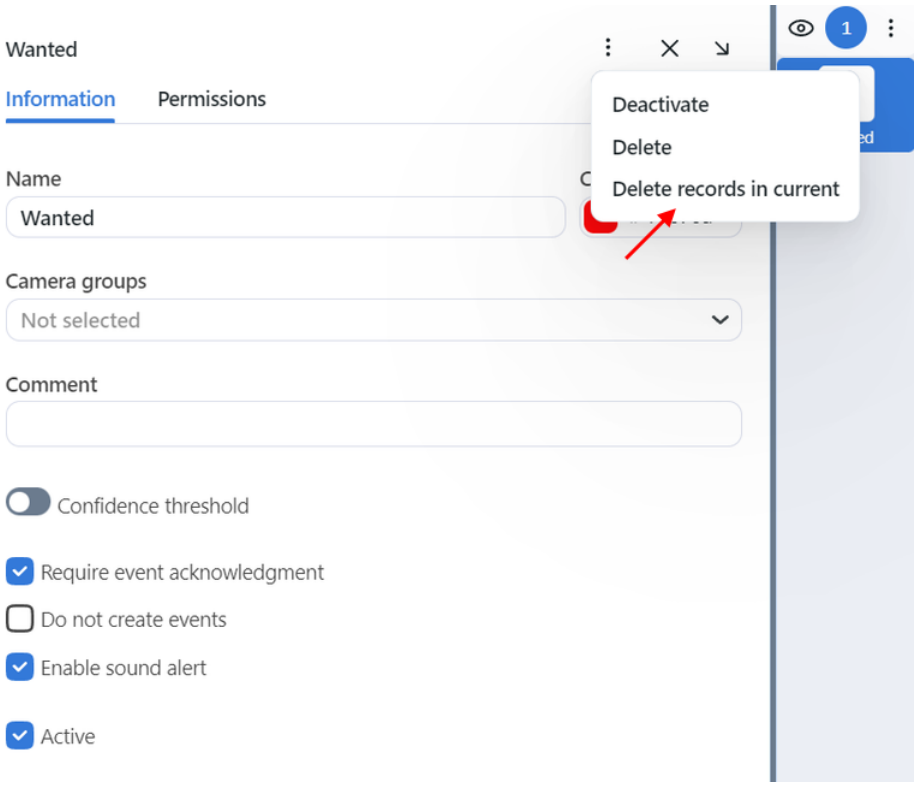
7. Enable *Require event acknowledgment* if it is mandatory that events associated with the list be manually acknowledged.
8. Enable *Do not create events* to avoid creating events once an object is detected.
9. Enable *Enable sound alert* to turn on sound notifications for the list if needed.
10. On the *Permissions* tab, assign privileges on the watch list, specifying which user roles are allowed to change/view the watch list settings.



11. Activate and save the watch list.

Remove Watch List

To remove a custom watch list, delete records associated with it first. Otherwise, the system will return an error and will not delete the watch list.



Default Watch List and *Unmatched* watch list cannot be removed from the system.

2.4.7 Configure Saving Images in Reports

When building *reports*, 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 `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.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 /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.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,
```

Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

2.4.8 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 *Settings* tab. Click *Camera Groups*.
3. Open the camera group settings.
4. Select *Deduplicate events with interval* 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 record 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.

2.4.9 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 `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. Disable saving unmatched events by setting `'IGNORE_UNMATCHED': True`.

```
...

FFSECURITY = {
    ...

    # do not save unmatched events (GDPR support)
    'IGNORE_UNMATCHED': True,

    ...

}
```

3. For events with matches, enable blurring all unmatched objects in full frames. To do so, set `'BLUR_UNMATCHED_OBJECTS': True`. Optionally, you can modify the default JPEG quality of those frames.

```
...

FFSECURITY = {
    ...
    # blur all unmatched objects on the full frame of the matched event (GDPR␣
↪support)
    'BLUR_UNMATCHED_OBJECTS': True,

    # full frame jpeg quality when `BLUR_UNMATCHED_OBJECTS` is enabled
    'BLURRED_FULLFRAME_JPEG_QUALITY': 85,
    ...

}
```

4. Enable blurring all unmatched objects on the *Video Wall*. To do so, set `"gdpr": True` in the `FFSECURITY_UI_CONFIG` -> `video_player` section.

```
FFSECURITY_UI_CONFIG = {
    ...
```

(continues on next page)

(continued from previous page)

```
   "video_player": {
      "overlay": {
        ...
        "gdpr": True
      }
   }
    ...
}
```

5. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

See also:

Video Wall

2.4.10 Configure Video Recorder

The FindFace Core internal *architecture* allows for embedding Video Recorder, an additional functionality that records, stores, and plays back video data from cameras.

This section is about the Video Recorder configuration.

In this section:

Enable Video Recorder

To enable Video Recorder, do the following:

1. Enable the `findface-video-worker` service to transfer video chunks to the `findface-video-storage` service.

   Important: This setting makes the Video Recorder functionality available over *HTTP API*. It's also obligatory if you want to harness Video Recorder as part of the FindFace Multi *web interface*.

   Do the following:

   1. Open the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration to enable `findface-video-worker` service that will supply Video Recorder with video.

      ```
      sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.
      ↪yaml
      ```

2. Set `enabled: true` in the `recorder` section.

```
recorder:
  enabled: true
  ...
```

3. Restart `findface-multi-findface-video-worker-1` container.

```
sudo docker container restart findface-multi-findface-video-worker-1
```

2. Configure Video Recorder to work as part of the FindFace Multi web interface.

Note: Omit the following steps if you do not need the Video Recorder tools to appear in the FindFace Multi web interface.

Do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.
↪py
```

2. Find the `FFSECURITY_UI_CONFIG` -> `vms` section.

   The list of vms parameters to configure is the following:

Parameter	Description
`vms -> "enabled"`	Set `True` to add the *Record video* checkbox to *camera settings*.
`vms -> "video_player"`	Set `True` to enable the video player *camera settings* to open in event notifications and *camera preview* (instead of static frames). This setting requires `"enabled": True`.

3. Find the `FFSECURITY_UI_CONFIG` -> `video_player` section.

   The list of video player parameters to configure is the following:

Parameter	Description
"overlay" -> "objects"	Defines the settings for displaying bbox and attribute data to the objects "faces", "bodies", "cars". Set True or False to enable/disable bbox and attribute data display for the relevant objects.
"overlay" -> "gdpr"	Set True to blurring all unmatched objects displayed on the video player. To fully comply with the personal data protection laws, follow *this guide*.
"timeline" -> "min_zoom"	Defines the largest possible scale on the *video player timeline*, seconds per pixel.
"timeline" -> "max_zoom"	Defines the smallest possible scale on the timeline, seconds per pixel.
"timeline" -> "objects" -> "events" "faces" -> "enabled"	Set True to enable marking face events on the timeline.
"timeline" -> "objects" -> "events" "faces" -> "limit"	The maximum number of face events simultaneously marked on the timeline. If there is more face events than that, you will be asked to zoom in.
"timeline" -> "objects" -> "events" "faces" -> "matchedColor"	Color of the matched face events.
"timeline" -> "objects" -> "events" "faces" -> "unmatchedColor"	Color of the unmatched face events.
"timeline" -> "objects" -> "events" "bodies" -> "enabled"	Set True to enable marking bodies events on the timeline.
"timeline" -> "objects" -> "events" "bodies" -> "limit"	The maximum number of body events simultaneously marked on the timeline. If there is more body events than that, you will be asked to zoom in.
"timeline" -> "objects" -> "events" "bodies" -> "matchedColor"	Color of the matched body events.
"timeline" -> "objects" -> "events" "bodies" -> "unmatchedColor"	Color of the unmatched body events.
"timeline" -> "objects" -> "events" "cars" -> "enabled"	Set True to enable marking vehicle events on the timeline.
"timeline" -> "objects" -> "events" "cars" -> "limit"	The maximum number of vehicle events simultaneously marked on the timeline. If there is more vehicle events than that, you will be asked to zoom in.
"timeline" -> "objects" -> "events" "cars" -> "matchedColor"	Color of the matched vehicle events.
"timeline" -> "objects" -> "events" "cars" -> "unmatchedColor"	Color of the unmatched vehicle events.
"timeline" -> "objects" -> "episodes" "humans" -> "enabled"	Set True to enable marking human episodes on the timeline.
"timeline" -> "objects" -> "episodes" "humans" -> "limit"	The maximum number of humans episodes simultaneously marked on the timeline. If there is more face episodes than that, you will be asked to zoom in.
"timeline" -> "objects" -> "episodes" "humans" -> "matchedColor"	Color of the matched human episodes.
"timeline" -> "objects" -> "episodes" "humans" -> "unmatchedColor"	Color of the unmatched human episodes.

```
FFSECURITY_UI_CONFIG = {

"video_player": {
 "overlay": {
     "objects": {
         "faces": {
           "bbox": True,
           "info": True
         },
         "bodies": {
           "bbox": True,
           "info": True
         },
         "cars": {
           "bbox": True,
           "info": True
         }
     },
     "gdpr": False
 },
 "timeline": {
     "min_zoom": 0.2,
     "max_zoom": 200,
     "objects": {
         "events": {
             "faces": {
                 "enabled": True,
                 "limit": 500,
                 "matchedColor": "rgba(6,193,103,0.8)",
                 "unmatchedColor": "rgba(232,92,74,0.8)",
             },
             "bodies": {
                 "enabled": False,
                 "limit": 500,
                 "matchedColor": "rgba(6,193,103,0.8)",
                 "unmatchedColor": "rgba(232,92,74,0.8)",
             },
             "cars": {
                 "enabled": False,
                 "limit": 500,
                 "matchedColor": "rgba(6,193,103,0.8)",
                 "unmatchedColor": "rgba(232,92,74,0.8)",
             },
         },
         "episodes": {
             "humans": {
                 "enabled": False,
                 "limit": 500,
                 "matchedColor": "rgba(6,193,103,0.8)",
                 "unmatchedColor": "rgba(232,92,74,0.8)",
             },
             "cars": {
                 "enabled": False,
```

(continues on next page)

(continued from previous page)

```
                "limit": 500,
                "matchedColor": "rgba(6,193,103,0.8)",
                "unmatchedColor": "rgba(232,92,74,0.8)",
            },
        },
    },
 },
},
"vms": {
   "enabled": True,
   "video_player": True,
},
```

4. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

Nuances of Disabling Video Recorder

If Video Recorder is running and recording video on selected cameras, and you need to disable it, be sure to disable video recording on the cameras first, before proceeding with the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` and `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration files.

2.4.11 Remove Video Records

See also:

Configure Video Recorder

There are two methods of video archive cleanup available in the system:

1. Regular cleanup
2. Manual cleanup via a console command

In this section:

Regular Video Cleanup

To configure the regular video cleanup, do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. Find the `SERVICES` section and set `"vms_cleanup": True`.

```
SERVICES = {
    "ffsecurity": {
        ...
        "vms_cleanup": True,
    }
}
```

3. Find the `VMS_CLEANUP_SETTINGS` section and configure what video to remove, using the following parameters:
   - `'CLEANUP_BETWEEN_TRACKS'`: set `True` to remove video intervals that do not contain recognition events
   - `'CLEANUP_ARCHIVE'`: set `True` to remove the entire video archive older than a given number of days
   - `'ARCHIVE_CLEANUP_AGE'`: the maximum age of video archive in the system, days. Applicable if `'CLEANUP_ARCHIVE': True`
   - `'CLEANUP_EVENTS_TYPES'`: event types that will be kept in the video archive, in the form of a dictionary, e.g., [‘face’, ‘body’, ‘car’]
   - `'KEEP_EVENT_BEFORE_AFTER'`: interval before and after a track with events, that won’t be removed in order to preserve meaningful information related to them, seconds. The value must be less than `CLEANUP_THRESHOLD`
   - `'CLEANUP_THRESHOLD'`: minimum interval between tracks with events, seconds. If the interval between tracks is shorter, it won’t be removed

   Tip: See the RRULE calculator here.

```
'VMS_CLEANUP_SETTINGS': {
    'CLEANUP_BETWEEN_TRACKS': True,
    'CLEANUP_ARCHIVE': True,
    'ARCHIVE_CLEANUP_AGE': 9,
    'CLEANUP_EVENTS_TYPES': ['car'],
    # Add `safe_time_interval` in seconds to each `track` subject to delete
    # Prevents deletion of essential data.
    # Should be lower than CLEANUP_THRESHOLD
    'KEEP_EVENT_BEFORE_AFTER': 10,  # minimal allowed value
    # Threshold between `intervals` in seconds.
    # If duration between `tracks` < `CLEANUP_THRESHOLD`
    # interval between tracks will not be added to deletion tasks
    'CLEANUP_THRESHOLD': 360,  # minimal allowed value
},
```

4. Find the `'VMS_CLEANUP_SERVICE_SCHEDULE'` parameter and specify a recurrence rule (RRULE) for scheduling video cleanup.

Tip: See the RRULE calculator here.

```
# rrule (recurrence rule) for scheduling `vms_cleanup` service
'VMS_CLEANUP_SERVICE_SCHEDULE': 'RRULE:FREQ=HOURLY;INTERVAL=3;WKST=MO',
```

5. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

Manual Video Cleanup

You can manually remove video chunks older than the given number of days by executing the command below. To set the number of days, use the –vms-videos-max-age argument.

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup_vms --vms-videos-max-age=30
```

2.4.12 Custom Tabs, Fields, and Filters in Record Index

See also:

To create custom fields in the feature vector database, refer to *Custom Metadata in Tarantool*.

To add custom tabs and fields to the records of individuals and vehicles, do the following:

1. Prepare the list of custom tabs and fields you want to add to the records.
2. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

3. Customize the records of individuals. To do so, uncomment the `FFSECURITY -> CUSTOM_FIELDS -> human_card` section and modify the exemplary content, considering the following:
   - `'items'`: the list of fields in a record. Describe each field with the following parameters:
     - `'name'`: field's internal name, string.
     - `'default'`: field's default value. If a default value exceeds `1e14 - 1`, use a string data type to specify it, for example, `"123123.."` instead of `123123..`.
     - `'label'`: field's label in a record, string.
     - `'tab'`: tab that features the field.
     - `'display'`: display format (`form` or `list`), string or array.
     - `'description'`: field's description, string.
     - `'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 record.

```
FFSECURITY = {

...

# -- Custom model fields --
# Edit CUSTOM_FIELDS -> `human_card` section to customize human card fields.
# Edit CUSTOM_FIELDS -> `car_card` section to customize car card fields.
...
  'CUSTOM_FIELDS': {
      'human_card': {
          'items': [
              {
                  'name': 'personid',
                  'default': '',
                  'label': 'PersonID',
                  'display': ['list', 'form'],
                  'description': 'Sigur person ID',
                  'editable': False
              },
              {
                  'name': 'firstname',
                  'default': '',
                  'label': 'First Name',
                  'display': ['list', 'form'],
                  'description': 'Sigur first name',
```

(continues on next page)

(continued from previous page)

```
                'editable': False
            },
            {
                'name': 'lastname',
                'default': '',
                'label': 'Last Name',
                'display': ['list', 'form'],
                'description': 'Sigur last name',
                'editable': False
            },
            {
                'name': 'version',
                'default': '',
                'label': 'Version',
                'display': ['list', 'form'],
                'description': 'Sigur photo version',
                'editable': False
            }
        ],
        'filters': [
            {
                'name': 'personid',
                'label': 'Sigur person ID filter',
                'field': 'personid'
            }
        ]
    },
    'car_card': {}, # same fields are available
  },
}
```

4. Customize the vehicle records. To do so, duplicate the `human_card` section content into the `car_card` section and modify it by analogy.

5. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

You will see the custom content appear in the records.

2.4.13 Custom Metadata in Tarantool

It is often necessary to assign additional metadata to the objects extracted from images uploaded to the record index and now stored in the feature vector database.

In this section:

Customize Meta Fields of Face Objects

To assign custom meta fields to the face objects, do the following:

1. Prepare the list of custom meta fields to assign.
2. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

3. In the `FFSECURITY` section, uncomment the `CUSTOM_FIELDS -> face_object` section and modify the exemplary content, considering the following:
   - `field_name`: field's name;
   - `type`: data type (`uint`, `string` or `bool`);
   - `default`: field's 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 model fields --
    ...
    # Edit CUSTOM_FIELDS -> `face_object` section to customize face object fields.
    ...
    # 'CUSTOM_FIELDS': {
          ...
          'face_object': {
              'items': [
                  {
                      "field_name": "tag_name_1",
                      "type": "string",
                      "default": "change_me"
                  },
                  {
                      "field_name": "tag_name_2",
                      "type": "uint",
                      "default": 123
```

(continues on next page)

(continued from previous page)

```
                },
                {
                    "field_name": "tag_name_3",
                    "type": "bool",
                    "default": True
                },
            ]
        }
    },
}
```

4. *Add the new meta fields* to the feature vector database structure.
5. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

You can work with the new meta fields through *HTTP API* using the `objects/faces/` methods.

Customize Meta Fields of Body and Vehicle Objects

Assigning custom meta fields to the body and vehicle objects are similar to the faces. The only difference is that you will need to manually create the `CUSTOM_FIELDS -> body_object` and `CUSTOM_FIELDS -> car_object` sections. It is a good idea to duplicate the `CUSTOM_FIELDS -> face_object` section and use it as a starting point for further modifications.

```
FFSECURITY = {

...

    # -- Custom model fields --
    ...
    # Edit CUSTOM_FIELDS -> `face_object` section to customize face object fields.
    ...
    # 'CUSTOM_FIELDS': {
          ...
          'body_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
```

(continues on next page)

(continued from previous page)

```
                },
            ]
        }
        'car_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
                },
            ]
        }
    },
}
```

Similarly, you can work with the new meta fields using the `objects/bodies/` and `objects/cars/` methods of *HTTP API*.

Note: Assigning custom meta fields to the face, body and vehicle events are similar to the face, body and vehicle objects. You will also need to manually create the `CUSTOM_FIELDS` -> `face_event`, `CUSTOM_FIELDS` -> `body_event` and `CUSTOM_FIELDS` -> `car_event` sections by similar way.

See also:

To create custom tabs, fields, and filters in cards, refer to *Custom Tabs, Fields, and Filters in Record Index*.

2.4.14 Console Bulk Record Upload

In addition to the *web interface upload*, you can bulk-upload records to the record index via the **`uploader.py`** console utility. We recommend preferring this utility over the web interface if the number of uploaded photos is more than 10,000.

Tip: To view the **`uploader.py`** help, execute:

```
docker exec findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/python3 /
↪tigre_prototype/ffsecurity/uploader.py --help

Usage: uploader.py [OPTIONS] COMMAND [ARGS]...

Options:
```

(continues on next page)

(continued from previous page)

```
--job FILE        Job file (default: enroll-job.db)
--log-level TEXT  Log level
--fsync BOOLEAN   Call fsync() to prevent data loss on power failure
--help            Show this message and exit.

Commands:
add    Add items from CSV or TSV file to job
print  Print contents of job file as JSON
run    Run upload job
```

```
docker exec findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/python3 /
↪tigre_prototype/ffsecurity/uploader.py add --help

Usage: uploader.py add [OPTIONS] FILES...

Options:
--format [csv|tsv]  Input file format - CSV or TSV
--delimiter TEXT    Field delimiter - by default it's "\t" for TSV and ","
                    for CSV

--help              Show this message and exit.
```

```
 docker exec findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/python3 /
↪tigre_prototype/ffsecurity/uploader.py print --help

 Usage: uploader.py print [OPTIONS]

 Print contents of job file as JSON

 Options:
--failed  Show only failed images
--noface  Show only images without detection
--help    Show this message and exit.
```

```
docker exec findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/python3 /
↪tigre_prototype/ffsecurity/uploader.py run --help

Usage: uploader.py run [OPTIONS]

Run upload job

Options:
--parallel INTEGER        Number of enroll threads (default: 10)
--api TEXT                API url (default: http://127.0.0.1:80/)
                          [required]

--user TEXT               API username  [required]
--password TEXT           API password  [required]
--watch-lists TEXT        Comma-separated list of card list ids  [required]
--inactive                Mark new cards as inactive
--failed                  Include failed images
```

(continues on next page)

(continued from previous page)

```
--noface                    Include images without detection
--all-faces                 Enroll all found faces on each image
--detect-timeout INTEGER    Request timeout for detect photos
--logging-delta INTEGER     Logging period delta
--help                      Show this message and exit.
```

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, you can use a `script`.

   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, upload the script to your home directory, for example `/home/ubuntu`, and run the following command:

   ```
   sudo docker run -it --rm --network host --volume ${PWD}:/home/ubuntu/create_cards --
   ↪volume /home/ubuntu/photos:/home/ubuntu/photos docker.int.ntl/ntech/multi/multi/
   ↪legacy:ffmulti-2.0.0 sh -c "cd /home/ubuntu/create_cards && /opt/findface-
   ↪security/bin/python3 tsv_builder.py /home/ubuntu/photos"
   ```

   where `/home/ubuntu/photos` is a directory with your photos.

2. Create a job file out of a CSV or TSV file by using the `add` utility command. As a result, a file `enroll-job.db` will be created and saved in a current directory.

   ```
   sudo docker run -it --rm --network host --volume ${PWD}:/home/ubuntu/create_cards␣
   ↪docker.int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0 sh -c "cd /home/ubuntu/
   ↪create_cards && /opt/findface-security/bin/python3 /tigre_prototype/ffsecurity/
   ↪uploader.py add /home/ubuntu/create_cards/images.tsv"
   ```

   The `add` utility 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 specifying the path to the photos (for example, `/home/ubuntu/photos`) and passing the necessary arguments. Use the following command:

   ```
   sudo docker run -it --rm --network host --volume ${PWD}:/home/ubuntu/create_cards --
   ↪volume /home/ubuntu/photos:/home/ubuntu/photos docker.int.ntl/ntech/multi/multi/
   ```

(continues on next page)

(continued from previous page)

```
↪legacy:ffmulti-2.0.0 sh -c "cd /home/ubuntu/create_cards && /opt/findface-
↪security/bin/python3 /tigre_prototype/ffsecurity/uploader.py run --user admin --
↪password password --watch-lists 1"
```

The `run` utility 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.
- `--api`: `findface-security` API URL, `http://127.0.0.1:80/` by default. Mandatory option.
- `--user`: login. Mandatory option.
- `--password`: password. Mandatory option.
- `--watch-lists`: comma-separated list of the watch lists id's. Mandatory option.
- `--inactive`: mark new records as inactive.
- `--failed`: should an error occur during the job file processing, correct the mistake and try again with this option.
- `--noface`: by default, images classified as having no faces will be assigned the `NOFACE` status and automatically excluded from the upload. To attempt re-detecting faces in such images, re-run the job file with this option. If the re-detection gives a negative result again, an image will be skipped and a relevant record will appear in the upload log.
- `--all-faces`: upload all faces from a photo if it features several faces.
- `--detect-timeout`: request timeout for detect photos.
- `--logging-delta`: logging period delta.

4. (Optional) Print the job processing results as JSON. If necessary, you can print only failed images/ images without detected faces.

   The `print` utility options:

   - `--failed`: show only failed images.
   - `--noface`: show only images without detection.

2.5 Configure FindFace Multi Neural Networks

2.5.1 Neural Networks Summary

Here you can find a summary of neural network models created by our Lab and used in FindFace Multi.

You can find installed models at `/opt/findface-multi/models/`.

Important: The default face biometrics model upon a clean install is `mango_320`.

Face, vehicle, and body detection

```
ls /opt/findface-multi/models/detector/

body.gustav_normal.015.cpu.fnk  body.jasmine_fast.018.cpu.fnk  car.gustav_accurate.004.
↪cpu.fnk  car.jasmine_fast.005.cpu.fnk  face.jasmine_fast.003.cpu.fnk  headbodyface.
↪alpha000_normal.001.gpu.fnk
body.gustav_normal.015.gpu.fnk  body.jasmine_fast.018.gpu.fnk  car.gustav_accurate.004.
↪gpu.fnk  car.jasmine_fast.005.gpu.fnk  face.jasmine_fast.003.gpu.fnk
```

Face and body image normalization

```
ls /opt/findface-multi/models/facenorm/

bee.v3.cpu.fnk    bee_fast.gpu.fnk              crop2x.v2_maxsize400.cpu.fnk  crop2x.v2_
↪no_maxsize.gpu.fnk  facenorm.multicrop_full_center_size400.cpu.fnk  facenorm.multicrop_
↪full_crop2x_size400.gpu.fnk
bee.v3.gpu.fnk    crop1x.v2_maxsize400.cpu.fnk  crop2x.v2_maxsize400.gpu.fnk  cropbbox.
↪v2.cpu.fnk           facenorm.multicrop_full_center_size400.gpu.fnk
bee_fast.cpu.fnk  crop1x.v2_maxsize400.gpu.fnk  crop2x.v2_no_maxsize.cpu.fnk  cropbbox.
↪v2.gpu.fnk           facenorm.multicrop_full_crop2x_size400.cpu.fnk
```

Face recognition

```
ls /opt/findface-multi/models/face/

lime.v2.cpu.fnk  lime.v2.gpu.fnk  mango_320.cpu.fnk  mango_320.gpu.fnk
```

Face attribute recognition

```
ls /opt/findface-multi/models/faceattr/

age.v2.cpu.fnk    beard.v0.gpu.fnk     gender.v2.cpu.fnk    glasses3.v0.gpu.fnk ␣
↪liveness.goodwin.cpu.fnk  liveness.pacs.v2.gpu.fnk  medmask3.v2.cpu.fnk      quality_
↪fast.v1.gpu.fnk
age.v2.gpu.fnk    emotions.v1.cpu.fnk  gender.v2.gpu.fnk    headpose.v2.cpu.fnk ␣
↪liveness.goodwin.gpu.fnk  liveness.web.v0.cpu.fnk   medmask3.v2.gpu.fnk
beard.v0.cpu.fnk  emotions.v1.gpu.fnk  glasses3.v0.cpu.fnk  headpose.v2.gpu.fnk ␣
↪liveness.pacs.v2.cpu.fnk  liveness.web.v0.gpu.fnk   quality_fast.v1.cpu.fnk
```

Vehicle image normalization

```
ls /opt/findface-multi/models/carnorm/

anaferon.v5.cpu.fnk  anaferon.v5.gpu.fnk  anaferon.v7.cpu.fnk  anaferon.v7.gpu.fnk ␣
↪briacon.v0.cpu.fnk  briacon.v0.gpu.fnk
```

Vehicle recognition

```
ls /opt/findface-multi/models/carrec/

alonso.cpu.fnk  alonso.gpu.fnk
```

Vehicle attribute recognition

```
ls /opt/findface-multi/models/carattr/

carattr.categories.v0.cpu.fnk     carattr.license_plate.v7.gpu.fnk          carattr.
↪orientation.v0.cpu.fnk  carattr.quality.v0.gpu.fnk          carattr.weight_types7.v0.
↪cpu.fnk  description.v0.gpu.fnk
carattr.categories.v0.gpu.fnk     carattr.license_plate_quality.v1.cpu.fnk  carattr.
↪orientation.v0.gpu.fnk  carattr.special_types11.v1.cpu.fnk  carattr.weight_types7.v0.
↪gpu.fnk
carattr.license_plate.v7.cpu.fnk  carattr.license_plate_quality.v1.gpu.fnk  carattr.
↪quality.v0.cpu.fnk      carattr.special_types11.v1.gpu.fnk  description.v0.cpu.fnk
```

Body recognition

```
ls /opt/findface-multi/models/pedrec/

pedrec.clio.cpu.fnk  pedrec.clio.gpu.fnk
```

Body attribute recognition

```
ls /opt/findface-multi/models/pedattr/

pedattr.age_gender.v0.cpu.fnk  pedattr.bags.v0.cpu.fnk  pedattr.clothes_type.v0.cpu.fnk ␣
↪pedattr.color.v1.cpu.fnk  pedattr.protective.v1.cpu.fnk  pedattr.quality.v0.cpu.fnk
pedattr.age_gender.v0.gpu.fnk  pedattr.bags.v0.gpu.fnk  pedattr.clothes_type.v0.gpu.fnk ␣
↪pedattr.color.v1.gpu.fnk  pedattr.protective.v1.gpu.fnk  pedattr.quality.v0.gpu.fnk
```

2.5.2 Enable Face and Face Attribute Recognition

FindFace Multi allows you to recognize human faces and face attributes. Subject to your needs, you can enable recognition of such face attributes as age, gender, emotions, beard, glasses, medical masks, head position, or liveness.

Face and face attribute recognition can be automatically enabled and configured during the *FindFace Multi installation*. If you skipped this step, you can manually do it later. Face and face attribute recognition works on both GPU- and CPU-acceleration.

Face object recognition is enabled by default. In case you removed face as a recognition object during the installation, you can add it later by following the below steps. If face object recognition is already installed, and you only need to enable face attribute recognition, jump to the steps 1.5, 1.6 and 4.1, 4.2. Other steps should be skipped.

1. To enable face recognition, do the following:

   Specify neural network models for face object detection in the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` configuration file.

   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 `findface-extraction-api.yaml` configuration file.

      ```
      sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-
      ↪api.yaml
      ```

   2. Specify the face detector model in the `detectors -> models` section by pasting the following code:

      #### GPU

      ```
      detectors:

        ...
        models:
          ...
          face_jasmine:
            aliases:
            - face
            - nnd
            - cheetah
            model: detector/face.jasmine_fast.003.gpu.fnk
            options:
              min_object_size: 32
              resolutions:
              - 256x256
              - 384x384
              - 512x512
              - 768x768
              - 1024x1024
              - 1536x1536
              - 2048x2048

        ...
      ```

CPU

```
detectors:

  ...
  models:
    ...
    face_jasmine:
      aliases:
      - face
      - nnd
      - cheetah
      model: detector/face.jasmine_fast.003.cpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 256x256
        - 384x384
        - 512x512
        - 768x768
        - 1024x1024
        - 1536x1536
        - 2048x2048

  ...
```

3. Make sure that the `objects -> face` section contains the `quality_attribute: face_quality` and the `base_normalizer: facenorm/crop2x.v2_maxsize400.gpu.fnk` or the `base_normalizer: facenorm/crop2x.v2_maxsize400.cpu.fnk`, depending on your acceleration type:

GPU

```
objects:
  ...
  face:
    base_normalizer: facenorm/crop2x.v2_maxsize400.gpu.fnk
    quality_attribute: face_quality
  ...
```

CPU

```
objects:
  ...
  face:
    base_normalizer: facenorm/crop2x.v2_maxsize400.cpu.fnk
    quality_attribute: face_quality
  ...
```

4. Specify the face normalizer models in the `normalizers` section by pasting the following code:

GPU

```
normalizers:
  ...
  models:
    crop1x:
      model: facenorm/crop1x.v2_maxsize400.gpu.fnk
    crop2x:
      model: facenorm/crop2x.v2_maxsize400.gpu.fnk
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
    multicrop_full_center:
      model: facenorm/facenorm.multicrop_full_center_size400.gpu.fnk
    multicrop_full_crop2x:
      model: ''
    norm200:
      model: facenorm/bee.v3.gpu.fnk
  ...
```

CPU

```
normalizers:
  ...
  models:
    crop1x:
      model: facenorm/crop1x.v2_maxsize400.cpu.fnk
    crop2x:
      model: facenorm/crop2x.v2_maxsize400.cpu.fnk
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
    multicrop_full_center:
      model: facenorm/facenorm.multicrop_full_center_size400.cpu.fnk
    multicrop_full_crop2x:
      model: ''
    norm200:
      model: facenorm/bee.v3.cpu.fnk
  ...
```

5. **Note:** This step is required to enable face attribute recognition.

   To enable face attribute recognition, do the following:

   In the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` configuration file, specify the extraction 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.

GPU

```
extractors:
  ...
  models:
    face_age: faceattr/age.v2.gpu.fnk
    face_beard: faceattr/beard.v0.gpu.fnk
    face_beard4: ''
    face_countries47: ''
    face_emben: face/mango_320.gpu.fnk
    face_emotions: faceattr/emotions.v1.gpu.fnk
    face_eyes_attrs: ''
    face_eyes_openness: ''
    face_gender: faceattr/gender.v2.gpu.fnk
    face_glasses3: faceattr/glasses3.v0.gpu.fnk
    face_glasses4: ''
    face_hair: ''
    face_headpose: faceattr/headpose.v2.gpu.fnk
    face_headwear: ''
    face_highlight: ''
    face_liveness: faceattr/liveness.web.v0.gpu.fnk
    face_luminance_overexposure: ''
    face_luminance_underexposure: ''
    face_luminance_uniformity: ''
    face_medmask3: faceattr/medmask3.v2.gpu.fnk
    face_medmask4: ''
    face_mouth_attrs: ''
    face_quality: faceattr/quality_fast.v1.gpu.fnk
    face_scar: ''
    face_sharpness: ''
    face_tattoo: ''
    face_validity: ''
```

CPU

```
extractors:
  ...
  models:
    face_age: faceattr/age.v2.cpu.fnk
    face_beard: faceattr/beard.v0.cpu.fnk
    face_beard4: ''
    face_countries47: ''
    face_emben: face/mango_320.cpu.fnk
    face_emotions: faceattr/emotions.v1.cpu.fnk
    face_eyes_attrs: ''
    face_eyes_openness: ''
    face_gender: faceattr/gender.v2.cpu.fnk
    face_glasses3: faceattr/glasses3.v0.cpu.fnk
    face_glasses4: ''
    face_hair: ''
    face_headpose: faceattr/headpose.v2.cpu.fnk
```

(continues on next page)

(continued from previous page)

```
  face_headwear: ''
  face_highlight: ''
  face_liveness: faceattr/liveness.web.v0.cpu.fnk
  face_luminance_overexposure: ''
  face_luminance_underexposure: ''
  face_luminance_uniformity: ''
  face_medmask3: faceattr/medmask3.v2.cpu.fnk
  face_medmask4: ''
  face_mouth_attrs: ''
  face_quality: faceattr/quality_fast.v1.cpu.fnk
  face_scar: ''
  face_sharpness: ''
  face_tattoo: ''
  face_validity: ''
```

The following extraction models are available:

Extractor	Acceleration	Configure as follows
age	CPU	face_age: faceattr/age.v2.cpu.fnk
	GPU	face_age: faceattr/age.v2.gpu.fnk
beard	CPU	face_beard: faceattr/beard.v0.cpu.fnk
	GPU	face_beard: faceattr/beard.v0.gpu.fnk
individual face feature vector	CPU	face_emben: face/mango_320.cpu.fnk
	GPU	face_emben: face/mango_320.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
head position	CPU	face_headpose: faceattr/headpose.v2.cpu.fnk
	GPU	face_headpose: faceattr/headpose.v2.gpu.fnk
face liveness	CPU	face_liveness: faceattr/liveness.web.v0.cpu.fnk
	GPU	face_liveness: faceattr/liveness.web.v0.gpu.fnk
face mask	CPU	face_medmask3: faceattr/medmask3.v2.cpu.fnk
	GPU	face_medmask3: faceattr/medmask3.v2.gpu.fnk
face quality	CPU	face_quality: faceattr/quality_fast.v1.cpu.fnk
	GPU	face_quality: faceattr/quality_fast.v1.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:
  face_age: ''
  face_beard: ''
  face_beard4: ''
  face_countries47: ''
  face_emben: ''
  face_emotions: ''
  face_eyes_attrs: ''
  face_eyes_openness: ''
  face_gender: ''
  face_glasses3: ''
  face_glasses4: ''
  face_hair: ''
  face_headpose: ''
  face_headwear: ''
  face_highlight: ''
  face_liveness: ''
  face_luminance_overexposure: ''
  face_luminance_underexposure: ''
  face_luminance_uniformity: ''
  face_medmask3: ''
  face_medmask4: ''
  face_mouth_attrs: ''
  face_quality: ''
  face_scar: ''
  face_sharpness: ''
  face_tattoo: ''
  face_validity: ''
```

Important: The `face_liveness` extraction model `liveness.web.v0` is enabled by default. Do not disable it if you use *authentication* by face.

6. Restart the `findface-multi-findface-extraction-api-1` container.

```
sudo docker container restart findface-multi-findface-extraction-api-1
```

2. To enable face recognition, modify the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file.

    1. In the `models` section, specify the face neural network models by analogy with the example below:

GPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.
↪yaml

models:
  ...
  detectors:
    ...
    face:
      fnk_path: /usr/share/findface-data/models/detector/face.jasmine_fast.003.
↪gpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    face_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.
↪gpu.fnk
    face_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/crop1x.v2_maxsize400.
↪gpu.fnk
    ...
  extractors:
    ...
    face_quality:
      fnk_path: /usr/share/findface-data/models/faceattr/quality_fast.v1.gpu.fnk
      normalizer: face_norm_quality
```

CPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.
↪yaml

models:
  ...
  detectors:
    ...
    face:
      fnk_path: /usr/share/findface-data/models/detector/face.jasmine_fast.003.
↪cpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    face_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.
↪cpu.fnk
    face_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/crop1x.v2_maxsize400.
↪cpu.fnk
```

(continues on next page)

(continued from previous page)

```
  ...
extractors:
  ...
  face_quality:
    fnk_path: /usr/share/findface-data/models/faceattr/quality_fast.v1.cpu.fnk
    normalizer: face_norm_quality
```

2. Make sure that the `objects -> face` section is included:

```
objects:
  ...
  face:
    normalizer: face_norm
    quality: face_quality
    track_features: ''
```

3. Restart the `findface-multi-findface-video-worker-1` container.

```
sudo docker container restart findface-multi-findface-video-worker-1
```

3. To enable face recognition, open the `/opt/findface-multi/configs/findface-video-manager/findface-video-manager.yaml` configuration file and make sure it contains the `face` section in `detectors` that looks similar to the example below.

```
sudo vi /opt/findface-multi/configs/findface-video-manager/findface-video-manager.
↪yaml

detectors:
  ...
  face:
    filter_min_quality: 0.5
    filter_min_size: 60
    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
    tracker_type: simple_iou
    track_deep_sort_matching_threshold: 0.65
    track_deep_sort_filter_unconfirmed_tracks: true
```

(continues on next page)

(continued from previous page)

```
    track_object_is_principal: false
    track_history_active_track_miss_interval: 0
```

4. **Note:** This step is required to enable face attribute recognition.

   Enable recognition of face attributes in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

   1. In the `FFSECURITY` section, specify the face attributes that you want to display for the face recognition events.

      ```
      # available features: age, beard, emotions, gender, glasses, headpose, medmask
      'FACE_EVENTS_FEATURES': ['glasses', 'beard', 'age', 'gender', 'headpose',
      ↪'medmask', 'emotions'],
      ```

   2. Restart the `findface-multi-findface-multi-legacy-1` container.

      ```
      sudo docker container restart findface-multi-findface-multi-legacy-1
      ```

2.5.3 Headpose Recognition

Headpose feature refers to the ability of the camera to detect and track the orientation and movement of a person's head relative to the CCTV camera in real-time.

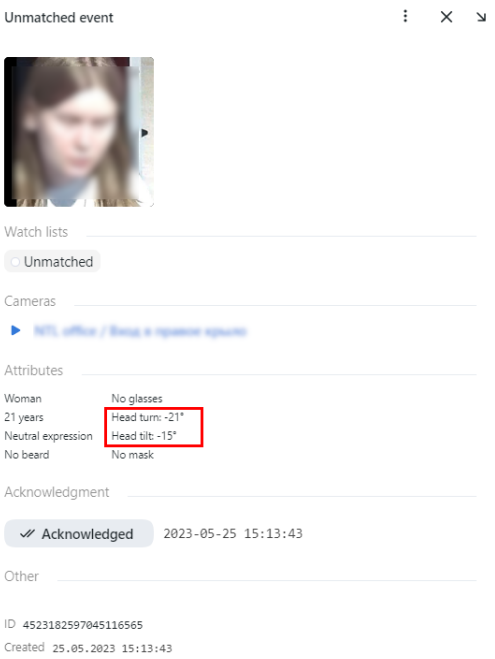
Warning: The headpose feature does not work when a person is wearing a medmask.

To detect the headpose (head turn and head tilt), FindFace Multi uses pitch and yaw.

- Pitch refers to the rotation of the head around its horizontal axis, which runs from ear to ear. Positive pitch indicates that the head is tilted forward, while negative pitch indicates that the head is tilted backward.
- Yaw refers to the rotation of the head around its vertical axis, which runs from top to bottom. Positive yaw indicates that the head is turned to the right, while negative yaw indicates that the head is turned to the left.

Pitch and yaw angles are measured relative to the camera and take on values from -90 to +90 degree.

You can see the value of head turn and head tilt in the attributes section of *Events* and filter face events by these parameters.



Possible scenarios of headpose recognition

Headpose recognition can be used in various scenarios where face recognition is used to improve accuracy and security, there are several of them:

- Physical Access Control System, PACS: improving employee access control systems by ensuring that the face of the employee matches the expected orientation. It means that if a person is near the camera, turns the head to the camera, but is not going to pass through the PACS, the access will not be provided, due to the pre-set value of the head position.
- Analytics of shopping center visitors: reducing the number of created low-quality face clusters. When a face cluster is forming, events corresponding to the specified values of the head rotation angles are taken into account. *See more for clusters*.

How to configure headpose

To configure headpose angle thresholds for creating clusters, do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. Enable the headpose threshold by setting `'FACE_CLUSTER_EVENT_HEADPOSE_THRESHOLDS_ENABLE': True,` .

3. If necessary, modify the lowest and highest threshold to pitch and yaw angles.

