FindFace Security

Release 4.0.2

NtechLab

Jun 16, 2023

Contents

FindFace Security is a video-based biometric identification system that automates Security and Hospitality Operations Management. Based on FindFace Enterprise Server, a cutting-edge AI facial recognition technology, FindFace Security 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 Security detects and identifies faces of the unwanted persons and VIP guests in video, and notifies security and hospitality managers about their arrival. It can also recognize such facial attributes as gender, age, emotions, glasses, and beard, and display this information in a face recognition event.

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.

Early recognition of the arrival of unwanted persons and VIP guests allows for solving the following problems:

- Operational losses due to fraudulent activity
- Reputational losses and conflicts
- Better catering to the needs of VIP guests
- Prevention of life-threatening situations

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

You are going to find this guide most useful if you are an expert of the following kind:

- FindFace Security administrator
- Security manager
- Hospitality manager
- Maintenance engineer
- System integration engineer who is going to integrate face recognition services into their system.

Administrator's Guide

1.1 Architecture

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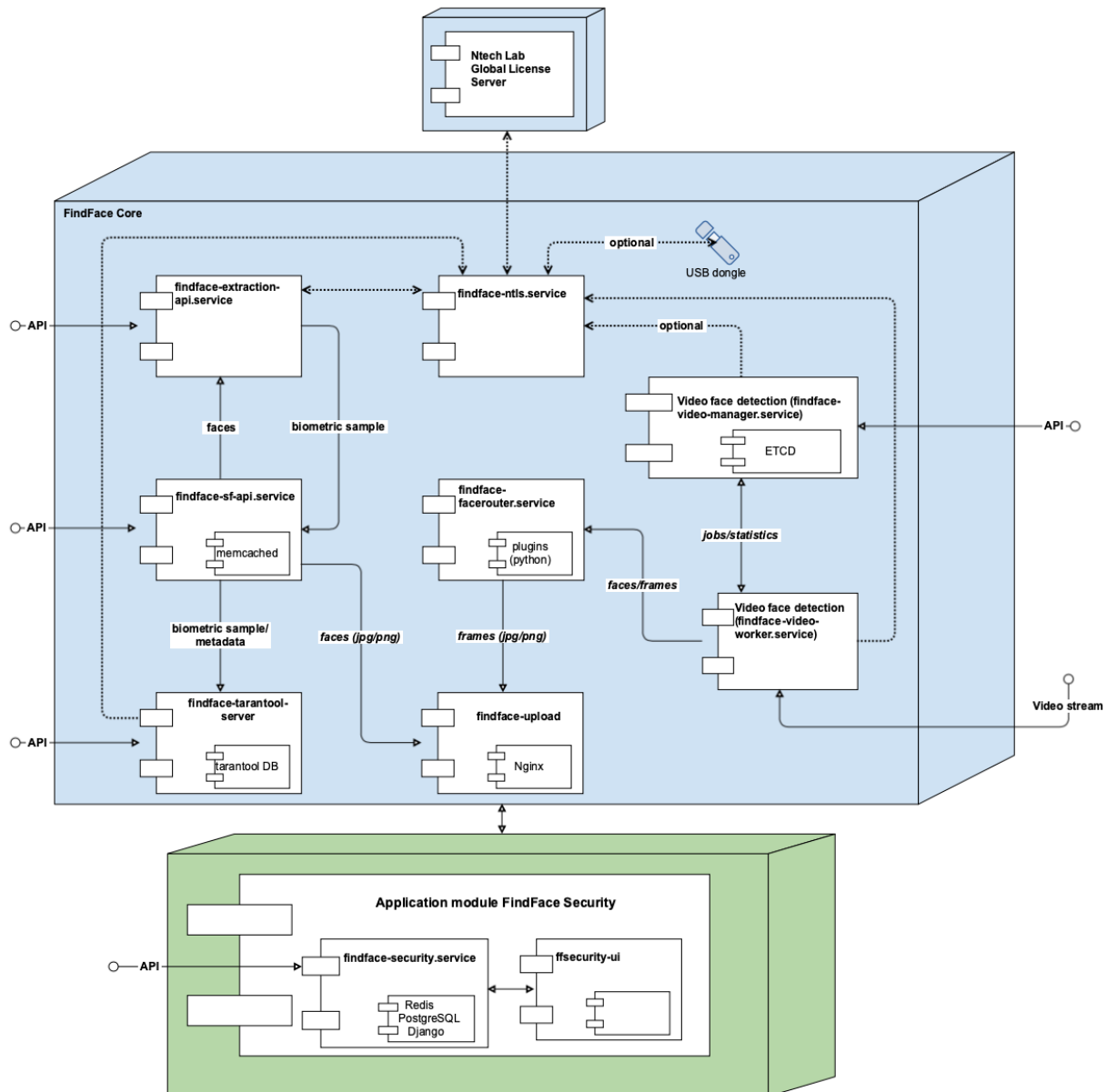
In this chapter:

- *Architectural Elements*
    - *Architecture scheme*
    - *FindFace Core*
    - *FindFace Security Application Module*
- *Single- and Multi-Host Deployment*
- *CPU- and GPU-acceleration*

1.1.1 Architectural Elements

FindFace Security consists of the following fundamental architectural elements:

- FindFace core, a cutting-edge AI-based face recognition technology that can be used as a separate product FindFace Enterprise Server.

- FindFace Security, which is a turnkey application module for FindFace Enterprise Server.

Architecture scheme

FindFace Core

The FindFace core includes the following components:

Com-po-nent	Description	Vendor
findface-extraction-api	Service that uses neural networks to detect a face in an image and extract a face biometric sample (feature vector). CPU- or GPU-acceleration.	Ntech Lab own deploy-ment
findface-sf-api	Service that implements HTTP API for face detection and face recognition.	
findface-tarantool-server	Service that provides interaction between the `findface-sf-api` service and the biometric database (database that stores face biometric samples) powered by Tarantool.	
findface-upload	NginX-based web server used as a storage for original images, thumbnails and normalized face images.	
findface-facerouter	Service used to define processing directives for detected faces. In FindFace Security, its functions are performed by `findface-security` (see *FindFace Security Application Module*). If necessary, you can still deploy and enable this component for integration purposes (see *Plugins*).	
findface-video-manager	Service, part of the video face detection module, that is used for managing the video face detection functionality, configuring the video face detector settings and specifying the list of to-be-processed video streams.	
findface-video-worker	Service, part of the video face detection module, that recognizes a face in the video and posts its normalized image, full frame and metadata (such as the camera ID and detection time) to the `findface-facerouter` service for further processing according to given directives. CPU- or GPU-acceleration.	
findface-ntls	License server which interfaces with the NtechLab Global License Server or a USB dongle to verify the *license* of your FindFace Security instance.	
Taran-tool	Third-party software which implements the biometric database that stores extracted biometric samples (feature vectors) and face identification events. The system data, dossiers, user accounts, and camera settings are stored in PostgreSQL (part of the FindFace Security application module).	Taran-tool
etcd	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 face detector with fault tolerance.	etcd
NginX	Third-party software which implements the system web interfaces.	nginx
mem-cached	Third-party software which implements a distributed memory caching system. Used by `findface-extraction-api` as a temporary storage for extracted face biometric samples before they are written to the biometric database powered by Tarantool.	mem-cached

FindFace Security Application Module

The FindFace Security application module includes the following components:

Component	Description	Vendor
findface-security	Component that serves as a gateway to the FindFace core. Provides interaction between the FindFace Core and the web interface, the system functioning as a whole, HTTP and web socket, biometric monitoring, event notifications, episodes, webhooks. Includes the following internal services: Monitoring updater, Unacknowledged event notifier, Webhook updater, NTLS checker, Event episodes manager.	Ntech Lab own deployment
ffsecurity ui	Main web interface that is used to interact with FindFace Security. Allows you to work with face identification events, search for faces, manage cameras, users, dossiers, and watch lists.	
PostgreSQL	Third party software which implements the main system database that stores detailed and categorized dossiers on particular persons, as well as data for internal use such as user accounts and camera settings. The face biometric data and face identification events are stored in Tarantool (part of the FindFace core).	PostgreSQL
Redis	Third-party software which implements a message broker inside `findface-security`.	Redis
Django	Third-party software which implements a web framework for the FindFace Security web interface.	Django

See also:

Components in Depth

1.1.2 Single- and Multi-Host Deployment

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

- Deploy FindFace Security standalone and distribute additional `findface-video-worker` components across multiple hosts.

- Distribute the FindFace Security components across multiple hosts. If necessary, set up load balancing.

See *Guide to Typical Cluster Installation* for details.

1.1.3 CPU- and GPU-acceleration

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

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

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

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

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

1.2 System Requirements

To calculate the FindFace Security host(s) characteristics, use the requirements provided in this chapter.

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

In this chapter:

- *Basic Configuration*
- *Benchmark Results*
    - *Testing Setup*
    - *Resource Consumption:* `findface-extraction-api` *and* `findface-extraction-api-gpu`
    - *Performance:* `findface-extraction-api` *and* `findface-extraction-api-gpu`
    - *Performance:* `findface-video-worker-cpu` *and* `findface-video-worker-gpu`
- *Examples of Hardware Configuration*
    - *CPU-based Server*
    - *GPU-based Server*

1.2.1 Basic Configuration

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

	Minimum	Recommended
CPU	Intel Core i5 CPU with 4 physical cores 2.8 GHz	Intel Xeon E5v3 with 6 physical cores, or higher or similar CPU
	The own needs of FindFace Security require 2 cores HT > 2.5 GHz. The characteristics also depend on the number of cameras in use. A single camera 720p@25FPS requires 2 cores >2.5 GHz. AVX support	
GPU (op-tional)	Nvidia Geforce® GTX 980 4GB	Nvidia Geforce® GTX 1080+ with 8+Gb RAM
	Supported series: GeForce (Maxwell, Pascal, Turing, and above), Tesla (Maxwell, Pascal, Volta v100, Turing, and above)	
RAM	10 Gb	16+ Gb
	The own needs of FindFace Security require 8 Gb. The RAM consumption also depends on the number of cameras in use. A single camera 720p@25FPS requires 2 GB RAM	
HDD	16 Gb	16+ Gb
	The own needs of the operating system and FindFace Security 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	
Oper-ating system	Ubuntu 16.04 x64 only	

Tip: For more accurate hardware selection, consult the FindFace Security resource consumption and performance *benchmark results*.

1.2.2 Benchmark Results

Here you can see the FindFace Security resource consumption and performance benchmark results. Use these data to select your hardware configuration.

Note: RAM usage and performance may slightly vary from test to test.

Warning: Strictly not recommended to use `face/elderberry_160` for work.

Testing Setup

Package versions:

- findface-extraction-api-cpu 2.6.999.1910+261.gebb8df6

- findface-extraction-api-gpu

- findface-video-worker 2.6.999.1910+261.gebb8df6

- findface-video-worker-gpu

- findface-tarantool-server 2.6.999.1910+261.gebb8df6

Hardware:

- Processor: Intel Core i5-8400 @ 3.60GHz (6 Cores)
- Motherboard: ASUS PRIME H370M-PLUS
- Memory: 2 x 8192 MB DDR4-2400MHz
- Graphics: Gigabyte NVIDIA GeForce GTX 1060 6GB

Software:

- OS: Ubuntu 16.04, Kernel: 4.15.0-29-generic (x86_64)
- Screen Resolution: 1920x1200

RAM consumption is calculated by:

- CPU: htop;
- GPU: nvidia-smi

CPU performance:

```
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 10000


Test execution summary:
    total time:                          9.1128s
    total number of events:              10000
    total time taken by event execution: 9.1112
    per-request statistics:
        min:                             0.82ms
        avg:                             0.91ms
        max:                             1.47ms
        approx.  95 percentile:          1.02ms

Threads fairness:
    events (avg/stddev):          10000.0000/0.00
    execution time (avg/stddev):  9.1112/0.00
```

GPU performance:

```
Unigine Heaven 4.0:
    pts/unigine-heaven-1.6.4 [Resolution: 1920 x 1080 - Mode: Windowed - Renderer:␣
→OpenGL]
    Test 1 of 2
    Estimated Trial Run Count:    3
    Estimated Test Run-Time:      15 Minutes
    Estimated Time To Completion: 29 Minutes
        Started Run 1 @ 17:54:37
        Started Run 2 @ 17:59:15
        Started Run 3 @ 18:03:52  [Std. Dev: 0.29%]

    Test Results:
        86.6473
        86.1475
        86.4553

    Average: 86.42 Frames Per Second


Unigine Heaven 4.0:
    pts/unigine-heaven-1.6.4 [Resolution: 1920 x 1080 - Mode: Fullscreen - Renderer:␣
→OpenGL]
    Test 2 of 2
    Estimated Trial Run Count:    3
    Estimated Time To Completion: 15 Minutes
        Started Run 1 @ 18:08:33
        Started Run 2 @ 18:13:09
        Started Run 3 @ 18:17:45  [Std. Dev: 1.37%]

    Test Results:
        87.7017
        89.5186
        90.023

    Average: 89.08 Frames Per Second
```