```
'FACE_CLUSTER_EVENT_YAW_ANGLE_LOWEST_THRESHOLD': -30,
'FACE_CLUSTER_EVENT_YAW_ANGLE_HIGHEST_THRESHOLD': 30,
```

(continues on next page)

(continued from previous page)

```
'FACE_CLUSTER_EVENT_PITCH_ANGLE_LOWEST_THRESHOLD': -60,
'FACE_CLUSTER_EVENT_PITCH_ANGLE_HIGHEST_THRESHOLD': 60,
```

4. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

2.5.4 Enable Vehicle and Vehicle Attribute Recognition

FindFace Multi allows you to recognize a single vehicle and its attributes.

The vehicle attributes are as follows:

- license plate number (for selected countries),

Note: Currently, the following countries are supported:

Europe: Russia, Lithuania, Latvia, Estonia, Finland, Czech Republic, Serbia, Belarus, Ukraine, Moldova, Georgia, Azerbaijan, Armenia.

Asia: the UAE, Kazakhstan, Kyrgyzstan, Tajikistan, Turkmenistan, Uzbekistan, Saudi Arabia, Vietnam, India, Pakistan, Thailand.

North America: Mexico.

South America: Argentina, Brazil.

- color,
- make,
- model,
- body style,
- vehicle category,
- vehicle orientation (front, rear, or side),
- *special vehicle type* (taxi, route transport, carsharing, ambulance, police, rescue service, gas service, military, road service, other special),
- vehicle weight and body size.

Note: Special vehicle recognition as well as vehicle weight and body size recognition work for selected countries only and may not work for your country. For more information please contact your manager or our support team (support@ntechlab.com).

Vehicle recognition along with the attributes choice can be configured at the *installation* level. This section describes how to enable vehicle and vehicle attribute recognition in case this step has been skipped during installation.

To enable recognition of vehicles and their attributes, do the following:

1. Specify neural network models for vehicle recognition and vehicle attribute recognition in the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` configuration file.

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 `findface-extraction-api.yaml` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-
↪api.yaml
```

2. Specify the vehicle detector model in the `detectors -> models` section by pasting the following code:

GPU

```
detectors:

  ...
  models:
    ...
    gustav:
      aliases:
      - car
      - efreitor
      model: detector/car.gustav_accurate.004.gpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 256x256
        - 384x384
        - 512x512
        - 768x768
        - 1024x1024
        - 1536x1536
        - 2048x2048
  ...
```

CPU

```
detectors:

  ...
  models:
    ...
    gustav:
      aliases:
      - car
      - efreitor
      model: detector/car.gustav_accurate.004.cpu.fnk
      options:
        min_object_size: 32
```

(continues on next page)

(continued from previous page)

```
      resolutions:
      - 256x256
      - 384x384
      - 512x512
      - 768x768
      - 1024x1024
      - 1536x1536
      - 2048x2048
  ...
```

3. Make sure that the `objects -> car` section contains the `quality_attribute: car_quality` and the `base_normalizer: facenorm/cropbbox.v2.gpu.fnk` or the `base_normalizer: facenorm/cropbbox.v2.cpu.fnk`, depending on your acceleration type:

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
  ...
```

4. Specify the normalizers required for the extractors. If you need license plate recognition, specify the `carlicplate` normalizer. For other extractors, specify the `cropbbox` normalizer.

Normalizer	Normalizer model	Used for extractors
carlicplate	`carnorm/briacon.v0.gpu.fnk` `carnorm/briacon.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`, `car_special_types11`, `car_categories`, `car_orientation`, `car_weight_types7`

GPU

```
normalizers:
  ...

  models:
    ...
    carlicplate:
      model: carnorm/briacon.v0.gpu.fnk
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
    ...
```

CPU

```
normalizers:
  ...

  models:
    ...
    carlicplate:
      model: carnorm/briacon.v0.cpu.fnk
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
    ...
```

5. Specify the extraction models in the `extractors -> models` section, subject to the extractors you want to enable:

GPU

```
extractors:
   ...
   models:
    car_categories: carattr/carattr.categories.v0.gpu.fnk
    car_color: ''
    car_container_number: ''
    car_description: carattr/description.v0.gpu.fnk
    car_emben: carrec/alonso.gpu.fnk
    car_license_plate: carattr/carattr.license_plate.v7.gpu.fnk
    car_license_plate_quality: carattr/carattr.license_plate_quality.v1.gpu.fnk
    car_license_plate_visibility: ''
    car_make: ''
    car_orientation: carattr/carattr.orientation.v0.gpu.fnk
    car_quality: carattr/carattr.quality.v0.gpu.fnk
    car_special_types: ''
    car_special_types11: carattr/carattr.special_types11.v1.gpu.fnk
    car_trash: ''
    car_weight_types7: carattr/carattr.weight_types7.v0.gpu.fnk
```

CPU

```
extractors:
   ...
   models:
    car_categories: carattr/carattr.categories.v0.cpu.fnk
    car_color: ''
    car_container_number: ''
    car_description: carattr/description.v0.cpu.fnk
    car_emben: carrec/alonso.cpu.fnk
    car_license_plate: carattr/carattr.license_plate.v7.cpu.fnk
    car_license_plate_quality: carattr/carattr.license_plate_quality.v1.cpu.fnk
    car_license_plate_visibility: ''
    car_make: ''
    car_orientation: carattr/carattr.orientation.v0.cpu.fnk
    car_quality: carattr/carattr.quality.v0.cpu.fnk
    car_special_types: ''
    car_special_types11: carattr/carattr.special_types11.v1.cpu.fnk
    car_trash: ''
    car_weight_types7: carattr/carattr.weight_types7.v0.cpu.fnk
```

The following extractors are available:

Extractor	Configure as follows
vehicle feature vector	`car_emben: carrec/alonso.cpu.fnk`
	`car_emben: carrec/alonso.gpu.fnk`
license plate number	`car_license_plate: carattr/carattr.license_plate.v7.cpu.fnk car_license_plate_quality: carattr/carattr.license_plate_quality.v1.cpu.fnk`
	`car_license_plate: carattr/carattr.license_plate.v7.gpu.fnk car_license_plate_quality: carattr/carattr.license_plate_quality.v1.gpu.fnk`
set of attributes: make / color / model / body style	`car_description: carattr/description.v0.cpu.fnk`
	`car_description: carattr/description.v0.gpu.fnk`
vehicle image quality	`car_quality: carattr/carattr.quality.v0.cpu.fnk`
	`car_quality: carattr/carattr.quality.v0.gpu.fnk`
special vehicle	`car_special_types: carattr/carattr.special_types11.v1.cpu.fnk`
	`car_special_types: carattr/carattr.special_types11.v1.gpu.fnk`
vehicle category	`car_categories: carattr/carattr.categories.v0.cpu.fnk`
	`car_categories: carattr/carattr.categories.v0.gpu.fnk`
vehicle weight and body size	`car_weight_types7: carattr/carattr.weight_types7.v0.cpu.fnk`
	`car_weight_types7: carattr/carattr.weight_types7.v0.gpu.fnk`
vehicle orientation	`car_orientation: carattr/carattr.orientation.v0.cpu.fnk`
	`car_orientation: carattr/carattr.orientation.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_categories: ''
    car_color: ''
    car_container_number: ''
    car_description: ''
    car_emben: ''
    car_license_plate: ''
    car_license_plate_quality: ''
    car_license_plate_visibility: ''
    car_make: ''
    car_orientation: ''
    car_quality: ''
    car_special_types: ''
    car_special_types11: ''
    car_trash: ''
    car_weight_types7: ''
```

6. Restart the `findface-multi-findface-extraction-api-1` container.

```
sudo docker container restart findface-multi-findface-extraction-api-1
```

2. Modify the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file.

    1. In the `models` section, specify the vehicle detector, normalizer, and extractor models as shown in the example below:

GPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.
↪yaml

models:
  ...
  detectors:
    ...
    car:
      fnk_path: /usr/share/findface-data/models/detector/car.jasmine_fast.005.
↪gpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    car_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
    car_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
    ...
```

(continues on next page)

(continued from previous page)

```
  extractors:
    ...
    car_quality:
      fnk_path: /usr/share/findface-data/models/carattr/carattr.quality.v0.gpu.
↪fnk
      normalizer: car_norm_quality
```

CPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.
↪yaml


models:
  ...
  detectors:
    ...
    car:
      fnk_path: /usr/share/findface-data/models/detector/car.jasmine_fast.005.
↪cpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    car_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    car_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    ...
  extractors:
    ...
    car_quality:
      fnk_path: /usr/share/findface-data/models/carattr/carattr.quality.v0.cpu.
↪fnk
      normalizer: car_norm_quality
```

2. Make sure that the `objects -> car` section is included:

```
objects:
  ...
  car:
    normalizer: car_norm
    quality: car_quality
    track_features: ''
```

3. Restart the `findface-multi-findface-video-worker-1` container.

```
sudo docker container restart findface-multi-findface-video-worker-1
```

3. Open the `/opt/findface-multi/configs/findface-video-manager/findface-video-manager.yaml` configuration file and make sure it contains the `car` section in `detectors` that looks similar to the

example below. Note that the `filter_min_quality` parameter is set to `0.65` by default. You can increase it to get more accurate results from cameras and other video sources. The best practice is to set this value to `0.73`.

```
sudo vi /opt/findface-multi/configs/findface-video-manager/findface-video-manager.
↪yaml

detectors:
  ...
  car:
    filter_min_quality: 0.65
    filter_min_size: 100
    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
    tracker_type: simple_iou
    track_deep_sort_matching_threshold: 0.65
    track_deep_sort_filter_unconfirmed_tracks: true
    track_object_is_principal: false
    track_history_active_track_miss_interval: 0
```

Note: Configuration of the `filter_min_quality` parameter affects the results that you get from video sources only and does not affect vehicle counting results and quality of the vehicle images that are used in the records. To get more accurate counting results or vehicle records with better quality images, configure the `MINIMUM_CAR_QUALITY` parameter individually in the `findface-multi-legacy.py` configuration file. This setting will be described below.

---

4. Enable recognition of vehicles and their attributes in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. Do the following:

   1. In the `FFSECURITY` section, set `'ENABLE_CARS': True`.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.
↪py

FFSECURITY = {
    ...
    # optional objects to detect
    'ENABLE_CARS': True,
```

(continues on next page)

(continued from previous page)

```
    ...
```

2. In the same section, you can configure the `MINIMUM_CAR_QUALITY` parameter. By default, it's set to `0.65`, which is the recommended minimum value for vehicle image quality, sufficient to make a record or use an object in a counter.

```
FFSECURITY = {
    ...
    # minimum car quality sufficient to add it to a card and use object in␣
↪counter
    # p.s. set this parameter equals to `0.73` for more accurate results
    'MINIMUM_CAR_QUALITY': 0.65,  # model: [carattr.quality.v0]
    ...
```

   If you need more accurate counting results or vehicle records with better quality images, you can increase the `MINIMUM_CAR_QUALITY` threshold. The best practice is to set this value to `0.73`.

3. In the `FFSECURITY` section, specify the attributes you want to display for vehicle recognition events.

```
# available features are: category, description, license_plate, orientation,␣
↪special_vehicle_type, weight_type
'CAR_EVENTS_FEATURES': ['license_plate', 'category', 'special_vehicle_type',
↪'description', 'weight_type', 'orientation'],
```

4. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

5. In the web interface, navigate to *Video Source*. Select a camera in the *Cameras* tab (or uploaded file the *Uploads* tab, or external detector in the corresponding tab). Navigate to the *General* tab. Check *Vehicles* in the Detectors section.

2.5.5 Interpret Vehicle Recognition Results

Vehicle recognition results can be accessed from the *Episodes & Events* tab of the web interface.

In certain scenarios vehicle recognition results lack description attributes and their values, or it is displayed that the attribute value is unknown.

- The attribute value is unknown. If you see in the recognition results that the attribute value is unknown, it means that the attribute recognition confidence is below the threshold, specified for the attribute in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. The system will not display the recognized value in this case. Instead, the recognized value will be lost.

  You can configure the threshold for the vehicle attribute in the `findface-multi-legacy.py` configuration file.

Important: The above does not apply to the license plate recognition. License plate recognition threshold is set to the minimum value by default. If you see in the recognition results that the license plate number is unknown, it means that it's hardly or not at all visible on the object. License plate country, region, and color recognition depends on the license plate number recognition results. The values of these attributes will remain unknown if the license plate number is unknown.

Note: License plate region and color are predicted for the United Arab Emirates (UAE) only. The values of these attributes will be marked as unknown for other countries.

- Lack of description attributes and their values. If you do not see description attributes (i.e., vehicle make, body, model, color) and their values in the recognition results, it happens for the following reasons:
  - Recognition of a vehicle category, as well as vehicle weight and body size recognition, is still under development. FindFace Multi accurately recognizes cars of category B and their attributes. When it comes to vehicle weight and body size, recognition of cars and trucks under 3.5 tons and their attributes is supported. However, the system can provide false positive results on other vehicle categories and weight classes and their description attributes. That's why description attributes and their values are displayed for a car category only, and, when it comes to vehicle weight classes, for cars and trucks under 3.5 tons. For other vehicle categories and weight classes, the description attributes and their recognition values are hidden. The recognition confidence of the description attribute does not matter in this case - it can be below or above the threshold.
  - The same thing happens if a vehicle category is unknown: description attributes along with recognition values will not be displayed. The recognition confidence of the description attribute is not a determining factor in this case.

  Subject to your needs, this behavior can be changed in the `CAR_EVENTS_FEATURES_TO_NULL` section of the `findface-multi-legacy.py` configuration file.

Read further to learn how to adjust the threshold for a vehicle attribute or to enable the display of description attributes for some or all vehicle categories and unknown vehicles.

In this section:

Configure the threshold for a vehicle attribute

The default threshold for each vehicle attribute is set to the optimum value. You can change it if necessary.

Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. In the `FFSECURITY` section, modify threshold values for selected attributes:

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

FFSECURITY = {
    ...

    'CAR_DESCRIPTION_THRESHOLD': {
        'make': 0.5,
        'model': 0.5,
        'body': 0.5,
        'color': 0.5,
    },  # model: [description.v0]
    'SPECIAL_VEHICLE_TYPE_THRESHOLD': 0.5,  # model: [special_types11.v1]
```

(continues on next page)

(continued from previous page)

```
    'CAR_CATEGORY_THRESHOLD': 0.6,  # model: [categories.v0]
    'CAR_WEIGHT_TYPE_THRESHOLD': 0.5,  # model: [weight-types7.v0]
    'CAR_ORIENTATION_THRESHOLD': 0.8,  # model  [orientation.v0]
    ...
```

Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

Configure the display of description attributes in the recognition results

Based on your needs, you can enable the display of description attributes for unsupported vehicle categories or unknown vehicles. By default, the display of description attributes is enabled only for cars of the category B and for vehicles of the weight categories B_light (cars under 3.5 tons) and B_heavy (trucks under 3.5 tons). Use information from the tables to match a vehicle and its category or weight class.

Table 1: Vehicle categories

Category	Vehicle
A	motorcycle (including moped), scooter, quad bike
B	car
BE	car with a trailer
C	truck
CE	truck with a trailer
D	bus
DE	articulated bus
other	a vehicle that doesn't fall within any of the above-listed types

Table 2: Vehicle weight classes

Weight class	Vehicle
B_light	car under 3.5 tons
B_heavy	truck under 3.5 tons
BE	car with a trailer
C_light	truck under 12 tons
C_heavy	truck over 12 tons
D_light	single-deck bus
D_long	articulated bus
other	a vehicle that doesn't fall within any of the above-listed types

Warning: The default setting is set to the optimum value. Please contact our technical experts (support@ntechlab.com) before making any adjustment.

To enable the display of description attributes for unsupported vehicle categories or unknown vehicles, open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. In the `FFSECURITY` section, find the `CAR_EVENTS_FEATURES_TO_NULL` setting:

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

(continues on next page)

(continued from previous page)

```
FFSECURITY = {
    ...

    'CAR_EVENTS_FEATURES_TO_NULL': {
        'category': {
            # features, that will be nulled for all categories
            'default': ['make', 'body', 'model', 'color'],
            'unknown': ['make', 'body', 'model', 'color'],
            # categories with other behavior
            'B': [],
        },
        'weight_type': {
            # features, that will be nulled for all weight types
            'default': ['make', 'body', 'model', 'color'],
            # weight types with other behavior
            'B_light': [],
            'B_heavy': [],
        },
    },
    ...
```

To enable the display of description attributes, remove them from the selected parameter. E.g., if you want to display description attributes for all vehicle categories, except for unknown vehicles, remove them from the `default` parameter. Note that unknown vehicles are excluded from the `default` parameter and are configured through the `unknown` parameter.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

FFSECURITY = {
    ...

    'CAR_EVENTS_FEATURES_TO_NULL': {
        'category': {
            # features, that will be nulled for all categories
            'default': [],
            'unknown': ['make', 'body', 'model', 'color'],
            # categories with other behavior
            'B': [],
        },
        'weight_type': {
            # features, that will be nulled for all weight types
            'default': [],
            # weight types with other behavior
            'B_light': [],
            'B_heavy': [],
        },
    },
    ...
```

To enable the display of description attributes for a specific vehicle category or for a vehicle of a certain weight class, add its name to the exception within the `categories with other behavior` or the `weight types with other behavior` sections. Use information from the tables *Vehicle categories* and *Vehicle weight classes* to match a vehicle and its category or weight class. E.g., if you want to add buses (category D) and single-deck buses (weight class

D_light) to the exception, do the following:

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

FFSECURITY = {
    ...

    'CAR_EVENTS_FEATURES_TO_NULL': {
        'category': {
            # features, that will be nulled for all categories
            'default': ['make', 'body', 'model', 'color'],
            'unknown': ['make', 'body', 'model', 'color'],
            # categories with other behavior
            'B': [],
            'D': [],
        },
        'weight_type': {
            # features, that will be nulled for all weight types
            'default': ['make', 'body', 'model', 'color'],
            # weight types with other behavior
            'B_light': [],
            'B_heavy': [],
            'D_light': [],
        },
    },
    ...
```

Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

2.5.6 Special Vehicle Recognition

FindFace Multi can recognize a special vehicle and its type. You can set up special vehicle recognition during the *installation* or enable it later, using the instruction in the *Enable Vehicle and Vehicle Attribute Recognition* section.

Note: Special vehicle recognition works for selected countries only and may not work for your country. For more information please contact your manager or our support team (support@ntechlab.com).

Recognition of the following vehicle types is supported:

- Taxi: taxi, including cargo taxi. In some cases, a cargo taxi can be defined as “other special”.
- Public road transport: route transport, i.e. buses, share taxis, trolleybuses.
- Carsharing: supported carsharing services are Citydrive, Yandex.Drive, Delimobil, BelkaCar. Carsharing services that look similar to those listed above, may also be recognized.
- Ambulance: ambulance, including mobile intensive care units (both white and yellow vehicles).
- Police: police, including traffic police cars.
- Rescue service: fire service and EMERCOM vehicles.
- Gas rescue and emergency services: overall yellow or white cars, including those with contrasting red doors, typical red stripes, and the `04` / `104` sign on board.

- Military: military vehicles, including the National Guard vehicles.
- Road service: road service vehicles, including municipal vehicles, construction equipment, and vehicles of the Center for the Organization of Road Traffic.
- Other special: a special vehicle that does not fall within the above-listed types.
- Not special: not a special vehicle.

2.5.7 Enable Body and Body Attribute Recognition

FindFace Multi allows you to recognize individual human bodies and body attributes.

The body attributes are as follows:

- gender:
  - male;
  - female;
- age (by group):
  - 0-16 years;
  - 17-35 years;
  - 36-50 years;
  - 50+ years;
- clothing type:
  - generalized category of upper body wear: long sleeves, short sleeves, no sleeve;
  - specific type of upper body wear: jacket, coat, sleeveless vest, sweatshirt, T-shirt, shirt, dress;
  - type of lower body wear: pants, skirt, shorts, nondescript;
  - type of headgear: hat/cap, hood/headscarf, none;
- clothing color (top/bottom);
- presence of personal protective equipment (PPE):
  - PPE item: vest, helmet;
  - PPE color;
  - PPE recognition score;
- whether a person has a bag:
  - on the back;
  - in hand(s).

Recognition of human bodies and their attributes can be configured at the *installation* level. This section describes how to enable body and body attribute recognition in case this step has been skipped during installation.

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 `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` configuration file.

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 `findface-extraction-api.yaml` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-
↪api.yaml
```

2. Specify the body detector model in the `detectors -> models` section by pasting the following code:

GPU

```
detectors:

  ...
  models:
    ...
    body_gustav:
      aliases:
      - body
      - edie
      - shiloette
      - glen
      model: detector/body.gustav_normal.015.gpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 256x256
        - 384x384
        - 512x512
        - 768x768
        - 1024x1024
        - 1536x1536
        - 2048x2048
  ...
```

CPU

```
detectors:

  ...
  models:
    ...
    body_gustav:
      aliases:
      - body
      - edie
      - shiloette
```

(continues on next page)

(continued from previous page)

```
      - glen
      model: detector/body.gustav_normal.015.cpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 256x256
        - 384x384
        - 512x512
        - 768x768
        - 1024x1024
        - 1536x1536
        - 2048x2048
  ...
```

3. Make sure that the `objects -> body` section contains the `quality_attribute: body_quality` and the `base_normalizer: facenorm/cropbbox.v2.gpu.fnk` or the `base_normalizer: facenorm/cropbbox.v2.cpu.fnk`, depending on your acceleration type:

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
  ...
```

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.

GPU

```
normalizers:
  ...

  models:
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
    ...
```

CPU

```
normalizers:
  ...

  models:
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
    ...
```

5. Specify the extraction models in the `extractors -> models` section, subject to the extractors you want to enable:

GPU

```
extractors:
   ...
   models:
    body_action_base6: ''
    body_action_car: ''
    body_action_fights: ''
    body_age_gender: pedattr/pedattr.age_gender.v0.gpu.fnk
    body_bags: pedattr/pedattr.bags.v0.gpu.fnk
    body_clothes: pedattr/pedattr.clothes_type.v0.gpu.fnk
    body_clothes34671: ''
    body_color: pedattr/pedattr.color.v1.gpu.fnk
    body_emben: pedrec/pedrec.clio.gpu.fnk
    body_fall: ''
    body_handface: ''
    body_protective_equipment: pedattr/pedattr.protective.v1.gpu.fnk
    body_quality: pedattr/pedattr.quality.v0.gpu.fnk
```

CPU

```
extractors:
   ...
   models:
    body_action_base6: ''
    body_action_car: ''
    body_action_fights: ''
    body_age_gender: pedattr/pedattr.age_gender.v0.cpu.fnk
    body_bags: pedattr/pedattr.bags.v0.cpu.fnk
    body_clothes: pedattr/pedattr.clothes_type.v0.cpu.fnk
    body_clothes34671: ''
    body_color: pedattr/pedattr.color.v1.cpu.fnk
    body_emben: pedrec/pedrec.clio.cpu.fnk
    body_fall: ''
    body_handface: ''
    body_protective_equipment: pedattr/pedattr.protective.v1.cpu.fnk
    body_quality: pedattr/pedattr.quality.v0.cpu.fnk
```

The following extractors are available:

Extractor	Configure as follows
age and gender	`body_age_gender: pedattr/pedattr.age_gender.v0.gpu.fnk`
	`body_age_gender: pedattr/pedattr.age_gender.v0.cpu.fnk`
presence of bag	`body_bags: pedattr/pedattr.bags.v0.gpu.fnk`
	`body_bags: pedattr/pedattr.bags.v0.cpu.fnk`
clothing type	`body_clothes: pedattr/pedattr.clothes_type.v0.gpu.fnk`
	`body_clothes: pedattr/pedattr.clothes_type.v0.cpu.fnk`
clothing color	`body_color: pedattr/pedattr.color.v1.gpu.fnk`
	`body_color: pedattr/pedattr.color.v1.cpu.fnk`
individual body feature vector	`body_emben: pedrec/pedrec.clio.gpu.fnk`
	`body_emben: pedrec/pedrec.clio.cpu.fnk`
presence of protective equipment	`body_protective_equipment: pedattr/pedattr.protective.v1.gpu.fnk`
	`body_protective_equipment: pedattr/pedattr.protective.v1.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.

Important: For body recognition to work properly, the `body_emben` and the `body_quality` extractors must be enabled.

GPU

```
extractors:
   ...
   models:
    body_action_base6: ''
    body_action_car: ''
    body_action_fights: ''
    body_age_gender: ''
    body_bags: ''
    body_clothes: ''
    body_clothes34671: ''
    body_color: ''
    body_emben: pedrec/pedrec.clio.gpu.fnk
    body_fall: ''
    body_handface: ''
    body_protective_equipment: ''
    body_quality: pedattr/pedattr.quality.v0.gpu.fnk
```

CPU

```
extractors:
   ...
   models:
    body_action_base6: ''
    body_action_car: ''
    body_action_fights: ''
    body_age_gender: ''
    body_bags: ''
    body_clothes: ''
    body_clothes34671: ''
    body_color: ''
    body_emben: pedrec/pedrec.clio.cpu.fnk
    body_fall: ''
    body_handface: ''
    body_protective_equipment: ''
    body_quality: pedattr/pedattr.quality.v0.cpu.fnk
```

6. Restart the `findface-multi-findface-extraction-api-1` container.

```
sudo docker container restart findface-multi-findface-extraction-api-1
```

2. Modify the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file.

    1. In the `models` section, specify the body neural network models by analogy with the example below:

GPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.
↪yaml

models:
  ...
  detectors:
    ...
    body:
      fnk_path: /usr/share/findface-data/models/detector/body.jasmine_fast.018.
↪gpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    body_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
    body_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
    ...
  extractors:
    ...
    body_quality:
      fnk_path: /usr/share/findface-data/models/pedattr/pedattr.quality.v0.gpu.
```

(continues on next page)

(continued from previous page)

```
↪fnk
      normalizer: body_norm_quality
```

CPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.
↪yaml

models:
  ...
  detectors:
    ...
    body:
      fnk_path: /usr/share/findface-data/models/detector/body.jasmine_fast.018.
↪cpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    body_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    body_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    ...
  extractors:
    ...
    body_quality:
      fnk_path: /usr/share/findface-data/models/pedattr/pedattr.quality.v0.cpu.
↪fnk
      normalizer: body_norm_quality
```

2. Make sure that the `objects -> body` section is included:

```
objects:
  ...
  body:
    normalizer: body_norm
    quality: body_quality
    track_features: ''
```

3. Restart the `findface-multi-findface-video-worker-1` container.

```
sudo docker container restart findface-multi-findface-video-worker-1
```

3. Open the `/opt/findface-multi/configs/findface-video-manager/findface-video-manager.yaml` configuration file and make sure it contains the `body` section in `detectors` that looks similar to the example below.

```
sudo vi /opt/findface-multi/configs/findface-video-manager/findface-video-manager.
↪yaml
```

(continues on next page)

(continued from previous page)

```
detectors:
  ...
  body:
    filter_min_quality: 0.6
    filter_min_size: 70
    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
    tracker_type: simple_iou
    track_deep_sort_matching_threshold: 0.65
    track_deep_sort_filter_unconfirmed_tracks: true
    track_object_is_principal: false
    track_history_active_track_miss_interval: 0
```

4. Enable recognition of bodies and body attributes in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. Do the following:

   1. In the `FFSECURITY` section, set `'ENABLE_BODIES': True`.

      ```
      sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.
      ↪py

      FFSECURITY = {
          ...

          # optional objects to detect
          'ENABLE_BODIES': True,
          ...
      ```

   2. To improve quality of body recognition, we recommend that you enable additional attribute analysis. In this case, the system compares not only the feature vectors of two bodies but also their attributes. A conclusion about the bodies' match is only made if both the feature vectors and attributes of the bodies coincide.

      You can use the following attributes for additional analysis:

      - `bottom_color`: color of lower body wear;
      - `top_color`: color of upper body wear;
      - `headwear`: type and absence/presence of headgear;

- `detailed_upper_clothes`: specific type of upper body wear, e.g., jacket;
- `upper_clothes`: generalized category of upper body wear: long sleeves, short sleeves, no sleeve;
- `lower_clothes`: type of lower body wear, e.g., pants;
- `bag_hand`: whether a person has a bag in hand(s);
- `bag_back`: whether a person has a bag on the back;
- `helmet_type`: helmet type by color, visibility, absence/presence;
- `vest_type`: vest type by color, visibility, absence/presence;
- `age_group`: belonging to any of four age groups: 0-16, 17-35, 36-50, 50+ years;
- `gender`: male or female.

To enable additional attribute analysis, set `True` in the `FFSECURITY -> EXTRA_BODY_MATCHING` section for the attributes that you want to compare. Set `min_confidence` value between 0 and 1.

```
FFSECURITY = {
    # use additional features for extra confidence when matching body by emben
    'EXTRA_BODY_MATCHING': {
        'bottom_color': {'enabled': False, 'min_confidence': 0},
        'top_color': {'enabled': False, 'min_confidence': 0},
        'headwear': {'enabled': False, 'min_confidence': 0},
        'detailed_upper_clothes': {'enabled': False, 'min_confidence': 0},
        'upper_clothes': {'enabled': False, 'min_confidence': 0},
        'lower_clothes': {'enabled': False, 'min_confidence': 0},
        'bag_hand': {'enabled': False, 'min_confidence': 0},
        'bag_back': {'enabled': False, 'min_confidence': 0},
        'bag_ground': {'enabled': False, 'min_confidence': 0},
        'helmet_type': {'enabled': False, 'min_confidence': 0},
        'vest_type': {'enabled': False, 'min_confidence': 0},
        'age_group': {'enabled': False, 'min_confidence': 0},
        'gender': {'enabled': False, 'min_confidence': 0},
    },
```

Note: The `bag_ground` attribute is not available for additional analysis yet.

Note: Contact our technical experts (support@ntechlab.com) for advice on `min_confidence` optimum value.

If you decide that you do not need additional attribute analysis, then skip this configuration and proceed to the next step.

3. In the `FFSECURITY` section, specify the body attributes that you want to display for the body recognition events.

```
# available features: age_gender, bags, clothes, color, protective_equipment
'BODY_EVENTS_FEATURES': ['protective_equipment', 'age_gender', 'bags', 'color',
↪'clothes'],
```

4. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

5. In the web interface, navigate to *Video Source*. Select a camera in the *Cameras* tab (or uploaded file the *Uploads* tab, or external detector in the corresponding tab). Navigate to the *General* tab. Check *Bodies* in the Detectors section.

2.5.8 Enable Face Liveness Detection

The FindFace Multi face liveness detector tells apart live faces from face representations, such as face images, videos, or masks. The liveness detector estimates 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.

The face liveness detector can be automatically enabled and configured during the *installation*. If you skipped this step, you can manually enable it later, following the instructions below.

Note: The face liveness detector functions on both GPU- and CPU-acceleration. However, it is much slower on CPU.

In this section:

Enable Face Liveness Detector

To enable the face liveness detector, do the following:

1. Open the `/opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml` configuration file. In the `liveness` section, specify the neural network models as shown in the example:

GPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml

liveness:
  fnk: /usr/share/findface-data/models/faceattr/liveness.pacs.v2.gpu.fnk
  norm: /usr/share/findface-data/models/facenorm/facenorm.multicrop_full_crop2x_
↪size400.gpu.fnk
...
```

CPU

```
sudo vi /opt/findface-multi/configs/findface-video-worker/findface-video-worker.yaml

liveness:
  fnk: /usr/share/findface-data/models/faceattr/liveness.pacs.v2.cpu.fnk
  norm: /usr/share/findface-data/models/facenorm/facenorm.multicrop_full_crop2x_
↪size400.cpu.fnk
...
```

2. Restart the `findface-multi-findface-video-worker-1` container.

```
sudo docker container restart findface-multi-findface-video-worker-1
```

Configure Liveness Threshold

If necessary, you can adjust the liveness threshold in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. The liveness detector will estimate 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 that you seek advice from our experts at support@ntechlab.com.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

FFSECURITY = {
    ...
    # feature specific confidence thresholds
    'LIVENESS_THRESHOLD': 0.674,  # model: [liveness.pacs.v2]
    ...
```

Restart the `findface-multi-findface-multi-legacy-1` container if you have configured the liveness threshold.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

2.5.9 Migrate Feature Vectors to a Different Neural Network Model

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

Important: In case, you are doing migration as part of the system update to a newer version, complete the *update* first. Only after that can you proceed to the migration.

This section is about how to migrate object feature vectors to another 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/ffmulti_dump`.

```
sudo docker exec -it findface-multi-findface-sf-api-1 bash -c "mkdir ffmulti_dump;␣
↪cd ffmulti_dump && /storage-api-dump -config /etc/findface-sf-api.ini"
sudo docker cp findface-multi-findface-sf-api-1:/ffmulti_dump /etc
```

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

   1. Navigate to the `/opt/findface-multi/configs/findface-tarantool-server/` directory and find out the number of shards by counting the number of configuration files `shard-00*.lua`.

      Note: There are eight shards in the example below.

```
cd /opt/findface-multi/configs/findface-tarantool-server

ls -l

shard-001.lua
shard-002.lua
shard-003.lua
shard-004.lua
shard-005.lua
shard-006.lua
shard-007.lua
shard-008.lua
```

   2. In the `/opt/findface-multi/configs/findface-tarantool-server/` directory, create the same number of new shards by copying the configuration files `shard-00*.lua`.

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

```
for i in {1..8}; do sudo cp shard-00$i.lua shard-01$i.lua; done
```

   3. Modify the following lines in each new shard's configuration file, subject to its name:

Old value	New value
listen = '127.0.0.1:32001'	Listen = '127.0.0.1:32011'
FindFace.start(«127.0.0.1», 8101, {	FindFace.start(«127.0.0.1», 8111, {

      You can do it by executing the following command:

```
for i in {1..8}; do sudo sed -i "s/    listen = '127.0.0.1:3200$i',/    listen␣
↪= '127.0.0.1:3201$i',/" shard-01$i.lua && sudo sed -i "s/FindFace.start(\"127.
↪0.0.1\", 810$i, {/FindFace.start(\"127.0.0.1\", 811$i, {/" shard-01$i.lua;␣
↪done
```

   4. Create directories that will host files of the new shards.

```
cd /opt/findface-multi/data/findface-tarantool-server

sudo mkdir -p shard-01{1..8}/{index,snapshots,xlogs}
```

3. Open the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` configuration file and replace the extraction models with the new ones in the `body_emben`, `car_emben`, and `face_emben` parameters, depending on the object types you want to migrate.

```
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.
↪yaml

extractors:
  ...
  models:
    ...
     body_emben: pedrec/<new_model_body>.cpu<gpu>.fnk
     ...
     car_emben: carrec/<new_model_car>.cpu<gpu>.fnk
     ...
     face_emben: face/<new_model_face>.cpu<gpu>.fnk
     ...
```

Restart the `findface-multi-findface-extraction-api-1` container.

```
cd /opt/findface-multi/
sudo docker-compose restart findface-extraction-api
```

4. In the `docker-compose.yaml` file, create a new service for each new shard. To do that, copy the existing service and replace the name of the shard.

```
sudo vi docker-compose.yaml

services:
  ...
  findface-tarantool-server-shard-**011**:
    depends_on: [findface-ntls]
    image: docker.int.ntl/ntech/universe/tntapi:ffserver-8.221216
    network_mode: service:pause
    restart: always
    volumes: ['./configs/findface-tarantool-server/shard-**011**.lua:/etc/tarantool/
↪instances.enabled/FindFace.lua:ro',
      './data/findface-tarantool-server/shard-**011**:/var/lib/tarantool/FindFace',
↪'./configs/findface-tarantool-server/tnt-schema.lua:/tnt_schema.lua:ro']
  ...
```

5. Start the new shards by upping the containers.

```
sudo docker-compose up -d
```

6. Create a configuration file with migration settings `migration.yaml` based on the example below.

```
sudo vi migration.yaml

extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
```

(continues on next page)

(continued from previous page)

```
    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: ''
    - master: http://127.0.0.1:8105/v2/
      slave: ''
    - master: http://127.0.0.1:8106/v2/
      slave: ''
    - master: http://127.0.0.1:8107/v2/
      slave: ''
    - master: http://127.0.0.1:8108/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: ''
    - master: http://127.0.0.1:8115/v2/
      slave: ''
    - master: http://127.0.0.1:8116/v2/
      slave: ''
    - master: http://127.0.0.1:8117/v2/
      slave: ''
    - master: http://127.0.0.1:8118/v2/
      slave: ''
workers_num: 3
faces_limit: 100
```

(continues on next page)

(continued from previous page)

```
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/
      slave: ''
    - master: http://127.0.0.1:8105/v2/
      slave: ''
    - master: http://127.0.0.1:8106/v2/
      slave: ''
    - master: http://127.0.0.1:8107/v2/
      slave: ''
    - master: http://127.0.0.1:8108/v2/
      slave: ''
    ...
```

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: ''
```

(continues on next page)

(continued from previous page)

```
    - master: http://127.0.0.1:8115/v2/
      slave: ''
    - master: http://127.0.0.1:8116/v2/
      slave: ''
    - master: http://127.0.0.1:8117/v2/
      slave: ''
    - master: http://127.0.0.1:8118/v2/
      slave: ''
    ...
```

7. Copy the `migration.yaml` file into the `findface-multi-findface-sf-api-1` container. Launch the `sf-api-migrate` utility with the `-config` option and provide the `migration.yaml` configuration file.

```
sudo docker cp migration.yaml findface-multi-findface-sf-api-1:/
sudo docker exec findface-multi-findface-sf-api-1 ./sf-api-migrate -config␣
↪migration.yaml
```

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

8. After the migration is complete, remove services for the old shards from the `docker-compose.yaml` file and stop their containers.

```
sudo docker-compose up -d --remove-orphans
```

9. Open the `/opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml` configuration file and adjust the shards ports, subject to the new shards settings. Restart the `findface-multi-findface-sf-api-1` container.

```
sudo vi /opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml

storage-api:
  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: ''
  - master: http://127.0.0.1:8115/v2/
    slave: ''
  - master: http://127.0.0.1:8116/v2/
    slave: ''
  - master: http://127.0.0.1:8117/v2/
    slave: ''
  - master: http://127.0.0.1:8118/v2/
    slave: ''

sudo docker-compose restart findface-sf-api
```

10. Migrate clusters as well if *this functionality* is enabled in your system. To do so, execute the following command:

    Note: List the object types to migrate as the command options: `--face`, `--body`, `--car`.

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/
↪bin/python3 /tigre_prototype/manage.py migrate_clusters --face --body --car --use-
↪best-event --use-thumbnail --force-clustering
```

    As a result, the system will regenerate feature vectors for the existing cluster events and automatically launch the scheduled clustering to rebuild clusters.

2.5.10 Deactivate `findface-liveness-api` installed with FindFace Multi

If you don't utilize the `findface-liveness-api` service installed with FindFace Multi and the face-based authentication does not apply to your system, we recommend deactivating `findface-liveness-api`.

Do the following:

1. Stop the `findface-multi-findface-liveness-api-1` container and disable its autostart by executing:

```
sudo docker container stop findface-multi-findface-liveness-api-1
sudo docker update --restart=no findface-multi-findface-liveness-api-1
```

2. Open the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.
↪yaml
```

3. Disable the neural network models used by the `findface-liveness-api` service. To do so, pass the empty value `''` to the `multicrop_full_crop2x` and `face_liveness` parameters.

    Note: Do not remove the parameter itself. Otherwise, the system will be searching for the default model.

```
normalizers:
  models:
    ...
    multicrop_full_crop2x: ''
    ...

extractors:
  models:
    ...
    face_liveness: ''
    ...
```

4. Restart the `findface-multi-findface-extraction-api-1` container.

```
sudo docker container restart findface-multi-findface-extraction-api-1
```

2.6 Maintenance and Troubleshooting

2.6.1 Update to FindFace Multi 2.0

Tip: If you use our product FindFace Security deployed on Ubuntu 18.04, upgrade it to FindFace Multi 1.2 and then update it to FindFace Multi 2.0.

The Incidents and VNS plugins, operable in FindFace CIBR, are not supported in FindFace Multi 2.0. That's why there's no need to enable `ffsecurity_incidents`, `ffsecurity_vns`, and `ffsecurity_puppeteer` plugins in the `findface-multi-legacy.py` configuration file after product update.

Integration with Axxon Next is included into *external VMS* integration in FindFace Multi 2.0 and configured through VMS integration plugin.

If you created person and car cards with custom fields in FindFace Multi 1.2 and want them to be displayed in the FindFace Multi 2.0 UI, then it is necessary to copy the `CUSTOM_FIELDS` section from the old configuration file in step #2 and paste it to the new configuration file in step #8.

To update FindFace Multi from 1.2 to 2.0, do the following:

1. Create a backup copy of the old schema of the Tarantool-based feature vector database:

   ```
   sudo cp /etc/findface-security/tnt_schema.lua /etc/findface-security/old_tnt_schema.
   ↪lua
   ```

   One of the most significant differences between FindFace Multi 2.0 and earlier versions of the product is related to the structure of the Tarantool biometric database (so called meta-schema). The new structure is created as a set of spaces, while in previous versions of the product there was only one space by default in the structure of the Tarantool-based database.

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

   ```
   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',
   ...
   # -- Custom model fields --
   # Edit CUSTOM_FIELDS -> `human_card` section to customize human card fields.
   # Edit CUSTOM_FIELDS -> `face_object` section to customize face object fields.
   # Below is an example with every field type possible.
   # 'CUSTOM_FIELDS': {
   #     'human_card': {
   #         'items': [
   #             {
   #                 'name': 'personid',
   ```

(continues on next page)

(continued from previous page)

```
#                 'default': '',
#                 'label': 'PersonID',
#                 'display': ['list', 'form'],
#                 'description': 'Sigur person ID',
#                 'editable': False
#             },
#             {
#                 'name': 'firstname',
#                 'default': '',
#                 'label': 'First Name',
#                 'display': ['list', 'form'],
#                 'description': 'Sigur first name',
#                 'editable': False
#             },
#             {
#                 'name': 'lastname',
#                 'default': '',
#                 'label': 'Last Name',
#                 'display': ['list', 'form'],
#                 'description': 'Sigur last name',
#                 'editable': False
#             },
#             {
#                 'name': 'version',
#                 'default': '',
#                 'label': 'Version',
#                 'display': ['list', 'form'],
#                 'description': 'Sigur photo version',
#                 'editable': False
#             }
#         ],
#         'filters': [
#             {
#                 'name': 'personid',
#                 'label': 'Sigur person ID filter',
#                 'field': 'personid'
#             }
#         ]
#     },
#     'face_object': {
#         'items': [
#             {
#                 "field_name": "tag_name_1",
#                 "type": "string",
#                 "default": "change_me"
#             },
#             {
#                 "field_name": "tag_name_2",
#                 "type": "uint",
#                 "default": 123
#             },
#             {
```

(continues on next page)

(continued from previous page)

```
#                 "field_name": "tag_name_3",
#                 "type": "bool",
#                 "default": True
#             },
#         ]
#     }
# },
}
```

3. Stop the `findface-security` service.

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

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

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

5. To avoid port conflicts, stop and disable all services before installing a new version.

Note: There are eight shards in the example below. If it differs with the number of shards in your system, adjust the below command accordingly. E.g., for the system with sixteen shards, replace `tarantool@shard-00{1..8}.service` with `tarantool@shard-0{01..16}.service`. Get the list of active shards with the `ls /etc/tarantool/instances.enabled/` command.

You may stop and disable the services one by one:

```
sudo systemctl stop postgresql.service
sudo systemctl stop postgresql@10-main
sudo systemctl stop findface-*.service
sudo systemctl stop pgbouncer.service
sudo systemctl stop tarantool@shard-00{1..8}.service
sudo systemctl stop nats-server.service
sudo systemctl stop etcd.service
sudo systemctl stop mongod.service
sudo systemctl stop mongodb.service
sudo systemctl stop memcached.service
sudo systemctl stop nginx.service

sudo systemctl disable postgresql.service
sudo systemctl disable postgresql@10-main
sudo systemctl disable pgbouncer.service
sudo systemctl disable findface-extraction-api.service
sudo systemctl disable findface-security.service
sudo systemctl disable findface-security-onvif.service
sudo systemctl disable findface-sf-api.service
sudo systemctl disable findface-ntls.service
sudo systemctl disable findface-video-manager.service
sudo systemctl disable findface-video-worker-cpu.service
sudo systemctl disable findface-video-worker-gpu.service
```

(continues on next page)

(continued from previous page)

```
sudo systemctl disable findface-counter.service
sudo systemctl disable findface-liveness-api.service
sudo systemctl disable findface-video-streamer-cpu.service
sudo systemctl disable findface-video-streamer-gpu.service
sudo systemctl disable findface-video-storage.service
sudo systemctl disable tarantool@shard-00{1..8}.service
sudo systemctl disable nats-server.service
sudo systemctl disable etcd.service
sudo systemctl disable mongod.service
sudo systemctl disable mongodb.service
sudo systemctl disable memcached.service
sudo systemctl disable nginx.service
```

Or you may use the following compact commands instead:

```
sudo systemctl stop postgresql.service postgresql@10-main findface-*.service␣
↪pgbouncer.service tarantool@shard-00{1..8}.service nats-server.service etcd.
↪service mongod.service mongodb.service memcached.service nginx.service

sudo systemctl disable postgresql.service postgresql@10-main pgbouncer.service␣
↪findface-extraction-api.service findface-security.service findface-security-onvif.
↪service findface-sf-api.service findface-ntls.service findface-video-manager.
↪service findface-video-worker-cpu.service findface-video-worker-gpu.service␣
↪findface-counter.service findface-liveness-api.service findface-video-streamer-
↪cpu.service findface-video-streamer-gpu.service findface-video-storage.service␣
↪tarantool@shard-00{1..8}.service nats-server.service etcd.service mongod.service␣
↪mongodb.service memcached.service nginx.service
```

6. *Install* the FindFace Multi 2.0 instance. Don't forget to prepare a server first:

   See:

   - *Ubuntu Server Preparation*
   - *CentOS 7 Server Preparation*
   - *Debian 11 Server Preparation*

7. After FindFace Multi installation, stop all the containers from the `/opt/findface-multi/` directory.

```
cd /opt/findface-multi/
sudo docker-compose stop
```

8. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file and paste the saved on step #2 values for the parameters `EXTERNAL_ADDRESS`, `SECRET_KEY`, `VIDEO_DETECTOR_TOKEN`, `ROUTER_URL`, and `CUSTOM_FIELDS` into it.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
...
# Use pwgen -sncy 50 1|tr "" "." to generate your own unique key
SECRET_KEY = '002231ccb690586f4d33e98322c591bb'
...
SERVICE_EXTERNAL_ADDRESS = 'http://172.20.77.58'
# EXTERNAL_ADDRESS is used to access objects created inside FFSecurity via external␣
↪links.
```

(continues on next page)

(continued from previous page)

```
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` and check it against the new version `/opt/findface-multi/configs/findface-ntls/findface-ntls.yaml`. Move all the custom parameters from the old version to the new one. Do the same for other components, e.g., for `findface-extract-api`, check `/etc/findface-extract-api.ini` against `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml`, for `findface-sf-api`, check `/etc/findface-sf-api.ini` against `/opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml`, etc.

```
sudo vi /etc/findface-ntls.cfg
sudo vi /opt/findface-multi/configs/findface-ntls/findface-ntls.yaml
sudo vi /etc/findface-extraction-api.ini
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.
↪yaml
sudo vi /etc/findface-sf-api.ini
sudo vi /opt/findface-multi/configs/findface-sf-api/findface-sf-api.yaml
```

Important: Follow these rules to transfer parameters from the old configuration file to the new one:

- If there is a new neural network model in the new configuration file, replace it with the previous one (considering that the previous model is still included in FindFace Multi 2.0), and if the previous model is missing in FindFace Multi 2.0, then do not change anything. In this case, you will have to do *migration to a different neural network model*.
- If a parameter had an empty value in the old configuration file, but has a certain value in the new configuration file, delete its value in the new configuration file.
- Keep as is those parameters that were not included in the old configuration file, but are present in the new configuration file.

---

10. Modify the Tarantool database structure by applying the `tnt_schema.lua` schema from FindFace Multi 2.0.

```
sudo docker run --rm --network host --volume '/opt/findface-multi/configs/findface-
↪multi-legacy/findface-multi-legacy.py:/etc/findface-security/config.py:ro' docker.
↪int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0 make-tnt-schema | sudo tee /etc/
↪findface-security/tnt_schema.lua
```

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

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

12. Copy the meta-schema of the `default` space from the `old_tnt_schema.lua` configuration file to the new `tnt_schema.lua` configuration file, so that the old meta-schema is still available. An easy way to do it is to follow these steps:

12.1. In the `/etc/findface-security/old_tnt_schema.lua` file, rename the following fields:

```
meta_scheme --> meta_scheme_default
meta_indexes --> meta_indexes_default
```

12.2. In the new configuration file `/etc/findface-security/tnt_schema.lua`, replace the following lines at the beginning of the file:

```
spaces = {
  default = {
    meta_scheme = {
      -- internal.normalized_id:
      {
          default = '',
          field_type = 'string',
          id = 1,
          name = 'normalized_id',
      },
      -- internal.feat:
      {
          default = '',
          field_type = 'string',
          id = 2,
          name = 'feat',
      },
    },
    meta_indexes = {}
  },
```

with these ones:

```
dofile("/etc/findface-security/old_tnt_schema.lua")
spaces = {
 default = {
    meta_scheme=meta_scheme_default,
    meta_indexes=meta_indexes_default
  },
```

13. Navigate to the directory with Tarantool configuration file(s) `/etc/tarantool/instances.available/`. Import new meta-schema `tnt_schema.lua` into each configuration file `shard-00*.lua`, as in the example below.

```
sudo vi /etc/tarantool/instances.available/shard-00*.lua

dofile("/etc/findface-security/tnt_schema.lua")
FindFace = require("FindFace")
FindFace.start("127.0.0.1", 8104, {
    license_ntls_server="127.0.0.1:3133",
    replication = replication_master,
    spaces = spaces
 })
```

14. Remove the default configuration file `FindFace.lua`, generated by the `findface-tarantool-server` package, as it will block the restart, required on the next step.

```
sudo rm -rf /etc/tarantool/instances*/FindFace.lua
```

15. 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
```

Upon completion of the above steps, the shards will still keep the old galleries created within the `default` space, but new spaces (e.g., `ffsec_body_objects_space`, `ffsec_face_clusters_space`, and so on) will also become available.

16. Restore old data from the backup. The data will be restored as it existed previously: all galleries will stay within the `default` space.

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

17. Migrate galleries from the `default` space to new spaces:

```
sudo systemctl start findface-sf-api.service
sudo systemctl start nginx.service
sudo docker run --rm --network host --volume '/opt/findface-multi/configs/findface-
↪multi-legacy/findface-multi-legacy.py:/etc/findface-security/config.py:ro' docker.
↪int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0 /opt/findface-security/bin/python3␣
↪/tigre_prototype/manage.py migrate_tnt_space
```

18. Perform PostgreSQL database migrations for FindFace Multi 2.0 compatibility. Do the following:

    18.1. Navigate to the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. In the `DATABASES -> default` section, temporarily replace `PASSWORD` with the old one, used in the `/etc/findface-security/config.py` configuration file.

    Important: Make sure to write down the password from the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. You will need it later.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-
↪legacy.py

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'DISABLE_SERVER_SIDE_CURSORS': True,
        'NAME': 'ffsecurity', 'HOST': '127.0.0.1', 'PORT': 5439, 'USER':
↪'ntech', 'PASSWORD': 'XXXXXXXXXXXXXXXX'
    }
}
```

    18.2. In the `/etc/pgbouncer/pgbouncer.ini` file, add the following line to the `databases` section:

```
ffsecurity_session = dbname=ffsecurity host=localhost port=5432 user=ntech␣
↪pool_mode=session pool_size=10
```

18.3. On the host system, perform the database migration:

```
sudo systemctl start postgresql.service
sudo systemctl start pgbouncer.service
sudo docker run --rm --network host --volume '/opt/findface-multi/configs/
↪findface-multi-legacy/findface-multi-legacy.py:/etc/findface-security/
↪config.py:ro' docker.int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0 /opt/
↪findface-security/bin/python3 /tigre_prototype/manage.py migrate
```

18.4. Back up an existing database with PostgreSQL, installed on the host system.

```
cd /opt/findface-multi/
sudo -u postgres pg_dump --verbose --disable-triggers ffsecurity | sudo tee␣
↪dump_ffsecurity.sql
sudo -u postgres pg_dump -t auth_group -t ffsecurity_adgroupguid -t␣
↪ffsecurity_deviceblacklistrecord -t ffsecurity_ffsecauthsession -t␣
↪ffsecurity_grouppermission -t ffsecurity_runtimesetting -t ffsecurity_
↪user -t ffsecurity_user_groups -t ffsecurity_user_user_permissions -t␣
↪ffsecurity_userkeyvalue -t knox_authtoken -t ffsecurity_
↪watchlistpermission -t ffsecurity_cameragrouppermission --data-only --
↪verbose --no-acl --no-owner --disable-triggers ffsecurity | sudo tee dump_
↪identity_provider.sql
```

18.5. Back up role permissions.

```
sudo docker run --rm --network host --volume '/opt/findface-multi/configs/
↪findface-multi-legacy/findface-multi-legacy.py:/etc/findface-security/
↪config.py:ro' docker.int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0 /opt/
↪findface-security/bin/python3 /tigre_prototype/manage.py dump_permissions␣
↪| sudo tee permissions.csv
```

18.6. **Change back the password, replaced in step #18.1**

18.7. Stop all the services.

```
sudo systemctl stop findface-sf-api.service nginx.service tarantool@shard-00
↪{1..8}.service postgresql.service pgbouncer.service
```

18.8. Open the `/opt/findface-multi/docker-compose.yaml` file and copy `POSTGRES_PASSWORD` value to use it in further commands.

18.9. Recreate the `ffsecurity` database to clean it up from the default data. Paste `{POSTGRES_PASSWORD}` value that you previously copied in step #18.8 into the command below:

```
sudo docker-compose up -d postgresql
sudo docker exec -it -u postgres findface-multi-postgresql-1 /bin/bash -c
↪"PGPASSWORD={POSTGRES_PASSWORD} dropdb ffsecurity"
sudo docker exec -it -u postgres findface-multi-postgresql-1 /bin/bash -c
↪"PGPASSWORD={POSTGRES_PASSWORD} createdb -O ntech --encoding=UTF-8 --lc-
↪collate=C.UTF-8 --lc-ctype=C.UTF-8 --template=template0 ffsecurity"
```

18.10. Restore data into the recreated `ffsecurity` database. Paste `{POSTGRES_PASSWORD}` value that you previously copied in step #18.8 into the command below:

```
sudo docker exec -i findface-multi-postgresql-1 /bin/bash -c "PGPASSWORD=
↪{POSTGRES_PASSWORD} psql --username postgres ffsecurity" < dump_
↪ffsecurity.sql
```

18.11. Recreate the `ffsecurity_identity_provider` database to clean it up from the default data. Paste `{POSTGRES_PASSWORD}` value that you previously copied in step #18.8 into the command below:

```
sudo docker exec -it -u postgres findface-multi-postgresql-1 /bin/bash -c
↪"PGPASSWORD={POSTGRES_PASSWORD} dropdb ffsecurity_identity_provider"
sudo docker exec -it -u postgres findface-multi-postgresql-1 /bin/bash -c
↪"PGPASSWORD={POSTGRES_PASSWORD} createdb -O ntech --encoding=UTF-8 --lc-
↪collate=C.UTF-8 --lc-ctype=C.UTF-8 --template=template0 ffsecurity_
↪identity_provider"
```

18.12. Run migration.

```
sudo docker-compose up -d pgbouncer
sudo docker run --rm --network host --volume '/opt/findface-multi/configs/
↪findface-multi-identity-provider/findface-multi-identity-provider.py:/etc/
↪findface-security/config.py:ro' docker.int.ntl/ntech/multi/multi/identity-
↪provider:ffmulti-2.0.0 /opt/findface-security/bin/python3 /tigre_
↪prototype/manage.py migrate
```

18.13. Restore data into the recreated `ffsecurity_identity_provider` database. Paste `{POSTGRES_PASSWORD}` value that you previously copied in step #18.8 into the command below:

```
sudo docker exec -i findface-multi-postgresql-1 /bin/bash -c "PGPASSWORD=
↪{POSTGRES_PASSWORD} psql --username postgres ffsecurity_identity_provider
↪" < dump_identity_provider.sql
```

18.14. Start all the services.

```
sudo docker-compose up -d
```

18.15. Run the command to create records in the outbox table for watch lists and camera groups. Paste `{POSTGRES_PASSWORD}` value that you previously copied in step #18.8 into the command below:

```
sudo docker exec -i findface-multi-postgresql-1 /bin/bash -c "PGPASSWORD=
↪{POSTGRES_PASSWORD} pg_dump --username postgres -f cg_wl_permissions.sql -
↪t ffsecurity_cameragrouppermission -t ffsecurity_watchlistpermission --
↪data-only ffsecurity_identity_provider"
sudo docker run --rm --network host --volume '/opt/findface-multi/configs/
↪findface-multi-legacy/findface-multi-legacy.py:/etc/findface-security/
↪config.py:ro' docker.int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0 /opt/
↪findface-security/bin/python3 /tigre_prototype/manage.py create_outbox
sudo docker exec -i findface-multi-postgresql-1 /bin/bash -c "PGPASSWORD=
↪{POSTGRES_PASSWORD} psql --username postgres -c 'TRUNCATE ffsecurity_
↪cameragrouppermission, ffsecurity_watchlistpermission RESTART IDENTITY;'␣
↪ffsecurity_identity_provider"
sudo docker exec -i findface-multi-postgresql-1 /bin/bash -c "PGPASSWORD=
↪{POSTGRES_PASSWORD} psql --username postgres ffsecurity_identity_provider
↪< cg_wl_permissions.sql"
```

18.16. Before you restore role permissions into the `identity_provider` service, examine the `/opt/findface-multi/permissions.csv` file. Make sure to replace `*_ffsecauthtoken` with `*_authtoken` if

any. This is mostly applicable to those cases when FindFace Multi 1.2 installation was an upgrade from earlier versions of the product.

After that, restore role permissions into the `identity_provider` service.

```
sudo docker run --rm --network host -v /opt/findface-multi/configs/findface-
↪multi-identity-provider/findface-multi-identity-provider.py:/etc/findface-
↪security/config.py:ro -v $(pwd)/permissions.csv:/var/permissions.csv:ro␣
↪docker.int.ntl/ntech/multi/multi/identity-provider:ffmulti-2.0.0 /opt/
↪findface-security/bin/python3 /tigre_prototype/manage.py load_permissions␣
↪/var/permissions.csv
```

18.17. Copy full frame photos, normalized images, and the license file. Copy files from the `/opt/ntech/license/` folder into the `/opt/findface-multi/data/findface-ntls/` folder, from the `/var/lib/findface-security/uploads/` folder into the `/opt/findface-multi/data/findface-multi-legacy/uploads/` folder, from the `/var/lib/ffupload/uploads/` folder into the `/opt/findface-multi/data/findface-upload/uploads/` folder.

```
sudo cp -r /opt/ntech/license/* /opt/findface-multi/data/findface-ntls/
sudo cp -r /var/lib/findface-security/uploads/* /opt/findface-multi/data/
↪findface-multi-legacy/uploads/
sudo cp -r /var/lib/ffupload/uploads/* /opt/findface-multi/data/findface-
↪upload/uploads/
sudo chmod 777 -R /opt/findface-multi/data/findface-upload/uploads/
sudo chown www-data:www-data -R /opt/findface-multi/data/findface-upload/
↪uploads/*
```

Alternatively, the above folders can be directly mounted into the relevant docker containers via the `docker-compose.yaml` file, like in the example below:

```
sudo vi /opt/findface-multi/docker-compose.yaml

findface-upload:
  image: docker.int.ntl/ntech/universe/upload:ffserver-8.221216
  logging: {driver: journald}
  network_mode: service:pause
  restart: always
  volumes: ['./configs/findface-upload/40-ffupload.sh:/docker-entrypoint.d/
↪40-ffupload.sh:ro',
    '/var/lib/ffupload:/var/lib/ffupload']
findface-multi-identity-provider:
  depends_on: [pgbouncer, nats, findface-sf-api, findface-liveness-api,␣
↪etcd]
  environment: {ADMIN_PASSWORD: <ADMIN_PASSWORD>}
  image: docker.int.ntl/ntech/multi/multi/identity-provider:ffmulti-2.0.0
  logging: {driver: journald}
  network_mode: service:pause
  restart: always
  volumes: ['./configs/findface-multi-identity-provider/findface-multi-
↪identity-provider.py:/etc/findface-security/config.py:ro',
    '/var/lib/findface-security/uploads:/var/lib/findface-security/uploads']
findface-multi-legacy:
  depends_on: [pgbouncer, nats, findface-sf-api, findface-counter, findface-
↪liveness-api, etcd]
```

(continues on next page)

(continued from previous page)

```
  environment: {ADMIN_PASSWORD: <ADMIN_PASSWORD>}
  image: docker.int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0
  logging: {driver: journald}
  network_mode: service:pause
  restart: always
  volumes: ['./configs/findface-multi-legacy/findface-multi-legacy.py:/etc/
↪findface-security/config.py:ro',
    '/var/lib/findface-security/uploads:/var/lib/findface-security/uploads']
findface-multi-ui:
  depends_on: [findface-multi-legacy]
  image: docker.int.ntl/ntech/multi/multi/ui:ffmulti-2.0.0
  logging: {driver: journald}
  network_mode: service:pause
  restart: always
  volumes: ['./configs/findface-multi-ui/nginx-site.conf:/etc/nginx/conf.d/
↪default.conf:ro',
    '/var/lib/findface-security/uploads:/var/lib/findface-security/uploads']
```

18.18. To move Tarantool data, do the following:

Stop all FindFace Multi containers:

```
sudo docker-compose down
```

Start the old shards and the `findface-sf-api` service again:

```
sudo systemctl start tarantool@shard-00{1..8}.service findface-sf-api.
↪service
```

Create a new backup of the feature vector database:

```
sudo mkdir -p /etc/findface_dump_final
sudo findface-storage-api-dump -output-dir=/etc/findface_dump_final -config␣
↪/etc/findface-sf-api.ini
```

Stop the rest of the services, clear the `instances.enabled` directory, start the containers again, and perform the storage-api-restore operation:

```
sudo systemctl stop tarantool@shard-00{1..8}.service findface-sf-api.
↪service findface-ntls.service
sudo rm /etc/tarantool/instances.enabled/*
sudo docker-compose up -d
sudo findface-storage-api-restore -config /opt/findface-multi/configs/
↪findface-sf-api/findface-sf-api.yaml /etc/findface_dump_final/*.json
```

The update has been completed, but the new version includes new neural network models, so it is also necessary to *migrate feature vectors to a different neural network model*, or you can use old neural network models by moving them from the `/usr/share/findface-data/models/` directory to the `/opt/findface-multi/models/` directory, specifying them in the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` file.

Note that object detection models in FindFace Multi 1.2 are stored in separate directories, i.e., face detection models are stored in the `/usr/share/findface-data/models/facedet/` directory, car and body detection models are stored in the `/usr/share/findface-data/models/cadet/` and `/usr/share/findface-data/models/pedet/`

directories respectively. In FindFace Multi 2.0 all object detection models can be found in the `/opt/findface-multi/models/detector/` directory. When moving old neural network models from the `/usr/share/findface-data/models/` directory to the `/opt/findface-multi/models/` directory, make sure to place all object detection models (`facedet`, `cadet`, `pedet`) to the `/opt/findface-multi/models/detector/` directory.

See for reference:

```
$ ls -lash /usr/share/findface-data/models
total 52K
4.0K drwxr-xr-x 13 root root 4.0K Jul 15 14:48 .
4.0K drwxr-xr-x  3 root root 4.0K Jul 15 14:48 ..
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 cadet
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 carattr
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 carnorm
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 carrec
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 face
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 faceattr
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 facedet
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 facenorm
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 pedattr
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 pedet
4.0K drwxr-xr-x  2 root root 4.0K Jul 15 14:48 pedrec


$ ls -lash /opt/findface-multi/models/
total 44K
4.0K drwxr-xr-x 11 root root 4.0K Jul 17 13:37 .
4.0K drwxr-xr-x  6 root root 4.0K Jul 19 15:31 ..
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 13:37 carattr
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 13:37 carnorm
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 13:36 carrec
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 16:20 detector
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 16:24 face
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 16:24 faceattr
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 13:37 facenorm
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 13:37 pedattr
4.0K drwxr-xr-x  2 root root 4.0K Jul 17 13:37 pedrec
```

Important: We highly recommend disabling the Ubuntu automatic update to preserve the FindFace Multi compatibility with the installation environment. 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
```

Note: You can additionally deploy Video Recorder. See *Deploy Video Recorder Step-by-Step*.

2.6.2 Back Up and Recover FindFace Multi and Its Data

You can back up FindFace Multi before uninstalling it to recover the product and its data later on.

In this section:

Back up FindFace Multi

To back up your FindFace Multi instance and its data, run the following commands:

```
sudo tar -cvzf ~/configs.tar.gz -C /opt/findface-multi/ configs
sudo tar -cvzf ~/data.tar.gz -C /opt/findface-multi/ data
```

Recover FindFace Multi and Its Data

To restore FindFace Multi and its data from the backup, do the following:

1. Download the installer file `findface-*.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.

   Note: Be sure to specify the actual file name instead of `findface-*`.

   ```
   chmod +x findface-*.run
   ```

4. Execute the `.run` file.

   ```
   sudo ./findface-*.run
   ```

5. Go through the installation process as described *here*.
6. After you have finished the installation, to restore FindFace Multi, its data and configuration files, stop all FindFace Multi containers.

   ```
   cd /opt/findface-multi
   sudo docker-compose stop
   ```

7. Remove new configuration files and data generated and created by the installer and restore them from the backup.

   ```
   sudo rm -r /opt/findface-multi/configs/*
   sudo tar -xvf ~/configs.tar.gz -C /opt/findface-multi/
   sudo rm -r /opt/findface-multi/data/*
   sudo tar -xvf ~/data.tar.gz -C /opt/findface-multi/
   ```

8. Restart FindFace Multi containers.

```
cd /opt/findface-multi
sudo docker-compose up -d
```

2.6.3 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 face metadata, and so on.

In this section:

About Database Structure

In FindFace Multi, the database structure is set via the `/opt/findface-multi/configs/findface-tarantool-server/tnt-schema.lua` file.

The structure is created as a set of spaces and 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 (`unsigned|string|set[string]|set[unsigned]`);
- `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. Create a backup of the Tarantool-based feature vector database in any directory of your choice, for example, `/etc/ffmulti_dump`.

```
sudo docker exec -it findface-multi-findface-sf-api-1 bash -c "mkdir ffmulti_dump;␣
↪cd ffmulti_dump && /storage-api-dump -config /etc/findface-sf-api.ini"
sudo docker cp findface-multi-findface-sf-api-1:/ffmulti_dump /etc
```

2. Modify the database structure by applying the new schema to the `tnt_schema.lua` file.

```
sudo docker run --rm --network host --volume '/opt/findface-multi/configs/findface-
↪multi-legacy/findface-multi-legacy.py:/etc/findface-security/config.py:ro' docker.
↪int.ntl/ntech/multi/multi/legacy:ffmulti-2.0.0 make-tnt-schema | sudo tee /opt/
↪findface-multi/configs/findface-tarantool-server/tnt-schema.lua
```

3. Navigate to the directory with Tarantool configuration file(s) `/opt/findface-multi/configs/findface-tarantool-server/`. For each shard `shard-00*.lua`, check whether it contains the `dofile` command. Make sure the `replication` and `spaces` are defined within the `FindFace.start` parameter, as in the example below.

```
sudo vi /opt/findface-multi/configs/findface-tarantool-server/shard-00*.lua

dofile("/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",
    replication = replication_master,
    spaces = spaces
})
```

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

```
sudo rm /opt/findface-multi/data/findface-tarantool-server/shard-*/{index,snapshots,
↪xlogs}/*
```

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

```
docker restart findface-multi-findface-tarantool-server-shard-001-1 findface-multi-
↪findface-tarantool-server-shard-002-1 findface-multi-findface-tarantool-server-
↪shard-003-1 findface-multi-findface-tarantool-server-shard-004-1 findface-multi-
↪findface-tarantool-server-shard-005-1 findface-multi-findface-tarantool-server-
↪shard-006-1 findface-multi-findface-tarantool-server-shard-007-1 findface-multi-
↪findface-tarantool-server-shard-008-1
```

6. Restore the Tarantool database from the backup.

```
sudo docker exec -it findface-multi-findface-sf-api-1 bash -c 'cd ffmulti_dump &&␣
↪for x in *.json; do /storage-api-restore -config /etc/findface-sf-api.ini < "$x";␣
↪done;'
```

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

See also:

Custom Metadata in Tarantool

2.6.4 Check Component Status

Check the status of containers once you have encountered a system problem, using the following commands:

```
docker ps
sudo docker container inspect <container_id>/<container_name>
sudo docker container stats <container_id>/<container_name>
```

2.6.5 Logging

Consulting logs is one of the first things you should do to identify a cause of a problem. By default, the FindFace Multi processes are logged to Docker container logs, which can be accessed via the `docker logs` and `docker service logs` commands. In addition, Docker uses the json-file logging driver, which caches container logs in JSON. You can configure Docker to use another logging driver, choosing from the multiple logging mechanisms available.

This section describes how to set up Docker to use the `journald` logging driver, which sends container logs to the `systemd` journal. In this case, log entries are retrieved using the `journalctl` command, through the `journal API`, or the `docker logs` command. You can configure the `systemd` journal as well by using the instructions below.

In this section:

Configure Journald

To configure the `systemd-journal` service, 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, ... bytes).

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

3. Restart the `journald` service.

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

Enabling Journald Logging Driver

To enable Docker to use the `journald` logging driver, do the following:

1. Add the following line to the `/etc/docker/daemon.json` configuration file.

Tip: This file may not be present on your system. Check whether it exists and if it does not, create the `/etc/docker` directory first and then the file.

```
sudo mkdir /etc/docker
sudo touch /etc/docker/daemon.json
```

```
sudo vi /etc/docker/daemon.json

{
  "log-driver": "journald"
}
```

2. Stop all Docker containers.

```
sudo docker stop $(sudo docker ps -a -q)
```

3. Restart the Docker service.

```
sudo systemctl restart docker
```

4. Start all Docker containers.

```
sudo docker start $(sudo docker ps -a -q)
```

Consult Logs

Use any convenient method to work with the `journald` logs. The following commands are a good place to start:

- Display all logs:

```
journalctl -fa
```

- Display logs by container ID:

```
journalctl CONTAINER_ID=<container_id> -f
```

- Display logs by container name:

```
journalctl CONTAINER_NAME=<container-name> -f
```

See also:

Audit Log

2.6.6 Troubleshoot Licensing and `findface-ntls`

When troubleshooting licensing and `findface-ntls` (see *Licensing*), 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 *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" -> "value": 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": ""
  },
```

(continues on next page)

(continued from previous page)


```
  "source": "/opt/ntech/license/import_
↪b68d7b7ec9a7310d18832035318cff0c9ddf11e3a9ab0ae962fbe48645e196d1.lic",
  "limits": [
    {
      "type": "time",
      "name": "end",
      "value": 1609161621
    },
    {
      "type": "number",
      "name": "faces",
      "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
    },
    {
```


(continues on next page)

(continued from previous page)


```
      "type": "boolean",
      "name": "beard",
      "value": false
    },
    {
      "type": "boolean",
      "name": "glasses",
      "value": false
    },
    {
      "type": "boolean",
      "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"
    }
  ]
}
```


2.6.7 Manually Purge Old Data from Database

Tip: To schedule automatic database cleanup, see *General Settings*.

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, vehicles) and related episodes,
- unmatched events (faces, bodies, vehicles) and related episodes,
- full frames of matched events (faces, bodies, vehicles),
- full frames of unmatched events (faces, bodies, vehicles),
- counter screenshots,
- cluster events (faces, bodies, vehicles),
- audit-logs,
- external VMS events.

The `cleanup` utility runs in the `findface-multi-findface-multi-legacy-1` container. To invoke the `cleanup` help message, execute:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --help

usage: manage.py cleanup [-h] [--as-configured]
                         [--face-events-max-fullframe-unmatched-age FACE_EVENTS_MAX_
↪FULLFRAME_UNMATCHED_AGE]
                         [--face-events-max-fullframe-matched-age FACE_EVENTS_MAX_
↪FULLFRAME_MATCHED_AGE]
                         [--face-events-max-unmatched-age FACE_EVENTS_MAX_UNMATCHED_AGE]
                         [--face-events-max-matched-age FACE_EVENTS_MAX_MATCHED_AGE]
                         [--body-events-max-fullframe-unmatched-age BODY_EVENTS_MAX_
↪FULLFRAME_UNMATCHED_AGE]
                         [--body-events-max-fullframe-matched-age BODY_EVENTS_MAX_
↪FULLFRAME_MATCHED_AGE]
                         [--body-events-max-unmatched-age BODY_EVENTS_MAX_UNMATCHED_AGE]
                         [--body-events-max-matched-age BODY_EVENTS_MAX_MATCHED_AGE]
                         [--car-events-max-fullframe-unmatched-age CAR_EVENTS_MAX_
↪FULLFRAME_UNMATCHED_AGE]
                         [--car-events-max-fullframe-matched-age CAR_EVENTS_MAX_
↪FULLFRAME_MATCHED_AGE]
                         [--car-events-max-unmatched-age CAR_EVENTS_MAX_UNMATCHED_AGE]
                         [--car-events-max-matched-age CAR_EVENTS_MAX_MATCHED_AGE]
                         [--car-cluster-events-max-age CAR_CLUSTER_EVENTS_MAX_AGE]
                         [--body-cluster-events-max-age BODY_CLUSTER_EVENTS_MAX_AGE]
                         [--face-cluster-events-max-age FACE_CLUSTER_EVENTS_MAX_AGE]
                         [--car-cluster-events-keep-best-max-age CAR_CLUSTER_EVENTS_KEEP_
↪BEST_MAX_AGE]
                         [--body-cluster-events-keep-best-max-age BODY_CLUSTER_EVENTS_
↪KEEP_BEST_MAX_AGE]
                         [--face-cluster-events-keep-best-max-age FACE_CLUSTER_EVENTS_
↪KEEP_BEST_MAX_AGE]
                         [--area-activations-max-age AREA_ACTIVATIONS_MAX_AGE]
                         [--audit-logs-max-age AUDIT_LOGS_MAX_AGE]
                         [--counter-records-max-age COUNTER_RECORDS_MAX_AGE]
                         [--external-vms-events-max-age EXTERNAL_VMS_EVENTS_MAX_AGE]
                         [--external-vms-send-events-status-max-age EXTERNAL_VMS_SEND_
↪EVENTS_STATUS_MAX_AGE]
                         [--remote-monitoring-events-max-age REMOTE_MONITORING_EVENTS_
↪MAX_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 clusters. Can't be used with other arguments.
```

(continues on next page)

(continued from previous page)

```
  --face-events-max-fullframe-unmatched-age FACE_EVENTS_MAX_FULLFRAME_UNMATCHED_AGE
                        face events max fullframe unmatched age to clean up
                        (in seconds)
  --face-events-max-fullframe-matched-age FACE_EVENTS_MAX_FULLFRAME_MATCHED_AGE
                        face events max fullframe matched age to clean up (in
                        seconds)
  --face-events-max-unmatched-age FACE_EVENTS_MAX_UNMATCHED_AGE
                        face events max unmatched age to clean up (in seconds)
  --face-events-max-matched-age FACE_EVENTS_MAX_MATCHED_AGE
                        face events max matched age to clean up (in seconds)
  --body-events-max-fullframe-unmatched-age BODY_EVENTS_MAX_FULLFRAME_UNMATCHED_AGE
                        body events max fullframe unmatched age to clean up
                        (in seconds)
  --body-events-max-fullframe-matched-age BODY_EVENTS_MAX_FULLFRAME_MATCHED_AGE
                        body events max fullframe matched age to clean up (in
                        seconds)
  --body-events-max-unmatched-age BODY_EVENTS_MAX_UNMATCHED_AGE
                        body events max unmatched age to clean up (in seconds)
  --body-events-max-matched-age BODY_EVENTS_MAX_MATCHED_AGE
                        body events max matched age to clean up (in seconds)
  --car-events-max-fullframe-unmatched-age CAR_EVENTS_MAX_FULLFRAME_UNMATCHED_AGE
                        car events max fullframe unmatched age to clean up (in
                        seconds)
  --car-events-max-fullframe-matched-age CAR_EVENTS_MAX_FULLFRAME_MATCHED_AGE
                        car events max fullframe matched age to clean up (in
                        seconds)
  --car-events-max-unmatched-age CAR_EVENTS_MAX_UNMATCHED_AGE
                        car events max unmatched age to clean up (in seconds)
  --car-events-max-matched-age CAR_EVENTS_MAX_MATCHED_AGE
                        car events max matched age to clean up (in seconds)
  --car-cluster-events-max-age CAR_CLUSTER_EVENTS_MAX_AGE
                        car cluster events max age to clean up (in seconds)
  --body-cluster-events-max-age BODY_CLUSTER_EVENTS_MAX_AGE
                        body cluster events max age to clean up (in seconds)
  --face-cluster-events-max-age FACE_CLUSTER_EVENTS_MAX_AGE
                        face cluster events max age to clean up (in seconds)
  --car-cluster-events-keep-best-max-age CAR_CLUSTER_EVENTS_KEEP_BEST_MAX_AGE
                        car cluster events keep best max age to clean up (in
                        seconds)
  --body-cluster-events-keep-best-max-age BODY_CLUSTER_EVENTS_KEEP_BEST_MAX_AGE
                        body cluster events keep best max age to clean up (in
                        seconds)
  --face-cluster-events-keep-best-max-age FACE_CLUSTER_EVENTS_KEEP_BEST_MAX_AGE
                        face cluster events keep best max age to clean up (in
                        seconds)
  --area-activations-max-age AREA_ACTIVATIONS_MAX_AGE
                        area activations max age to clean up (in seconds)
  --audit-logs-max-age AUDIT_LOGS_MAX_AGE
                        audit logs max age to clean up (in seconds)
  --counter-records-max-age COUNTER_RECORDS_MAX_AGE
                        counter records max age to clean up (in seconds)
  --external-vms-events-max-age EXTERNAL_VMS_EVENTS_MAX_AGE
```

(continues on next page)

(continued from previous page)

```
                        external vms events max age to clean up (in seconds)
  --external-vms-send-events-status-max-age EXTERNAL_VMS_SEND_EVENTS_STATUS_MAX_AGE
                        external vms send events status max age to clean up
                        (in seconds)
  --remote-monitoring-events-max-age REMOTE_MONITORING_EVENTS_MAX_AGE
                        remote monitoring events max age to clean up (in
                        seconds)
  --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
                        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 seconds, use the `--*-events-max-matched-age/--*-events-max-unmatched-age` options, subject to the object type. For example, to remove unmatched car events older than 5 days (432000 seconds), execute:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --car-events-max-unmatched-age 432000
```

To remove only matched car events older than 5 days (432000 seconds), execute:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --car-events-max-matched-age 432000
```

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

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --body-events-max-fullframe-matched-age␣
↪432000
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --body-events-max-fullframe-unmatched-age␣
↪432000
```

To remove only counter screenshots, execute:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --counter-records-max-age 432000
```

To remove only cluster events with faces, execute:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --face-cluster-events-max-age 432000
```

To remove only audit logs, execute:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py cleanup --audit-logs-max-age 432000
```

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

2.6.8 Reset Password

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

```
sudo docker exec -it findface-multi-findface-multi-identity-provider-1 /opt/findface-
↪security/bin/python3 /tigre_prototype/manage.py changepassword <username>
```

2.6.9 S3-Compatible Storage Configuration

S3 (Simple Storage Service) provides reliable and long-term storage of an unlimited number of files and data. It gives a way to avoid the limitations of the file system when it comes to large amounts of data.

Configure S3-Compatible Storage

In this section, we will describe how to configure S3-compatible MinIO storage for operation with FindFace Multi. If you prefer another S3-compatible storage, you are yourself responsible for its configuration.

1. *Prepare a server on Ubuntu*.

   Note: MinIO operation has been tested on Ubuntu only. Refer to the official MinIO documentation to find out whether other OS are supported.

2. To start a container with MinIO, execute the following command, specify the `MINIO_ROOT_USER` and `MINIO_ROOT_PASSWORD` login credentials:

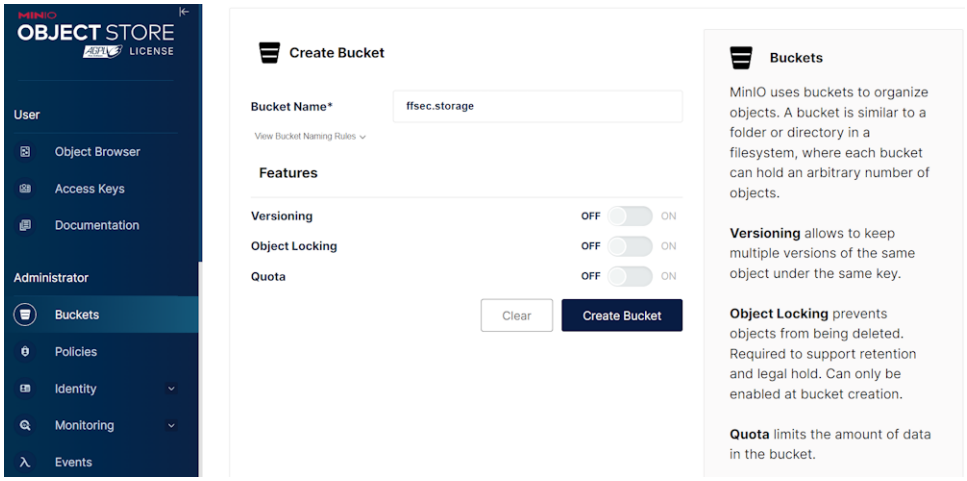
   ```
   docker run -d \
   -p 9003:9000 \
   -p 9004:9001 \
   -e "MINIO_ROOT_USER=admin" \
   -e "MINIO_ROOT_PASSWORD=admin_admin" \
   -v /path/to/storage_directory:/data \
   quay.io/minio/minio server /data --console-address ":9001"
   ```

   where `/path/to/storage_directory` is a directory, allocated for the S3 storage.

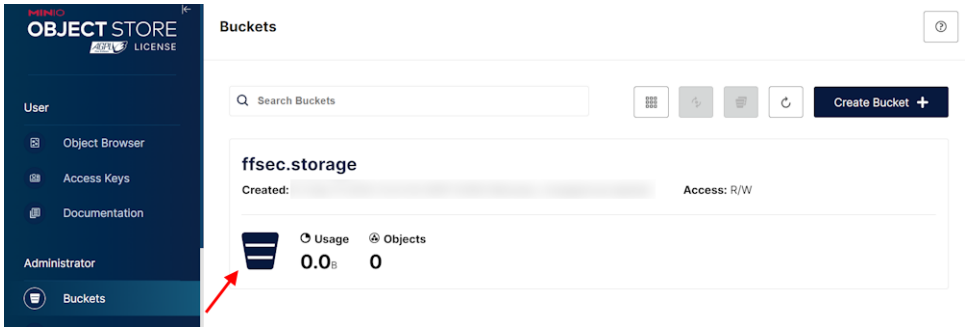
3. MinIO web interface address will depend on the interface that you have selected during FindFace Multi *installation* in step #4.5. The port in use is `9004`, e.g., `192.168.112.254:9004`. Log in to the MinIO web interface with `MINIO_ROOT_USER`, `MINIO_ROOT_PASSWORD` login credentials that you have specified in the previous step.

4. In the MinIO UI, create a bucket (*Buckets -> Create Bucket*). Assign a default name `ffsec.storage` to the bucket.

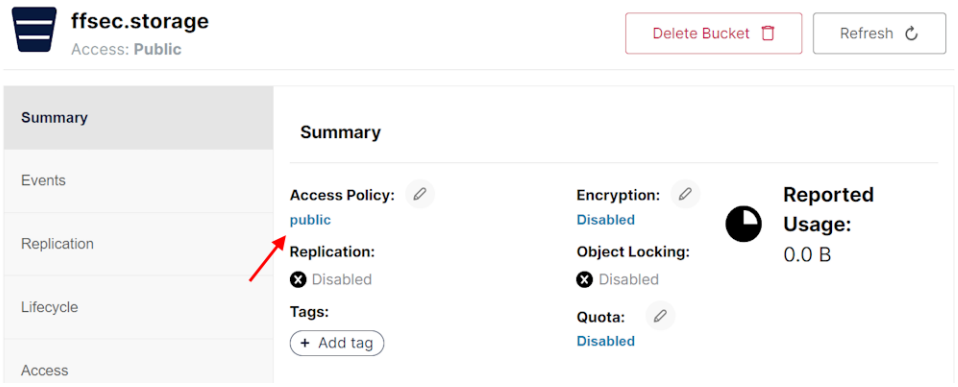
Note: We advise keeping the bucket name `ffsec.storage` as suggested by default, because it corresponds to the `'STORAGE_BUCKET_NAME'` parameter in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` and `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-identity-provider.py` configuration files. If you use another name for a bucket, make sure to adjust it in the above-mentioned configuration files accordingly.



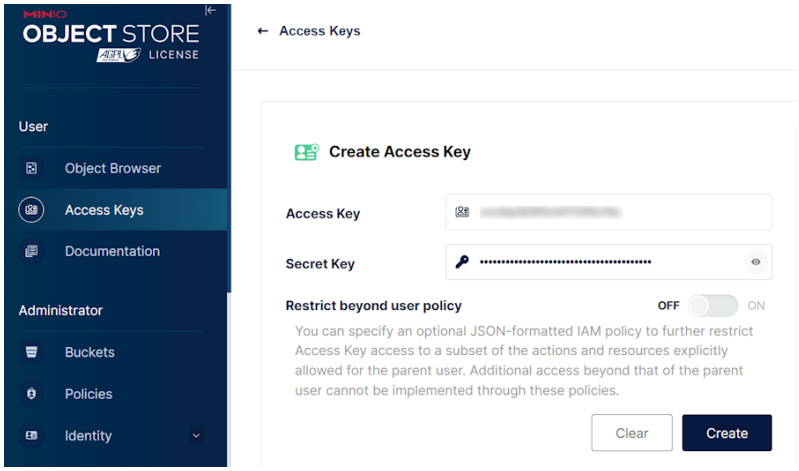
5. Click on the bucket to open its settings.



6. Change access policy for the bucket to `public` for the correct processing of video archives.



7. In the *User* -> *Access Keys* section, create an access key. **Write down the `Access key` and `Secret key` values as you will need them later for FindFace Multi configuration.**



Note: MinIO UI may differ. In the earlier versions, set up an access key in the *Identity* -> *Service Accounts* -> *Create service account* section.

Configure FindFace Multi to Employ S3-Compatible Storage

1. In the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file, locate the `EXTERNAL_STORAGE_CONFIG` section.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

EXTERNAL_STORAGE_CONFIG = {
    'STORAGE_TYPE': None,
    'STORAGE_BUCKET_NAME': 'ffsec.storage',
    'ACCESS_KEY_ID': 'access-key',
    'SECRET__KEY': 'secret-key',
    'REGION_NAME': 'eu-west-3',
    'LOCAL_S3_SETTINGS': {
```

(continues on next page)

(continued from previous page)

```
        'CONNECTION_ADDRESS': 'localhost:9003',
    },
}
```

Configure the EXTERNAL_STORAGE_CONFIG parameters:

1. The 'STORAGE_TYPE' parameter may take values 'LOCAL', 'AWS', None:
   - None – local file system (default)
   - 'LOCAL' – any S3-compatible storage (e.g., MinIO)
   - 'AWS' – Amazon S3 storage.

   Change the 'STORAGE_TYPE' parameter to 'LOCAL' to enable S3-compatible storage.
2. The values of the 'STORAGE_BUCKET_NAME', 'ACCESS_KEY_ID', and 'SECRET__KEY' parameters should correspond to those that were specified during MinIO initialization.
3. 'REGION_NAME' – the default value is eu-west-3. It can be changed in MinIO settings.
4. In the 'LOCAL_S3_SETTINGS' -> 'CONNECTION_ADDRESS', replace the localhost with MinIO IP host address, e.g., 192.168.112.254.

2. Configure the /opt/findface-multi/configs/findface-multi-identity-provider/findface-multi-identity-provider.py file in the same way. Locate the EXTERNAL_STORAGE_CONFIG section and apply changes, analogous to those made in the /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py file in the previous step.

```
sudo vi /opt/findface-multi/configs/findface-multi-identity-provider/findface-multi-
↪identity-provider.py

EXTERNAL_STORAGE_CONFIG = {
    'STORAGE_TYPE': None,
    'STORAGE_BUCKET_NAME': 'ffsec.storage',
    'ACCESS_KEY_ID': 'access-key',
    'SECRET__KEY': 'secret-key',
    'REGION_NAME': 'eu-west-3',
    'LOCAL_S3_SETTINGS': {
        'CONNECTION_ADDRESS': 'localhost:9003',
    },
}
```

3. In order for the UI to load content (thumbnails, full frame images, and so on), configure the /opt/findface-multi/configs/findface-multi-ui/nginx-site.conf file. In the server section, locate the add_header Content-Security-Policy and add {ip_host_s3}:9003/ address, e.g.:

```
sudo vi /opt/findface-multi/configs/findface-multi-ui/nginx-site.conf

...
server {
        listen 80 default_server;
        listen [::]:80 default_server;
...
        location / {
...
        }
```

(continues on next page)

(continued from previous page)

```
        add_header Content-Security-Policy "default-src 'self' 192.168.112.254:9003/
↪ ; img-src 'self' 192.168.112.254:9003/ blob: data:; media-src 'self' blob:;␣
↪style-src 'self' 'unsafe-inline';";
...
}
```

4. Rebuild all FindFace Multi containers from the `/opt/findface-multi` directory .

```
cd /opt/findface-multi

sudo docker-compose down

sudo docker-compose up -d
```

5. To make sure that the `FindFace Local S3 Storage` service is running, explore the `findface-multi-findface-multi-legacy-1` logs:

```
sudo docker logs findface-multi-findface-multi-legacy-1
```

Data Transfer Command

To transfer data to either remote or local storage, use the following command:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py transfer_data --to_local / --to_remote
```

- The argument `--to_remote` transfers data to remote storage.
- The argument `--to_local` transfers data to the local file system.

Before executing the above command, make sure that you have *configured FindFace Multi* to use S3-compatible storage.

2.6.10 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
```

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 and Video Storage

Note: We recommend that you get acquainted with *FindFace Multi Data Storages* before starting migration of storages.