Resource Consumption: `findface-extraction-api` and `findface-extraction-api-gpu`

RAM usage: `findface-extraction-api`

Model	# instances	RAM, MB	# instances	RAM, MB	# instances	RAM, MB
face/elderberry_576.cpu	5	3730	2	7450	3	11000
face/elderberry_160.cpu		1590		2800		4050
face/elderberry_576.cpu + features (faceattr/age.v1.cpu, faceattr/beard.v0.cpu, faceattr/emotions.v1.cpu, faceattr/gender.v2.cpu, faceattr/glasses3.v0.cpu)		5568		10800		•
face/elderberry_160.cpu + features (faceattr/age.v1.cpu, faceattr/beard.v0.cpu, faceattr/emotions.v1.cpu, faceattr/gender.v2.cpu, faceattr/glasses3.v0.cpu)		3473		6250		9400
Features only (faceattr/age.v1.cpu, faceattr/beard.v0.cpu, faceattr/emotions.v1.cpu, faceattr/gender.v2.cpu, faceattr/glasses3.v0.cpu)		2270		3900		5800

RAM usage: `findface-extraction-api-gpu`

Note: `findface-extraction-api-gpu` allows only 1 model instance.

Model	RAM, MB
face/elderberry_576.gpu	~2200 (up to 4.5 Gb on initialization)
face/elderberry_160.gpu	~850 (up to 1.8 Gb on initialization)
face/elderberry_576.gpu + features (faceattr/age.v1.gpu, faceattr/beard.v0.gpu, faceattr/emotions.v1.gpu, faceattr/gender.v2.gpu, faceattr/glasses3.v0.gpu)	~3100 (up to 6.3 Gb on initialization)
face/elderberry_160.gpu + features (faceattr/age.v1.gpu, faceattr/beard.v0.gpu, faceattr/emotions.v1.gpu, faceattr/gender.v2.gpu, faceattr/glasses3.v0.gpu)	~1871 (up to 4 Gb on initialization)
Features only (faceattr/age.v1.gpu, faceattr/beard.v0.gpu, faceattr/emotions.v1.gpu, faceattr/gender.v2.gpu, faceattr/glasses3.v0.gpu)	1200

Performance: `findface-extraction-api` and `findface-extraction-api-gpu`

Speed: `findface-extraction-api`

Model	Time, ms (i5-8400)
face/elderberry_576.cpu	620
face/elderberry_160.cpu	350
face/elderberry_576.cpu + features (faceattr/age.v1.cpu, faceattr/beard.v0.cpu, faceattr/emotions.v1.cpu, faceattr/gender.v2.cpu, faceattr/glasses3.v0.cpu)	655
face/elderberry_160.cpu + features (faceattr/age.v1.cpu, faceattr/beard.v0.cpu, faceattr/emotions.v1.cpu, faceattr/gender.v2.cpu, faceattr/glasses3.v0.cpu)	380
Features only (faceattr/age.v1.cpu, faceattr/beard.v0.cpu, faceattr/emotions.v1.cpu, faceattr/gender.v2.cpu, faceattr/glasses3.v0.cpu)	300

Speed: `findface-extraction-api-gpu`

Model	Time, ms (1060TI)
face/elderberry_576.gpu	240
face/elderberry_160.gpu	225
face/elderberry_576.gpu + features (faceattr/age.v1.gpu, faceattr/beard.v0.gpu, faceattr/emotions.v1.gpu, faceattr/gender.v2.gpu, faceattr/glasses3.v0.gpu)	260
face/elderberry_160.gpu + features (faceattr/age.v1.gpu, faceattr/beard.v0.gpu, faceattr/emotions.v1.gpu, faceattr/gender.v2.gpu, faceattr/glasses3.v0.gpu)	235

Performance: `findface-video-worker-cpu` and `findface-video-worker-gpu`

CPU/RAM consumption and speed: `findface-video-worker-cpu`

Stream	RAM, MB	CPU utilization,% (i5-8400 6 cores)	Processing speed, FPS* (i5-8400)
1x 720p25FPS	370	230	62
2x 720p25FPS	755	500	56
3x 720p25FPS	1040	580	45
4x 720p25FPS	1437	600	36
5x 720p25FPS	1900	600	24
8x 720p25FPS	2650	600	18
1x 1080p25FPS	502	250	41
2x 1080p25FPS	1023	508	37
3x 1080p25FPS	1529	590	30
4x 1080p25FPS	2031	594	23
1x 720p25FPS + 1x 1080p25FPS	890	453	38
2x 720p25FPS + 2x 1080p25FPS	1750	590	21

Important: If video processing speed is less than the number of FPS in video, it means that the system is running low on resources and the lack of resources causes the video face detector to drop frames. Avoid this situation as it can lead to missing out on faces, instability and potential failures.

To check your resource consumption, execute:

```
sudo journalctl -f -a -u findface-video-worker-cpu | grep dropped
```

The following lines indicate that the system has less resources than necessary:

```
findface-video-worker[28882]: [2] 2 frames dropped!
findface-video-worker[28882]: [1] 6 frames dropped!
```

In this case, consider changing component settings or hardware configuration.

GPU RAM consumption and speed: `findface-video-worker-gpu`

Stream	GPU RAM, MB	Processing speed, FPS* (1060TI)
Without streams	600	•
1x 720p25FPS	656	254
2x 720p25FPS	738	126
4x 720p25FPS	858	63
8x 720p25FPS	1117	30
1x 1080p25FPS	735	202
2x 1080p25FPS	935	96
4x 1080p25FPS	1185	48
8x 1080p25FPS	2650	48
1x 720p25FPS + 1x 1080p25FPS	803	453
2x 720p25FPS + 2x 1080p25FPS	1100	54
4x 720p25FPS + 4x 1080p25FPS	1500	26
8x 720p25FPS + 8x 1080p25FPS	2300	11

Important: If video processing speed is less than the number of FPS in video, it means that the system is running low on resources and the lack of resources causes the video card to accumulate frames in its memory. Avoid this situation as it can lead to instability and potential failures.