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

1. Inside the mount point, create directories (e.g., `/mnt/ffdata/ffupload` and `/mnt/ffdata/findface-security`) to store photos and recorded video chunks (if the Video Recorder was deployed). Then, move contents of the `/opt/findface-multi/data/findface-upload/uploads/` and the `/opt/findface-multi/data/findface-multi-legacy/uploads/` directories to the newly created directories.

```
sudo mkdir /mnt/ffdata/ffupload
sudo mkdir /mnt/ffdata/findface-security
sudo cp -ax /opt/findface-multi/data/findface-upload/uploads/ -R /mnt/ffdata/
↪ffupload/
sudo cp -ax /opt/findface-multi/data/findface-multi-legacy/uploads/ -R /mnt/ffdata/
↪findface-security/
```

(continues on next page)

(continued from previous page)

```
sudo rm -r /opt/findface-multi/data/findface-multi-legacy/uploads/
sudo rm -r /opt/findface-multi/data/findface-upload/uploads/
```

2. Mount the created directories (`/mnt/ffdata/ffupload` and `/mnt/ffdata/findface-security` in the example) into the corresponding containers. To do so, open the `/opt/findface-multi/docker-compose.yaml` configuration file and list them in the volumes of the sections mentioned in the example instead of the default directories.

```
sudo vi /opt/findface-multi/docker-compose.yaml

findface-upload:
  ...
  volumes: ['./configs/findface-upload/40-ffupload.sh:/docker-entrypoint.d/40-
↪ffupload.sh:ro',
    '/mnt/ffdata/ffupload:/var/lib/ffupload']
...
findface-multi-identity-provider:
  ...
  volumes: ['./configs/findface-multi-identity-provider/findface-multi-identity-
↪provider.py:/etc/findface-security/config.py:ro',
      '/mnt/ffdata/findface-security/uploads:/var/lib/findface-security/uploads']
...
findface-multi-legacy:
  ...
  volumes: ['./configs/findface-multi-legacy/findface-multi-legacy.py:/etc/findface-
↪security/config.py:ro',
      '/mnt/ffdata/findface-security/uploads:/var/lib/findface-security/uploads']
...
findface-multi-ui:
  ...
  volumes: ['./configs/findface-multi-ui/nginx-site.conf:/etc/nginx/conf.d/default.
↪conf:ro',
      '/mnt/ffdata/findface-security/uploads:/var/lib/findface-security/uploads']
```

3. Rebuild all FindFace Multi containers.

```
cd /opt/findface-multi

sudo docker-compose down

sudo docker-compose up -d
```

Migrate Main Database (PostgreSQL)

To migrate the PostgreSQL database, do the following:

1. Stop all FindFace Multi containers:

```
cd /opt/findface-multi

sudo docker-compose down
```

2. Move the `/opt/findface-multi/data/postgresql` directory to the `/mnt/ffdata` directory.

```
sudo mv /opt/findface-multi/data/postgresql /mnt/ffdata
```

3. Specify the new path of the `postgresql` directory (`/mnt/ffdata/postgresql` in the example) to mount into the `findface-multi-postgresql-1` container. To do so, open the `/opt/findface-multi/docker-compose.yaml` configuration file and list the `/mnt/ffdata/postgresql` directory in the volumes of the `postgresql` section instead of the default `/opt/findface-multi/data/postgresql` directory.

```
sudo vi /opt/findface-multi/docker-compose.yaml

postgresql:
...
  volumes: ['./configs/postgresql/40-init.sql:/docker-entrypoint-initdb.d/40-init.
↪sql:ro',
    '/mnt/ffdata/postgresql:/bitnami/postgresql']
```

4. Start all FindFace Multi containers.

```
sudo docker-compose up -d
```

2.6.11 Disable Services

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

- episodes
- video archive queue manager
- webhooks
- Active Directory
- VMS cleanup

To do so, open the `/opt/findface-multi/config/findface-multi-legacy/findface-multi-legacy.py` configuration file and modify the `SERVICES` section, setting `False` for the services that are no longer in use. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo vi /opt/findface-multi/config/findface-multi-legacy/findface-multi-legacy.py

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

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

After that, relevant sections will disappear from the web interface.

Note: A relevant section will remain if there are some entities on it (for example, webhooks on the *Webhooks* tab). However, the service will stop working and generating data.

2.6.12 Hide Menu Items

To hide specific menu items, do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. In the `FFSECURITY_UI_CONFIG` section, find the `menu` section. List the menu items that you want to hide.

```
FFSECURITY_UI_CONFIG = {
    "menu": {
        "disabled_items": [
            "areas",
            "player",
            "cases"
        ],
    },
```

You can hide the following items:

Menu item	Configure as follows
Search	`"search"`
Interaction Tracking	`"relations"`
Video Wall	`"video-wall"`
Counters	`"counters"`
Video Sources	`"data_sources"`
Episodes and Events	`"events_episodes"`
Clusters	`"clusters"`
Audience Analysis	`"analytics"`
Settings	`"settings"`
Record Index	`"cards"`
Video Player	`"player"`
Verify	`"verify"`
Audit Log	`"audit-logs"`
Reports	`"reports"`
Users	`"users"`
Sessions	`"sessions"`
Blocklist records	`"blocklistRecords"`
Camera groups	`"camera-groups"`
Watch lists	`"watch-lists"`
License	`"license"`
External VMS	`"external-vms"`
Webhooks	`"hooks"`
Documentation	`"documentation"`
API documentation	`"api_doc"`
menu items activated by custom plugins	Contact our support team for details about your plugin.

3. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

2.7 Appendices

2.7.1 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 recognition events: `/opt/findface-multi/data/findface-tarantool-server`
- PostgreSQL-based main system database that stores internal system data, records, and user accounts: `/opt/findface-multi/data/postgresql`
- The directory that stores photos uploaded to records, video files, full frames of events and counters, and object thumbnails: `/opt/findface-multi/data/findface-upload/uploads`
- The directory that stores such event artifacts as normalized object images: `/opt/findface-multi/data/findface-multi-legacy/uploads`

  Note: It also stores video chunks if you are using *Video Recorder*.

- (Only with Video Recorder) MongoDB-based database storage: `/opt/findface-multi/data/mongodb/` that stores meta-information of the video chunks, including their exact location in the `/opt/findface-multi/data/findface-multi-legacy/uploads`.

Feature Vector Database Galleries

There are the following default 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 body images uploaded to the record index.
- `ffsec_body_clusters`: centroids of body clusters.
- `ffsec_car_events`: feature vectors extracted from vehicles detected in the video.
- `ffsec_car_objects`: feature vectors extracted from vehicle images uploaded to the record index.
- `ffsec_car_clusters`: centroids of vehicle clusters.
- `ffsec_face_events`: feature vectors extracted from faces detected in the video.

- `ffsec_face_objects`: feature vectors extracted from face images uploaded to the record index.
- `ffsec_face_clusters`: centroids of face clusters.
- `ffsec_user_face`: feature vectors extracted from the FindFace Multi users' photos for face-based authentication.

CHAPTER

THREE

USER'S GUIDE

This chapter describes how to work with the FindFace web interface, including its advanced possibilities, and will be of interest to administrators, operators, and other users.

3.1 Getting Started

Once you have successfully *deployed and configured* FindFace Multi, it's time to 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:

3.1.1 Gear Up for Work

Perform the primary configuration of your system:

1. Log into your system.
2. Choose the language and dark or light theme.

3.1.2 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*.

3.1.3 Organize Watch Lists and Record Index

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, vehicles) by arbitrary criteria, e.g., persona non grata, wanted, VIP, staff, etc.
2. Upload records and add them to the watch list either *manually*, *in bulk via the web interface*, or use the *console bulk upload* function.

You may also need:

Customize record content. Create additional fields, tabs, and search filters.

3.1.4 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. *Enable automatic clustering of objects of the same origin*: face/body images belonging to the same person, images of the same vehicle.
2. Support laws related to the processing of personal data of individuals (GDPR and similar). *Learn more*.

3.1.5 Organize Video Surveillance

Create a camera layout for essential video surveillance.

You may also need:

For advanced video surveillance, enable *Video Recorder*.

3.1.6 Count People and Vehicles. Measure Distance between People

Set up *counters* to count faces, bodies, and vehicles on connected cameras. You can also configure counters to measure distance between bodies. The Counters possibilities can apply to a wide range of situations, such as people counting in queues and waiting areas, monitoring public gatherings, crowding prevention, health protocol enforcement, traffic jam detection, and more.

3.1.7 People-Related Analytics

FindFace Multi provide a set of people-related analytical tools:

1. *Analyze* interactions. Examine a circle of people with whom a person has previously been in contact.
2. *Analyze audience* by the number of visitors, their gender, average age, most frequently visited zones, and the character of visits (first-timers or returners).

3.1.8 FindFace Multi in Action

1. *Automatically identify objects (faces, bodies, vehicles) in live video* and check them against watch lists. Work with the event history by using various filters.
2. Harness the *episodes*. An episode is a set of identification events that feature objects of the same origin (face and body images of the same person and images of the same vehicle) 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 objects 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 card index. *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, clusters, counters, cameras, card index, audience, or audit logs.

3.1.9 Basic Maintenance

1. *Configure* automatic cleanup of events, episodes, full frames, and other old data.
2. Manually *purge* old data.
3. Regularly *back up* the database.
4. *Harness* the FindFace Multi comprehensive and searchable audit logs to enhance your system protection.

3.1.10 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*.

See also:

- *User Management and System Security*
- *Video Sources*
- *Record Index*
- *Face, Body, Vehicle Counters. Distance Measurement*
- *Face, Body, Vehicle Clusters*
- *People-Related Analytics*
- *Maintenance and Troubleshooting*

3.2 Web Interface Basics

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.

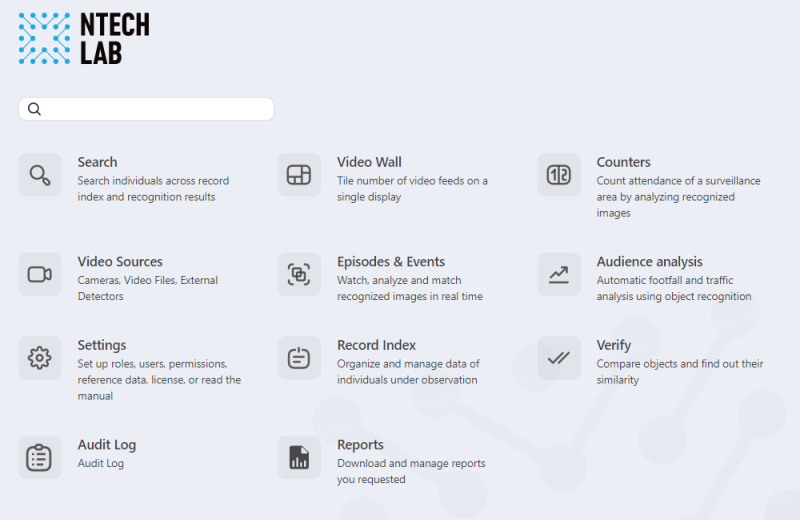
Note: The basic address is set during *deployment*.

Important: To log in for the first time, use the admin account created during *deployment*. To create more users, refer to *Role and User Management*.

Tip: Take your system security up a notch with *face-based authentication*.

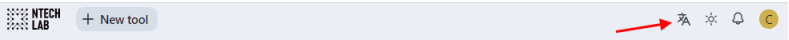
3.2.1 Navigation

There are a different number of items in the navigation bar depending on the user's role.



3.2.2 Web Interface Language and Theme

To change the system language and a theme click on a top panel.



3.3 Record Index

Record index stores records of individuals, including their photo, biometric data, or related documents and vehicle records, including their photo, license plate number, or related documents.

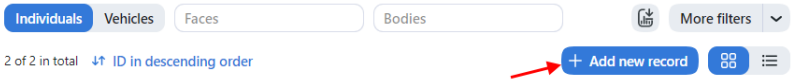
To create records in bulk, use the *console bulk record upload* functionality.

In this section:

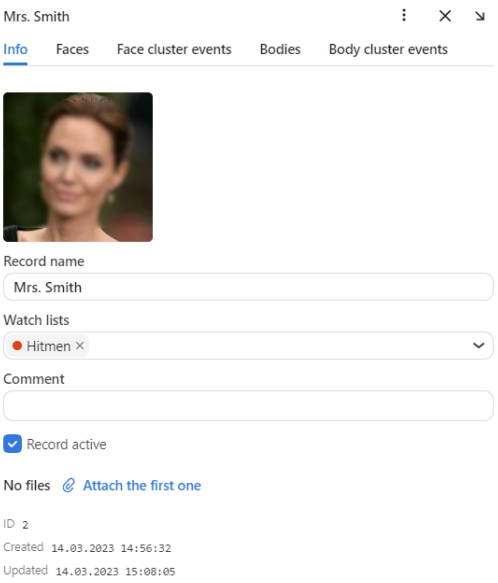
3.3.1 Create Record

To create a record manually, do the following:

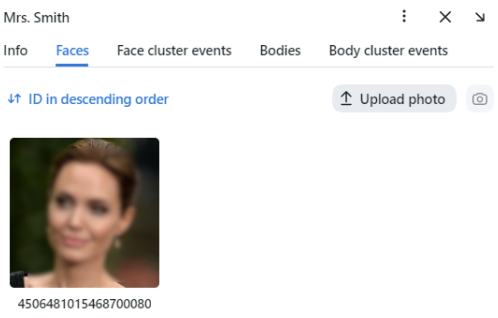
1. Navigate to the *Record Index* tab.
2. Click + *Add new record*.



3. Specify the record name. If necessary, add a comment.
4. From the *Watch lists* drop-down menu, select a watch list for the record (or several watch lists, one by one).
5. Check *Record active*. If a record is inactive, it is excluded from monitoring.
6. Save the record. You will see additional tabs appear.
7. On the same tab *Info*, attach related files.



8. For the individual records on the *Faces* tab attach images of the individual's face and on the *Bodies* tab attach images of the individual's body. Supported formats: WEBP, JPG, BMP, PNG.

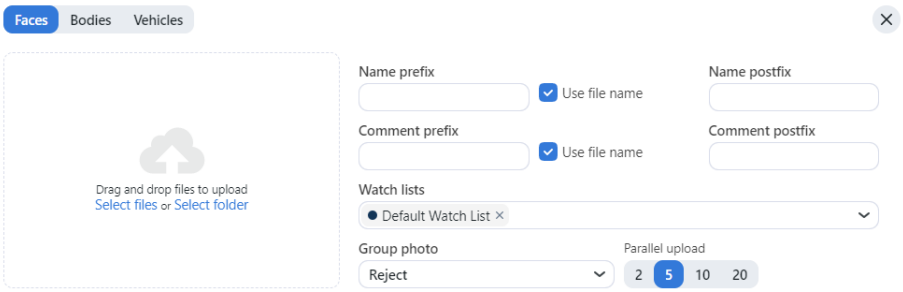


9. For the vehicle records on the *Vehicles* tab attach corresponding images.

3.3.2 Batch Record Upload

If there are too many records to create, you can bulk load records into the record index. Do the following:

1. Navigate to the *Record Index* tab.
2. Click + *Add new record* -> *batch record upload*.
3. Select files or folder to drop and drag to upload.



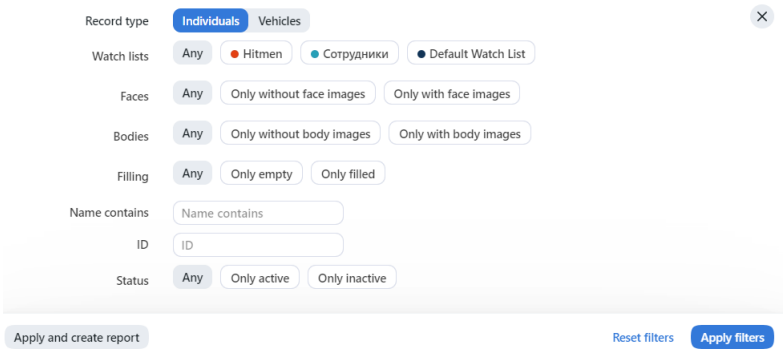
4. Specify the name. If necessary, add a comment.
5. Click *Start*.

3.3.3 Filter Records

The most frequently used filters for the record index are available in the upper part of the window.

To display the entire set of filters, click the *More filters* button. Here it is:

- *Record type*: display records by given type (individuals or vehicles).
- *Watch lists*: display records from selected watch lists.
- *Faces*: filter records by presence of a face biometric data.
- *Bodies*: filter records by presence of a bodies biometric data.
- *Vehicles*: filter records by presence of a vehicle data.
- *License plate number*: filter vehicle records by license plate number.
- *Filling*: display only empty records, only filled or any records.
- *Name contains*: filter records by name.
- *ID*: display a record with a given ID.
- *Status*: filter records by status.

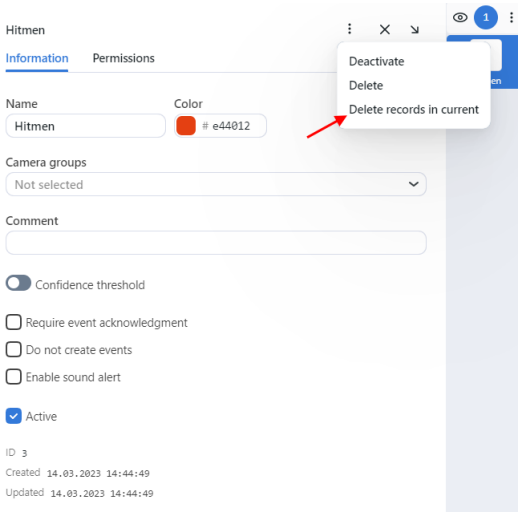


You can sort out records on the list by ID.

3.3.4 Purge Record Index

You can purge the record index entirely or by watch lists in one click. Do the following:

1. Navigate *Settings -> Watch Lists*.
2. Select one or several watch lists.
3. Click *Delete records in current*.



3.4 Video Sources

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 *Role and User Management*).

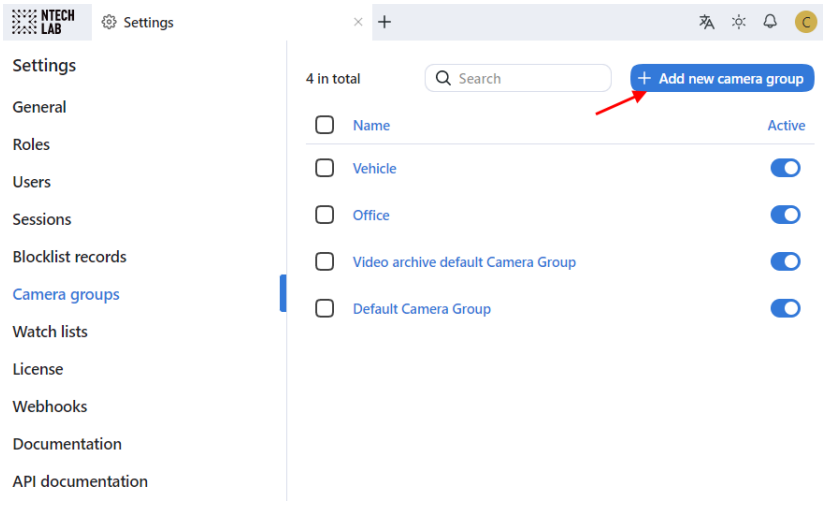
In this chapter:

3.4.1 Create Camera Group

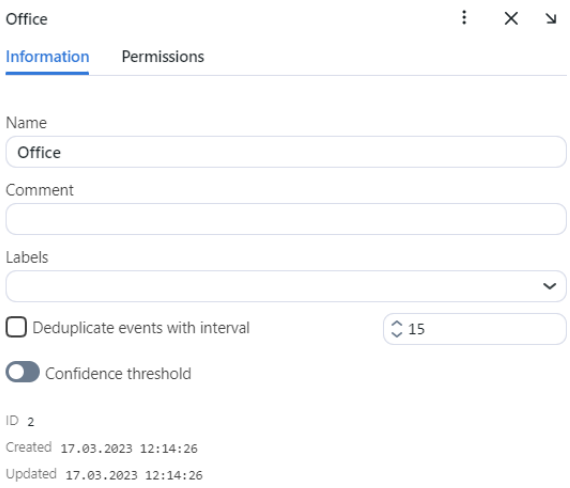
Tip: A default preconfigured camera group is available in the system.

To create a group of cameras, do the following:

1. Navigate to the *Settings* tab. Click *Camera Groups*.
2. Click + *Add new camera group*.



3. On the *Information* tab, specify the group name. Add a comment if needed.



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 `findface-video-worker.yaml` configuration file. See *Allocate findface-video-worker to Camera Group* for details.

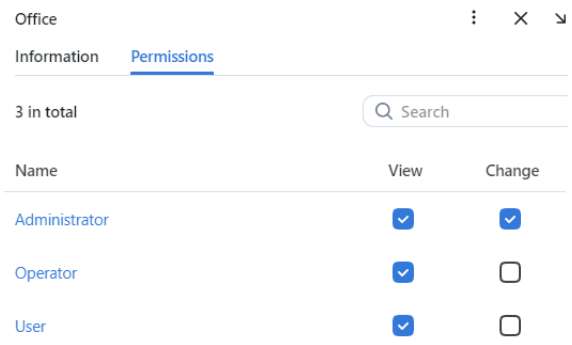
5. If you want to deduplicate events from cameras that belong to the same group, i.e. exclude coinciding events, check *Deduplicate events with interval* and specify the deduplication interval (interval between 2 consecutive checks for event uniqueness).

   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 prior (support@ntechlab.com).

7. Click *Save*.
8. Check *Activate*.
9. On the *Permissions* tab, assign privileges on the camera group, specifying which user roles are allowed to change/view the camera group settings.

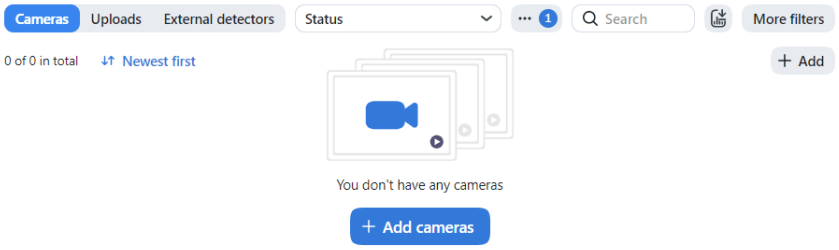


10. Click *Save*.

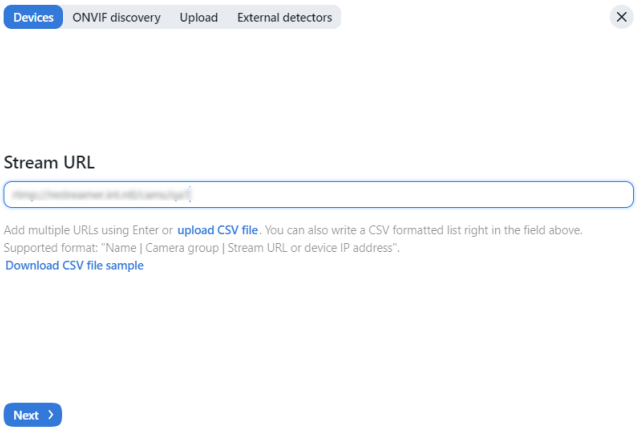
3.4.2 Add Camera

To add a camera, do the following:

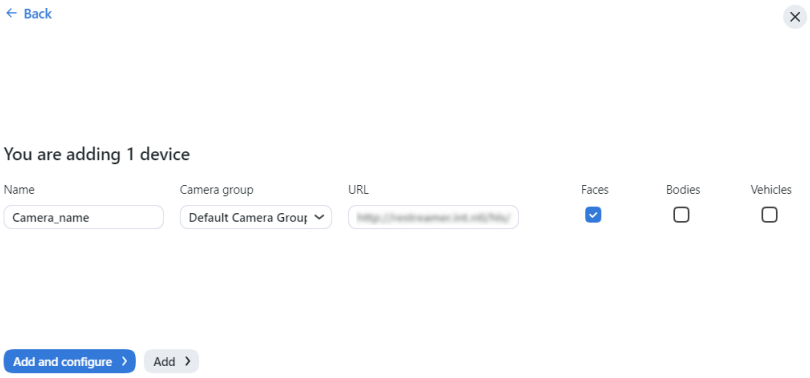
1. Navigate to the *Video Sources* -> *Cameras*.
2. Click + *Add*.



3. On the *Devices* tab, enter the stream URL and click *Next*.

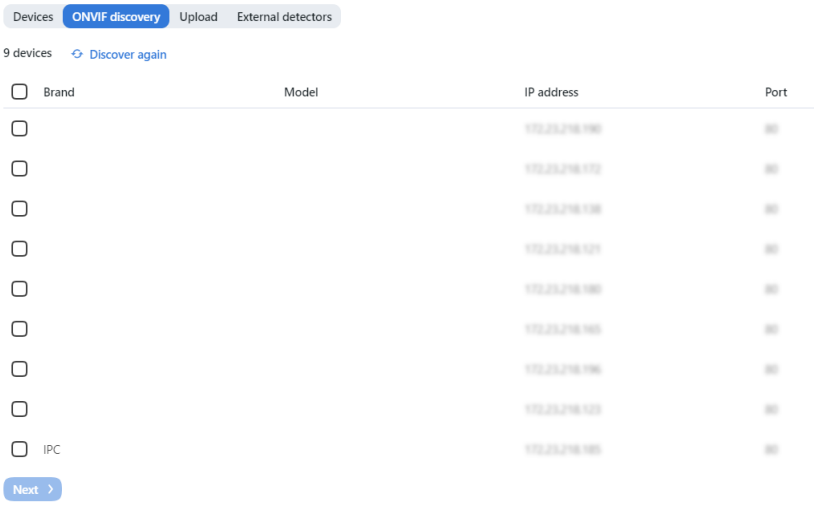


- Enter the camera name.
- Add the camera to a camera group.
- Check detectors that you want to enable on this camera: faces, bodies, vehicles.



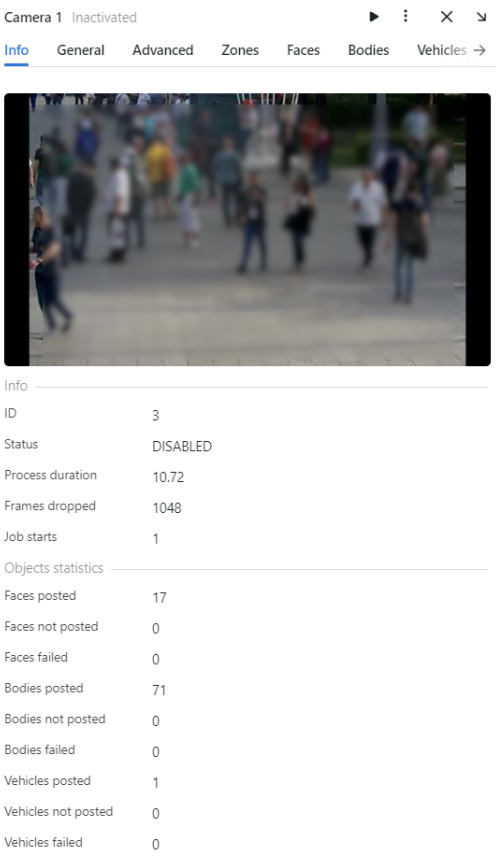
- Click *Add and configure*. You will see additional tabs appear.

4. Check the camera on the list to open the processing configuration wizard. Set up the *video processing parameters*.
5. If the camera is ONVIF, select it from the list of detected devices to automatically load available settings and streams.

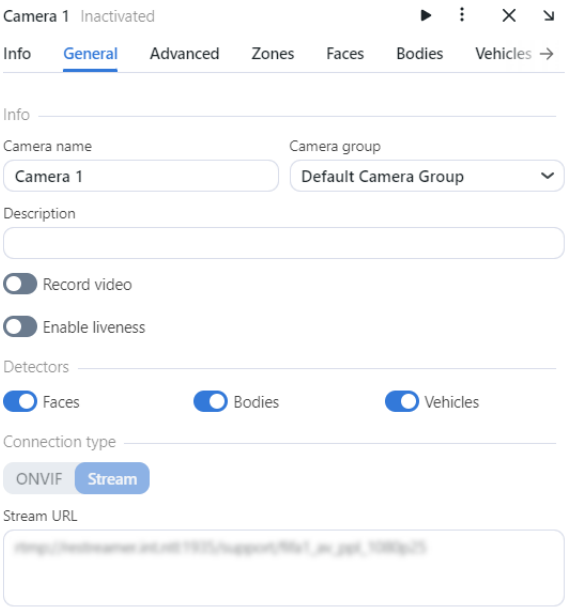


3.4.3 Video processing parameters

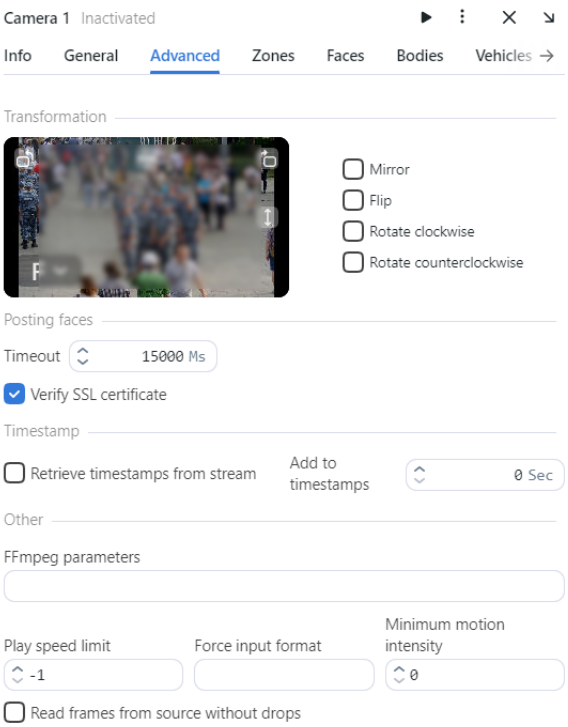
1. For each camera, you will be provided with complete statistics such as current session duration, the number of successfully posted objects, the number of objects processed with errors after the last job restart, the number of frame drops, and other data. To consult these data, click the camera and go to the *Info* tab.



2. On the *General* tab, you can change Camera Name, Camera group add Description. You can record video and enable liveness. Check detectors that you want to enable on this camera: faces, bodies, vehicles.



3. On the *Advanced* tab, fine-tune the camera:

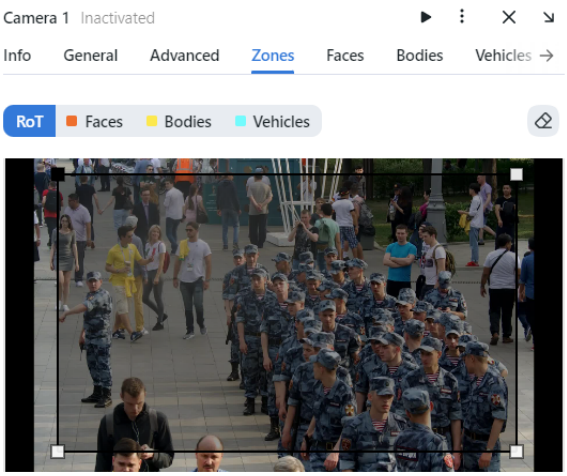


- If needed, change the video orientation.

Important: Be aware that the `findface-multi-legacy` server rotates the video using post-processing tools. It can negatively affect performance. Rotate the video via the camera functionality wherever possible.

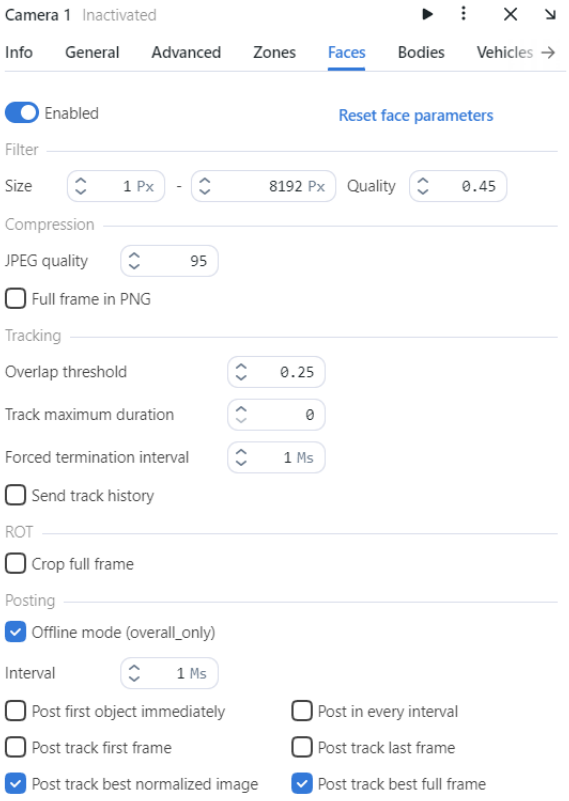
- *Timeout*: Specify the timeout in milliseconds for posting detected objects.
- *Verify SSL certificate*: Check to enable verification of the server SSL certificate when the object tracker posts objects to the server over https. Uncheck the option if you use a self-signed certificate.
- *Retrieve timestamps 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 parameters*: FFMPEG options for the video stream 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`. Not applicable for live-streaming.
- *Force input format*: Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.
- *Minimum motion intensity*: Minimum motion intensity to be detected by the motion detector.
- *Read frames from source without groups*: By default (false), 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 (true) `findface-video-worker` puts excessive frames on a waiting list to process them later. Set a variable to true when processing files as cameras.

4. Specify the region of tracking within the camera field and region of interest (*Zones*). Click *Save*.



The region of tracking enables detecting and tracking faces/bodies and vehicles only inside a clipping rectangle. You can use this option to reduce the video object detector load. The region of interest enables posting objects detected only within its boundaries.

5. On the *Faces*, *Bodies* and *Vehicles* tabs, specify settings for each object type detector.



- *Size*: Minimum object size in pixels to post and maximum object size in pixels to post.
- *Quality*: The minimum quality of the face image for detection. The allowed range is from 0 to 1. The recommended reference value is 0.45, which corresponds to object images of satisfying quality. Do not

change the default value without consulting with our technical experts (support@ntechlab.com).

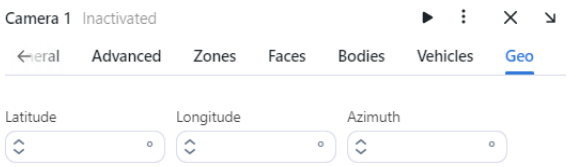
- *JPEG quality*: Full frame compression quality.
- *Full frame in PNG*: Send full frames in PNG and not in JPEG as set by default. Do not enable this parameter without supervision from our team as it can affect the entire system functioning.
- *Overlap threshold*: The percentage of overlap of bboxes between two serial frames so that these bboxes are considered as one track. The range of values is from 0 to 1. Do not change the default value without consulting with our technical experts (support@ntechlab.com).
- *Track maximum duration*: 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 faces from advertising media.
- *Forced termination interval*: Terminate the track if no new object has been detected within the specified time (in seconds).
- *Send track history*: Send array of bbox coordinates along with the event. May be applicable for external integrations to map the path of an object.
- *Crop full frame*: Check to crop the full frame to the size of the ROT area before sending it for recognition. The size of the full frame will be equal to the size of ROT area.
- *Offline mode (overall_only)*: By default, the system uses the offline mode to process the video, i.e., it posts one snapshot of the best quality per track to save disk space. Disable it to receive more face snapshots if needed. If the offline mode is on, the parameters of the real-time mode are off.

Real-time mode parameters:

Note: These parameters are non-functional if the offline mode is on.

- *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 `realtime_post_interval` to complete in real-time mode. Uncheck the option to post the first object after the first `realtime_post_interval` completes.
- *Post track first frame*: At the end of the track, the first frame of the track will be additionally sent complimentary to the overall frame of the track. May be applicable for external integrations.
- *Post track best normalized image*: Send best normalized images for detected objects.
- *Post in every interval*: Check to post the best snapshot within each time interval (`realtime_post_interval`) in real-time mode. Uncheck the option to post the best snapshot only if its quality has improved comparing to the posted snapshot.
- *Post track last frame*: At the end of the track, the last frame of the track will be additionally sent complimentary to the overall frame of the track. May be applicable for external integrations.
- *Post track best full frame*: Send best full frames of detected objects.

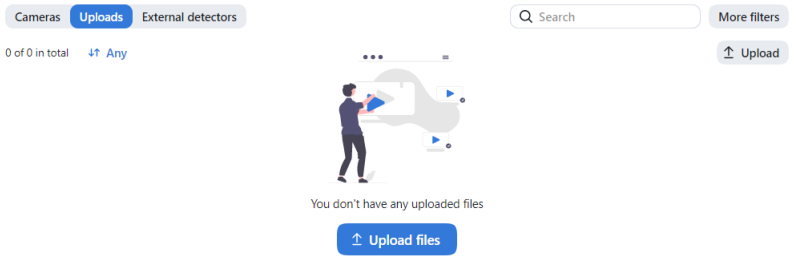
6. (Optional) On the *GEO* tab, specify the camera geographical location.



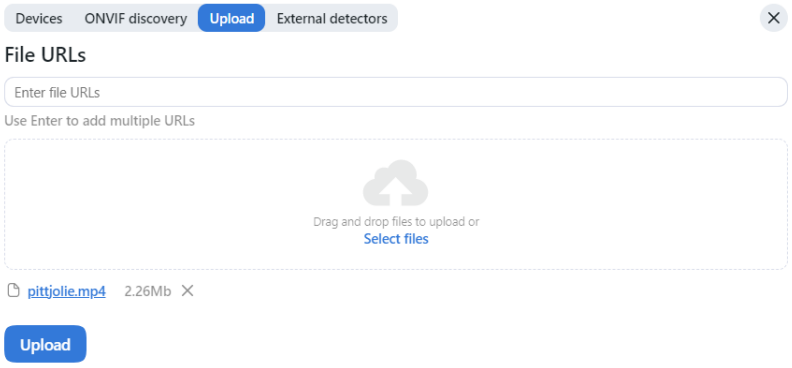
3.4.4 Upload and Process Video File

To upload and process a video, do the following:

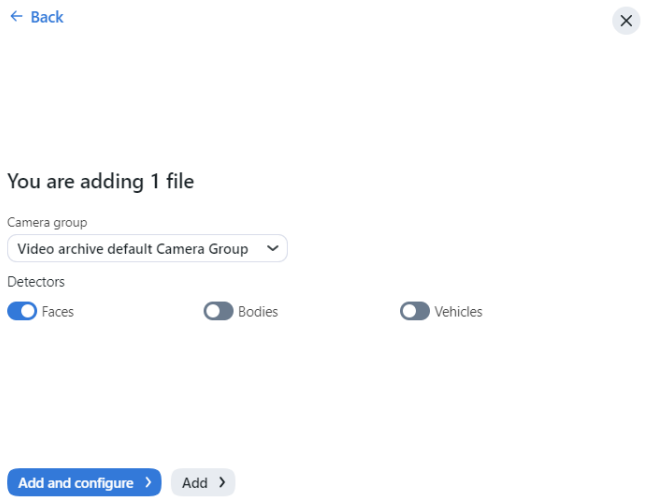
1. Navigate to the *Video Sources* -> *Uploads*.
2. Click *Upload*.



3. Specify a URL to the online storage, or select a file. Click *Upload*.

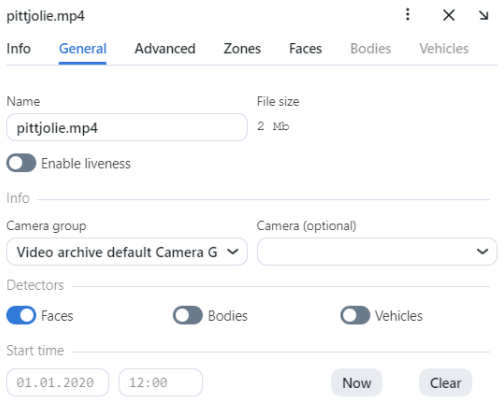


4. Designate a camera group to which the system will attribute the object recognition events from the video. The `Video archive default Camera Group` is perfect for this task. You can also create a *new camera group* with basic settings specifically for this video file. Check detectors that you want to enable on this video: faces, bodies, vehicles.



Click *Add and configure*. The video will be uploaded and shown in the source list.

5. Click the video on the list to open the processing configuration wizard. Specify parameters of *video processing* in the same way as for cameras.

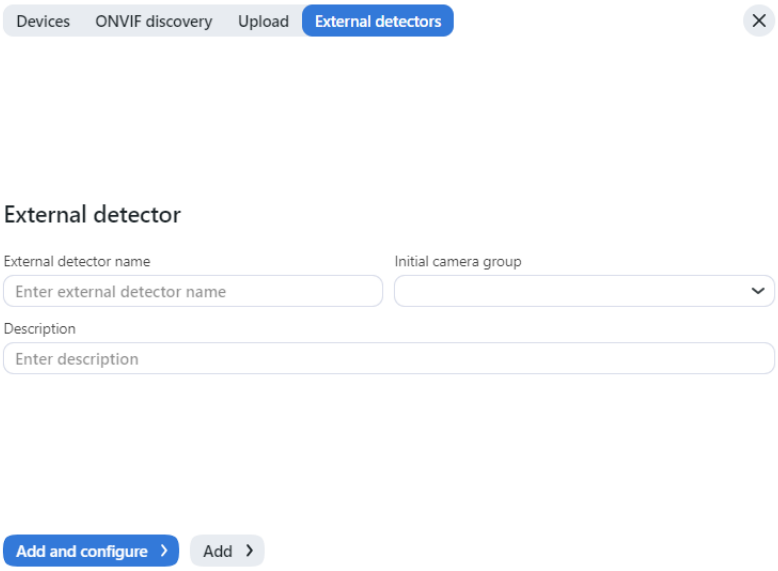


   - (Optional) Select a camera within that camera group to tag the object recognition events from this video more precisely.
   - (Optional) Configure the timestamps for object recognition events.

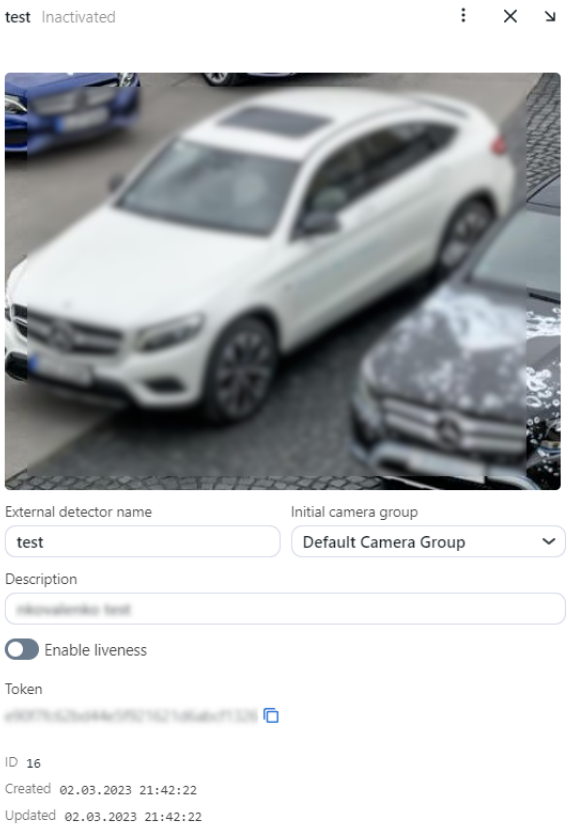
6. Click three dots -> *Process current* to start object identification. You can view object identification events on the *Events* and *Episodes* tabs by filtering the list of events by the camera group/camera associated with the video.