To view the current processing speed, execute the following command on the `findface-video-manager` host console:

```
curl -s http://127.0.0.1:18810/jobs | jq -r '.[]|("id="+(.id|tostring)+" url="+.
↪stream_url+" FPS="+(.statistic.processing_fps|tostring))'
```

In the response, you will find each video stream processing speed. For example, enough amount of resources when processing 7 video streams with characteristics **h264 (High) ([27][0][0][0] / 0x001B), yuvj420p(pc, bt709), 1920x1080, 25 fps, 25 tbr, 90k tbn, 180k tbc** will result in the following response:

```
id=10 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=27.189745
id=5 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=27.189854
id=6 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=27.589714
id=3 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=27.189857
id=4 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=27.189857
id=7 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=27.389784
id=9 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=27.189857
```

Lack of resources when processing 8 video streams with the same characteristics will give FPS (processing speed) less than the video's 25 fps:

```
id=8 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=23.772333
id=10 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=23.772415
id=5 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=23.772339
id=6 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=23.372803
id=3 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=23.772339
id=4 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=23.772339
id=7 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.775822
id=9 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=22.573729
```

Even smaller values will be registered when processing 10 video streams with the same characteristics:

```
id=7 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.380642
id=9 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.380642
id=2 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.380646
id=8 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.380642
id=10 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=9.984919e-05
id=5 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.380642
id=6 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.380642
id=1 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.380651
id=3 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=20.180836
id=4 url=http://restreamer.int.ntl/hls/openspace.m3u8 FPS=19.581406
```

Important: If `findface-video-worker-gpu` processes video streams of equal FPS, the number of processed streams doesn't severely affect the GPU memory consumption, as all the streams are processed by the same worker. On the other hand, if `findface-video-worker-gpu` processes video streams of different FPS, it severely increases the memory consumption as different streams have to be processed by different workers.

1.2.3 Examples of Hardware Configuration

Important: The exemplary hardware configurations in this section are only for reference. Do not use these data to select your production instance configuration. To select the optimal configuration, ask advice from our experts by support@ntechlab.com.

Resource consumption may vary depending on the following factors:

- The number of HTTP requests per second, sent to `findface-extraction-api` (depends on the number of faces in a camera field of view, the number of user search requests, etc.).
- Video quality (video interference, colourful video background take up more resources).
- Motion intensity in video.

The following examples are given for standard component configuration.

Important: Changes in component settings may result in significant changes in resource consumption.

CPU-based Server

Cam-eras	CPU	RAM, GB	Extraction
1x720p25fps	Intel Core i5 - 6400 (4 cores 2700MHz)	8	elderberry_160 + features* model_instances = 1 or elderberry_576 model_instances = 1
2x720p25fps	FPS: Intel Core i7 - 6700 (4 core 3400MHz); recommended: Intel Core i7 - 6850K (6 cores 3600MHz)	12	elderberry_160 + features* model_instances = 2 or elderberry_576 + features* model_instances = 2
4x720p25fps	FPS: Intel Core i7 - 8700K (6 cores 3700MHz); recommended: Intel Core i9 - 9900K (8 cores 3600MHz)	16	elderberry_576 + features* model_instances = 2 or elderberry_576 model_instances = 3
1x1080p25fps	FPS Intel Core i7 - 6700 (4 cores 3400MHz); recommended: Intel Core i7 - 6850K (6 core 3600MHz)	32	elderberry_576 + features* model_instances = 1 or elderberry_576 model_instances = 2

GPU-based Server

Cameras	CPU	RAM, GB	GPU	Installation	Extraction	Video
1x720p 25FPS	Intel Core i5 - 6400 (4 cores 2700MHz)	8	nVidia GeForce GTX1060 6Gb	extraction-api on CPU video-worker on GPU	elderberry_160 + features* model_instances = 1 or elderberry_576.cpu model_instances = 1	basic
				extraction-api on GPU video-worker on CPU	basic	basic
2x720p 25FPS	Intel Core i5 - 6400 (4 cores 2700MHz)	12	nVidia GeForce GTX1060 6Gb	extraction-api on CPU video-worker on GPU	elderberry_160 + features* model_instances = 2 or elderberry_576.cpu + features model_instances = 1 or elderberry_576.cpu model_instances = 2	basic
				extraction-api on GPU video-worker on CPU	basic	basic
4x720p 25FPS	Intel Core i5 - 8400 (4 cores 2800MHz)	16	nVidia GeForce GTX1060 6Gb	extraction-api on CPU video-worker on GPU	elderberry_576.cpu + features* model_instances = 2	basic
8x720p 25FPS	Intel Core i5 - 8400 (4 cores 2800MHz) Intel Core i7 - 6700 (4 cores 3400MHz)	16	nVidia GeForce GTX1060 TI 6Gb	extraction-api on CPU video-worker on GPU	elderberry_576.cpu + features* model_instances = 2	basic
16x720p 25FPS	Intel Core i7 - 6700 (4 cores 3400MHz) Intel Core i7 - 8700K (6 cores 3700MHz) Intel Core i9 - 9900K (8 cores 3600MHz)	32	2x nVidia GeForce GTX1060 TI 6Gb	extraction-api on CPU video-worker on GPU	elderberry_576.cpu + features* model_instances = 4 or	basic

1.3 Licensing Principles

FindFace Security is licensed by the following criteria:

1. The number of extracted biometric samples and biometric samples under monitoring (in watch lists). In the

course of the FindFace Security functioning, biometric samples are extracted from faces detected in the video, and from dossier photos. Overall, the licensing scheme is as follows:

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

- Dossier: 1 photo in a dossier = 2 faces in a license (face extraction + face monitoring).

2. The number of cameras in use.

3. The number of the `findface-extraction-api` model instances in use.

4. Face features recognition: gender/age/emotions/glasses/beard.

5. Face liveness detection.

6. Integration with partners.

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

- The online licensing requires a stable internet connection. Upon being disconnected from the internet, the system will continue working off-grid for about 1 hour.

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

To provide the system functioning, one `findface-ntls` instance should be enough. If for some reason, your system requires more license servers, contact your Ntech Lab manager beforehand to prevent your system from being blocked.