3.4.5 External Detector

A camera object can also be used for integrating an *External Detector*.

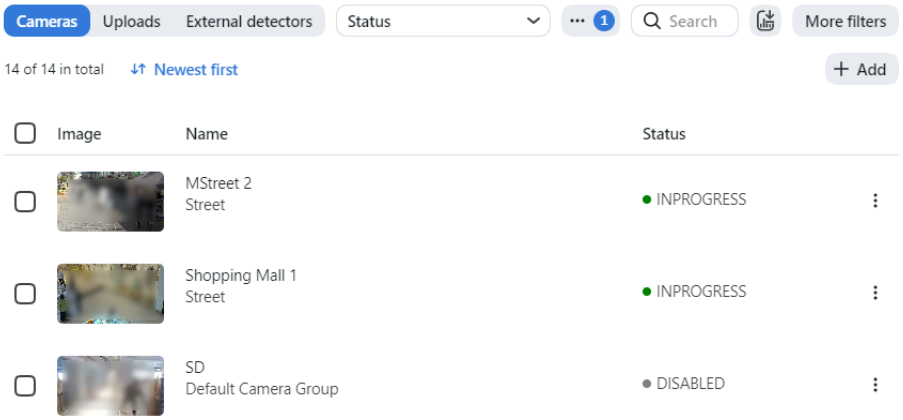


If necessary, change the External detector name. Specify the Initial camera group and Description. Check to enable liveness.



3.4.6 Monitor Camera Operation

To monitor the operation of cameras, navigate to the *Cameras* tab.



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 `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file:

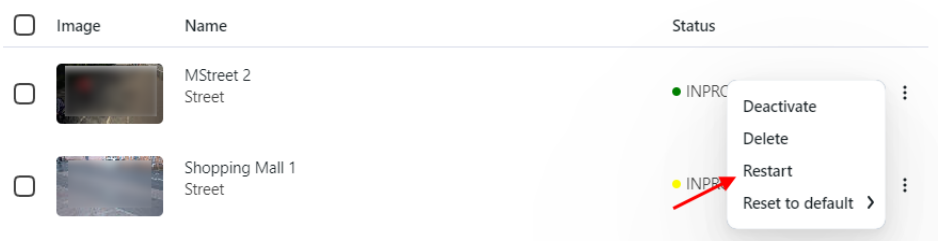
```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

FFSECURITY = {
   ...

   # max camera frames_dropped percent
   'MAX_CAMERA_DROPPED_FRAMES': {'yellow': 0.1, 'red': 0.3},
   # max camera objects failed percent
   'MAX_CAMERA_FAILED_OBJECTS': {'yellow': 0.1, 'red': 0.3},

   ...
   }
```

Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

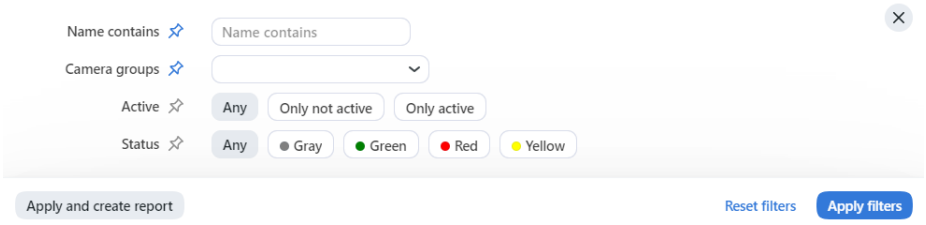
Each created camera is associated with a so called job, a video processing task that contains configuration settings and stream data and is assigned to `findface-video-worker`. This task can be restarted.

To restart a job, click *Restart*. In this case, the number of errors will be reset to `0`.



If there are a large number of cameras in the system, use the following filters:

- *Name contains*,
- *Camera groups*,
- *Active*,
- *Status*.



3.5 Events and Episodes of Object Recognition

To monitor the real-time object identification in live videos, navigate to the *Episodes & Events* and use the *Episodes* and *Events* tabs. Besides monitoring, both tabs allow you to access the history of identification events.

Tip: To perform the object identification in archived videos, see *Upload and Process Video File*.

3.5.1 Work with Events

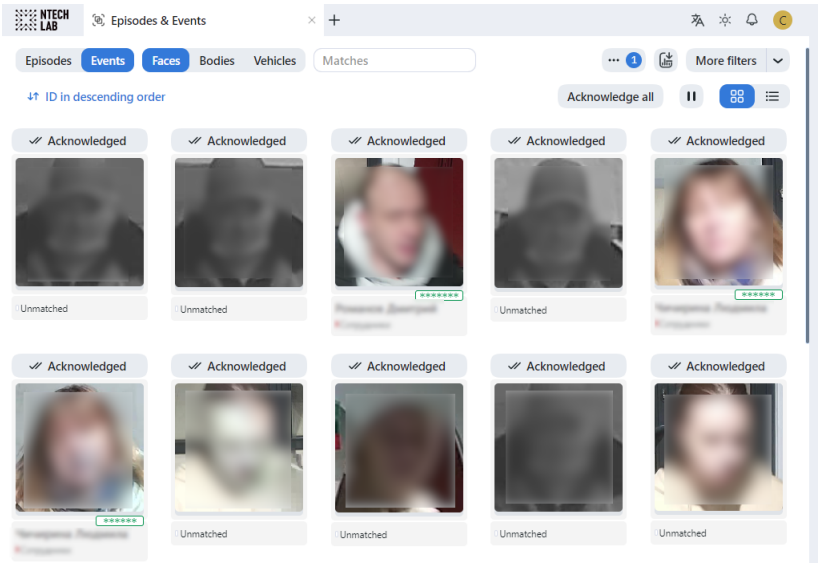
This section is about the *Events* tab.

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:

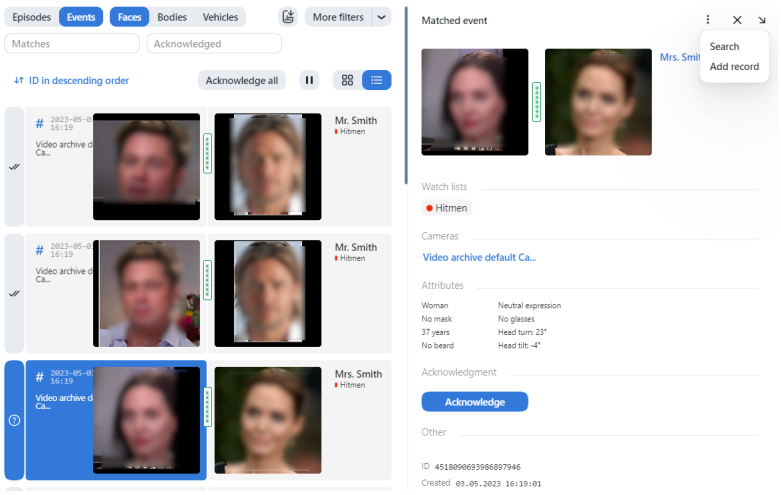
View Events

Once an object is detected, you will see a notification on one of the event lists: *Faces*, *Bodies*, or *Vehicles*, subject to the object type.



A notification can feature different pieces of information, depending on whether a detected object has a match in the card index:

- Match not found: the normalized object image, detection date and time, camera group name, watch lists.
- Match found: the normalized object image, reference object photo from the related card, name, the similarity between matched objects, comment from the card, watch lists, detection date and time, camera group, attributes.



Important: In order to pause the notification 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.

- *Object*: display events only for faces, bodies or vehicles.
- *Matches*: display events only with/without matches, or all events.
- *Watch lists*: display events only for a selected watch list.
- *Acknowledged*: display only acknowledged/unacknowledged events, or all events.
- *Camera groups*: display only events from a selected group of cameras.
- *Cameras*: display only events from a selected camera.
- *Date and time*: display only events that occurred within a certain period.
- *Video Archive ID*: display events from the video archive with a given ID.
- *Record name*: display events with a given record name.
- *Episode ID*: display events from the episode with a given ID.
- *Event ID*: display an event with a given ID.
- *Event's best shot*: display all events of a track, only events with real-time snapshots, only one event with the best snapshot at the end of a track.

Specific filters for faces

- *Age*: display events with people of a given age.
- *Beard*: filter events by the fact of having a beard.
- *Emotions*: display events with given emotions.
- *Gender*: display events with people of a given gender.
- *Glasses*: filter events by the fact of wearing glasses.
- *Liveness*: filter events by face liveness.
- *Face mask*: filter events by the fact of wearing a face mask.
- *Head turn*: filter events by degree of head turn.
- *Head tilt*: filter events by degree of head tilt.

Specific filters for bodies

- *Gender by body*: display only events with people of a given gender or all events.
- *Age by body*: display only events with people of a given age.
- *Headwear*: display only events with a person wearing headgear of a given type: hat/cap, hood/headscarf, none.
- *Vest*: display only events with a person wearing a vest of a given color.
- *Vest score*: display only events with a person wearing a vest within a given score.
- *Helmet*: display only events with a person wearing a helmet of a given color.
- *Helmet score*: display only events with a person wearing a helmet within a given score.
- *Upper clothes color*: display only events with a person wearing a top of a given color.

- *Lower clothes color*: display only events with a person wearing a bottom of a given color.
- *Upper clothes type*: display only events with a person wearing upper body wear of a given specific type: jacket, coat, sleeveless vest, sweatshirt, T-shirt, shirt, dress.
- *Lower body clothes*: display only events with a person wearing lower body wear of a given type: pants, skirt, shorts, obscured.
- *Upper body clothes*: display only events with a person wearing upper body wear of a given generalized category: long sleeves, short sleeves, no sleeve.
- *Bag on the back*: display only events with a person wearing/not wearing a bag on the back.
- *Bag in hand*: display only events with a person wearing/not wearing a bag in hand.

Specific filters for vehicles

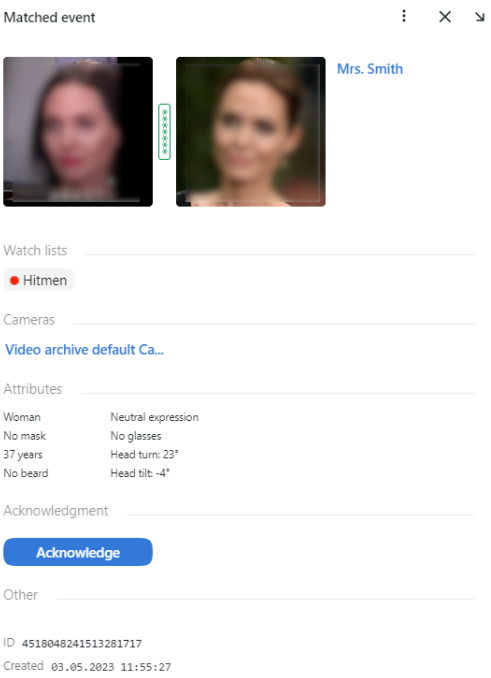
- *Make*: filter vehicle events by vehicle make.
- *Model*: filter vehicle events by vehicle model.
- *Vehicle body type*: display only events with vehicles of a given body type: minivan, limousine.
- *Vehicle body color*: display only events with vehicles of a given color.
- *Country*: display only events with vehicles registered in a given country.
- *License plate number*: display an event with a given plate number.
- *Region*: display only events with vehicles registered in a given region.
- *License plate color*: display only events with a given license plate color.
- *Special vehicle*: display only events with vehicles belonging to a given type: police, fire service and EMERCOM vehicles, gas rescue and emergency services, military, municipal vehicles, and others.
- *Vehicle category*: display only events with vehicles belonging to a given category: motorcycle, scooter, car, car with a trailer, truck, truck with a trailer, bus, articulated bus, and others.
- *Vehicle weight and body size*: display only events with vehicles of a given weight and body size.
- *Vehicle orientation*: display only events with vehicles of a given orientation.

Note: License plate region and color are predicted for the United Arab Emirates (UAE) only. The values of these attributes will be marked as unknown for other countries.

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.

An event ticket contains the same data as a relevant *notification*. It also allows for acknowledging the event. To do so, click *Acknowledge* to change the event acknowledgment status to *Acknowledged*.



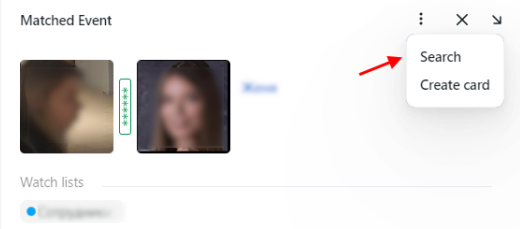
Tip: If a detected object has a match in the card index, you can navigate to the relevant card by clicking on the name in the event ticket.

Tip: In order to acknowledge all events, click *Acknowledge all* 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. To navigate from an event ticket to the search tab, click *Search*.



See also:

- *Search Objects in System*

Work Time Reports on Events

See *Reports*.

3.5.2 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 objects of the same origin (face and body images of the same person and images of the same vehicle) detected within a specific period. 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 objects' similarity and detection time. This allows for easy processing of diverse events, even in large numbers.

In this chapter:

About Episodes

There are two stages of an episode lifecycle:

- LIVE: an episode is currently active, to which new events can be possibly added.
- Closed: an episode is closed, no events can be added.

Episodes are classified as individual episodes and vehicle episodes. Individual episodes feature face and body images of people, detected within a specific period, while vehicle episodes contain vehicle images and license plate.

Grant Rights for Episodes

A user receives a notification of a new episode if they have rights to the first event. Viewing new events in the episode also requires proper rights.

The right to an event consists of the rights for a corresponding camera and watch list.

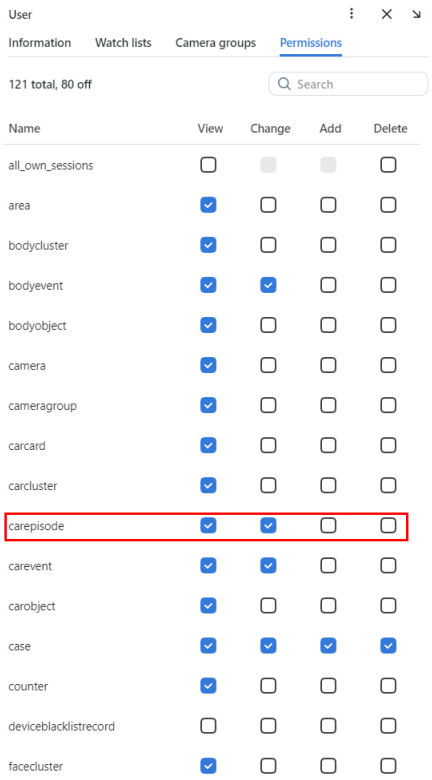
Note: To see unmatched events, you only need the rights to a camera.

To manage the rights of a role for the entire `Episode` entity, open permissions for this role and adjust the following permissions:

- `humanepisode`: individuals episodes

- `carepisode`: vehicle episodes

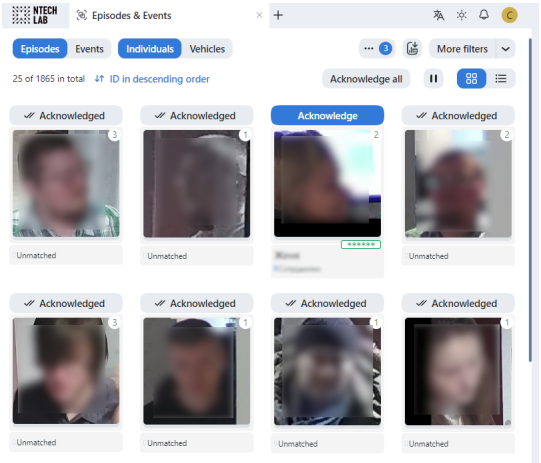
Tip: See *Role and User Management*.



View Episodes

You can find the list of episodes with filters and statistics on the *Episodes & Events* -> *Episodes* tab. Episodes are sorted subject to the object's origin: there are individual episodes and vehicle episodes.

Once an object is detected, it is either added to an existing LIVE episode or used as a starting point for a new episode. Each episode is assigned an identifier which can be later used to filter events and episodes.



When working with the episodes, the following default filters may come in handy:

Note: Some filters from the list below may be hidden, subject to enabled recognition features.

- *Object*: display events only for individuals or vehicles.
- *Matches: faces*: display episodes only with/without face matches, or all episodes.
- *Matches: bodies*: display episodes only with/without body matches, or all episodes.
- *Matches*: display episodes only with/without vehicle matches, or all episodes.
- *Watch lists*: display episodes only for a selected watch list.
- *Acknowledged*: display only acknowledged/unacknowledged episodes, or all episodes.
- *Camera groups*: display only episodes from a selected group of cameras.
- *Cameras*: display only episodes from a selected camera.
- *Date and time*: display only episodes that occurred within a certain period.
- *Video Archive ID*: display episodes related to the video archive with a given ID.
- *Record name*: display only episodes with a given record name.
- *Episode ID*: display an episode with a given ID.
- *Count events*: display only episodes with a given number of events.

Specific filters for faces

- *Age*: display episodes with people of a given age.
- *Beard*: filter episodes by the fact of having a beard.
- *Emotions*: display episodes with given emotions.
- *Gender*: display episodes with people of a given gender.
- *Glasses*: filter episodes by the fact of wearing glasses.
- *Liveness*: filter episodes by face liveness.
- *Face mask*: filter episodes by the fact of wearing a face mask.

Specific filters for bodies

- *Gender by body*: display episodes with a body of a given gender.
- *Age by body*: display episodes with a body of a given age.
- *Headwear*: display only episodes with a person wearing headgear of a given type: hat/cap, hood/scarf, none.
- *Vest*: display only episodes with a person wearing a vest of a given color.
- *Vest score*: display only episodes with a person wearing a vest within a given score.
- *Helmet*: display only episodes with a person wearing a helmet of a given color.
- *Helmet score*: display only episodes with a person wearing a helmet within a given score.
- *Upper clothes color*: display only episodes with a person wearing a top of a given color.

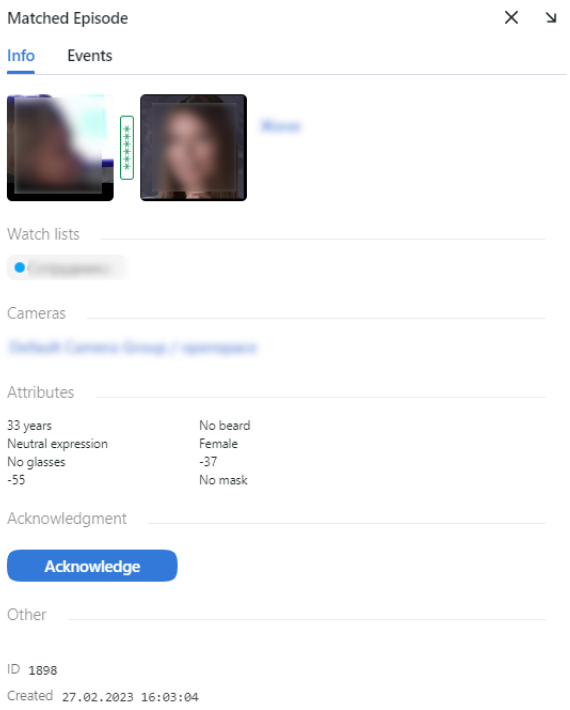
- *Lower clothes color*: display only episodes with a person wearing a bottom of a given color.
- *Upper clothes type*: display only episodes with a person wearing upper body wear of a given specific type: jacket, coat, sleeveless, sweatshirt, T-shirt, shirt, dress.
- *Lower body clothes*: display only episodes with a person wearing lower body wear of a given type: pants, non-descript, skirt, shorts.
- *Upper body clothes*: display only episodes with a person wearing upper body wear of a given generalized category: long sleeves, short sleeves, no sleeve.
- *Bag on the back*: display only episodes with a person wearing/not wearing a bag on the back.
- *Bag in hand*: display only episodes with a person wearing/not wearing a bag in hand.

Specific filters for vehicle episodes

- *Make*: filter vehicle episodes by vehicle make.
- *Model*: filter vehicle episodes by vehicle model.
- *Vehicle body type*: display only episodes with vehicles of a given body type: minivan, limousine.
- *Vehicle body color*: display only episodes with vehicles of a given color.
- *Country*: display only episodes with vehicles registered in a given country.
- *License plate number*: display an episode with a given plate number.
- *Region*: display only episodes with vehicles registered in a given region.
- *License plate color*: display only episodes with a given license plate color.
- *Special vehicle*: display only episodes with vehicles belonging to a given type: taxi, public road transport, car-sharing, ambulance, police, fire service and emergencies ministry vehicles, gas rescue and emergency services, military, municipal vehicles, other or not special.
- *Vehicle category*: display only episodes with vehicles belonging to a given category: motorcycle, scooter, car, car with a trailer, truck, truck with a trailer, bus, articulated bus, other.
- *Vehicle weight and body size*: display only episodes with vehicles of a given weight and body size.

Note: License plate region and color are predicted for the United Arab Emirates (UAE) only. The values of these attributes will be marked as unknown for other countries.

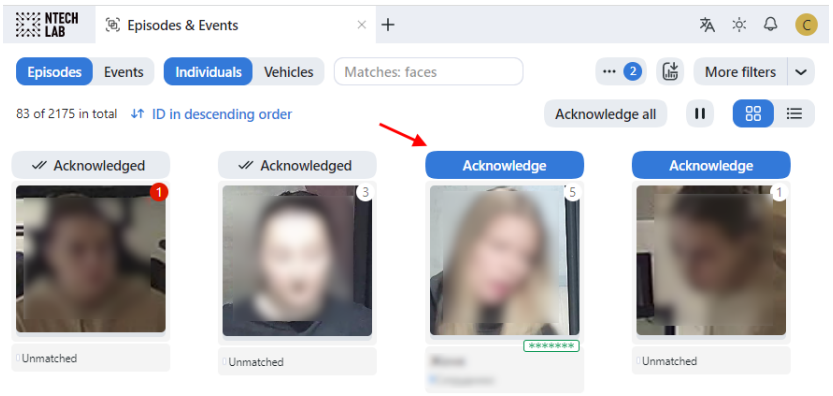
To view episode events, click the episode on the list. You will be redirected to the *Matched Episode* or *Unmatched Episode* tab with the episode information and related events.



Work with the *Events* tab as described in *Work with Events*.

Event and Episode Acknowledging

To acknowledge an entire episode, click *Acknowledge* for this episode or 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.

Episode Settings

To configure the episodes, use the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. Find the following parameters in the `FFSECURITY` section:

- `FACE_EPISODES_THRESHOLD`, `BODY_EPISODES_THRESHOLD`, `CAR_EPISODES_THRESHOLD`: Similarity thresholds that the system are using when searching for recent events in order to construct an episode. The default values are set to the optimum. If necessary, you can change them. Be sure to consult with our technical experts prior (support@ntechlab.com).
- `FACE_EPISODE_SEARCH_INTERVAL`, `BODY_EPISODE_SEARCH_INTERVAL`, `CAR_EPISODE_SEARCH_INTERVAL`: The period preceding an event, for which the system searches the feature vector database for already existing events with similar objects. If no such event is found, the system creates a new episode. Otherwise, it sorts out the 100 most recent similar objects and picks up the most relevant event from a LIVE episode.
- `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 object. Use this option to save disk space.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

...

FFSECURITY = {
      ...

      # episodes specific matching threshold that is used to join faces in an episode
      'FACE_EPISODES_THRESHOLD': 0.673,  # FAR = 1.00E-06  # model: [mango_320]
      'BODY_EPISODES_THRESHOLD': 0.75,  # model: [clio]
      'CAR_EPISODES_THRESHOLD': 0.61,  # model: [alonso]
      ...
      # when closing episode, delete all events except the best episode event
      'EPISODE_KEEP_ONLY_BEST_EVENT': False,
      # delete episode events after delay in seconds
      'EPISODE_DELETE_EVENTS_DELAY': 60,
      ...
      # maximum event age in seconds than could be added to an episode.
      'FACE_EPISODE_SEARCH_INTERVAL': 60,
      'BODY_EPISODE_SEARCH_INTERVAL': 60,
      'CAR_EPISODE_SEARCH_INTERVAL': 60,
      # maximum episode duration (episode is closed after)
      'EPISODE_MAX_DURATION': 300,
      # if no new event added to an episode during this timeout, episode will be closed.
      'EPISODE_EVENT_TIMEOUT': 30,
      ...
}

...
```

Restart the `findface-multi-findface-multi-legacy-1` container.

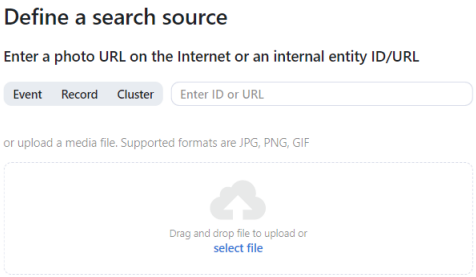
```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

3.6 Search Objects in System

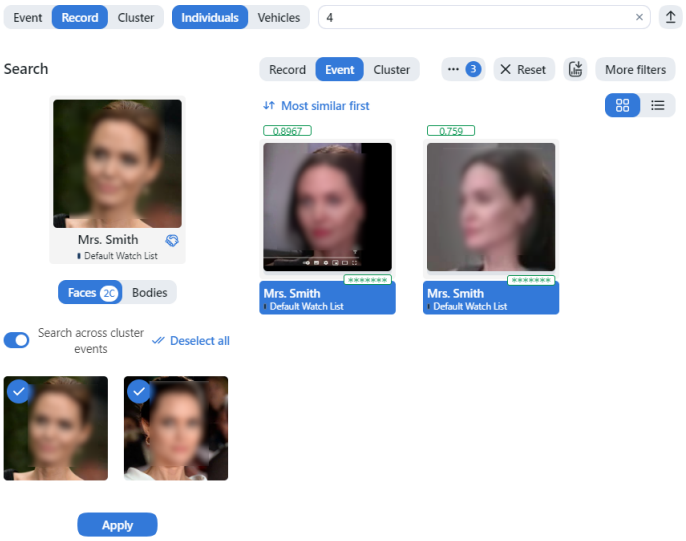
FindFace Multi allows you to search for individuals throughout the entire system.

To find an individual, do the following:

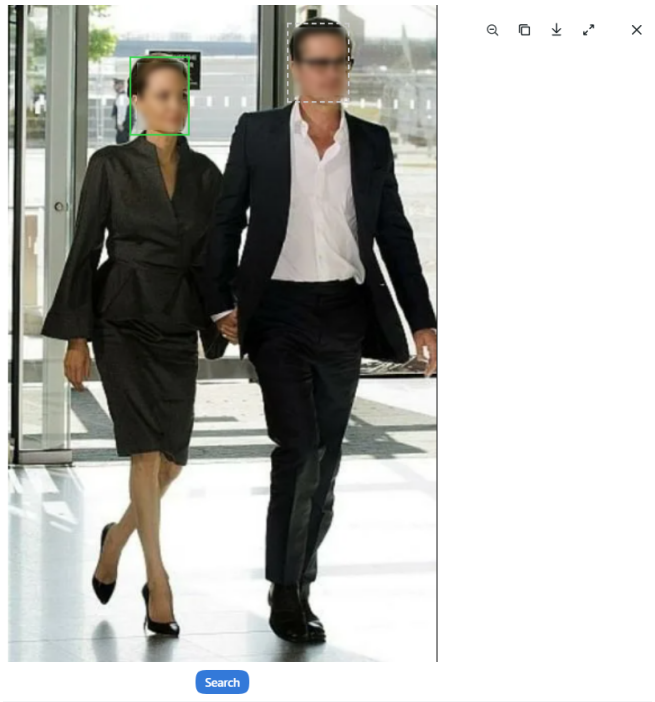
1. Navigate to the *Search* tab.
2. Specify an object to search for in one of the following ways:
   - by event's URL or ID
   - by record's URL or ID
   - by cluster's URL or ID
   - by uploading a photo



3. If you specified a record URL, select a photo from it. If there are multiple photos, you can select some or all of them. Click the *Apply* button.



4. If you uploaded a photo, it will be displayed in the new window. If there are multiple objects in the image, select the one of your interest. Click *Search*.

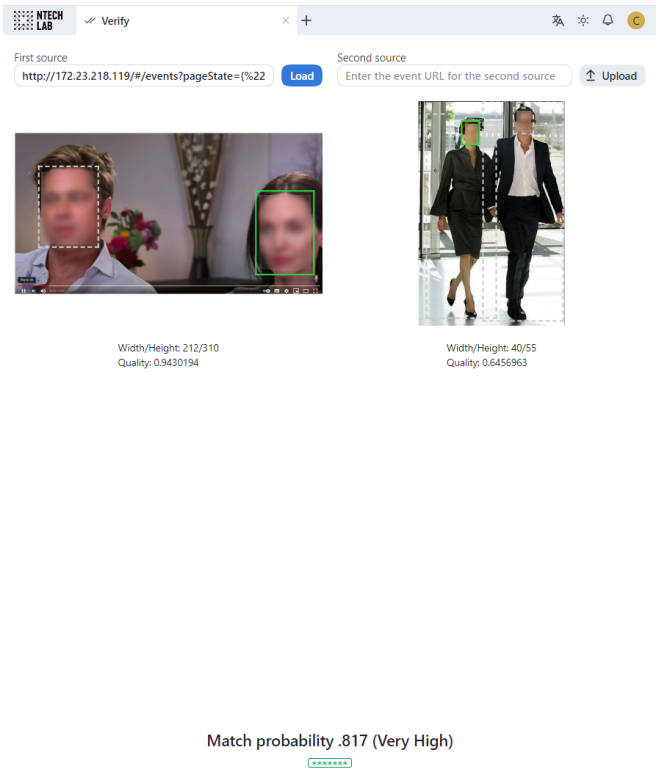


5. You will see the search results appear. If necessary, you can narrow down your search by specifying a watch list, similarity threshold, etc.

3.7 Verify Two Objects

FindFace Multi allows you to compare two objects and verify that they match. Do the following:

1. Navigate to the *Verify* tab.
2. Specify two objects to verify in one of the following ways:
   - by event's URL
   - by uploading a photo
3. If there are multiple objects in the image, select the one of your interest.



4. You will see the probability that the objects match.

3.8 Face, Body, Vehicle Counters. Distance Measurement

Important: To be able to count human bodies (silhouettes) or vehicles, you first have to enable *body detection* or *vehicle detection*.

FindFace Multi allows you to count faces, bodies and vehicles 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, traffic jam detection, and more.

The counting method is based on time slices, which means that the system counts faces, bodies and vehicles in static screenshots taken with a given `count interval`.

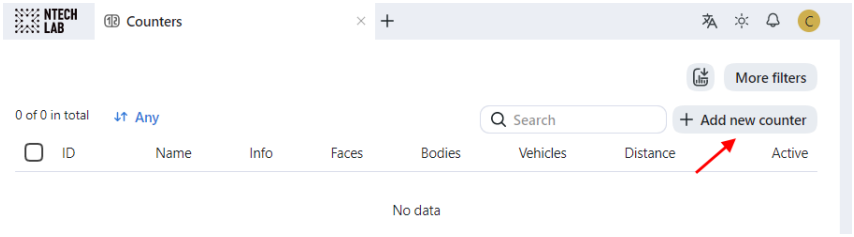
You can opt for counting faces/bodies/vehicles either on each camera independently, or collectively on all selected cameras.

In this section:

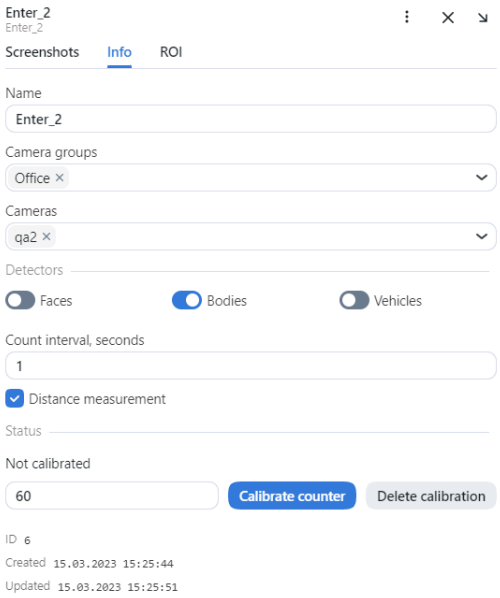
3.8.1 Create Counter

To set up a counter, do the following:

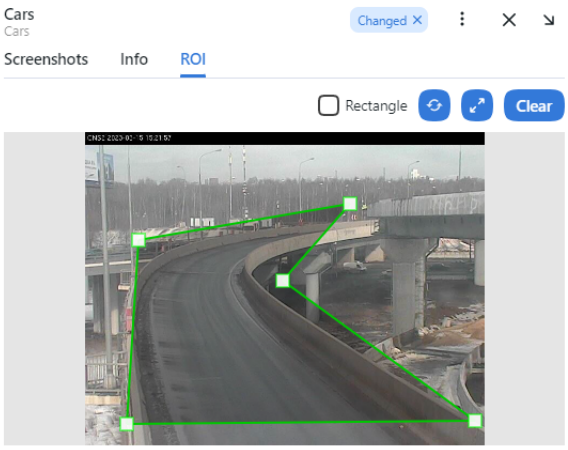
1. Navigate to the *Counters* tab.
2. Click + *Add new counter*.



3. Specify the counter name.
4. Select one or several camera groups for counting.
5. Select one or several cameras for counting.
6. Check *Faces* to count faces.
7. Check *Bodies* to count bodies. Body detection has to be *enabled*.
8. Check *Vehicles* to count vehicles. Vehicle detection has to be *enabled*.
9. Specify the interval in seconds between two consecutive screenshots used for counting.
10. Click *Save*.
11. Check *Distance measurement* to count the distance between bodies. This option becomes available only if *Bodies* detector is checked.



12. Make sure that the counter is activated.
13. Click *Save*.
14. (Optional) Navigate to the *ROI* tab to specify the face/body/vehicle tracking region within the camera(s) field. Select a camera and specify a region of interest. Click *Save*.



3.8.2 Calibrate Counter for Distance Measurement

If *Distance measurement* is enabled, you should perform the counter calibration.

For counter calibration:

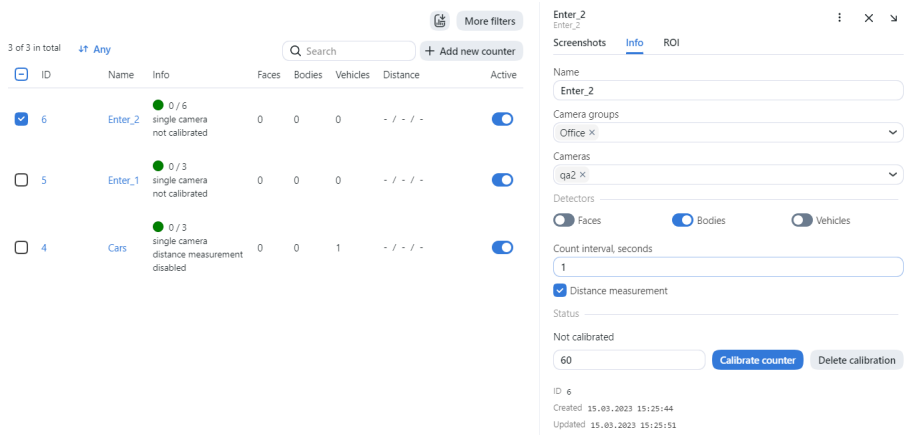
1. Ask a person to stand before the selected camera at full height.
2. Ask a person to walk through the area visible by the camera, along which the distances will be calculated. To achieve the best quality, the person must walk on a flat surface.
3. Enter the counter calibration timeframe (from 15s to 300s) and click *Calibrate counter*.

If you need to calibrate the counter for several cameras, perform steps 1-3 for each camera selected in the *Cameras* field.

If you are unsatisfied by the result, click *Delete calibration* and perform the calibration once more.

3.8.3 Monitor Counter Operation

To monitor the operation of counters, navigate to the *Counters* tab.



Counter statuses:

- Green: the counter is running without errors, or the number of errors doesn't pass the acceptable threshold.
- Yellow: the number of errors exceeds the threshold.
- Red: the number of errors is critical.
- Grey: the counter is disabled.

Tip: You can configure the yellow and red statuses based on the portion of failed counter records and change the time window duration between two consecutive checks of the counter health status. To do so, modify the following parameters in the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file:

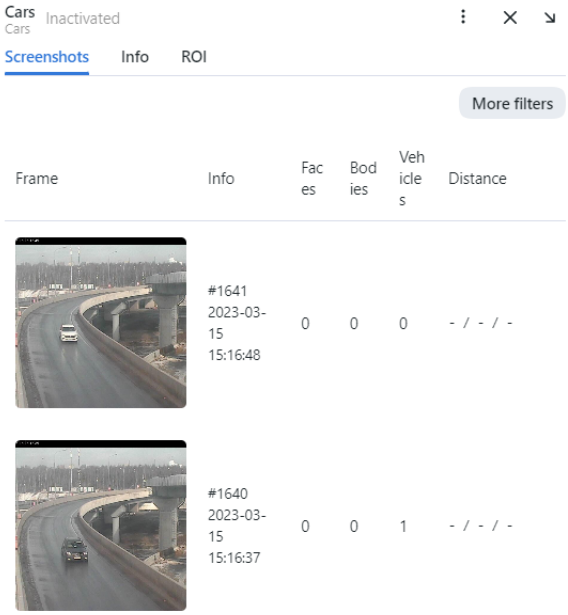
```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

FFSECURITY = {
    ...
    # Counter health status config:
    # max percent of camera records with errors
    'MAX_COUNTER_ERROR_RECORDS': {'yellow': 0.3, 'red': 0.5},
    # time window for computing health status (in seconds)
    'COUNTER_HEALTH_STATUS_TIME_WINDOW': 30,
    ...
}
```

3.8.4 Work with Counter Records

Static screenshots taken by a counter, with the number of faces, bodies and vehicles in them, are saved as counter entries. If you have enabled the distance measurement, each record will also contain a minimum, average, and maximum detected distance in meters.

If the counter is running with errors, the system will be creating blank records with an error message.

To see the counter records, navigate to the *Counters* tab. Click on the counter. Navigate to the *Screenshots* tab.



You can scale the screenshots by clicking on them. If distance measurement is enabled, the screenshot will contain boxes around bodies and the distance between them.

To work with the counter list, use the following filters:

- Name contains
- Counter ID
- Cameras
- Camera groups
- Distance

To work with counter screenshots, use the following filters:

- Cameras
- Date and time
- Count faces
- Count bodies
- Count vehicles
- Counter entry ID
- Minimum distance
- Maximum distance

- Average distance

3.8.5 Set Webhook for Counter

To take it up a notch, *configure a webhook* for counter records with a specific number of faces, bodies and vehicles.

See also:

- *Enable Body and Body Attribute Recognition*
- *Enable Vehicle and Vehicle Attribute Recognition*
- *Webhooks*

3.8.6 Configure Counters

To configure counters, open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file and modify the following parameters:

- `COUNTERS_DEDUP_OPTIONS`: a set of options that help avoid object duplication in multi-camera counters, i.e., the situation when the same object is counted on several cameras at the same time, leading to the overly increased counting result. Two options are available for each object type (`face`, `body`, `car`): `enabled` - enables object deduplication, `threshold` - defines the minimum level of similarity between objects for the system to consider them duplicates.
- `COUNTERS_SAVE_FULLFRAME` determines saving options of full frames in counters: `always`, `detect` - only save if faces, bodies or vehicles have been detected, `never`.
- `COUNTERS_FULLFRAME_JPEG_QUALITY`: JPEG quality of full frames.
- `COUNTERS_THUMBNAIL_JPEG_QUALITY`: JPEG quality of thumbnails.
- `COUNTERS_ROI_INTERSECTION_THRESHOLD`: required percentage of the rectangle around the object (a.k.a. `bbox`) intersection with the region of interest defined for the counter.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

# deduplicate objects on multi-camera counters
 'COUNTERS_DEDUP_OPTIONS': {
     'face': {
         'enabled': False,
         'threshold': 0.714,  # model: [mango_320]
     },
     'body': {
         'enabled': False,
         'threshold': 0.65,  # model: [clio]
     },
     'car': {
         'enabled': False,
         'threshold': 0.65,  # model: [alonso]
     },
},
# counters full frame saving options:
# `always` - save always
# `detect` - save only if faces or bodies have been detected
# `never` - never save full frames
```

(continues on next page)

(continued from previous page)

```
'COUNTERS_SAVE_FULLFRAME': 'always',
'COUNTERS_FULLFRAME_JPEG_QUALITY': 75,
'COUNTERS_THUMBNAIL_JPEG_QUALITY': 75,
# required percentage of bbox intersection with ROI
'COUNTERS_ROI_INTERSECTION_THRESHOLD': 0.75,
...
```

Be sure to restart the `findface-multi-findface-multi-legacy-1` container after making changes.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

3.9 Face, Body, Vehicle Clusters

FindFace Multi supports automatic clustering of objects of the same origin:

- Face images belonging to the same person form a face cluster.
- Body images belonging to the same person form a body cluster.
- Images of the same vehicle constitute a vehicle cluster.

Aggregated cluster galleries of faces, bodies, and vehicles are available on the *Clusters* tab.

Note: If a face cluster or a body cluster matches a *person record*, such a cluster will automatically appear in that record. Similarly, a vehicle cluster will be saved to the corresponding vehicle record.

Important: By default, the object clustering is disabled. *Enable and configure it* via the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

In this section:

3.9.1 Clustering Algorithms

FindFace Multi uses the following object clustering algorithms:

- Real-time clustering. This clustering algorithm processes episodes to select suitable object images and clusters the selected ones. It works on the fly after an episode is closed. The real-time clustering results are dynamically displayed on the *Clusters* tab and in the relevant *record*.

  Not all episodes are used for clustering. If an episode meets all requirements (see the details *below*), the system forms a cluster in the following way:

  - Selects the best quality event.
  - Creates a new entity `cluster event` in the main system database **PostgreSQL**. The entity contains the selected event metadata, a link to the parent episode, an object feature vector, and a thumbnail.
  - Searches for a matching object centroid in the `cluster_events` gallery of the **Tarantool** feature vector database. An object centroid is a virtual feature vector averaged across all twin objects that have been detected so far (for example, a face centroid is a feature vector averaged across all face images of the same person). The system updates a matching centroid using the new event if such a centroid was found or creates a new centroid otherwise.

- Scheduled clustering. This clustering algorithm works over and revises the cluster events created during the real-time clustering. This algorithm improves the cluster centroid quality as the centroid will be averaged across a more extensive array of accumulated feature vectors. The results of the scheduled clustering are displayed on the *Clusters* tab and in the relevant *record* after each scheduled iteration.

  Use the RRULE format to define the schedule in the `CLUSTERS_CLUSTERIZATION_SCHEDULE` parameter at `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py`. Late-night hours are preferred as the scheduled clustering takes up a lot of CPU resources and time.

Important: The scheduled clustering completely overwrites the cluster galleries, including ids, unless you pin specific clusters by enabling the `CLUSTERS_AUTO_PIN_HEURISTICS` and `PIN_MATCHED_CLUSTERS` parameters (see *below*).

3.9.2 Enable and Configure Clustering

By default, the clustering is disabled. To enable and configure it, do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file. Find the `Clusters configuration` section.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

 # -- Clusters configuration --
 'ENABLE_NIGHT_CLUSTERIZATION': False,
 'ENABLE_REALTIME_CLUSTERIZATION': True,
 # rrule (recurrence rule) for scheduling "night" clusterization
 'CLUSTERIZATION_SCHEDULE': 'RRULE:FREQ=DAILY;INTERVAL=1;WKST=MO;BYHOUR=0;BYMINUTE=0
↪',
 # clusterize only selected objects types (for realtime and nightly clusterization)
 # available are: face, body, car
 'CLUSTERIZE_OBJECT_TYPES': ['face'],
 # keep no more than N the best quality events in centroid (None to disable)
 'CENTROID_MAX_SIZE': None,
```

(continues on next page)

(continued from previous page)

```
 # save cluster events without emben and images (only keep thumbnail for the best␣
↪event)
 'LONG_LIVING_CLUSTER_EVENTS': False,
 # skip clusterization if unpinned cluster events count is greater than this value
 'CLUSTERIZATION_MAX_CLUSTER_EVENTS': None,
 # create cluster only from cluster events in current case or only from cluster␣
↪events without case
 'ISOLATE_CASE_CLUSTERS': True,
 # cluster event to cluster matching confidence threshold
 'FACE_CLUSTER_CONFIDENCE_THRESHOLD': 0.714,  # model: [mango_320]
 'BODY_CLUSTER_CONFIDENCE_THRESHOLD': 0.65,  # model: [clio]
 # minimum required event quality for cluster creation
 'FACE_CLUSTER_EVENT_MIN_QUALITY': 0.5,  # model: [quality_fast.v1]
 'BODY_CLUSTER_EVENT_MIN_QUALITY': 0.6,  # model: [pedattr.quality.v0]
 'CAR_CLUSTER_EVENT_MIN_QUALITY': 0.73,  # model: [carattr.quality.v0]
 # discard cluster event if `max_centroids` similar centroids found with confidence␣
↪greater than `confidence`
 'FACE_CLUSTER_MAX_N_SIMILAR': {'enabled': False, 'max_centroids': 5, 'confidence':␣
↪0.714},
 'BODY_CLUSTER_MAX_N_SIMILAR': {'enabled': False, 'max_centroids': 5, 'confidence':␣
↪0.65},
 # minimum required object size in pixels for cluster creation
 'FACE_CLUSTER_EVENT_MIN_SIZE': 50,
 'BODY_CLUSTER_EVENT_MIN_SIZE': 50,
 'CAR_CLUSTER_EVENT_MIN_SIZE': 50,
 # minimum required number events in episode for cluster creation
 'FACE_CLUSTER_EVENT_MIN_EPISODE_EVENTS': 1,
 'BODY_CLUSTER_EVENT_MIN_EPISODE_EVENTS': 1,
 'CAR_CLUSTER_EVENT_MIN_EPISODE_EVENTS': 1,
 # age feature threshold for cluster creation
 'FACE_CLUSTER_EVENT_MIN_AGE_THRESHOLD': 16,
  ....
 # pinned clusters keep their id and events after reclusterization
 'CLUSTERS_AUTO_PIN_HEURISTICS': {
    'face': {
        # pin clusters with `value` minimum cluster events
        'min_events': {'enabled': True, 'value': 10},
        # cluster's centroid similarity confidence is less than
        'max_centroid_similarity_threshold': {'enabled': True, 'value': 0.54},  #␣
↪mango_320
        # minimum average event's quality
        'min_average_events_quality': {'enabled': True, 'value': 0.45},
    },
    'body': {},
    'car': {},
    },
 # always pin clusters with matched events (not affected by heuristics above)
 'PIN_MATCHED_CLUSTERS': False,
 ...
```

2. Enable the real-time clustering by setting `ENABLE_REALTIME_CLUSTERIZATION: True`.
3. If necessary, enable the scheduled clustering by setting `ENABLE_NIGHT_CLUSTERIZATION: True`.