See also:

Licensing

1.4 Deploy FindFace Security

For your convenience, we offer you several deployment options:

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

1.4.1 Deploy from Console Installer

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

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

Important: The FindFace Security 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
```

(continues on next page)

(continued from previous page)

```
gateway 192.168.112.254
dns-nameservers 192.168.112.254
```

Restart networking.

```
sudo service networking restart
```

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

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

1. Download the installer file `findface-security-and-server-4.0.2.run`.

2. Put the `.run` file into some directory on the designated host (for example, `/home/username`).

3. From this directory, make the `.run` file executable.

```
chmod +x findface-security-and-server-4.0.2.run
```

4. Execute the `.run` file.

```
sudo ./findface-security-and-server-4.0.2.run
```

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

1. Product to install: FindFace Security.

2. Installation type:

   - `1`: install FindFace Security standalone.

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

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

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

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

   - `4`: fully customized installation.

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

     ---

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

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

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

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

Important: In the case of a clean install, the installer will automatically configure `findface-extraction-api` to use the `grapefruit_480` neural network. Otherwise, you will be able to choose between `grapefruit_480` and the previous model. It is strictly not recommended to use the installer to update the system. See *Update FindFace Security to 4.x* for the instructions.

Service	Configuration
postgresql-9.5	Installed and started.
redis-server	Installed and started.
etcd	Installed and started.
mem-cached	Installed and started.
nginx	Installed and started.
django	Installed and started as a web framework for the FindFace Security web interface.
findface-ntls	Installed and started.
findface-tarantool-server	Installed and started. The number of instances (shards) is calculated using the formula: $N = $ `max(min(mem_mb // 2000, cpu_cores), 1)`, i.e. it is equal to the RAM size in MB divided by 2000, or the number of CPU physical cores (but at least 1 shard).
findface-extraction-api	Installed and started.
findface-sf-api	Installed and started.
findface-upload	Installed.
findface-video-manager	Installed and started (CPU/GPU-acceleration).
findface-video-worker-*	Installed and started.
findface-data-*	Neural network models for face and face features recognition (gender, age, emotions, glasses, beard). Installed.
findface-gpudetector-data/	NTechLab gpudetector data. Installed.
python3-ntech.ffsecurity-client	NtechLab FindFace Security API python client library. Installed.
findface-security	Installed and started.
jq	Installed. Used to pretty-print API responses from FindFace Security.

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

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

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

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

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

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

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

```
sudo ./findface-security-and-server-4.0.2.run -f /tmp/<findface-installer-*>.json
```

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

1.4.2 Deploy Step-by-Step from Repository

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

In this section:

- *Install APT Repository*
- *Prerequisites*
- *Provide Licensing*
- *Deploy Main Database*
- *Deploy FindFace Core*
- *Deploy FindFace Security Application Module and Biometric Database*

Install APT Repository

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

1. Download the installer file `findface-security-and-server-4.0.2.run`.

2. Put the `.run` file into some directory on the designated host (for example, `/home/username`).

3. From this directory, make the `.run` file executable.

```
chmod +x findface-security-and-server-4.0.2.run
```

4. Execute the `.run` file.

```
sudo ./findface-security-and-server-4.0.2.run
```

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

1. Product to install: `FindFace Security`.

2. Installation type: `repo: Don't install anything, just set up the APT repository`.

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

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

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

Prerequisites

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

1. Install the prerequisite packages as such:

```
sudo apt update
sudo apt install -y postgresql-9.5 redis-server etcd memcached
```

2. Open the `memcached` configuration file. Set the maximum memory to use for items in megabytes: `-m 512`. Set the max item size: `-I 16m`. If one or both of these parameters are absent, simply add them in the file.

```
sudo vi /etc/memcached.conf

-m 512
-I 16m
```

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

```
sudo systemctl enable postgresql@9.5-main.service redis-server etcd.service␣
↪memcached.service
sudo systemctl start postgresql@9.5-main.service redis-server etcd.service␣
↪memcached.service
```

Provide Licensing

See also:

Licensing Principles

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

The FindFace Security licensing is provided as follows:

1. Deploy `findface-ntls`, license server in the FindFace core.

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

   Tip: In the `findface-ntls` configuration file, you can change the license folder and specify your proxy server IP address if necessary. You can also change the `findface-ntls` web interface remote access settings. See *findface-ntls* for details.

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

2. Upload the license file via the `findface-ntls` web interface in one of the following ways:

   • Navigate to the `findface-ntls` web interface `http://<NTLS_IP_address>:3185/#/`. Upload the license file.

     Tip: Later on, use the FindFace Security main web interface to consult your license information, and upgrade or extend your license (*Settings -> License*).

   • Directly put the license file into the license folder (by default, `/ntech/license`, can be changed in the `/etc/findface-ntls.cfg` configuration file).

3. For the on-premise licensing, insert the USB dongle into a USB port.

4. If the licensable components are installed on remote hosts, specify the IP address of the `findface-ntls` host in their configuration files. See *findface-extraction-api*, *findface-tarantool-server*, *Video face detection: findface-video-manager and findface-video-worker* for details.

See also:

View and Update License

Deploy Main Database

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

1. Using the **PostgreSQL** console, create a new user `ntech` and a database `ffsecurity` in PostgreSQL.

```
sudo -u postgres psql

postgres=# CREATE ROLE ntech WITH LOGIN;

postgres=# CREATE DATABASE ffsecurity WITH OWNER ntech ENCODING 'UTF-8' LC_
→COLLATE='en_US.UTF-8' LC_CTYPE='en_US.UTF-8' TEMPLATE template0;
```

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

---

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

```
echo 'local all ntech peer' | sudo tee -a /etc/postgresql/9.5/main/pg_hba.conf

sudo systemctl restart postgresql@9.5-main.service
```

Deploy FindFace Core

To deploy the FindFace core, do the following:

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

---

1. Install the FindFace core components:

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

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

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

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

---

2. Open the `findface-extraction-api` configuration file (CPU or GPU service). Enable the `quality_estimator` to be able to estimate the face quality in a dossier.

Note: The *minimum face quality* in a dossier photo is set as `MINIMUM_DOSSIER_QUALITY` in `/etc/ffsecurity/config.py`.

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

quality_estimator: true
```