Important: Enabling the scheduled clustering only makes sense if the real-time clustering is enabled. Otherwise, the system won't form any new clusters, since only the real-time clustering produces **unique** cluster events.

```
...
# -- Clusters configuration --
    'ENABLE_NIGHT_CLUSTERIZATION': True,
    'ENABLE_REALTIME_CLUSTERIZATION': True,
...
```

4. If necessary, specify a recurrence rule (RRULE) for the scheduled clustering. If the RRULE is not specified, the clustering automatically starts at 00:00 GMT.

Tip: See the RRULE calculator here.

```
# rrule (recurrence rule) for scheduling clusters clusterization
'CLUSTERIZATION_SCHEDULE': 'RRULE:FREQ=DAILY;INTERVAL=1;WKST=MO;BYHOUR=0;BYMINUTE=0
↪',
```

5. By default, the system forms only face clusters. To enable cluster formation of bodies and vehicles, add relevant object types to the following line:

```
# available are: face, body, car
'CLUSTERIZE_OBJECT_TYPES': ['face','body','car'],
```

6. If necessary, modify the minimum number of events in the episodes used for clustering. It's 1 by default. Do so separately for each object type.

```
# minimum required number events in episode for cluster creation
'FACE_CLUSTER_EVENT_MIN_EPISODE_EVENTS': 3,
'BODY_CLUSTER_EVENT_MIN_EPISODE_EVENTS': 3,
'CAR_CLUSTER_EVENT_MIN_EPISODE_EVENTS': 2,
```

7. If necessary, modify the minimum quality of the object images used for clustering. Do so separately for each object type.

Note: As this setting requires a high level of expertise and knowledge, we highly recommend consulting with our technical experts prior.

```
# minimum required event quality for cluster creation
'FACE_CLUSTER_EVENT_MIN_QUALITY': 0.5,  # model: [quality_fast.v1]
'BODY_CLUSTER_EVENT_MIN_QUALITY': 0.6,  # model: [pedattr.quality.v0]
'CAR_CLUSTER_EVENT_MIN_QUALITY': 0.73,  # model: [carattr.quality.v0]
```

8. If necessary, modify the confidence threshold for matching a cluster event and a cluster.

Warning: Be sure to consult with our experts by support@ntechlab.com before altering this parameter.

```
# cluster event to cluster matching confidence threshold
'FACE_CLUSTER_CONFIDENCE_THRESHOLD': 0.714,  # model: [mango_320]
'BODY_CLUSTER_CONFIDENCE_THRESHOLD': 0.65,  # model: [clio]
```

9. The scheduled clustering completely overwrites all created clusters. You can "pin" specific clusters, i.e. keep them and associated cluster events, including their IDs, intact. To do so, use the following settings:

Note: These settings are independent. Apply both if necessary.

Note: These settings do not affect the real-time clustering. It will continue to create new cluster events for the pinned clusters.

- `CLUSTERS_AUTO_PIN_HEURISTICS`: set `True` or `False` for the following options and specify the corresponding values:

  Note: Do so for each object type if applicable.

  - `min_events`: pin a cluster when the number of associated cluster events exceeds the given minimum value.
  - `max_centroid_similarity_threshold`: pin a cluster if the similarity between its centroid and centroids of other clusters is less than the given threshold. If a cluster looks similar to some other clusters, chances are these clusters belong to the exact person/vehicle. In this case, the system won't pin such a cluster to have an opportunity to re-cluster it. On the contrary, dissimilar clusters will be pinned.
  - `min_average_events_quality`: pin a cluster if the average quality of associated cluster events is greater than the given minimum value.

```
# pinned clusters keep their id and events after reclusterization
'CLUSTERS_AUTO_PIN_HEURISTICS': {
    'face': {
        # pin clusters with `value` minimum cluster events
        'min_events': {'enabled': True, 'value': 10},
        # cluster's centroid similarity confidence is less then
        'max_centroid_similarity_threshold': {'enabled': True, 'value': 0.54},
        # minimum average event's quality
        'min_average_events_quality': {'enabled': True, 'value': 0.45},
    },
    'body': {},
    'car': {},
},
```

- Enable the `PIN_MATCHED_CLUSTERS` parameter to pin matched clusters and the associated cluster events.

```
# always pin clusters with matched events (not affected by heuristics above)
'PIN_MATCHED_CLUSTERS': True,
```

10. If necessary, specify the maximum number of cluster events in the clusters that remain non-pinned. After this number is reached, the scheduled clustering will be automatically disabled.

```
# skip clusterization if unpinned cluster events count is greater than this value
'CLUSTERIZATION_MAX_CLUSTER_EVENTS': None,
```

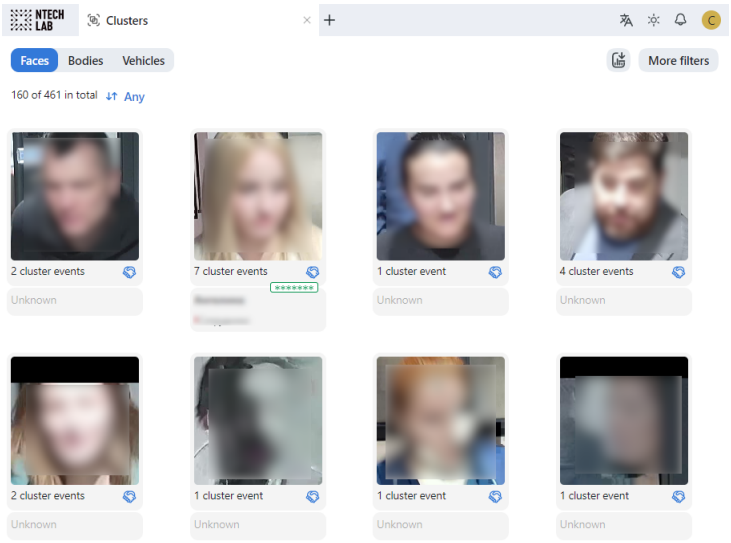
11. Restart the `findface-multi-findface-multi-legacy-1` container. You will see the *Clusters* tab appear in the FindFace Multi web interface.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

3.9.3 Work with Cluster Galleries

View and Filter Clusters

To view the cluster galleries, navigate to the *Clusters* tab.



When working with the cluster galleries, the following filters may come in handy:

Note: Some filters from the list below may be hidden, depending on which recognition features are enabled.

- *Object*: display clusters only for faces, bodies or vehicles.
- *Matches*: display clusters only with/without matches or any.
- *Watch lists*: display only clusters for a selected watch list.
- *Camera groups*: display only clusters from a selected group of cameras.
- *Cameras*: display only clusters from a selected camera.
- *Record name*: display only clusters with a given name.
- *Date and time*: display only clusters formed within a certain period.
- *First cluster event*: display only the first cluster event within a certain period.
- *Cluster event*: display only cluster event within a certain period.
- *ID*: display a cluster with a given ID.

Specific filters for face clusters

- *Age*: display clusters with people of a given age.
- *Beard*: filter clusters by the fact of having a beard.
- *Emotions*: display clusters with given emotions.
- *Gender*: display clusters with people of a given gender.
- *Glasses*: filter clusters by the fact of wearing glasses.
- *Liveness*: filter clusters by face liveness.
- *Face mask*: filter clusters by the fact of wearing a face mask.

Specific filters for body clusters

- *Gender by body*: display only events with people of a given gender or all events.
- *Age by body*: display only events with people of a given age.
- *Headwear*: display only events with a person wearing headgear of a given type: hat/cap, hood/scarf, none.
- *Vest*: display only events with a person wearing a vest of a given color.
- *Vest score*: display only events with a person wearing a vest within a given score.
- *Helmet*: display only events with a person wearing a helmet of a given color.
- *Helmet score*: display only events with a person wearing a helmet within a given score.
- *Upper clothes color*: display only events with a person wearing a top of a given color.
- *Lower clothes color*: display only events with a person wearing a bottom of a given color.
- *Upper clothes type*: display only events with a person wearing upper body wear of a given specific type: jacket, coat, sleeveless, sweatshirt, T-shirt, shirt, dress.
- *Lower body clothes*: display only events with a person wearing lower body wear of a given type: pants, nondescript, skirt, shorts.
- *Upper body clothes*: display only events with a person wearing upper body wear of a given generalized category: long sleeves, short sleeves, no sleeve.

Specific filters for vehicle clusters

- *Make*: filter vehicle events by vehicle make.
- *Model*: filter vehicle events by vehicle model.
- *Vehicle body type*: display only events with vehicles of a given body type: minivan, limousine.
- *Vehicle body color*: display only events with vehicles of a given color.
- *Country*: display only events with vehicles registered in a given country.
- *License plate number*: display an event with a given plate number.
- *Region*: display only events with vehicles registered in a given region.
- *License plate color*: display only events with a given license plate color.
- *Special vehicle*: display only events with vehicles belonging to a given type: police, fire service and emergencies ministry vehicles, gas rescue and emergency services, military, municipal vehicles, other.

- *Vehicle category*: display only events with vehicles belonging to a given category: unknown, motorcycle, scooter, car, car with a trailer, truck, truck with a trailer, bus, articulated bus, other.
- *Vehicle weight and body size*: display only events with vehicles of a given weight and body size.

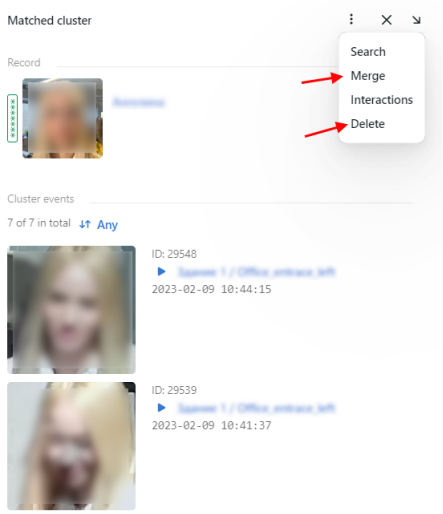
Note: License plate region and color are predicted for the United Arab Emirates (UAE) only. The values of these attributes will be marked as unknown for other countries.

Click a cluster to see the associated cluster events. You will be redirected to the *Cluster events* page.

Merge and Delete Clusters

To manually merge several clusters, select them one by one and click *Merge*.

To delete a cluster, select it and click *Delete*.



3.9.4 Make Cluster Gallery Static

Sometimes it's necessary to finalize the object clustering at a certain point in time and then operate with a static gallery of formed clusters.

To display the *Clusters* tab upon the disabled clustering, do the following:

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file.

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py
```

2. Add the line `"clusters": True` to the `SERVICES` section, as shown in the example below:

```
...
SERVICES = {
    "ffsecurity": {
        ...
        "clusters": True,
    }
...
```

3. Disable the real-time and scheduled clustering processes.

```
...
'ENABLE_NIGHT_CLUSTERIZATION': False,
'ENABLE_REALTIME_CLUSTERIZATION': False,
```

4. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

3.9.5 Manual Clustering

To manually launch the clustering process, use the `run_clusterization` utility.

You can invoke the `run_clusterization` help message by executing:

```
 sudo docker -it exec findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py run_clusterization --help

 usage: manage.py run_clusterization [-h]
                                     [--object-types OBJECT_TYPES [OBJECT_TYPES ...]]
                                     [--force] [--configuration CONFIGURATION]
                                     [--version] [-v {0,1,2,3}]
                                     [--settings SETTINGS]
                                     [--pythonpath PYTHONPATH] [--traceback]
                                     [--no-color] [--force-color]
                                     [--skip-checks]

 optional arguments:
-h, --help            show this help message and exit
--object-types OBJECT_TYPES [OBJECT_TYPES ...]
                      Clusterize selected object types. Uses
                      CLUSTERIZE_OBJECT_TYPES from config if not provided.
                      Allowed types: face, body, car
--force               Force clusterization even if
                      CLUSTERIZATION_MAX_CLUSTER_EVENTS condition is met
--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
                      A directory to add to the Python path, e.g.
                      "/home/djangoprojects/myproject".
--traceback           Raise on CommandError exceptions
```

(continues on next page)

(continued from previous page)

```
--no-color            Don't colorize the command output.
--force-color         Force colorization of the command output.
--skip-checks         Skip system checks.
```

With this utility, it is possible to separately launch clustering of faces, bodies, and vehicles and perform force clustering when the maximum number of cluster events exceeds the value of the CLUSTERIZATION_MAX_CLUSTER_EVENTS parameter (see *Enable and Configure Clustering*). For example, to force start face clustering, execute:

```
sudo docker exec -it findface-multi-findface-multi-legacy-1 /opt/findface-security/bin/
↪python3 /tigre_prototype/manage.py run_clusterization --object-types face --force
```

See also:

- *Record Index*
- *Interaction Tracking*
- *Audience analysis*

3.10 Reports

In this chapter:

3.10.1 Report Types

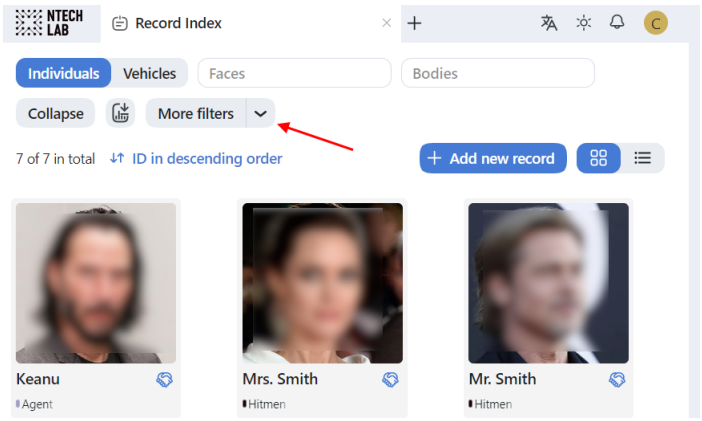
The following reports are available:

- Standard reports on the system entities:
  - *events*
  - *episodes*
  - *search*
  - *clusters*
  - *video sources*
  - *records*
  - *analytical data*
  - *audit logs*
- Work time reports, that allow you to regard the exact moments of staff entering and exiting an enterprise or a designated area and calculate the total time spent at it, can be built on the *Events* tab.

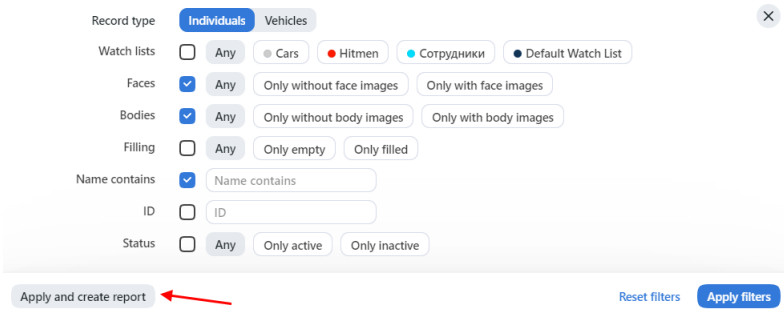
3.10.2 Build Standard Report

To build a standard report on a system entity, do the following:

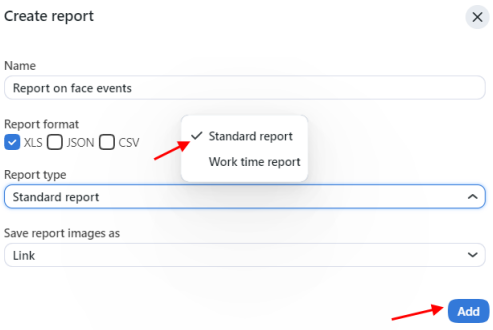
1. Navigate to the tab associated with the required entity: *Search*, *Episodes & Events*, *Clusters*, *Video Sources*, *Record index*, *Audience analysis*, *Audit Logs*.
2. Click the *More filters* button. Set filters for the report.



3. Click *Apply and create report*.



4. Specify the report name.
5. If you are on the *Events* tab, select the report type *Standard report* (as there are two types of reports available).
6. If applicable, choose whether to save the report images as links, thumbnails, or full frames.
7. Click *Add*. The report will be available for download on the *Reports* tab.

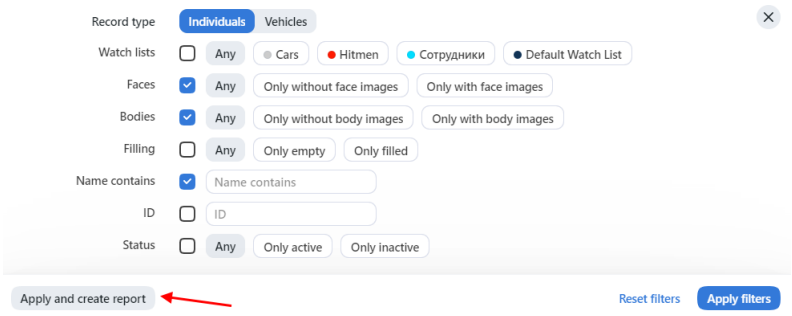


3.10.3 Build Work Time Report

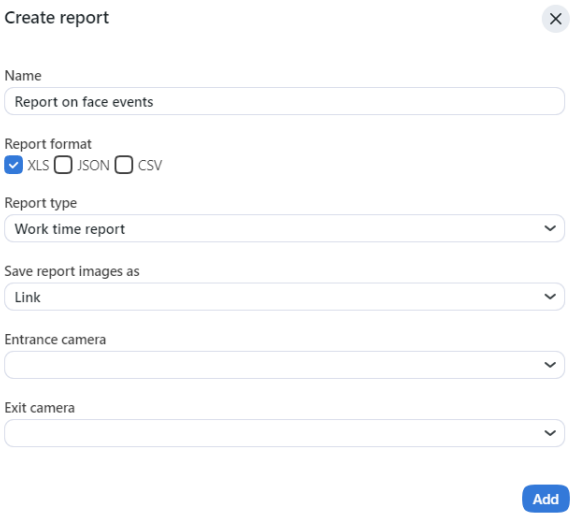
A work time report can be built on the *Events* tab. It requires specifying entrance and exit cameras.

To build a work time report, do the following:

1. Set filters for the report.
2. Click *Apply and create report*.



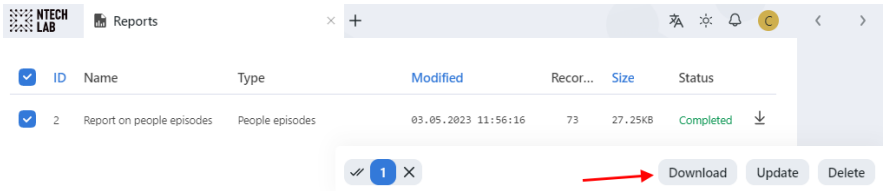
3. Specify the report name.
4. Select the report format.
5. Select the report type: *Work time report*.
6. If applicable, choose whether to save the report images as links, thumbnails, or full frames.
7. Select the entrance camera.
8. Select the exit camera.
9. Click *Add*. The report will be available for download on the *Reports* tab.



3.10.4 Work with Reports

You can access reports previously created in the system on the *Reports* tab. The following operations are available:

- Download selected reports.
- Update selected reports.
- Delete selected reports.

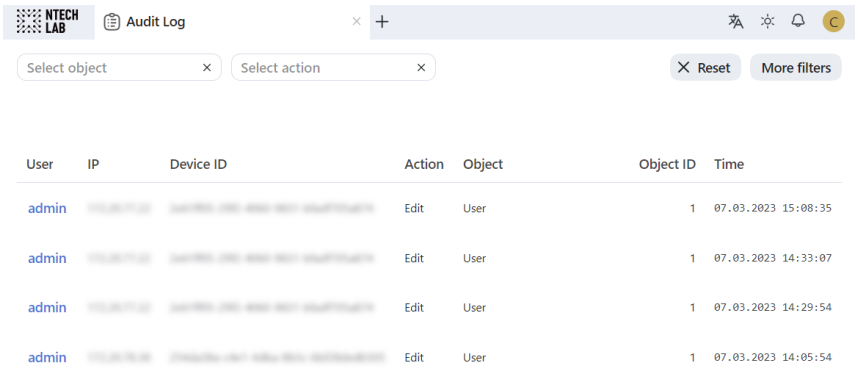


See also:

Configure Saving Images in Reports

3.11 Audit Log

The FindFace Multi comprehensive and searchable audit log is 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 Log* tab.



Each record provides the following data:

- username 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 record
- object identifier
- details, subject to the action type
- timestamp

Use the filter panel above to set up the search conditions.

3.12 People-Related Analytics

FindFace Multi is an ideal tool to gather people-related analytics. Enable face and body *clusters* first and then make the most of them with our analytical features.

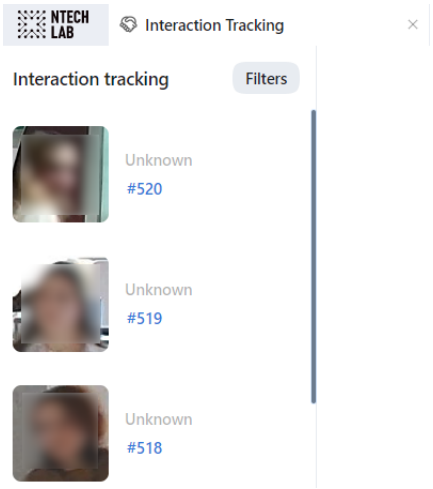
3.12.1 Interaction Tracking

FindFace Multi can provide interaction tracking and analysis. To harness this functionality, you must enable *face clustering*.

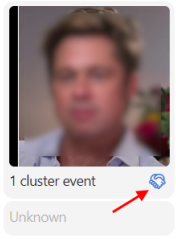
When tracking interactions, the system automatically determines a circle of people with whom a person has previously been in contact. For each person from the first circle, the system builds another circle of connected people, and so on. Overall, interaction analysis is three-circles deep.

Note: You can track interactions between bodies and vehicles as well. Access this functionality via *HTTP API*. Note that body and vehicle clustering must be enabled beforehand.

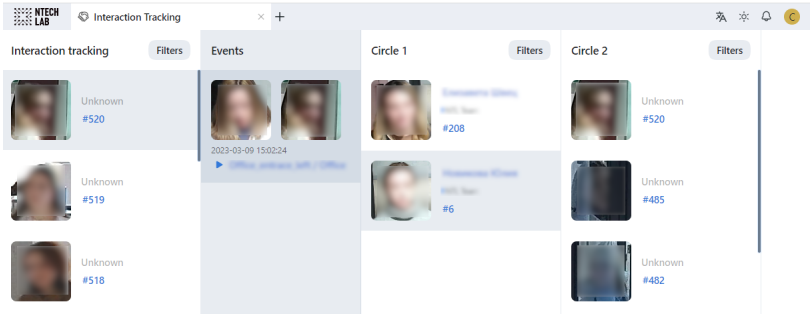
The interaction analysis is available on the *Interaction Tracking* tab.



Tip: You can also display the circle of connected people right from the *Clusters* tab and *Record index* tab by clicking on the handshake icon.



On the *Interaction Tracking* tab, click on a person to display their first circle of interactions and associated events. Keep on to unveil the entire tree of interactions.



You can apply the available filters to the interaction tracking:

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.

- *Matches*: display clusters only with/without matches, or all events.
- *Watch lists*: display clusters only for a selected watch list.
- *Camera groups*: display only clusters from a selected group of cameras.
- *Cameras*: display only clusters from a selected camera.
- *Record name*: display clusters with a given record name.
- *Date and time*: display only clusters formed within a certain period.
- *First cluster even*: display only the first cluster event within a certain period.
- *Cluster event*: display only cluster event within a certain period.
- *ID*: display a cluster with a given ID.

Specific filters for face clusters

- *Age*: display clusters with people of a given age.
- *Beard*: filter clusters by the fact of having a beard.
- *Emotions*: display clusters with given emotions.
- *Gender*: display clusters with people of a given gender.
- *Glasses*: filter clusters by the fact of wearing glasses.
- *Liveness*: filter clusters by face liveness.
- *Face mask*: filter clusters by the fact of wearing a face mask.

When searching through a circle of interactions, apply the following additional settings:

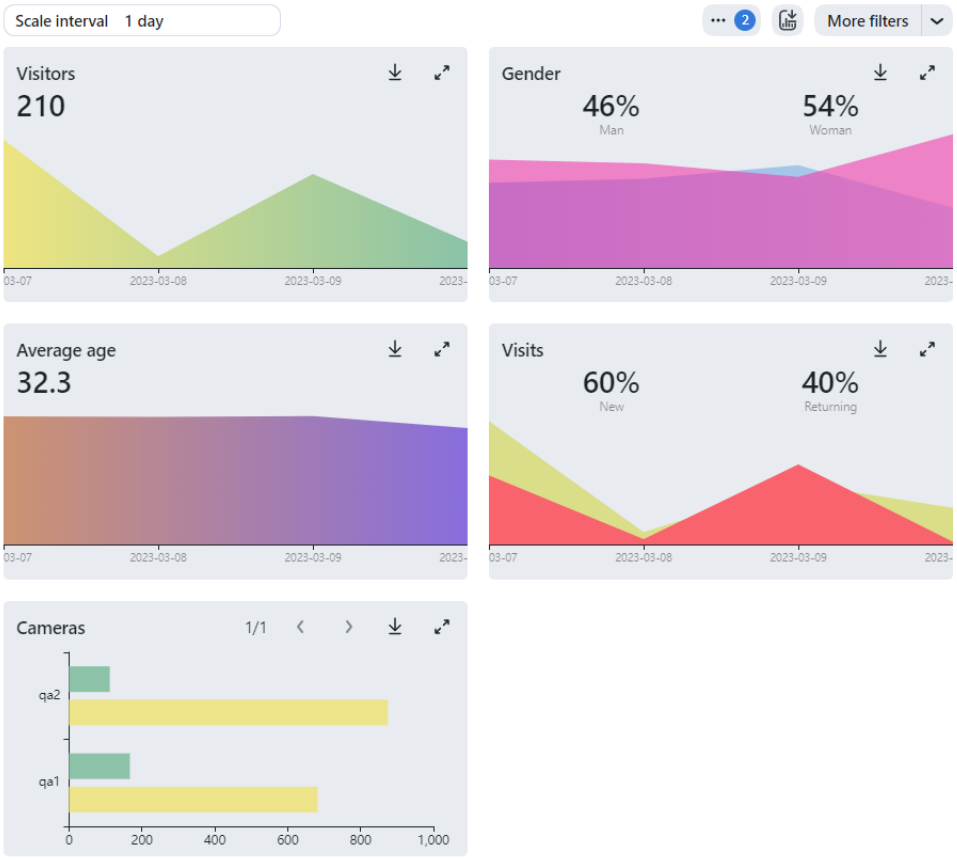
- *Interaction search based on the episode last event*: check to analyze contacts between individuals using the last event of an episode. In this case, having found truly associated people is most probable as they simultaneously leave a camera's field of view. If the option is disabled, the system will use the best event of an episode for interaction search.
- *Maximum gap between the appearance of individuals to consider them connected, seconds*: maximum time in seconds between the appearance of individuals to consider them related.

3.12.2 Audience analysis

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).

The analytical data charts are available on the *Audience analysis* tab.

Important: The analytics are built only when the *face clustering* is enabled.



To work with the analytical data, use the following filters:

- Date and time
- Scale interval
- Minimum time between visits
- Number of returns
- Cameras
- Watch lists
- Age
- Gender

See also:

- *Face, Body, Vehicle Clusters*

3.13 Video Surveillance

In this chapter:

3.13.1 Video Wall

The FindFace Multi Video Wall allows for basic video surveillance. Use it to display the video image from cameras and video files.

Important: Advanced video surveillance is available with *Video Recorder*.

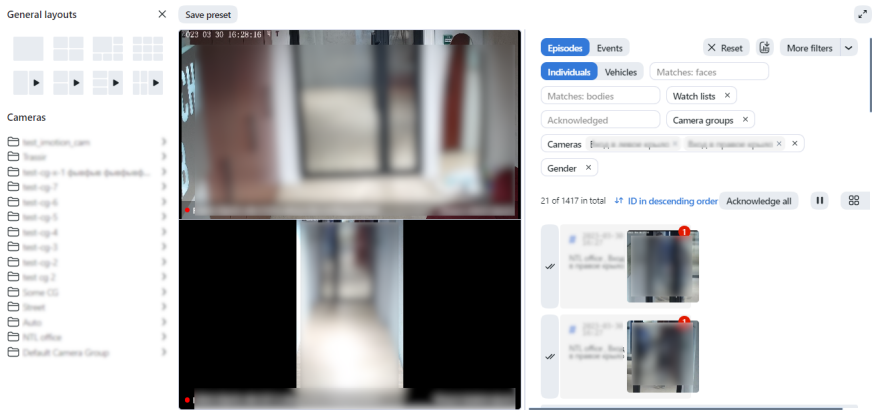
Display Video

The Video Wall offers two modes, four predefined layouts in each:

- video streaming
- video streaming with object detection and episode feed

To display video on the Video Wall, do the following:

1. Navigate to the *Video Wall* tab.
2. Select a mode and camera layout.



3. Drag-n-drop cameras of your choice to the Video Wall.

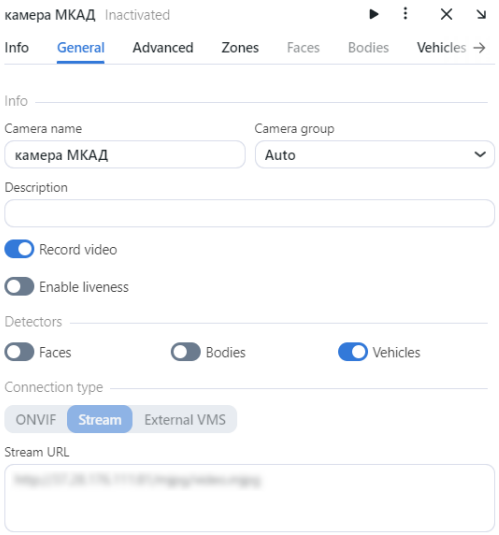
You can work with the episode and event feed on the Video Wall in the *same manner* as with the *Episodes & Events* tab, including the same filters that are available for episodes and events.

Note: See the full list of *episode filters* and *event filters* applicable to the Video Wall.

3.13.2 Video Recorder

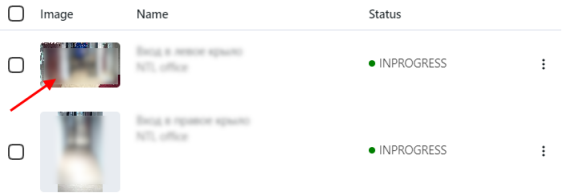
Enable Video Recording

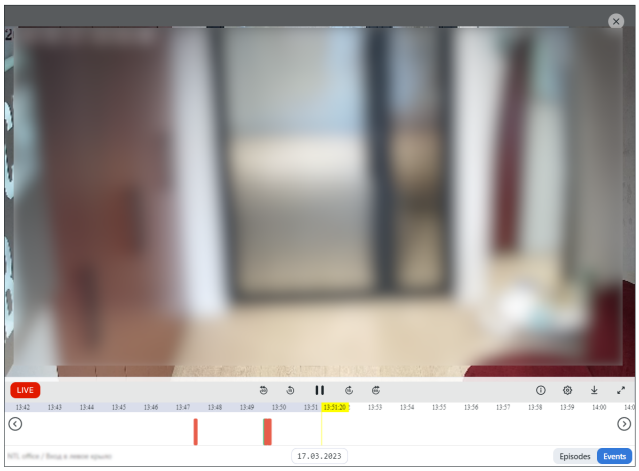
If a Video Recorder is *deployed* and *configured*, you will see the *Record video* checkbox appear in the main camera settings. Check it to enable recording video chunks from a camera and sending them over to Video Recorder for further processing.



View Camera Video in Video Player

Under *specific settings*, clicking on the camera preview opens up not a static frame, but a video player, streaming live video from the camera.





The video player has a highly intuitive design. It provides the following possibilities:

1. Stream live video from the camera.
2. Watch video chunks recorded from the camera. They will be marked in purple on the timeline. To quickly switch from archived video to the camera live stream, click the *Live* button.
3. Visual indication of moments corresponding to face or vehicle events (if enabled) as colorful markers on the timeline. The unmatched events are marked in pink, while the matched ones are light green.

Note: This feature is optional and must be pre-configured. See *Configure Video Recorder*.

Important: The time on the timeline is displayed in the spectator's time zone. For example, if an event happened at 2 p.m. in Abu Dhabi, its marker would be at 12 p.m. for a spectator in Paris.

---

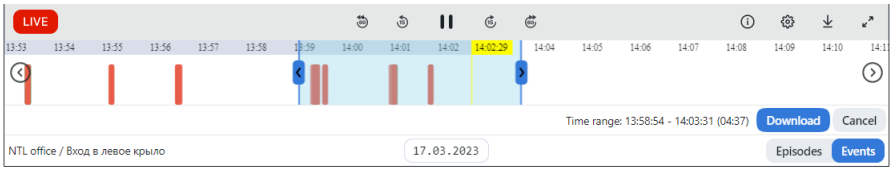
4. Navigate to the past and future over the timeline, with the possibility of zooming it in and out.

Tip: You can navigate along the timeline by using the < / > buttons, or by moving the cursor along the timeline with the right mouse button held down.

Tip: To zoom in and out, use the mouse wheel with Ctrl.

---

5. To export selected clips, select the interval and click *Download*.



CHAPTER

FOUR

INTEGRATIONS

This chapter is all about integration with FindFace. In the current version of FindFace, you can only integrate your system via HTTP API.

4.1 HTTP API

Detailed interactive documentation on the FindFace HTTP API is available after installation at `http://<findface-ip:port>/api-docs`. Learn and try it out.

Tip: `<findface-ip:port>` must be substituted into your ip address for FindFace Multi.

Tip: You can also find it by navigating to *Settings -> API Documentation* in the web interface.

4.1.1 Overview

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

Authentication

All API methods require a simple token-based HTTP Authentication. To authenticate, put the word "Token" and a token, separated by whitespace, into the Authorization HTTP header: `Authorization: Token 000...` All requests that fail to provide a valid authentication token will result in an HTTP 401 Unauthorized response.

Parameters Format

There are two ways to pass parameters to the API methods:

- `application/json`: parameters are represented by a JSON contained in the body.
- `multipart/form-data`: parameters are encoded into separate parts. This way supports uploading a photo image file in the same request.

Additional Information

Standard extraction limits:

- Image formats: JPEG, PNG, WEBP
- Maximum photo file size: 10 MB
- Maximum photo resolution: 6000 pixels on the biggest side
- Minimal size of a face: 50x50 pixels

Check `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` for custom definition.

4.1.2 Getting 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:

Tip: `<findface-ip:port>` must be substituted into your ip address for FindFace Multi. `<token>` must be substituted into your token.

Gear Up for Work

Perform the primary configuration of your system:

1. *Deploy FindFace Multi* and learn API documentation at `http://<findface-ip:port>/api-docs`.
2. *Authorize* with HTTP Basic auth by sending credentials to the login form.
3. *Create an authorization token* with your session UUID and optionally face authorization.
4. Paste a token into the `API Key` field in the authorization form: `Token <token>` or put it into the Authorization HTTP header.

Organize Watch Lists and Card Index

1. *Create a new watch list*, using `POST` method, or use the default one.
2. *Find watch list ID* to make a `POST` request for *creating a new card*.
3. Perform *face (car, body) detection* on the attached photo and copy `id` of the necessary object.
4. Attach an image of the individual's *face*, using `detection id`.

Upload and Process Video File

1. Organize Video Surveillance by *uploading a video file*.
2. Find `id` of the video archive by *listing video archives* or *adding new one*.
3. *Attach source video file* with `id` of the video archive.
4. *Start video archive processing*.

Data Sources

To configure video-based object monitoring, add cameras to FindFace Multi, grouping them subject to their location.

1. A default preconfigured camera group is available in the system. *List available camera group*.
2. If necessary, *create a new camera group*.
3. Use camera group `id` to make a `POST` request for *creating a new camera*.

FindFace Multi in Action

1. Search for objects in the database of detected objects and card index. *Learn more*.
2. *Compare two objects* and verify that they match.

4.1.3 Authentication

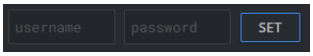
API requests to FindFace Multi are to be sent to `http://<findface-ip:port>`.

HTTP Basic authentication

HTTP basic authentication is a simple challenge and response mechanism with which a server can request authentication information (a username and password).

Put your credentials to the login form:

- Username – username of your FindFace Multi account.
- Password – password for your FindFace Multi account.



This data are to be sent in the Authorization request header as base64 code.

```
Authorization: Basic <base64(username:password)>
```

API Key authorization

To create an authorization token, use the following method:

```
POST /auth/login/
```

You should provide basic authorization credentials (username, password) earlier.

This method returns a structure containing an authorization `token`, that you can use in all other methods.

If the system uses face and password authorization, together with the Basic authorization header, additionally pass the token received during face authorization in the `video_auth_token` field.

The REQUEST BODY is required and contains application/json object with the following parameters:

Tip: * - means required parameters. – read only.

Option	Schema	Description
`video_auth_token`	string	Constraints: Min 1 chars. Inactive token from face authorization. Required when `face_and_password` is set.
`uuid*`	string	Constraints: 1 to 256 chars. Session unique identifier on the device.
`mobile`	boolean	Device is mobile.
`device_info`	<any-key>: strintfloat-boolobjec-tarraynull	Device information.

Request example

```
{
 "video_auth_token": "A",
 "uuid": "A",
 "mobile": false,
 "device_info": {}
}
```

CURL Example

```
curl -X POST "http://<findface-ip:port>/auth/login/" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Basic <base64(username:password)>" \
 -H "Content-Type: application/json" \
 -d '{"uuid":"A"}' \
```

Returns:

- 200 on success.
- 401 if response body is unspecified.

If the response is successful (OK: 200), it returns an object that contains the following *parameters*. *Example*.

Response example

Option	Schema	Description
`token*`	string	Authorization token.
`user*`	{...}	User info.
`token_expiration_datetime*`	date-time	Token expiration datetime.

`user` contains the following parameters:

- `id*` : integer
- `active*` : boolean
- `created_date*` : date-time
- `groups` : [integer]
- `modified_date*` : date-time – Object modification date.
- `permissions*` : [string]
- `real_name*` : string – Constraints: Max 80 chars.
- `name*` : string – Constraints: Max 80 chars.
- `comment` : string – Extended description of the user, up to 2048 chars.
- `camera_group_permissions*` : {<any-key>: string}

- `watch_list_permissions*` : {<any-key>: string}
- `group_permissions*` : {<any-key>: string}
- `primary_group*` : integer
- `language` : enum – Allowed: en-usesru. Backend message language for the user.
- `has_face*` : boolean – User photo is provided.
- `face_cover` : stringnull – Constraints: Max 32 chars.
- `ad_user*` : boolean – User is registered in Active Directory.

Response example


```
{
 "token": "string",
 "user": {
   "id": 0,
   "active": false,
   "created_date": "1970-01-01T00:00:00.000Z",
   "groups": [
     0
   ],
   "modified_date": "1970-01-01T00:00:00.000Z",
   "permissions": [
     "string"
   ],
   "real_name": "AAAAAA",
   "name": "AAAAAA",
   "comment": "AAAAAA",
   "camera_group_permissions": {},
   "watch_list_permissions": {},
   "group_permissions": {},
   "primary_group": 0,
   "language": "en-us",
   "has_face": false,
   "face_cover": "AAAAAA",
   "ad_user": false
 },
 "token_expiration_datetime": "1970-01-01T00:00:00.000Z"
}
```


Paste this obtained token into the `API Key` field in the authorization form: `Token <token>` and click *SET* or put it into the Authorization HTTP header: `Authorization: Token 000...`. All requests that fail to provide a valid authentication token will result in an HTTP 401 Unauthorized response.

To disable applied keys, click *REMOVE* or *CLEAR ALL API KEYS* or send a request `POST /auth/logout/`.

4.1.4 Watch lists

List watch lists

To list watch lists, use the following method:

```
GET /watch-lists/
```

The REQUEST contains the following QUERY-STRING PARAMETERS:

Name	Schema	Description
`limit`	integer	Number of results to return.
`ordering`	string	Available fields: id, created_date, modified_date.

To look for watch lists `id` point out some value of `limit` and for `ordering`.

CURL example

```
curl -X GET "http://<findface-ip:port>/watch-lists/?limit=2&ordering=id" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
```

If the response is successful (OK: 200), it returns an object that contains array of results with *the following parameters*. *Example*.

Tip: * – means required parameters. – read only.

Name	Schema	Description
id*	integer	Watch list ID.
created_date*	date-time	Object creation date.
modified_date*	date-time	Object modification date.
active	boolean	true if a watch list is enabled.
name*	string	Short watch list name, up to 256 characters.
comment	string	Extended description, up to 2048 characters.
color	string	Color of the object label in hex. Constraints: Max 6 chars.
notify	boolean	true if the notifications on the watch list enabled.
acknowledge	boolean	Require manual acknowledgment of the events that match with the watch list.
permissions	{<any-key>: string}	Permissions.
camera_groups	[integer]	List of the camera groups used to monitor the watch list.
face_threshold	number-null	Watch list face confidence threshold. Constraints: Min 0Max 1.
body_threshold	number-null	Watch list body confidence threshold. Constraints: Min 0Max 1.
car_threshold	number-null	Watch list car confidence threshold. Constraints: Min 0Max 1.
ignore_events	boolean	Events won't be created if set to True.
send_events_to_external_vms	boolean	true if events matched with this watch list should be sent to an external VMS.
active_after	date-timenull	Data-time information.
active_before	date-timenull	Data-time information.
disable_schedule	{...}	Serializer mixin that raises ValidationError if excess fields are presented. Can be used in nested serializers.
recount_schedule_on	date-time	Data-time information.
origin	string	Constraints: Max 256 chars bit.

Response example


```
{
 "results": [
   {
     "id": -1,
     "created_date": "2023-01-16T13:44:36.407610Z",
     "modified_date": "2023-01-16T13:44:36.407750Z",
     "active": true,
     "name": "Unmatched",
     "comment": "Default list for unmatched events",
     "color": "ffffff",
     "notify": false,
     "acknowledge": false,
     "permissions": {
       "1": "edit",
       "2": "view",
       "3": "view"
```


(continues on next page)

(continued from previous page)


```
    },
    "camera_groups": [],
    "face_threshold": null,
    "body_threshold": null,
    "car_threshold": null,
    "ignore_events": false,
    "send_events_to_external_vms": false,
    "active_after": null,
    "active_before": null,
    "disable_schedule": {},
    "recount_schedule_on": null,
    "origin": "ffsecurity"
  },
  {
    "id": 1,
    "created_date": "2023-01-16T13:44:36.394542Z",
    "modified_date": "2023-01-16T13:44:36.394577Z",
    "active": true,
    "name": "Default Watch List",
    "comment": "",
    "color": "123456",
    "notify": false,
    "acknowledge": false,
    "permissions": {
      "1": "edit",
      "2": "view",
      "3": "view"
    },
    "camera_groups": [],
    "face_threshold": null,
    "body_threshold": null,
    "car_threshold": null,
    "ignore_events": false,
    "send_events_to_external_vms": false,
    "active_after": null,
    "active_before": null,
    "disable_schedule": {},
    "recount_schedule_on": null,
    "origin": "ffsecurity"
  }
 ]
}
```

Create a new watch list

To add a new watch list, use the following method:

```
POST /watch-lists/
```

The REQUEST BODY is required and contains application/json object with the *watch lists parameters*.

Name	Schema	Description
active	boolean	true if a watch list is enabled.
name*	string	Short watch list name, up to 256 characters.
comment	string	Extended description, up to 2048 characters.
color	string	Color of the object label in hex. Constraints: 1 to 6 chars.
notify	boolean	true if the notifications on the watch list enabled.
acknowledge	boolean	Require manual acknowledgment of the events that match with the watch list.
permissions	<any-key>: string	Constraints: Min 1 chars.
camera_groups	[integer]	List of the camera groups used to monitor the watch list.
face_threshold	number-null	Watch list face confidence threshold. Constraints: Min 0Max 1.
body_threshold	number-null	Watch list body confidence threshold. Constraints: Min 0Max 1.
car_threshold	number-null	Watch list car confidence threshold. Constraints: Min 0Max 1.
ignore_events	boolean	Events won't be created if set to True.
send_events_to_external_vms	boolean	true if events matched with this watch list should be sent to an external VMS.
active_after	date-timenull	Date-time information.
active_before	date-timenull	Date-time information.
disable_schedule	{...}	Serializer mixin that raises ValidationError if excess fields are presented. Can be used in nested serializers.
origin	string	Constraints: 1 to 256 chars bit.

Request example

Tip: This example is given for reference only, substitute your values in the corresponding fields. You may fill in only the required fields, and the others will be by default.

```
{
 "active": false,
 "name": "A",
 "comment": "AAAAAA",
 "color": "A",
 "notify": false,
 "acknowledge": false,
 "permissions": {},
 "camera_groups": [
```

(continues on next page)

(continued from previous page)


```
    0
  ],
  "face_threshold": 0,
  "body_threshold": 0,
  "car_threshold": 0,
  "ignore_events": false,
  "send_events_to_external_vms": false,
  "active_after": "1970-01-01T00:00:00.000Z",
  "active_before": "1970-01-01T00:00:00.000Z",
  "disable_schedule": {
    "monday": [
      [
        "A"
      ]
    ],
    "tuesday": [
      [
        "A"
      ]
    ],
    "wednesday": [
      [
        "A"
      ]
    ],
    "thursday": [
      [
        "A"
      ]
    ],
    "friday": [
      [
        "A"
      ]
    ],
    "saturday": [
      [
        "A"
      ]
    ],
    "sunday": [
      [
        "A"
      ]
    ]
  },
  "origin": "A"
}
```


You may send these parameters:

```
{
```

(continues on next page)

(continued from previous page)

```
 "active": true,
 "name": "Test_list",
 "comment": "AAAAAA",
 "color": "35a2ee"
}
```

CURL example

```
curl -X POST "http://<findface-ip:port>/watch-lists/" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
 -H "Content-Type: application/json" \
 -d '{"active":true,"name":"Test_list","comment":"AAAAAA","color":"35a2ee"}' \
```

If the response is successful (Created: 201), it returns an object that contains *parameters*. *Example*.

Response example

```
{
 "id": 2,
 "created_date": "2023-01-16T14:07:52.424520Z",
 "modified_date": "2023-01-16T14:07:52.424549Z",
 "active": true,
 "name": "Test_list",
 "comment": "AAAAAA",
 "color": "35a2ee",
 "notify": false,
 "acknowledge": false,
 "permissions": {
   "1": "edit"
 },
 "camera_groups": [],
 "face_threshold": 0,
 "body_threshold": 0,
 "car_threshold": 0,
 "ignore_events": false,
 "send_events_to_external_vms": false,
 "active_after": null,
 "active_before": null,
 "disable_schedule": {},
 "recount_schedule_on": null,
 "origin": "ffsecurity"
}
```

Use watch list `id` to make a `POST` request for *creating a new card*.

Useful requests

```
GET /watch-lists/
POST /watch-lists/
GET /watch-lists/{id}/
DELETE /watch-lists/{id}/
PATCH /watch-lists/{id}/
POST /watch-lists/{id}/purge/
GET /watch-lists/count/
POST /watch-lists/purge_all/
```

4.1.5 Record

Create a new record

To create a new human record, use the following method:

```
POST /cards/humans/
```

Tip: For vehicle records instead `humans` use `cars`.

The REQUEST BODY is required and contains application/json object with the *following parameters*.

Tip: * – means required parameters. – read only.

Name	Schema	Description
`active`	boolean	`true` if the object is enabled.
`name*`	string	Record name, up to 256 characters.
`comment`	string	Extended description, up to 2048 characters.
`watch_lists*`	[integer]	Array of the related Watch lists ID.
`meta`	<any-key>	Meta data.
`active_after`	date-timenull	Date-time information.
`active_before`	date-timenull	Date-time information.
`disable_schedule`	{...}	Serializer mixin that raises ValidationError if excess fields are presented. Can be used in nested serializers.

Request example

Tip: This example is given for reference only, substitute your values in the corresponding fields. You may fill in only the required fields, and the others will be by default.

```
{
 "active": false,
```

(continues on next page)

(continued from previous page)


```
 "name": "A",
 "comment": "AAAAAA",
 "watch_lists": [
   0
 ],
 "meta": {},
 "active_after": "1970-01-01T00:00:00.000Z",
 "active_before": "1970-01-01T00:00:00.000Z",
 "disable_schedule": {
   "monday": [
     [
       "A"
     ]
   ],
   "tuesday": [
     [
       "A"
     ]
   ],
   "wednesday": [
     [
       "A"
     ]
   ],
   "thursday": [
     [
       "A"
     ]
   ],
   "friday": [
     [
       "A"
     ]
   ],
   "saturday": [
     [
       "A"
     ]
   ],
   "sunday": [
     [
       "A"
     ]
   ]
 }
}
```


For example, you may send these parameters:


```
{
 "active": true,
 "name": "Angelina Jolie",
```


(continues on next page)

(continued from previous page)

```
 "comment": " ",
 "watch_lists": [
   2
 ]
}
```

CURL example

```
curl -X POST "http://<findface-ip:port>/cards/humans/" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
 -H "Content-Type: application/json" \
 -d '{"active":true,"name":"Angelina Jolie","comment":" ","watch_lists":[2]}' \
```

If the response is successful (Created:201), it returns an object that contains the following *parameters*. *Example*.

Name	Schema	Description
id*	integer	Record ID.
active	boolean	true if the object is enabled.
filled*	boolean	true if record is filled.
created_date*	date-time	Object creation date.
modified_date*	date-time	Object modification date.
name*	string	Record name, up to 256 characters.
comment	string	Extended description, up to 2048 characters.
watch_lists*	[integer]	Array of the related Watch lists ID.
looks_like* • confidence* • collection* • matched_object*	• number • stringnull • stringnull	• Confidence. Constraints: Min 0Max 1. • Collection. • Matched object.
meta	<any-key>	Meta data.
looks_like_confidence*	number	Looks like confidence.
active_after	date-timenull	Data-time information.
active_before	date-timenull	Data-time information.
disable_schedule	array	Serializer mixin that raises ValidationError if excess fields are presented. Can be used in nested serializers.
recount_schedule_on*	date-time	Data-time information.
face_objects*	integer	Linked face objects counter.
body_objects*	integer	Linked body objects counter.
face_cluster*	integer	Face claster.
body_cluster*	integer	Body cluster.
links_to_relations* • id* • name* • created_date* • card* • relation*	• integer • string • date-time • integer • integer	• ID • Short name, up to 256 characters. • RelationLink creation date. • Record. • Relation.

Note: Relations are not supported in FindFace Multi 2.0 and are kept in API only.

Response example


```
{
 "id": 1,
 "active": true,
 "filled": true,
 "created_date": "2023-01-17T07:43:56.363330Z",
 "modified_date": "2023-01-17T07:43:56.363354Z",
 "name": "Angelina Jolie",
 "comment": " ",
```


(continues on next page)

(continued from previous page)

```
 "watch_lists": [
   2
 ],
 "meta": {},
 "active_after": null,
 "active_before": null,
 "disable_schedule": {},
 "recount_schedule_on": null,
 "face_objects": 0,
 "body_objects": 0,
 "face_cluster": null,
 "body_cluster": null,
 "links_to_relations": []
}
```

List records

To list human records, use this method:

```
GET /cards/humans/
```

The REQUEST contains the QUERY-STRING PARAMETERS. All parameters you can see in `http://<findface-ip:port>/api-docs`.

You may use this method to *search faces* in the system. Specify the value of `detection:<detection id>`, received by *detection of objects on a photo*, in the field `looks_like`.

CURL example

```
curl -X GET "http://<findface-ip:port>/cards/humans/?looks_like=detection
↪%3Acf4ffvmv54rotim9jt60" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
```

If the response is successful (OK: 200), it returns an object that contains the following parameters:

Name	Schema	Description
`next_page`	stringnull	Next page.
`prev_page`	stringnull	Previous page.
`results`	[{...}]	Contain the following *parameters*.

Response example


```
{
"next_page": null,
"prev_page": null,
"results": [
  {
    "id": 1,
    "active": true,
    "filled": true,
    "created_date": "2023-01-17T07:43:56.363330Z",
    "modified_date": "2023-01-17T11:56:42.496871Z",
    "name": "Angelina Jolie",
    "comment": " ",
    "watch_lists": [
      2
    ],
    "looks_like": {
      "confidence": 0.8078,
      "collection": "face_objects",
      "matched_object": "4493493039043981648"
    },
    "meta": {},
    "looks_like_confidence": 0.8078,
    "active_after": null,
    "active_before": null,
    "disable_schedule": {},
    "recount_schedule_on": null,
    "face_objects": 1,
    "body_objects": 0,
    "face_cluster": 11,
    "body_cluster": null,
    "links_to_relations": []
  }
 ]
}
```


Useful requests

```
GET /cards/cars/
POST /cards/cars/
GET /cards/cars/{id}/
DELETE /cards/cars/{id}/
PATCH /cards/cars/{id}/
GET /cards/cars/count/
GET /cards/humans/
POST /cards/humans/
GET /cards/humans/{id}/
DELETE /cards/humans/{id}/
PATCH /cards/humans/{id}/
GET /cards/humans/count/
```

4.1.6 Detect objects on a photo

To detect objects on a photo, use this method:

```
POST /detect
```

The REQUEST BODY is required and contains multipart/form-data with the following parameters.

Name	Schema	Description
`photo`	binary	Source image file.
`attributes`	object	Attributes of `face`, `car` and `body`.

`attributes` might be empty or contain the objects of `face`, `car` and `body` with the following parameters for each one:

- age: boolean
- beard: boolean
- emotions: boolean
- glasses: boolean
- gender: boolean
- medmask: boolean
- headpose: boolean

Attach a source image file and send the `POST` request.

CURL example

```
curl -X POST "http://<findface-ip:port>/detect" \
-H "Accept: application/json" \
-H "Content-Language: ru" \
-H "Accept-Language: ru" \
-H "Authorization: Token <token>" \
-H "Content-Type: multipart/form-data" \
-F "photo=@_01.png" \
-F "attributes={
 "face": {
   "age": false,
   "beard": false,
   "emotions": false,
   "glasses": false,
   "gender": false,
   "medmask": false,
   "headpose": false
 },
 "car": {
   "description": false,
   "license_plate": false,
   "special_vehicle_type": false,
   "category": false,
   "weight_type": false,
```

(continues on next page)

(continued from previous page)

```
    "orientation": false
  },
  "body": {
    "color": false,
    "clothes": false,
    "bags": false,
    "protective_equipment": false,
    "age_gender": false
  }
}" \
```

If the response is successful (OK: 200), it returns an object that contains the following *parameters*. *Example*.

Name	Schema	Description
orientation*	integer	EXIF orientation of the photo.
objects	<any-key>: strintfloat-boolobjec-tarraynull	Returned objects with requested attributes.

Response example

```
{
  "orientation": 1,
  "objects": {
    "face": [
      {
        "id": "cf0mbqev54rqhngnq940",
        "bbox": {
          "left": 451,
          "top": 235,
          "right": 645,
          "bottom": 502
        },
        "detection_score": 0.80645436,
        "low_quality": false,
        "features": {}
      },
      {
        "id": "cf0mbqev54rqhngnq94g",
        "bbox": {
          "left": 757,
          "top": 79,
          "right": 948,
          "bottom": 353
        },
        "detection_score": 0.90099674,
        "low_quality": false,
        "features": {}
```

(continues on next page)

(continued from previous page)

```
      }
    ]
  }
}
```

In the response you will get ID of the object and the coordinates of bbox of the object (face, car, body). Copy the value of `id` of the necessary object to use it for *adding an object* to the card or to *search an object*.

4.1.7 Objects

Add face object

```
POST /objects/faces/
```

This method creates new `Face object` that contains source photo, thumbnail, and other attributes.

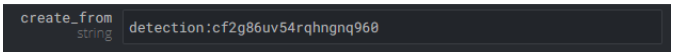
Tip: For car or body objects instead `faces` use `cars` or `body`.

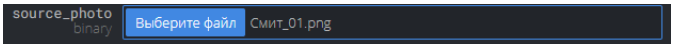
The REQUEST BODY is required and contains multipart/form-data with the following *parameters*:

Name	Schema	Description
`create_from`	string	This field can contain one of the following references: • `detection:<detection id>`: use a `Detection` object (obtained with the `POST /detect`) to select a object on `source_photo`. (`source_photo` must contain the same image that was used for `POST /detect`). • `{face, body, or car}event:<event id>`: create new Object from an Event (source_photo must be empty). • `{face, body, or car}object:<object id>`: use another Object as a template for this object (source_photo must be empty).
`detect_id`	string	Auxiliary parameter.
`mf_selector`	enum	This parameter defines the FindFace Multi default behaviour when there are multiple objects are present in `source_photo`, and `create_from` is not set. Default: `reject`. Allowed: • `reject` – Reject `source_photo` with multiple objects. • `biggest` – Select the biggest object in `source_photo`.
`upload_list`	integer	Add object to this upload list.
`source_photo`	binary	Source photo (required when `create_from` points to a `Detection` or empty).
`frame_coords_left`	integer	Left border of object's bounding box.
`frame_coords_top`	integer	Top border of object's bounding box.
`frame_coords_right`	integer	Right border of object's bounding box.
`frame_coords_bottom`	integer	Bottom border of object's bounding box.
`active`	boolean	`true` if object is enabled. Default: true.
`card`	integer	Related `card` ID.

Attach the photofile to the `source_photo`, point out the related `card id` and paste `detection:<detection id>` into `create_from`, where `detection id` you have got from *detection objects on a photo*. `source_photo` must

contain the same image that was used for `POST /detect`.





CURL example

```
curl -X POST "http://<findface-ip:port>/objects/faces/" \
  -H "Accept: application/json" \
  -H "Content-Language: ru" \
  -H "Accept-Language: ru" \
  -H "Authorization: Token <token>" \
  -H "Content-Type: multipart/form-data" \
  -F "create_from=detection:cf2g86uv54rqhngnq960" \
  -F "source_photo=@_01.png" \
  -F "card=3" \
```

If the response is successful (Created: 201), it returns an object that contains the following *parameters*. *Example*.

Name	Schema	Description
id*	string	ID.
created_date*	date-time	Object creation date.
modified_date*	date-time	Object modification date.
source_photo_name*	string	Filename supplied for source_photo on object creation.
frame_coords_left	integer	Left border of object's bounding box.
frame_coords_top	integer	Top border of object's bounding box.
frame_coords_right	integer	Right border of object's bounding box.
frame_coords_bottom	integer	Bottom border of object's bounding box.
thumbnail*	uri	Object thumbnail.
active	boolean	Default: true. `true` if object is enabled.
features*	<any-key>: strintfloat-boolobjec-tarraynull	Features.
card*	integer	Related card ID.

Response example


```
{
 "card": 3,
 "created_date": "2023-01-16T08:12:55+00:00",
 "modified_date": "1970-01-01T00:00:00+00:00",
 "source_photo_name": "_01.png",
 "source_photo": "http://172.23.218.94/uploads/cards/7w/3/face_%D0%A1%D0%BC%D0%B8%D1%82_
↪01_ftgY5K.png",
 "thumbnail": "http://172.23.218.94/uploads/cards/TD/3/face_%D0%A1%D0%BC%D0%B8%D1%82_01_
↪thumbnail_VyVsIj.png",
 "frame_coords_left": 757,
 "frame_coords_top": 79,
 "frame_coords_right": 948,
 "frame_coords_bottom": 353,
 "active": true,
 "features": {},
 "id": "4493225067924944019",
 "meta": {}
}
```


Useful requests

```
GET /objects/bodies/
POST /objects/bodies/
GET /objects/bodies/{id}/
DELETE /objects/bodies/{id}/
PATCH /objects/bodies/{id}/
GET /objects/cars/
POST /objects/cars/
GET /objects/cars/{id}/
DELETE /objects/cars/{id}/
PATCH /objects/cars/{id}/
GET /objects/faces/
POST /objects/faces/
GET /objects/faces/{id}/
DELETE /objects/faces/{id}/
PATCH /objects/faces/{id}/
```

4.1.8 Videos

List video archives

To list video archives, use the following method:

```
GET /videos/
```

The REQUEST contains *QUERY-STRING PARAMETERS*.

Name	Schema	Description
case_in	array of integer	Select videos included in these cases.
created_date_gt	date-time	Select objects with created_date greater than this value.
created_date_gte	date-time	Select objects with created_date greater than or equal to this value.
created_date_last_n_days	number	Select objects with created_date in last N days.
created_date_lt	date-time	Select objects with created_date less than this value.
created_date_lte	date-time	Select objects with created_date less than or equal to this value.
created_date_nth_full_week	number	Select objects with created_date in last Nth week (including Sunday and Saturday).
created_date_nth_work_week	number	Select objects with created_date in last Nth week (only working days, i.e. excluding Sunday and Saturday).
limit	string	Number of results to return per page.
name	string	Select video archives with this name field.
ordering	string	Available fields: id, created_date, name.
page	string	Pagination cursor value.
save_to	string	Select video archives with this save_to field.

For example, select some number of results for limit.

CURL example

```
curl -X GET "http://<findface-ip:port>/videos/?limit=2" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
```

If the response is successful (OK: 200), it returns an object that contains with *the following parameters*. *Example*.

Name	Schema
next_page	stringnull
prev_page	stringnull
results	array

The results contain an array with the following parameters:

Name	Schema	Description
id*	integer	ID of the video archives.
camera_group*	integer	Camera group.
name*	stringnull	Name of the video archives. Constraints: Max 256 chars.
url	stringnull	URL.
camera	integernull	Camera.
processing_start_date*	date-time	Video processing start date.
active*	boolean	Processing is active.
screenshot*	uri	Screenshot URL.
stream_settings	{...}	Serializer mixin that raises ValidationError if excess fields are presented. Can be used in nested serializers.
source_len*	number	Source length in seconds.
health_status*	{...}	Additional status information.
finished*	boolean	`true` if the video processing is finished.
queued*	boolean	`true` if the video is in the processing queue.
face_count*	integer	Number of created faces.
file_size*	integer	Video file size in the archive.
created_date*	date-time	Object creation date.
body_count*	integer	Number of created bodies.
car_count*	integer	Number of created cars.
case	integernull	Case.
face_cluster_count*	integernull	Count of face clusters created from this video.
body_cluster_count*	integernull	Count of body clusters created from this video.
car_cluster_count*	integernull	Count of car clusters created from this video.

Note: Cases are not supported in FindFace Multi 2.0 and are kept in API only.

Response example


```
{
 "next_page": null,
 "prev_page": null,
 "results": [
   {
     "id": 2,
     "camera_group": 1,
     "name": "file.mp4",
     "url": null,
     "camera": null,
     "processing_start_date": null,
     "active": false,
     "screenshot": "http://<findface-ip:port>/videos/2/screenshot/",
     "stream_settings": {
       "detectors": {
         "face": {
           "filter_max_size": 8192,
           "filter_min_quality": 0.45,
           "filter_min_size": 60,
```


(continues on next page)

(continued from previous page)


```
          "fullframe_crop_rot": false,
          "fullframe_use_png": false,
          "jpeg_quality": 95,
          "overall_only": true,
          "post_best_track_frame": true,
          "post_best_track_normalize": true,
          "post_first_track_frame": false,
          "post_last_track_frame": false,
          "realtime_post_every_interval": false,
          "realtime_post_first_immediately": false,
          "realtime_post_interval": 1,
          "roi": "",
          "track_interpolate_bboxes": true,
          "track_max_duration_frames": 0,
          "track_miss_interval": 1,
          "track_overlap_threshold": 0.25,
          "track_send_history": false,
          "tracker_type": "simple_iou",
          "track_deep_sort_matching_threshold": 0.65,
          "track_deep_sort_filter_unconfirmed_tracks": true
        },
        "body": null,
        "car": null
      },
      "disable_drops": true,
      "ffmpeg_format": "",
      "ffmpeg_params": [],
      "imotion_threshold": 0,
      "play_speed": -1,
      "rot": "",
      "router_timeout_ms": 15000,
      "router_verify_ssl": true,
      "start_stream_timestamp": 0,
      "stream_data_filter": "",
      "use_stream_timestamp": false,
      "video_transform": "",
      "enable_recorder": false,
      "enable_liveness": false
    },
    "source_len": null,
    "health_status": {
      "enabled": false,
      "status": "DISABLED",
      "msg": "",
      "statistic": {
        "processed_duration": 0,
        "faces_posted": 0,
        "faces_failed": 0,
        "faces_not_posted": 0,
        "processing_fps": 0,
        "frames_dropped": 0,
        "frames_processed": 0,
```


(continues on next page)

(continued from previous page)


```
        "frames_imotion_skipped": 0,
        "decoding_soft_errors": 0,
        "frame_width": 0,
        "frame_height": 0,
        "last_stream_timestamp": 0,
        "objects": null,
        "job_starts": 0
      },
      "code": "gray",
      "code_desc": "   "
    },
    "finished": false,
    "queued": false,
    "face_count": 0,
    "file_size": 2259950,
    "created_date": "2023-01-12T08:57:36.811305Z",
    "body_count": 0,
    "car_count": 0,
    "case": null,
    "face_cluster_count": 0,
    "body_cluster_count": 0,
    "car_cluster_count": 0
  }
 ]
}
```


Add a new video archive

To add video archives, use the following method:

```
POST /videos/
```

The REQUEST BODY is required and contains application/json object with the following *parameters*:

Name	Schema	Description
camera_group*	integer	ID value of camera group.
name*	stringnull	Name of the video archive. Constraints: 1 to 256 chars.
url	stringnull	Constraints: Min 1 chars.
camera	integernull	Camera.
stream_settings	{...}	Serializer mixin that raises ValidationError if excess fields are presented. Can be used in nested serializers.
case	integernull	Case.

Note: Cases are not supported in FindFace Multi 2.0 and are kept in API only.

Request example

Tip: This example is given for reference only, substitute your values in the corresponding fields. You may fill in only the required fields, and the others will be by default.


```
{
 "camera_group": 0,
 "name": "A",
 "url": "A",
 "camera": 0,
 "stream_settings": {
   "detectors": {
     "face": {
       "filter_max_size": 0,
       "filter_min_quality": 0,
       "filter_min_size": 0,
       "fullframe_crop_rot": false,
       "fullframe_use_png": false,
       "jpeg_quality": 0,
       "overall_only": false,
       "post_best_track_frame": false,
       "post_best_track_normalize": false,
       "post_first_track_frame": false,
       "post_last_track_frame": false,
       "realtime_post_every_interval": false,
       "realtime_post_first_immediately": false,
       "realtime_post_interval": 0,
       "roi": "string",
       "track_interpolate_bboxes": false,
       "track_max_duration_frames": 0,
       "track_miss_interval": 0,
       "track_overlap_threshold": 0,
       "track_send_history": false,
       "tracker_type": "string",
       "track_deep_sort_matching_threshold": 0,
       "track_deep_sort_filter_unconfirmed_tracks": false
     },
     "body": {
       "filter_max_size": 0,
       "filter_min_quality": 0,
       "filter_min_size": 0,
       "fullframe_crop_rot": false,
       "fullframe_use_png": false,
       "jpeg_quality": 0,
       "overall_only": false,
       "post_best_track_frame": false,
       "post_best_track_normalize": false,
       "post_first_track_frame": false,
       "post_last_track_frame": false,
       "realtime_post_every_interval": false,
       "realtime_post_first_immediately": false,
```


(continues on next page)

(continued from previous page)


```
      "realtime_post_interval": 0,
      "roi": "string",
      "track_interpolate_bboxes": false,
      "track_max_duration_frames": 0,
      "track_miss_interval": 0,
      "track_overlap_threshold": 0,
      "track_send_history": false,
      "tracker_type": "string",
      "track_deep_sort_matching_threshold": 0,
      "track_deep_sort_filter_unconfirmed_tracks": false
    },
    "car": {
      "filter_max_size": 0,
      "filter_min_quality": 0,
      "filter_min_size": 0,
      "fullframe_crop_rot": false,
      "fullframe_use_png": false,
      "jpeg_quality": 0,
      "overall_only": false,
      "post_best_track_frame": false,
      "post_best_track_normalize": false,
      "post_first_track_frame": false,
      "post_last_track_frame": false,
      "realtime_post_every_interval": false,
      "realtime_post_first_immediately": false,
      "realtime_post_interval": 0,
      "roi": "string",
      "track_interpolate_bboxes": false,
      "track_max_duration_frames": 0,
      "track_miss_interval": 0,
      "track_overlap_threshold": 0,
      "track_send_history": false,
      "tracker_type": "string",
      "track_deep_sort_matching_threshold": 0,
      "track_deep_sort_filter_unconfirmed_tracks": false
    }
  },
  "disable_drops": false,
  "ffmpeg_format": "string",
  "ffmpeg_params": [
    "A"
  ],
  "imotion_threshold": 0,
  "play_speed": 0,
  "rot": "string",
  "router_timeout_ms": 0,
  "router_verify_ssl": false,
  "start_stream_timestamp": 0,
  "stream_data_filter": "string",
  "use_stream_timestamp": false,
  "video_transform": "string",
  "enable_recorder": false,
```


(continues on next page)

(continued from previous page)

```
    "enable_liveness": false
  },
  "case": 0
}
```

For example, you may send these parameters.

```
{
  "camera_group": 1,
  "name": "Pitt&Jolie"
}
```

CURL example

```
curl -X POST "http://<findface-ip:port>/videos/" \
  -H "Accept: application/json" \
  -H "Content-Language: ru" \
  -H "Accept-Language: ru" \
  -H "Authorization: Token <token>" \
  -H "Content-Type: application/json" \
  -d '{"camera_group":1,"name":"Pitt&Jolie"}' \
```

If the response is successful (Created: 201), it returns an object that contains *parameters*. *Example*.

Response example

```
{
  "id": 6,
  "camera_group": 1,
  "name": "Pitt&Jolie",
  "url": null,
  "camera": null,
  "processing_start_date": null,
  "active": false,
  "screenshot": "http://<findface-ip:port>/videos/6/screenshot/",
  "stream_settings": {
    "detectors": {
      "face": {
        "filter_max_size": 8192,
        "filter_min_quality": 0.45,
        "filter_min_size": 60,
        "fullframe_crop_rot": false,
        "fullframe_use_png": false,
        "jpeg_quality": 95,
        "overall_only": true,
        "post_best_track_frame": true,
        "post_best_track_normalize": true,
        "post_first_track_frame": false,
        "post_last_track_frame": false,
        "realtime_post_every_interval": false,
```

(continues on next page)

(continued from previous page)


```
      "realtime_post_first_immediately": false,
      "realtime_post_interval": 1,
      "roi": "",
      "track_interpolate_bboxes": true,
      "track_max_duration_frames": 0,
      "track_miss_interval": 1,
      "track_overlap_threshold": 0.25,
      "track_send_history": false,
      "tracker_type": "simple_iou",
      "track_deep_sort_matching_threshold": 0.65,
      "track_deep_sort_filter_unconfirmed_tracks": true
    },
    "body": {
      "filter_max_size": 8192,
      "filter_min_quality": 0.6,
      "filter_min_size": 70,
      "fullframe_crop_rot": false,
      "fullframe_use_png": false,
      "jpeg_quality": 95,
      "overall_only": true,
      "post_best_track_frame": true,
      "post_best_track_normalize": true,
      "post_first_track_frame": false,
      "post_last_track_frame": false,
      "realtime_post_every_interval": false,
      "realtime_post_first_immediately": false,
      "realtime_post_interval": 1,
      "roi": "",
      "track_interpolate_bboxes": true,
      "track_max_duration_frames": 0,
      "track_miss_interval": 1,
      "track_overlap_threshold": 0.25,
      "track_send_history": false,
      "tracker_type": "simple_iou",
      "track_deep_sort_matching_threshold": 0.65,
      "track_deep_sort_filter_unconfirmed_tracks": true
    },
    "car": {
      "filter_max_size": 8192,
      "filter_min_quality": 0.73,
      "filter_min_size": 100,
      "fullframe_crop_rot": false,
      "fullframe_use_png": false,
      "jpeg_quality": 95,
      "overall_only": true,
      "post_best_track_frame": true,
      "post_best_track_normalize": true,
      "post_first_track_frame": false,
      "post_last_track_frame": false,
      "realtime_post_every_interval": false,
      "realtime_post_first_immediately": false,
      "realtime_post_interval": 1,
```


(continues on next page)

(continued from previous page)


```
      "roi": "",
      "track_interpolate_bboxes": true,
      "track_max_duration_frames": 0,
      "track_miss_interval": 1,
      "track_overlap_threshold": 0.25,
      "track_send_history": false,
      "tracker_type": "simple_iou",
      "track_deep_sort_matching_threshold": 0.65,
      "track_deep_sort_filter_unconfirmed_tracks": true
    }
  }
},
"source_len": null,
"health_status": {
  "enabled": false,
  "status": "WAITING_FOR_SYNC",
  "msg": "",
  "statistic": {},
  "code": "red",
  "code_desc": "    .  ."
},
"finished": false,
"queued": false,
"face_count": 0,
"file_size": 0,
"created_date": "2023-01-18T08:38:52.119129Z",
"body_count": 0,
"car_count": 0,
"case": null,
"face_cluster_count": 0,
"body_cluster_count": 0,
"car_cluster_count": 0
}
```


Then point its `id` to upload the source file with `PUT /videos/{id}/upload/source_file/`.

Upload video file

To upload a video file, use the following method:

```
PUT /videos/{id}/upload/source_file/
```

The request contains required PATH PARAMETERS:

Name	Schema	Description
id*	integer	A unique integer value identifying this video.

In REQUEST BODY attach your video source file with `id` of the video archive.

Returns:

- Created: 201 – in success.

- Not Found: 404 – if fail.

CURL example

```
curl -X PUT "http://<findface-ip:port>/videos/6/upload/source_file/" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
 -H "Content-Type: video/mp4" \
 --data-binary @pittjolie.mp4 \
```

Start video archive processing

To start video archive processing, use the following method:

```
POST /videos/{id}/process/
```

The REQUEST contains `id` in PATH PARAMETERS, identifying unique integer value of the video archive.

CURL example

```
curl -X POST "http://<findface-ip:port>/videos/6/process/" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
```

Returns:

- OK: 200 – in success.
- Not Found: 404 – if fail.

Useful requests

```
GET /videos/
POST /videos/
GET /videos/{id}/
PUT /videos/{id}/
DELETE /videos/{id}/
POST /videos/{id}/process/
GET /videos/{id}/screenshot/
POST /videos/{id}/screenshot/
POST /videos/{id}/stop/
PUT /videos/{id}/upload/source_file/
GET /videos/count/
```

4.1.9 Camera groups

Camera Group is a collection of Cameras, set up in same location or used for similar purposes.

List camera groups

To list camera groups, use the following method:

```
GET /camera-groups/
```

The REQUEST contains the following QUERY-STRING PARAMETERS:

Name	Schema	Description
`limit`	integer	Number of results to return.
`ordering`	string	Available fields: `id`, `created_date`, `modified_date`.

To look for camera groups `id` specify some value of `limit` and `ordering`.

CURL example

```
curl -X GET "http://<findface-ip:port>/camera-groups/?limit=3&ordering=id" \
  -H "Accept: application/json" \
  -H "Content-Language: ru" \
  -H "Accept-Language: ru" \
  -H "Authorization: Token <token>" \
```

If the response is successful (OK: 200), it returns an object that contains array of results with *the following parameters*. *Example*.

Tip: * – means required parameters. – read only.

Name	Schema	Description
id*	integer	ID of the camera group.
created_date*	date-time	Object creation date.
modified_date*	date-time	Object modification date.
active	boolean	Process events from this camera group.
name*	string	Short group name, up to 256 characters.
comment	string	Extended description of the user, up to 2048 chars.
deduplicate	boolean	True if events from this camera group are to be deduplicated.
deduplicateDelay	integer	Event deduplication timeout. Constraints: Min 0 Max 10000. Default value is 15.
labels	<any-key>: string	Labels used to allocate a certain findface-video-worker instance to process video streams from this camera group.
permissions	<any-key>: string	Permissions on this camera group.
face_threshold	number-null	Camera group face confidence threshold. Constraints: Min 0Max 1.
body_threshold	number-null	Camera group body confidence threshold. Constraints: Min 0Max 1.
car_threshold	number-null	Camera group car confidence threshold. Constraints: Min 0Max 1.

Response example


```
{
 "results": [
   {
     "id": -1,
     "created_date": "2023-01-16T13:44:36.531501Z",
     "modified_date": "2023-01-16T13:44:36.531519Z",
     "active": true,
     "name": "Video archive default Camera Group",
     "comment": "",
     "deduplicate": false,
     "deduplicateDelay": 15,
     "labels": {},
     "permissions": {
       "1": "edit",
       "2": "view",
       "3": "view"
     },
     "face_threshold": null,
     "body_threshold": null,
     "car_threshold": null
   }
 ]
}
```

Create Camera Group

To add a new camera group, use the following method:

```
POST /camera-groups/
```

The REQUEST BODY is required and contains application/json object with *the following parameters*. *Example*.

Name	Schema	Description
active	boolean	Process events from this camera group.
name*	string	Short group name, up to 256 characters.
comment	string	Extended description of the user, up to 2048 chars.
deduplicate	boolean	True if events from this camera group are to be deduplicated.
deduplicateDelay	integer	Event deduplication timeout. Constraints: Min 0Max 10000. Default value is 15.
labels	<any-key>: string	Labels used to allocate a certain findface-video-worker instance to process video streams from this camera group.
permissions	<any-key>: string	Permissions on this camera group.
face_threshold	number-null	Camera group face confidence threshold. Constraints: Min 0Max 1.
body_threshold	number-null	Camera group body confidence threshold. Constraints: Min 0Max 1.
car_threshold	number-null	Camera group car confidence threshold. Constraints: Min 0Max 1.

CURL example

```
curl -X POST "http://<findface-ip:port>/camera-groups/" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
 -H "Content-Type: application/json" \
 -d '{"active":false,"name":"Test_camera_group","comment":"AAAAAA","deduplicate":false,
↪"deduplicateDelay":0,"labels":{},"permissions":{},"face_threshold":0,"body_threshold
↪":0,"car_threshold":0}' \
```

Request example

Tip: This example is given for reference only, substitute your values in the corresponding fields. You may fill in only the required fields, and the others will be by default.

You may send these parameters:

```
{
 "active": false,
 "name": "Test_camera_group",
 "comment": "AAAAAA",
 "deduplicate": false,
 "deduplicateDelay": 0,
 "labels": {},
 "permissions": {},
 "face_threshold": 0,
 "body_threshold": 0,
 "car_threshold": 0
}
```

If the response is successful (Created: 201), it returns an object that contains *parameters*. *Example*.

Response example

```
{
 "id": 2,
 "created_date": "2023-01-23T12:14:09.885006Z",
 "modified_date": "2023-01-23T12:14:09.885030Z",
 "active": false,
 "name": "Test_camera_group",
 "comment": "AAAAAA",
 "deduplicate": false,
 "deduplicateDelay": 0,
 "labels": {},
 "permissions": {
   "1": "edit"
 },
 "face_threshold": 0,
 "body_threshold": 0,
 "car_threshold": 0
}
```

Use camera group `id` to make a `POST` request for *creating a new camera*.

Useful requests

```
GET /camera-groups/
POST /camera-groups/
GET /camera-groups/{id}/
DELETE /camera-groups/{id}/
PATCH /camera-groups/{id}/
GET /camera-groups/count/
```

4.1.10 Cameras

Create a new camera

To add a new camera, use this method:

```
POST /cameras/
```

The REQUEST BODY contains application/json object with the following *required parameters*:

Name	Schema	Description
group*	integer	Camera group.
name*	string	Camera name. Constraints: 1 to 256 chars.

The other parameters you can see at `http://<findface-ip:port>/api-docs`.

You may send the request with these parameters substituting with your values:

Request example

```
{
 "group": 1,
 "active": false,
 "name": "Camera_name",
 "comment": "AAAAAA",
 "url": "url"
}
```

CURL example

```
curl -X POST "http://<findface-ip:port>/cameras/" \
 -H "Accept: application/json" \
 -H "Content-Language: ru" \
 -H "Accept-Language: ru" \
 -H "Authorization: Token <token>" \
 -H "Content-Type: application/json" \
 -d '{"group":1,"active":false,"name":"Camera_name","comment":"AAAAAA","url":"url"}' \
```

Returns

- Created: 201 – if success.

4.1.11 Search Faces in System

FindFace Multi allows you to search for individuals throughout the entire system.

You may specify a face to search for in one of the following ways:

- by uploading a photo
- by event ID
- by object ID
- by cluster ID

To find an individual by uploading a photo, do the following:

1. Upload a photo with the `POST /detect` *method*.
2. Attach source image file and send the `POST` request.
3. In the response you will get `id` of the object and the coordinates of bbox of the face.
4. If there are multiple faces in the image, select the one of your interest. Copy the value of `id` of the necessary object to use it for search with `GET /cards/humans/` *method* in the `looks_like` field in this way: `detection:<detection id>`.
5. You may also use:
   - `event:<event id>` (face, body, or car)
   - `object:<object id>` (face, body or car)
   - `cluster:<cluster id>`
6. You will see the search results in response. *Example*. If necessary, you can narrow down your search by specifying a watch list, similarity threshold, etc.

4.1.12 Compare Two Faces

FindFace Multi allows you to compare two faces and verify that they belong to the same individual.

You may compare two objects from different sources or object with an object within the same card and return the similarity between them.

Do the following:

1. Perform *face (car, body) detection* on the attached photo and copy `id` of the necessary object.
2. Use the `GET /verify` method.