3. In the `findface-extraction-api` configuration file, enable recognition models for face features such as gender, age, emotions, glasses3, and/or beard, subject to your needs. 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. See *Face Features Recognition* for details.

```
models:
  age: faceattr/age.v1.cpu.fnk
  emotions: faceattr/emotions.v1.cpu.fnk
  face: face/grapefruit_480.cpu.fnk
  gender: faceattr/gender.v2.cpu.fnk
  beard: faceattr/beard.v0.cpu.fnk
  glasses3: faceattr/glasses3.v0.cpu.fnk
```

The following models are available:

Face feature	Acceleration	Configuration file parameter
face (biometry)	CPU	`face:  face/grapefruit_480.cpu.fnk        face:  face/grapefruit_160.cpu.fnk`
	GPU	`face:  face/grapefruit_480.gpu.fnk        face:  face/grapefruit_160.gpu.fnk`
age	CPU	`age:  faceattr/age.v1.cpu.fnk`
	GPU	`age:  faceattr/age.v1.gpu.fnk`
gender	CPU	`gender:  faceattr/gender.v2.cpu.fnk`
	GPU	`gender:  faceattr/gender.v2.gpu.fnk`
emotions	CPU	`emotions:  faceattr/emotions.v1.cpu.fnk`
	GPU	`emotions:  faceattr/emotions.v1.gpu.fnk`
glasses3	CPU	`glasses3:  faceattr/glasses3.v0.cpu.fnk`
	GPU	`glasses3:  faceattr/glasses3.v0.gpu.fnk`
beard	CPU	`beard:  faceattr/beard.v0.cpu.fnk`
	GPU	`beard:  faceattr/beard.v0.gpu.fnk`

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

```
models:
  gender: ""
  age: ""
  emotions: ""
```

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

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

mgr-static=127.0.0.1:18811

capacity=10
```

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

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

Deploy FindFace Security Application Module and Biometric Database

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

1. Install the `findface-security` and `ffsecurity-ui` components.

```
sudo apt update
sudo apt install -y ffsecurity ffsecurity-ui
```

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

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

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

3. Create a structure of the Tarantool-based biometric database.

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

4. Import the `meta_scheme` variable from the `tnt_schema.lua` file. Open the `/etc/tarantool/instances.enabled/FindFace.lua` configuration file. Before the `FindFace.start` section, add a line `dofile("/etc/ffsecurity/tnt_schema.lua")`. In the `FindFace.start` parameters, define `meta_scheme=meta_scheme`.

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

dofile("/etc/ffsecurity/tnt_schema.lua")

FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    facen_size=480,
    meta_scheme = meta_scheme
 })
```

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

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

6. Open the `/etc/ffsecurity/config.py` configuration file. Specify the following parameters:

   - `SERVICE_EXTERNAL_ADDRESS`: FindFace Security 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 Security web interface. Once this parameter not specified, the system auto-detects it as the external IP address. To access FindFace Security, you can use both the auto-detected and specified IP addresses.

   - `VIDEO_DETECTOR_TOKEN`: to authorize the video face detection module, come up with a token and specify it here.

   - `VIDEO_MANAGER_ADDRESS`: IP address of the `findface-video-manager` host.

   - `NTLS_HTTP_URL`: IP address of the `findface-ntls` host.

   - `ROUTER_URL`: IP address of the `findface-security` host that will receive detected faces from the `findface-video-worker` instance(s). Specify either external or internal IP address, subject to the network through which `findface-video-worker` interacts with `findface-security`.

   - `SF_API_ADDRESS`: IP address of the `findface-sf-api` host.

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

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

Tip: It is recommended to change the `MINIMUM_DOSSIER_QUALITY` default value. This parameter determines the minimum quality of a face in a dossier photo. Photos containing faces of worse quality will be rejected when uploading to a dossier. Upright faces in frontal position are considered the best quality. They result in values around 0, mostly negative (such as -0.00067401276, for example). Inverted faces and large face angles are estimated with negative values some -5 and less. By default, `'MINIMUM_DOSSIER_QUALITY'`: `-2` which is the average quality.

Important: If you enabled recognition models in the `findface-extraction-api` configuration file, add the following line in the `FFSECURITY` section: 'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'], subject to the list of enabled models. This line must be placed between `SF_API_ADDRESS` and `LIVENESS_THRESHOLD` as shown in the example below. See *Face Features Recognition* for details.

```
sudo vi /etc/ffsecurity/config.py

MEDIA_ROOT = "/var/lib/ffsecurity/uploads"
STATIC_ROOT = "/var/lib/ffsecurity/static"

# SERVICE_EXTERNAL_ADDRESS prioritized for webhooks and genetec
```

```python
SERVICE_EXTERNAL_ADDRESS = 'http://localhost'
EXTERNAL_ADDRESS = "http://172.20.77.58"


DEBUG = False


LANGUAGE_CODE = 'en-us'


TIME_ZONE = 'UTC'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'ffsecurity',
    }
}

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

FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '381b0f4a20495227d04185ab02f5085f',
    'CONFIDENCE_THRESHOLD': 0.75,
    'MINIMUM_DOSSIER_QUALITY': -2,
    'IGNORE_UNMATCHED': False,
    'EXTRACTION_API': 'http://127.0.0.1:18666/',
    'VIDEO_MANAGER_ADDRESS': 'http://127.0.0.1:18810',
    'EVENTS_MAX_AGE': 30,
    'NTLS_HTTP_URL': 'http://127.0.0.1:3185',
    'ROUTER_URL': 'http://172.20.77.58',
    'MONITORING_UPDATE_INTERVAL': 60,
    'SF_API_ADDRESS': 'http://127.0.0.1:18411',
    'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'],
    'LIVENESS_THRESHOLD': 0.75,
    'BEARD_THRESHOLD': 0.7,
}


ASGI_THREADS = 16


UVICORN_SETTINGS = {
    'workers': 4,
    'host': 'localhost',
    'port': 8002,
 }


FFSECURITY_UI_CONFIG = {
    "event": {
        "features": {
            "f_gender_class": ["male", "female"],
            "age": {
                "f_age_gte": "",
                "f_age_lte": ""
            },
            "f_emotions_class": ["angry", "disgust", "fear", "happy", "sad",
→"surprise"],
            "f_glasses_class": ["none", "eye", "sun"],
            "f_beard_class": ["none", "beard"],
            "f_liveness_class": ["real", "fake"],
```

```
        }
    }
}

# integration plugins
INSTALLED_APPS.append('ffsecurity_genetec')  # remove or comment out this line to
→disable genetec integration
```

7. Generate a signature key for the session encryption (used by Django) by executing: `pwgen -sncy 50 1|tr
"'" "."`. Specify this key as `SECRET_KEY`.