```
GET /verify
```

The REQUEST contains QUERY-STRING PARAMETERS:

Name	Schema	Description
card_id	string	Min 1 chars. This field accepts the id of the card that holds object to compare.
*object1	string	Min 1 chars. This field can contain one of the following references: • detection:<detection id> – use a Detection object • faceevent:<event id> – use FaceEvent • carevent:<event id> – use CarEvent • bodyevent:<event id> – use BodyEvent • faceobject:<face id> – use FaceObject • carobject:<face id> – use CarObject • bodyobject:<face id> – use BodyObject
object2	string	Min 1 chars. Same as object1.

CURL example

```
curl -X GET "http://<findface-ip:port>/verify?card_id=1&object1=detection
↪%3Acf58e4uv54rotim9jtd0&object2=detection%3Acf58g1mv54rotim9jtdg" \
  -H "Accept: application/json" \
  -H "Content-Language: ru" \
  -H "Accept-Language: ru" \
  -H "Authorization: Token <token>" \
```

If the response is successful (OK: 200), it returns an object that contains confidence.

Name	Schema	Description
confidence*	number	Face similarity score. Constraints: Min 0Max 1.

Response example

```
{
 "confidence": {
   "face_objects": {
     "4493493039043981648": 0.7896046
   },
   "average_conf": 0.7896046
 }
}
```

4.2 Webhooks

You can set up FindFace Multi to automatically send notifications about specific events, episodes, and counter screenshots to a given URL. To do so, create and configure a webhook. In this case, when such an event, episode, or a counter screenshot 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:

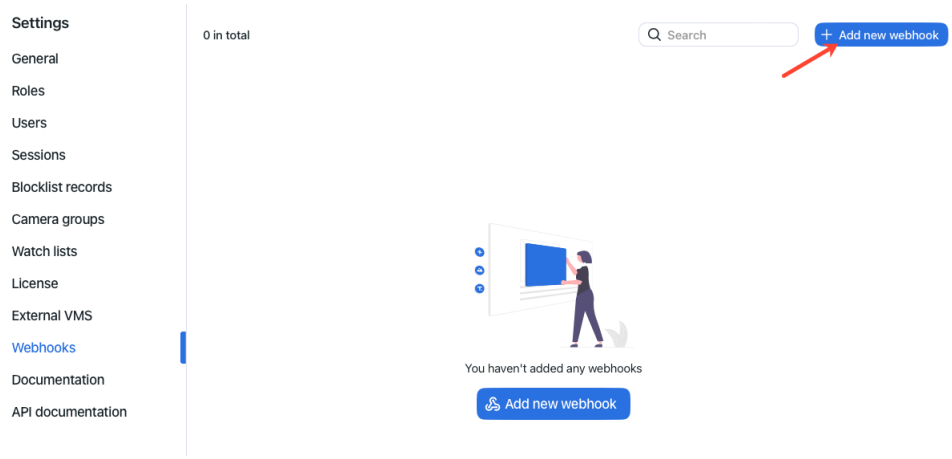
4.2.1 Configure Webhook

Important: You need Administrator privileges to create a webhook.

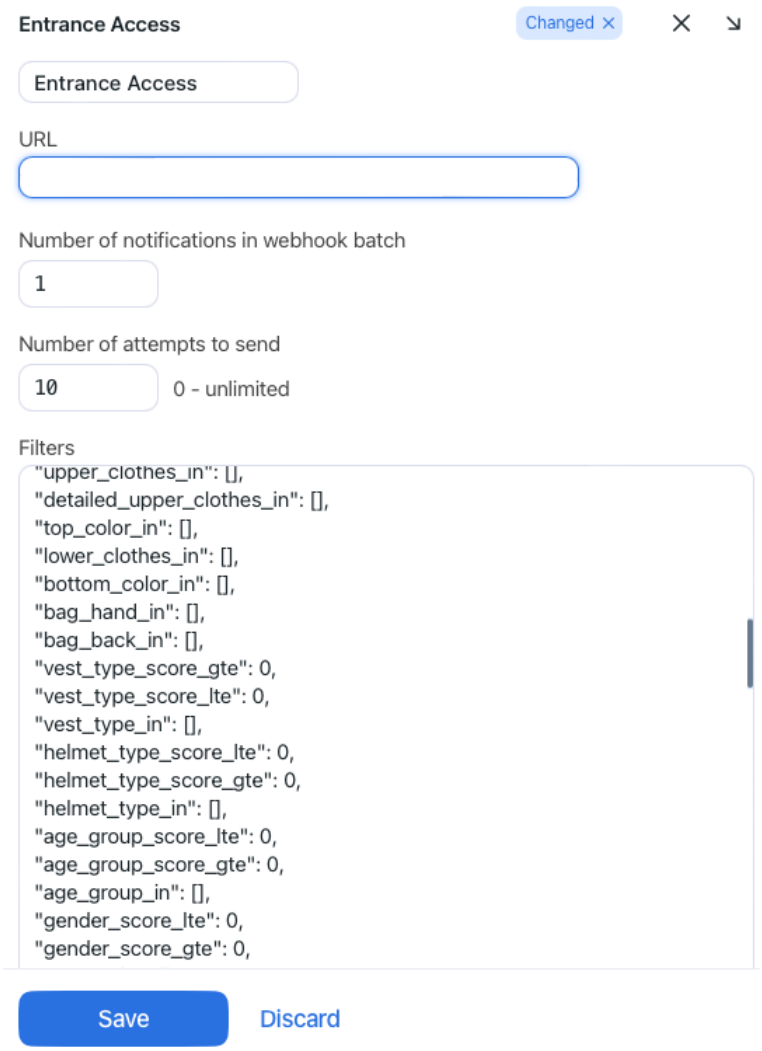
Note: To use the webhooks, make sure that at least one of the following parameters is specified in `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py`: `SERVICE_EXTERNAL_ADDRESS` or `EXTERNAL_ADDRESS`.

To create and configure a webhook, do the following:

1. Navigate to the *Settings* tab. Click *Webhooks*.
2. Click + *New webhook*.



3. Specify the webhook header.



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 counter screenshots that match given filters. The following filters are available:

Generic filters for recognition events (face, body, vehicle):

Sections: `face_events`, `body_events`, `car_events`.

- `"allowed_bs_types"`: object tracking mode, possible values: `overall`, `realtime`.
- `"camera_group_in"`: camera group id, number.
- `"camera_in"`: camera id, number.
- `"confidence_gte"`: minimum confidence, number.
- `"matched_lists_in"`: watch list id, number.
- `"matched_card_in"`: matched record id, number.
- `"matched"`: the `matched` status. Set `true` or `false` if only matched or unmatched events must trigger sending a notification, boolean.

Filters specific for face recognition events:

Section: `face_events`.

- `"liveness_gte"`: minimum algorithm confidence that a face belongs to a live person, decimal number.
- `"headpose_yaw_angle_gte"`: minimum head turn, number (degrees). If a head is turned fully to the left, the yaw angle is `-180`. If it is turned fully to the right, it's `180`. The neutral head position is `0`.
- `"headpose_yaw_angle_lte"`: maximum head turn, number (degrees). If a head is turned fully to the left, the yaw angle is `-180`. If it is turned fully to the right, it's `180`. The neutral head position is `0`.
- `"headpose_pitch_angle_gte"`: minimum head tilt, number (degrees). If a head is tilted forward with the chin touching the chest, it's `-180`. If a head is tilted backward facing the sky, it's `180`. The neutral head position is `0`.
- `"headpose_pitch_angle_lte"`: maximum head tilt, number (degrees). If a head is tilted forward with the chin touching the chest, it's `-180`. If a head is tilted backward facing the sky, it's `180`. The neutral head position is `0`.
- `"gender_in"`: gender, dictionary of strings. Possible values: `"male"`, `"female"`.
- `"age_lte"`: maximum age, number.
- `"age_gte"`: minimum age, number.
- `"glasses_in"`: glasses, dictionary of strings. Possible values: `"none"`, `"eye"` (eyeglasses), `"sun"` (sunglasses).
- `"emotions_in"`: emotions, dictionary of strings. Possible values: `"any"`, `"angry"`, `"fear"`, `"sad"`, `"neutral"`, `"disgust"`, `"happy"`, `"surprise"`.
- `"beard_in"`: beard, dictionary of strings. Possible values: `"none"`, `"beard"`.
- `"medmask_in"`: face mask, dictionary of strings. Possible values: `"correct"`, `"incorrect"`, `"none"`.
- `"liveness_in"`: liveness, dictionary of strings. Possible values: `"real"`, `"fake"`.

Filters specific for body recognition events:

Section: `body_events`.

- "`headwear_in`": type of headgear (hat/cap, hood/headscarf, none), dictionary of strings. Possible values: "`hat`", "`hood`", "`none`".
- "`upper_clothes_in`": generalized category of upper body wear (long sleeves, short sleeves, no sleeve), dictionary of strings. Possible values: "`long_sleeves`", "`short_sleeves`", "`without_sleeves`".
- "`detailed_upper_clothes_in`": specific type of upper body wear (jacket, coat, sleeveless vest, sweatshirt, T-shirt, shirt, dress), dictionary of strings. Possible values: "`jacket`", "`coat`", "`sleeveless`", "`sweatshirt`", "`t-shirt`", "`shirt`", "`dress`".
- "`top_color_in`": top clothing color, dictionary of strings. Possible values: `white`, "`black`", "`grey`", "`brown`", "`red`", "`orange`", "`yellow`", "`green`", "`lightblue`", "`blue`", "`purple`", "`pink`", "`beige`", "`violet`".
- "`lower_clothes_in`": type of lower body wear (pants, skirt, shorts, nondescript), dictionary of strings. Possible values: "`pants`", "`obscured`", "`skirt`", "`shorts`".
- "`bottom_color_in`": bottom clothing color, dictionary of strings. Possible values: `white`, "`black`", "`grey`", "`brown`", "`red`", "`orange`", "`yellow`", "`green`", "`lightblue`", "`blue`", "`purple`", "`pink`", "`beige`", "`violet`".
- "`bag_hand_in`": whether a person has a bag in hand(s), dictionary of strings. Possible values: "`none`", "`hand`".
- "`bag_back_in`": whether a person has a bag on the back, dictionary of strings. Possible values: "`none`", "`back`".
- "`vest_type_score_gte`": minimum recognition confidence of a vest presence, number.
- "`vest_type_score_lte`": maximum recognition confidence of a vest presence, number.
- "`vest_type_in`": presence of personal protective equipment in the form of a vest, dictionary of strings. Possible values: "`green/yellow`", "`orange`", "`not_visible`", "`none`".
- "`helmet_type_score_lte`": maximum recognition confidence of a helmet presence, number.
- "`helmet_type_score_gte`": minimum recognition confidence of a helmet presence, number.
- "`helmet_type_in`": presence of personal protective equipment in the form of a helmet, dictionary of strings. Possible values: "`red/orange`", "`white`", "`other`", "`not_visible`", "`none`".
- "`age_group_score_lte`": maximum age group recognition confidence, number.
- "`age_group_score_gte`": minimum age group recognition confidence, number.
- "`age_group_in`": age by group, dictionary of strings. Possible values: "`0-16`", "`17-35`", "`36-50`", "`50+`".
- "`gender_score_lte`": maximum gender recognition confidence, number.
- "`gender_score_gte`": minimum gender recognition confidence, number.
- "`gender_in`": gender, dictionary of strings. Possible values: "`male`", "`female`".

Filters specific for vehicle recognition events:

Section: `car_events`.

- `"category_confidence_gte"`: minimum recognition confidence of the vehicle category, number.
- `"category_confidence_lte"`: maximum recognition confidence of the vehicle category, number.
- `"category_type_in"`: vehicle category, dictionary of strings. Possible values: `"unknown"`, `"A"`, `"B"`, `"BE"`, `"C"`, `"CE"`, `"D"`, `"DE"`, `"other"`.
- `"color_in"` vehicle color, dictionary of strings. Possible values: `"beige"`, `"black"`, `"blue"`, `"brown"`, `"cyan"`, `"gold"`, `"green"`, `"grey"`, `"orange"`, `"pink"`, `"purple"`, `"red"`, `"violet"`, `"white"`, `"yellow"`, `"coming_soon"`, `"unknown"`.
- `"body_in"`: vehicle body style, dictionary of strings. Possible values: `"suv"`, `"sedan"`, `"crossover"`, `"convertible"`, `"coupe"`, `"wagon"`, `"cab"`, `"minibus"`, `"minivan"`, `"limousine"`, `"coming_soon"`, `"unknown"`.
- `"make_in"`: vehicle make, dictionary of strings. Check out the corresponding event filter to see what makes are supported in the current version.
- `"model_in"`: vehicle model, dictionary of strings. Check out the corresponding event filter to see what models are supported in the current version.
- `"license_plate_number_in"`: license plate number, dictionary of strings. Wildcards are not supported.
- `"license_plate_color_in"`: license plate color, dictionary of strings. Possible values: `"unknown"`, `"white"`, `"yellow"`, `"blue"`, `"green"`, `"grey"`.
- `"license_plate_color_confidence_lte"`: maximum recognition confidence of the license plate color, number.
- `"license_plate_color_confidence_gte"`: minimum recognition confidence of the license plate color, number.
- `"license_plate_country_in"`: license plate country, dictionary of strings. Check out the corresponding event filter to see what countries are supported in the current version.
- `"license_plate_region_in"`: license plate region, dictionary of strings. Check out the corresponding event filter to see what regions are supported in the current version.
- `"special_vehicle_type_in"`: special vehicle type, dictionary of strings. Possible values: `"taxi"`, `"route_transport"`, `"car_sharing"`, `"ambulance"`, `"police"`, `"rescue_service"`, `"gas_service"`, `"military"`, `"road_service"`, `"other_special"`, `"not_special"`, `"unknown"`.
- `"weight_type_in"`: vehicle weight type. Possible values: `"B_light"`, `"B_heavy"`, `"C_light"`, `"C_heavy"`, `"D_light"`, `"D_long"`, `"other"`.
- `"weight_type_confidence_lte"`: maximum recognition confidence of the vehicle weight type, number.
- `"weight_type_confidence_gte"`: minimum recognition confidence of the vehicle weight type, number.
- `"orientation_in"`: vehicle orientation, dictionary of strings. Possible values: `"front"`, `"back"`, `"side"`.
- `"orientation_confidence_lte"`: maximum recognition confidence of the vehicle orientation, number.
- `"orientation_confidence_gte"`: minimum recognition confidence of the vehicle orientation, number.

Episodes with people:

Section: `human_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.
- `"face_matched"`: the `matched` status of face events in an episode, boolean. Set `true` if only matched faces or `false` if only unmatched faces must trigger sending a notification.
- `"body_matched"`: the `matched` status of body events in an episode, boolean. Set `true` if only matched bodies or `false` if only unmatched bodies must trigger sending a notification.
- `"events_count_gte"`: minimum number of events in an episode, number.
- `"events_count_lte"`: maximum number of events in an episode, number.

Episodes with vehicles:

Section: `car_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.
- `"car_matched"`: the `matched` status of car events in an episode, boolean. Set `true` if only matched cars or `false` if only unmatched cars must trigger sending a notification.
- `"events_count_gte"`: minimum number of events in an episode, number.
- `"events_count_lte"`: maximum number of events in an episode, number.

Counter screenshots:

Section: `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 screenshot, number.
- `"faces_lte"`: maximum number of faces in a counter screenshot, number.
- `"silhouettes_gte"`: minimum number of bodies in a counter screenshot, number.
- `"silhouettes_lte"`: maximum number of bodies in a counter screenshot, number.
- `"cars_gte"`: minimum number of vehicles in a counter screenshot, number.
- `"cars_lte"`: maximum number of vehicles in a counter screenshot, number.

- "proximity_min_lte": send a notification if the minimum detected distance in meters is less than this value, number.
- "proximity_min_gte": send a notification if the minimum detected distance in meters is greater than this value, number.
- "proximity_avg_lte": send a notification if the average detected distance in meters is less than this value, number.
- "proximity_avg_gte": send a notification if the average detected distance in meters is greater than this value, number.
- "proximity_max_lte": send a notification if the maximum detected distance in meters is less than this value, number.
- "proximity_max_gte": send a notification if the maximum detected distance in meters is greater than this value, number.

```
{
"face_events": {
"matched_lists_in": [],
"camera_group_in": [],
"camera_in": [],
"matched_card_in": [],
"matched": true,
"confidence_gte": 0.75,
"allowed_bs_types": [
"overall",
"realtime"
],
"liveness_gte": 0.65,
"headpose_yaw_angle_gte": -180,
"headpose_yaw_angle_lte": 180,
"headpose_pitch_angle_gte": -180,
"headpose_pitch_angle_lte": 180,
"gender_in": [],
"age_lte": 0,
"age_gte": 0,
"glasses_in": [],
"emotions_in": [],
"beard_in": [],
"medmask_in": [],
"liveness_in": []
},
"body_events": {
"matched_lists_in": [],
"camera_group_in": [],
"camera_in": [],
"matched_card_in": [],
"matched": true,
"confidence_gte": 0.75,
"allowed_bs_types": [
"overall",
"realtime"
],
"headwear_in": [],
```

(continues on next page)

(continued from previous page)


```
"upper_clothes_in": [],
"detailed_upper_clothes_in": [],
"top_color_in": [],
"lower_clothes_in": [],
"bottom_color_in": [],
"bag_hand_in": [],
"bag_back_in": [],
"vest_type_score_gte": 0,
"vest_type_score_lte": 0,
"vest_type_in": [],
"helmet_type_score_lte": 0,
"helmet_type_score_gte": 0,
"helmet_type_in": [],
"age_group_score_lte": 0,
"age_group_score_gte": 0,
"age_group_in": [],
"gender_score_lte": 0,
"gender_score_gte": 0,
"gender_in": []
},
"car_events": {
"matched_lists_in": [],
"camera_group_in": [],
"camera_in": [],
"matched_card_in": [],
"matched": true,
"confidence_gte": 0.75,
"allowed_bs_types": [
"overall",
"realtime"
],
"category_confidence_gte": 0,
"category_confidence_lte": 0,
"category_type_in": [],
"color_in": [],
"body_in": [],
"make_in": [],
"model_in": [],
"license_plate_number_in": [],
"license_plate_color_in": [],
"license_plate_color_confidence_lte": 0,
"license_plate_color_confidence_gte": 0,
"license_plate_country_in": [],
"license_plate_region_in": [],
"special_vehicle_type_in": [],
"weight_type_in": [],
"weight_type_confidence_lte": 0,
"weight_type_confidence_gte": 0,
"orientation_in": [],
"orientation_confidence_lte": 0,
"orientation_confidence_gte": 0
},
```


(continues on next page)

(continued from previous page)

```
"human_episodes": {
"allowed_types": [
"episode_open",
"episode_event",
"episode_close"
],
"matched_lists_in": [],
"camera_in": [],
"camera_group_in": [],
"events_count_gte": 0,
"events_count_lte": 999,
"face_matched": true,
"body_matched": true },
"car_episodes": {
"allowed_types": [
"episode_open",
"episode_event",
"episode_close"
],
"matched_lists_in": [],
"camera_in": [],
"camera_group_in": [],
"events_count_gte": 0,
"events_count_lte": 999,
"car_matched": true },
"counters": {
"counter_in": [],
"camera_in": [],
"camera_group_in": [],
"faces_gte": 1,
"faces_lte": 100,
"silhouettes_gte": 1,
"silhouettes_lte": 100,
"cars_gte": 1,
"cars_lte": 100,
"proximity_min_lte": 100,
"proximity_min_gte": 0,
"proximity_avg_lte": 100,
"proximity_avg_gte": 0,
"proximity_max_lte": 100,
"proximity_max_gte": 0
}
}
```

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 vehicle events:

```
{ "car_events": {} }
```

Example #2. Get notifications of the opening of matched episodes that feature human faces and bodies:

```
{ "human_episodes": { "allowed_types": ["episode_open"], "face_matched": true,
↪"body_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 a body event from the camera group 1 or 3 if a matched record is 12 or 25.

```
{
        "body_events": {
                "camera_group_in": [1, 3],
                "matched_card_in": [12, 25],
        },
}
```

8. Check *Active*.
9. Click *Save*.

4.2.2 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 screenshots.

To view a webhook pulling status, execute:

```
sudo journalctl CONTAINER_NAME=findface-multi-findface-multi-legacy-1 -f | grep 'Webhook'
```

4.2.3 Verbose Webhooks

By default, webhook notifications contain only ids of such entities as cards, watch lists, cameras, and camera groups. It is possible to get the full content of the mentioned entities by switching webhooks to the verbose mode.

To do so, open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` configuration file and set `'VERBOSE_WEBHOOKS': True`:

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py

...
FFSECURITY = {
    ...
    # send serialized cards, card-lists, camera and camera groups in webhooks
    'VERBOSE_WEBHOOKS': True,
    ...
}
...
```

Be sure to restart the `findface-multi-findface-multi-legacy-1` container after making changes.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

4.3 External VMS

On the FindFace Multi side, a mechanism of integration with third-party VMS is implemented, allowing you to add cameras from the latter to FindFace Multi and perform their processing. When a camera from a VMS is added, all the actions available for a regular FindFace Multi camera become available for it.

Integration with third-party VMS is possible through a plug-in layer. Interaction between the plugin and FindFace Multi is performed via HTTP. The plugin interacts with the VMS, FindFace Multi interacts with the plugin.

In this section:

This is the step-by-step configuration guide for installing and configuring External VMS feature in FindFace Multi. The reader should have strong experience of administering Linux OS and Docker containers.

4.3.1 Creating a service account for VMS integration plugin

Create a user (username: **ntech**) with specific set of permissions. This will be used as the service account for the integration plugin. To do so, execute the command below:

```
adduser --system --disabled-password --disabled-login --home /var/empty \
                --no-create-home --quiet --force-badname --group ntech
```

4.3.2 FindFace Multi

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` and locate `External VMS Integration settings` section.
2. Change the `ENABLED` parameter to `True`.
3. Take a note of the `TOKEN` value, as you will need it to configure the plugin later. (The default value is `VmsPluginToken PLUGIN_TOKEN`).

```
...
# -- External VMS integration settings --
   # cleanup settings
   'EXTERNAL_VMS_EVENTS_MAX_AGE': 0,
   'EXTERNAL_VMS_SEND_EVENTS_STATUS_MAX_AGE': 0,
   'EXTERNAL_VMS': {
       'ENABLED': True,
       'PLUGIN_ADDRESS': 'http://127.0.0.1:18333',
       'TOKEN': 'VmsPluginToken PLUGIN_TOKEN',
       'EVENT_SENDER': {
           'ENABLED': True,
           'ALLOWED_TYPES': ['face'],
           'SENDER_TASKS': 1,
           'MAX_SEND_ATTEMPTS': 1,
           'MIN_EVENT_SEND_TIMEOUT': 0.1,
           'MAX_EVENT_SEND_TIMEOUT': 100,
           'RESPONSE_TIMEOUT': 10,
       },
   },
...
```

Tip: In our example, we are installing the plugin on the same host where FindFace Multi installation resides. In the case of using the separate machine, please consider specifying the proper address of the server where the VMS integration plugin resides in the `PLUGIN_ADDRESS` field.

Note: If you need to send the events to VMS, please also set the ENABLED value to `True` in the `EVENT_SENDER` section.

4. Restart the `findface-multi-findface-multi-legacy-1` container.

```
sudo docker container restart findface-multi-findface-multi-legacy-1
```

Brief description of the parameters available below:

Parameter	Description
EXTERNAL_VMS_EVENTS_MAX_AGE	Time to store the events received from an external VMS system (days). 0 – unlimited.
EXTERNAL_VMS_SEND_EVENTS_STATUS_AGE	Time to store the events sent to an external VMS system (days). 0 – unlimited.
EXTERNAL_VMS: ENABLED	The status of FindFace VMS Integration module. If enabled, the `External VMS` option will appear in the user interface.
PLUGIN_ADDRESS	Plugin IP address.
TOKEN	Token required for VMS Integration plugin to authenticate in Findface Multi (it must match the token in the plugin configuration file).
EVENT_SENDER:ENABLED	If `True`, all matched events in Findface Multi will be sent to an external VMS.
ALLOWED_TYPES	Objects types to send. At present, only face objects supported.
SENDER_TASKS	The number of simultaneous tasks to send events to the plugin. Default value = 1. It can be increased if required, please consult NtechLab Support Team (support@ntechlab.com).
MAX_SEND_ATTEMPTS	The number of attempts to send before it is considered unsuccessful.
MIN_EVENT_SEND_TIMEOUT MAX_EVENT_SEND_TIMEOUT	The time between sending attempts, if a failed sending occurs. It will grow exponentially from minimum to maximum as long as the number of attempts MAX_SEND_ATTEMPTS is increasing.
RESPONSE_TIMEOUT	Time to wait for a response from the plugin when sending an event.

4.3.3 PostgreSQL

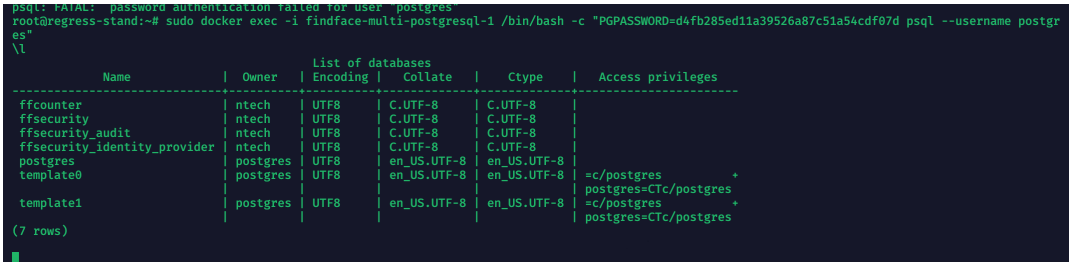
1. Open the file `/opt/findface-multi/docker-compose.yaml` and take a note of `POSTGRES_PASSWORD` value. We will need this for the next step.

```
sudo vi /opt/findface-multi/docker-compose.yaml
...
 postgresql:
   environment: {POSTGRESQL_ALLOW_REMOTE_CONNECTIONS: 'no', POSTGRES_
↪PASSWORD: POSTGRES_PASSWORD}
...
```

2. Sign in to the `findface-multi-postgresql-1` container via executing the following command:

```
sudo docker exec -i findface-multi-postgresql-1 /bin/bash -c "PGPASSWORD=
↪{POSTGRES_PASSWORD} psql --username postgres"
```

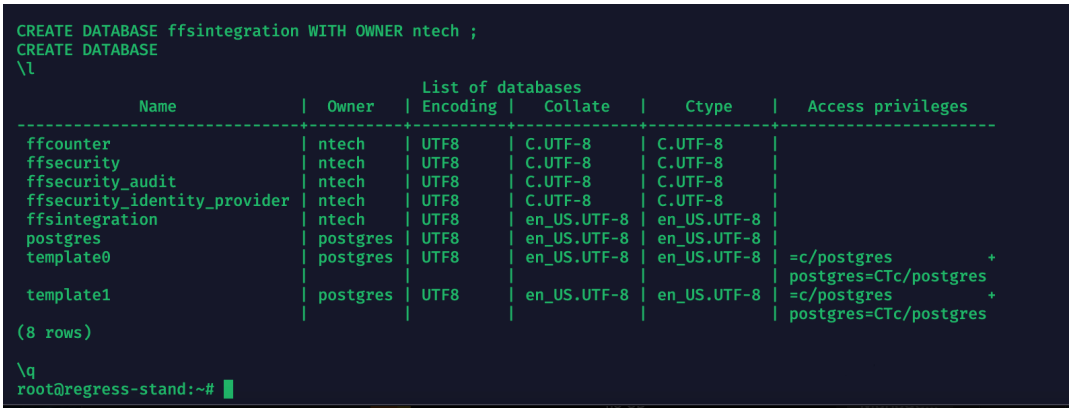
To ensure you've signed in successfully, use the `\l` command to view the list of databases:



3. Run the following command to create the new database required for external VMS functionality:

```
CREATE DATABASE ffsintegration WITH OWNER ntech ;
```

4. Validate the database creation via executing `\l` again. Once completed, use `\q` parameter to exit PostgreSQL.



4.3.4 Installing and configuring FindFace VMS Integration Plugin

1. Open the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` file and locate `DATABASES` section. Take a note of the `PASSWORD` value for user `ntech`:

```
sudo vi /opt/findface-multi/configs/findface-multi-legacy/findface-multi-
↪legacy.py


...
# camera groups, watchlists and so on. Only PostgreSQL is supported.
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'DISABLE_SERVER_SIDE_CURSORS': True,
        'NAME': 'ffsecurity', 'HOST': '127.0.0.1', 'PORT': 5439, 'USER':
↪'ntech', 'PASSWORD': 'PASSWORD'
```

(continues on next page)

(continued from previous page)

```
    }
}
```

2. Download FindFace plugin installation file (.deb) using the link from NtechLab representative.
3. Install it using the command:

```
sudo dpkg -i findface-vms-integration-plugin_0.1.0_amd64.deb
```

4. Open the config file of the plugin, located at `/etc/findface-vms-integration-plugin.conf`. Uncomment and insert the following data here:

```
sudo vi /etc/findface-vms-integration-plugin.conf
```

   - POSTGRES_SERVER (specify the server, where postgresql container is running)
   - POSTGRES_PORT (use the value different from the one specified in config.py, for example, use `5432`)
   - POSTGRES_USER (specify `ntech`)
   - POSTGRES_PASSWORD (use the `PASSWORD` value for user `ntech` from the `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` file, see *1.*)
   - POSTGRES_DB (specify `ffsintegration`)
   - PUBLIC_URL (default: http://127.0.0.1:18333)
   - FFS_PUBLIC_URL (default: http://127.0.0.1)
   - FFS_TOKEN (specify the token as in the section `EXTERNAL_VMS` → `TOKEN` in `/opt/findface-multi/configs/findface-multi-legacy/findface-multi-legacy.py` without prefix `VmsPluginToken`)

   The resulting configuration file will look `as follows`:

   Tip: In our example, we are installing the plugin on the same host where FindFace Multi installation resides. In case of using the separate machine, please consider specifying the proper address of FindFace Multi server in the `POSTGRES_SERVER` field and the FindFace Multi URL in the `FFS_PUBLIC_URL` field.

   The rest parameters are optional and not required for the initial configuration.

5. Update the database using the following command:

```
sudo /opt/findface-vms-integration-plugin/bin/vms-integration-plugin-
↪alembic upgrade head
```

   In case of success, the output will look as follows:

```
root@regress-stand:/home/presale# sudo /opt/findface-vms-integration-plugin/bin/vms-integration-plugin-alembic upgrade head
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 809ccbb29324, init
root@regress-stand:/home/presale#
```

6. To start the plugin service and ensure the service is launching automatically, execute the following command:

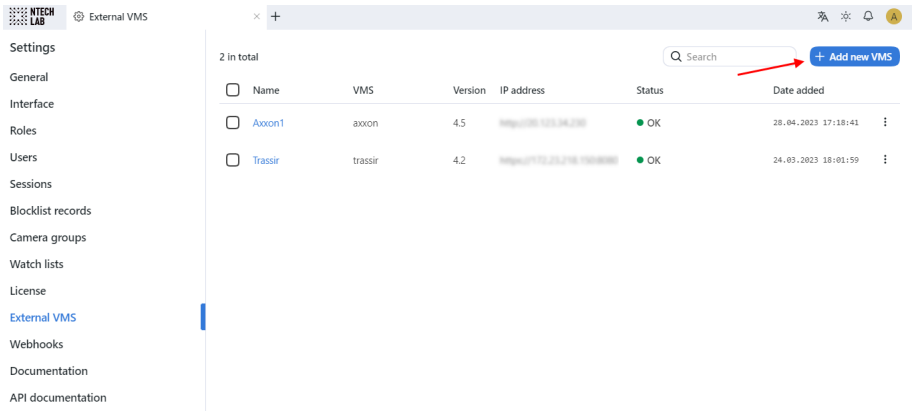
```
sudo systemctl start findface-vms-integration-plugin && sudo systemctl␣
↪enable findface-vms-integration-plugin && sudo systemctl status findface-
↪vms-integration-plugin
```

```
root@regress-stand:~# systemctl start findface-vms-integration-plugin
root@regress-stand:~# systemctl enable findface-vms-integration-plugin
Created symlink /etc/systemd/system/multi-user.target.wants/findface-vms-integration-plugin.service → /lib/systemd/system/findface-vms-integration-plu
gin.service.
root@regress-stand:~#
```

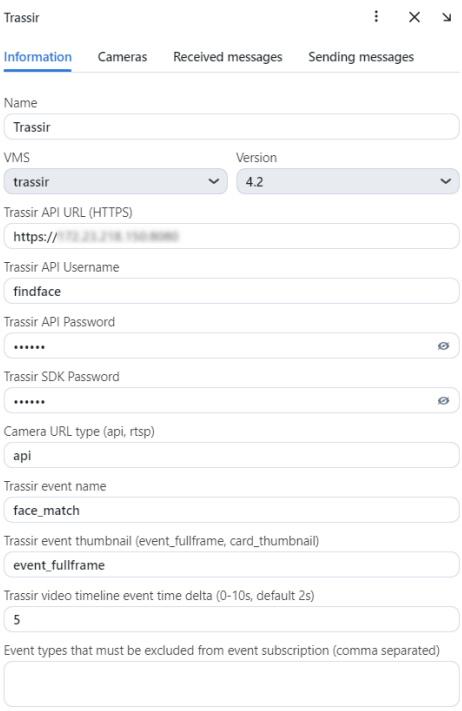
Warning: It is necessary to set the same time zone on the external VMS server as on the FindFace Multi server for correct playback of video archives from the external VMS.

4.3.5 Adding External VMS in FindFace Multi

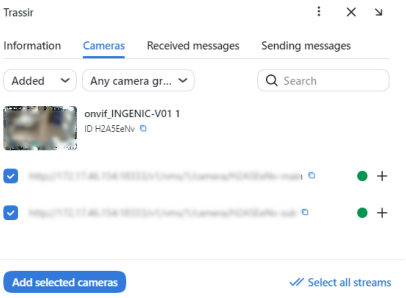
1. Sign in to FindFace Multi in your web browser, go to *Settings* -> *External VMS* section. Click on the *Add new VMS* button.



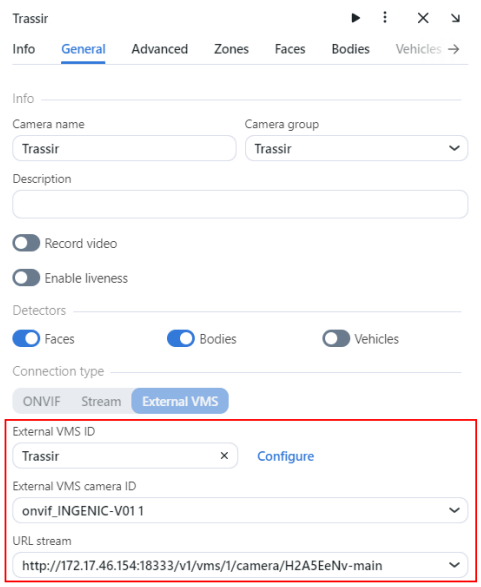
2. Specify your VMS parameters in the opened windows and click on the *Save* button once ready:



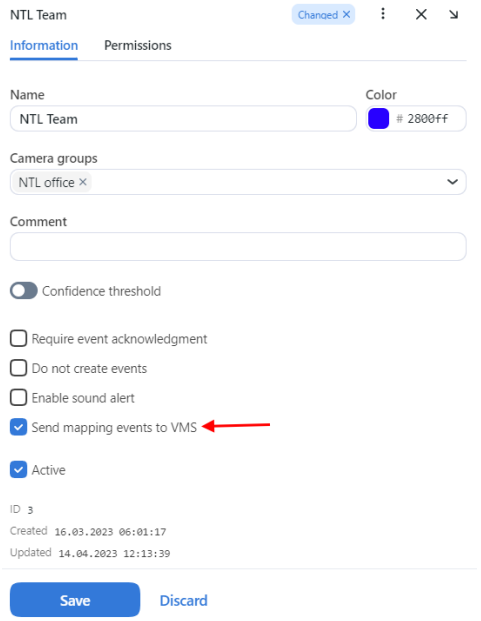
3. Once completed now you can manage the streams from your VMS in the Cameras tab. For instance, you may add all or selected cameras in FindFace Multi instantly.



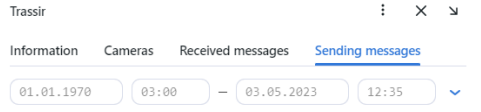
4. Should you need to send the events to an external VMS, In the Camera settings added from VMS, check if it configured to send the events to the external VMS:



5. In the required watch list properties, activate the checkbox `Send mapping events to VMS` and click on *Save* button to apply the changes.



6. Sent and received messages can be viewed in the corresponding tabs of the target VMS:



Now, your FindFace Multi installation is enhanced by external VMS support. You may connect as many VMS systems

and required and add existing video streams in FindFace seamlessly within just a few clicks.

4.4 External Detectors

It is possible to integrate FindFace Multi with the external detectors that can source frames for object recognition, e.g., Access Control terminals. In this case, once FindFace Multi receives a frame from an external detector, it will initiate the extraction of the object's feature vector and event creation. You can work with these events by analogy with the *events* from CCTV cameras.

Important: To use an external detector for object recognition, liveness detection must be enabled. If you have not enabled it during *installation*, you can do it now, using information from the *Enable Face Liveness Detection* section.

The liveness model `liveness.web.v0` comes in the default configuration and is used for authentication by face. If your use case implies connection to an external detector for object recognition, replace the `liveness.web.v0` model with the `liveness.pacs.v2` model in the `findface-extraction-api.yaml` configuration file. Since both models employ the same normalizer `facenorm.multicrop_full_crop2x_size400.cpu.fnk` there is no need to change it.

Replace the liveness model in the `/opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml` configuration file:

GPU

```
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml

extractors:
  ...
  models:
    face_liveness: faceattr/liveness.pacs.v2.gpu.fnk
    ...
```

CPU

```
sudo vi /opt/findface-multi/configs/findface-extraction-api/findface-extraction-api.yaml

extractors:
  ...
  models:
    face_liveness: faceattr/liveness.pacs.v2.cpu.fnk
    ...
```

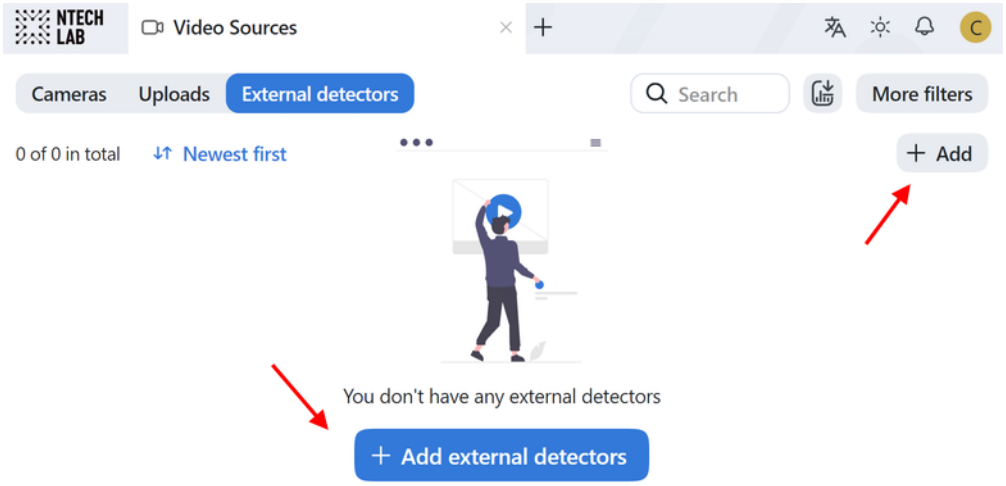
Restart the `findface-multi-findface-extraction-api-1` container.

```
sudo docker container restart findface-multi-findface-extraction-api-1
```

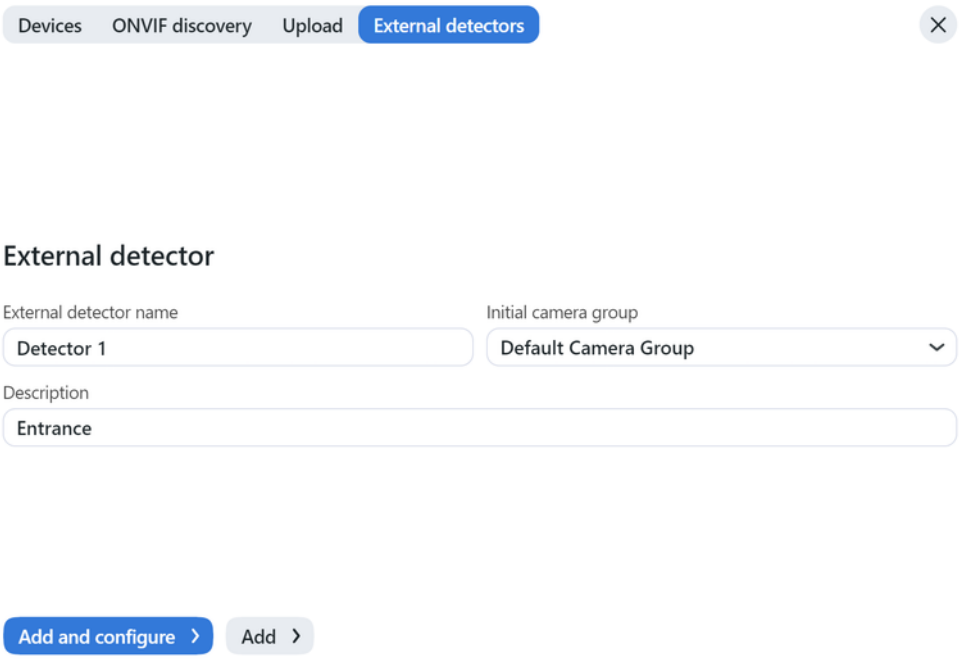
The external detector integration is done through *HTTP API*. After the primary configuration, FindFace Multi will issue a token. Specify this token in every API request sent by the external detector to FindFace Multi to authorize the device.

To integrate an external detector with FindFace Multi, do the following:

1. Navigate to the *Video Sources -> External Detectors*.
2. Click + *Add* or *Add external detectors* if you don't have any detectors yet.

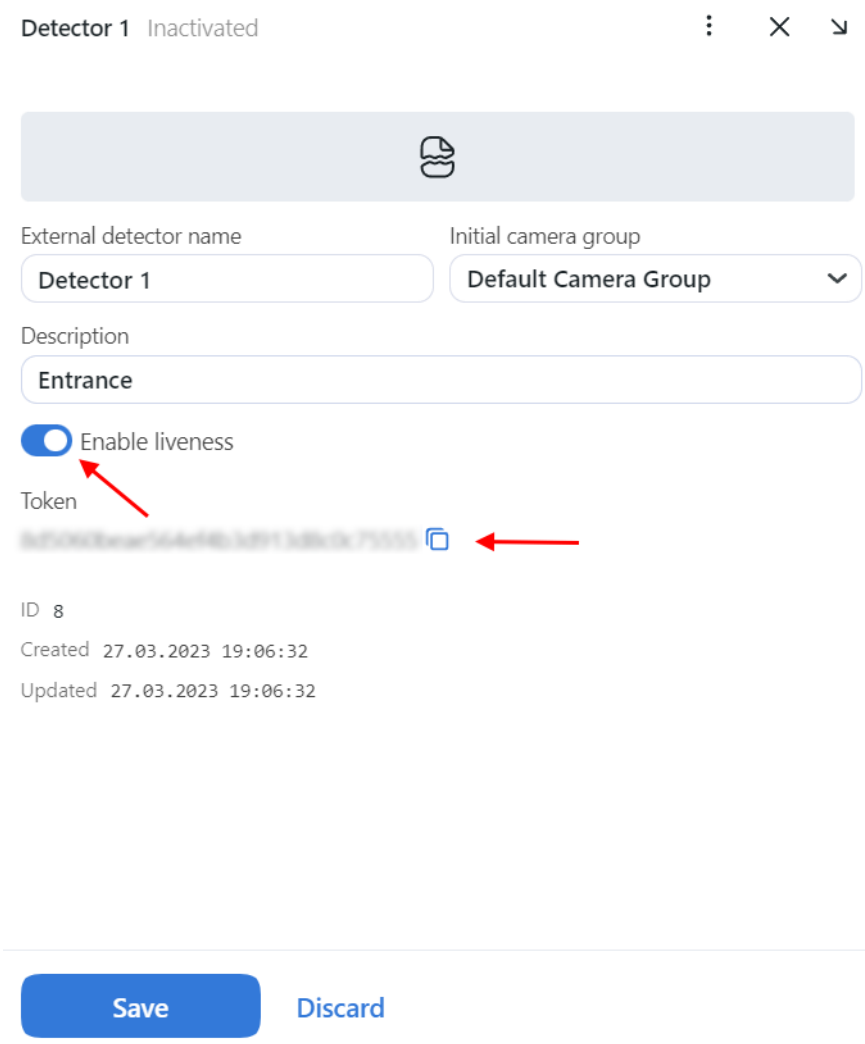


3. In the opened tab, specify the external detector name and description. Add the external detector to a camera group, so it will be more convenient to filter events from this device later.

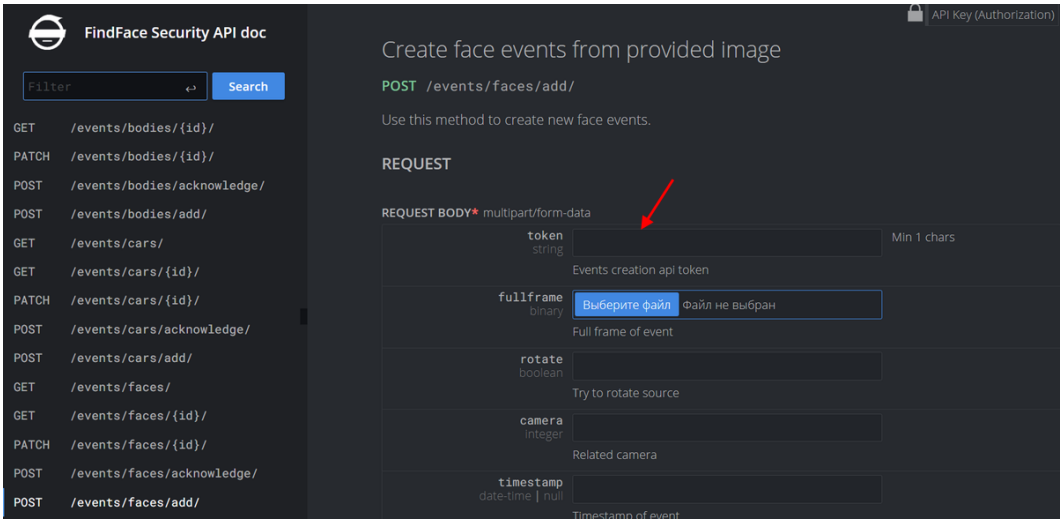


Tip: You can allot a separate camera group specifically to external detectors.

4. On the *Add and configure* tab, enable liveness. On the same tab you will see a token.



5. Specify this token in every API request that the external detector sends to FindFace Multi to create an event. As a result, frames passed in the requests will be associated with the external detector's relevant camera and processed by analogy with the frames from CCTV cameras.



Detailed interactive documentation on the FindFace Multi HTTP API is available after installation at `http://<findface-ip:port>/api-docs`. Learn and try it out.

Tip: You can also find it by navigating to *Settings -> API Documentation* in the web interface.