8. Start the services.

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

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

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

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

sudo nginx -s reload
```

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

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

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

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

---

1. Download the installer file `findface-security-and-server-4.0.2.run`.

2. Put the `.run` file into some directory on the designated host (for example, `/home/username`).

3. From this directory, make the `.run` file executable.

```
chmod +x findface-security-and-server-4.0.2.run
```

4. Execute the `.run` file.

```
sudo ./findface-security-and-server-4.0.2.run
```

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

1. Product to install: FindFace Video Worker.

2. Type of `findface-video-worker` package: CPU or GPU.

3. IP address of the `ffsecurity` host.

After that, the installation process will automatically begin.

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

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

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

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

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

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

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

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

```
sudo ./findface-security-and-server-4.0.2.run -f /tmp/<findface-installer-*>.json
```

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

1.4.4 Neural Network Models Installation

To detect and identify faces and face features (gender, age, emotions, beard, glasses, etc.), `findface-extraction-api` uses neural networks.

If you have to manually initiate the models installation, use the console installer as follows:

1. Execute the prepared `findface-security-and-server-4.0.2.run` file.

   ```
   sudo ./findface-security-and-server-4.0.2.run
   ```

2. Select the installation type: `Fully customized installation.`

3. Select a FindFace Security 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): `1`. Enter done to save your selection and proceed to another step.

4. In the same manner, select models to install. After that, the installation process will automatically begin.

Note: You can find installed face recognition models at `/usr/share/findface-data/models/face/`, face features recognition models at `/usr/share/findface-data/models/faceattr/`.

```
ls /usr/share/findface-data/models/face/
grapefruit_480.cpu.fnk  grapefruit_480.gpu.fnk grapefruit_160.cpu.fnk  grapefruit_160.
↪gpu.fnk

ls /usr/share/findface-data/models/faceattr/
age.v1.cpu.fnk  age.v1.gpu.fnk  beard.v0.cpu.fnk  beard.v0.gpu.fnk  emotions.v1.cpu.
↪fnk  emotions.v1.gpu.fnk  gender.v2.cpu.fnk  gender.v2.gpu.fnk  glasses3.v0.cpu.fnk␣
↪ glasses3.v0.gpu.fnk  liveness.v3.gpu.fnk
```

1.4.5 Fully Customized Installation

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

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

- Product to install: `FindFace Security`.

- Installation type: `Fully customized installation`.

- FindFace Security components to install: whenever you have to make a selection, first deselect all the listed components by entering `-*` in the command line, then select required components by entering their sequence number (keyword), for example: `1 7` (`findface-data`, `findface-extraction-api`), `13` (`findface-tarantool-server`), or `9` (findface-upload). Enter `done` to save your selection and proceed to another step.

- Related questions such as about the acceleration type: CPU or GPU.

1.4.6 Guide to Typical Cluster Installation

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

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

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

- Necessity to distribute the video processing high load.

- 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

- Necessity to distribute the biometric sample extraction high load.

- Large number of faces to search through, that requires implementation of a distributed face database.

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

- One principal server with the following components: `findface-ntls`, `findface-security`, `findface-sf-api`, `findface-video-manager`, `findface-upload`, `findface-video-worker`, `findface-extraction-api`, `findface-tarantool-server`, and third-parties.

- Several additional video processing servers with installed `findface-video-worker`.

- (If needed) Several additional biometric servers with installed `findface-extraction-api`.

- (If needed) Additional database servers with multiple Tarantool shards.

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

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

- *Deploy Principal Server*
- *Deploy Video Processing Servers*
- *Deploy Biometric Servers*
- *Distribute Load across Biometric Servers*
- *Distribute Database*
- *Configure Network*

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 Security from installer as follows:

   - Product to install: `FindFace Security`.

   - Installation type: `Single server, multiple video workers`. In this case, FindFace Security will be installed and configured to interact with additional remote `findface-video-worker` instances.

   - Type of the `findface-video-worker` acceleration (on the principal server): CPU or GPU, subject to your hardware configuration.

   - Type of the `findface-extraction-api` acceleration (on the principal server): CPU or GPU, subject to your hardware configuration.

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

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

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

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

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

---

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

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

# Listen address of NTLS server where services will connect to.
# The format is IP:PORT
# Use 0.0.0.0:PORT to listen on all interfaces
# This parameter is mandatory and may occur multiple times
# if you need to listen on several specific interfaces or ports.
listen = 0.0.0.1:3133
```

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 Security 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-security-and-server-4.0.2.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 `findface-video-worker`
configuration file after the installation.

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

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

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

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

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

Deploy Biometric Servers

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

- Product to install: `FindFace Security`.

- Installation type: `Fully customized installation`.

- FindFace Security components to install: `findface-extraction-api` and `findface-data`. To make a selection, first deselect all the listed components by entering `-*` in the command line, then select `findface-extraction-api` and `findface-data` by entering their sequence number (keyword): `1 7`. Enter `done` to save your selection and proceed to another step.

- Type of `findface-extraction-api` acceleration: CPU or GPU.

- Modification of the `findface-extraction-api` 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 for gender, age, emotions, glasses and/or beard recognition. To make a selection, first deselect all the listed models by entering `-*` in the command line, then select required models by entering their sequence number (keyword), for example, `8 2` to select the GPU-models for biometric sample extraction and age recognition. Enter `done` to save your selection and proceed to another step. Be sure to choose the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models. See *Face Features Recognition* for details.

  The following models are available:

Face feature	Acceleration	Package
face (biometry)	CPU	findface-data-grapefruit-480-cpu_3.0.0_amd64.deb findface-data-grapefruit-160-cpu_3.0.0_amd64.deb
	GPU	findface-data-grapefruit-480-gpu_3.0.0_amd64.deb findface-data-grapefruit-160-gpu_3.0.0_amd64.deb
age	CPU	findface-data-age.v1-cpu_3.0.0_amd64.deb
	GPU	findface-data-age.v1-gpu_3.0.0_amd64.deb
gender	CPU	findface-data-gender.v2-cpu_3.0.0_amd64.deb
	GPU	findface-data-gender.v2-gpu_3.0.0_amd64.deb
emotions	CPU	findface-data-emotions.v1-cpu_3.0.0_amd64.deb
	GPU	findface-data-emotions.v1-gpu_3.0.0_amd64.deb
glasses3	CPU	findface-data-glasses3.v0-cpu_3.0.0_amd64.deb
	GPU	findface-data-glasses3.v0-gpu_3.0.0_amd64.deb
beard	CPU	findface-data-beard.v0-cpu_3.0.0_amd64.deb
	GPU	findface-data-beard.v0-gpu_3.0.0_amd64.deb

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-security-and-server-4.0.2.run -f /tmp/<findface-installer-*>.
↪json
```

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

Distribute Load across Biometric Servers

To distribute load across several biometric 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 Security principal server (`172.168.1.9`), and 2 on additional remote servers (`172.168.1.10`, `172.168.1.11`). Should you have more biometric 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 Security principal server (recommended) or any other server with nginx as a gateway to all the biometric servers.

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

   Tip: You can install nginx as such:

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

   ---

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

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

3. Insert the following entry into the newly created configuration file. In the `upstream` directive (`upstream extapibackends`), substitute the exemplary IP addresses with the actual IP addresses of the biometric servers. In the `server` directive, specify the gateway server listening port as `listen`. You will have to enter this port when configuring the FindFace Security *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;
   ```
   (continues on next page)

```
                proxy_next_upstream error;
        }
        access_log /var/log/nginx/extapi.access_log;
        error_log /var/log/nginx/extapi.error_log;
}
```

4. Enable the load balancer in nginx.

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

5. Restart nginx.

```
sudo service nginx restart
```

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

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

listen: 172.168.1.10:18666
```

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

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

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

Distribute Database

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

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

- Product to install: `FindFace Security`.

- Installation type: `Fully customized installation`.

- FindFace Security components to install: `findface-tarantool-server`. To make a selection, first deselect all the listed components by entering `-*` in the command line, then select `findface-tarantool-server` by entering its sequence number (keyword): `13`. Enter `done` to save your selection and proceed to another step.

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

```
sudo ./findface-security-and-server-4.0.2.run -f /tmp/<findface-installer-*>.json
```

As a result of the installation, `findface-tarantool-server` shards will be automatically installed in the amount of $N = max(min(mem\_mb\ //\ 2000,\ cpu\_cores),\ 1)$, i.e. equal to the RAM size in MB divided by 2000, or the number of CPU physical cores (but at least 1 shard).

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

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

You will get the following result:

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

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

Configure Network

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

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

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

   Specify the following parameters:

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

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

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

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

(continues on next page)

```
    slave: ''
  - master: http://172.168.1.12:8101/v2/
    slave: ''
  - master: http://172.168.1.12:8102/v2/
    slave: ''
  - master: http://172.168.1.13:8102/v2/
    slave: ''
  - master: http://172.168.1.13:8102/v2/
    slave: ''
```

2. Open the `/etc/ffsecurity/config.py` configuration file.

```
sudo vi /etc/ffsecurity/config.py
```

Specify the following parameters:

Parameter	Description
SERVICE_EXTERNAL_ADDRESS	FindFace Security 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 Security web interface. Once this parameter not specified, the system auto-detects it as the external IP address. To access FindFace Security, you can use both the auto-detected and specified IP addresses.
VIDEO_DETECTOR_TOKEN	To authorize the video face detection module, come up with a token and specify it here.
VIDEO_MANAGER_ADDRESS	IP address of the findface-video-manager host.
NTLS_HTTP_URL	IP address of the findface-ntls host.
ROUTER_URL	External IP address of the findface-security host that will receive detected faces from the findface-video-worker instance(s).
SF_API_ADDRESS	IP address of the findface-sf-api host.
EXTRACTION_API	IP address and listening port of the *gateway biometric server* with set up load balancing.

```
sudo vi /etc/ffsecurity/config.py

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


...
FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '7ce2679adfc4d74edcf508bea4d67208',
    ...
    'EXTRACTION_API': 'http://172.168.1.9:18667/',
    'VIDEO_MANAGER_ADDRESS': 'http://127.0.0.1:18810',
    ...
    'NTLS_HTTP_URL': 'http://127.0.0.1:3185',
    'ROUTER_URL': 'http://172.168.1.9',
    ...
    'SF_API_ADDRESS': 'http://127.0.0.1:18411',
    ...
}
```

The FindFace Security components interaction is now set up.

1.5 First Steps after Deployment

Once FindFace Security is successfully deployed, it is 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 the entire functionality of your system.

In this chapter:

- *Organize Cameras*
- *Organize Watch Lists and Dossiers*
- *Create Users and Grant Them Privileges*
- *Start Monitoring Faces*
- *Organize Video Surveillance*
- *FindFace Security in Action*
- *Basic Maintainance*
- *Go Further*

1.5.1 Organize Cameras

1. *Create a camera group*. 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 facial recognition events among cameras belonging to the same group. *Learn more*.

1.5.2 Organize Watch Lists and Dossiers

1. *Create a watch list*. A watch list is an entity that allows you to classify people by arbitrary criteria: black list, wanted, VIP, staff, etc.

2. Upload dossiers and add them in the watch list either *manually*, *in bulk via the web interface*, or use the *console bulk upload* function.

1.5.3 Create Users and Grant Them Privileges

1. Check out the list of *predefined user roles* and *create new roles* if necessary.

2. *Add users* into the system and grant them privileges.

1.5.4 Start Monitoring Faces

By default, FindFace Security is monitoring only *unmatched faces*. To enable a custom watch list monitoring, simply 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. Make events more informative by enabling recognition of gender, age, emotions, beard and glasses. *Learn more*.

2. Protect your system from spoofing by enabling the Face Liveness Detection functionality. *Learn more*.

1.5.5 Organize Video Surveillance

Create a camera layout for the basic video surveillance.

1.5.6 FindFace Security in Action

1. *Automatically identify faces 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 faces of the same person, detected within a certain period of time. As events on the *Events* tab show up in an arbitrary order, a large number of miscellaneous events can make the work difficult and unproductive. With the Episodes, the system uses AI to organize incoming events based on the faces similarity and detection time. This allows for easy processing of diverse events, even in large numbers.

3. Search for faces in the following databases:

   - Database of detected faces. *Learn more*.
   - Dossier database. *Learn more*.

4. *Search archived videos* for faces in the watch lists.

5. Manually *compare 2 faces* and verify that they belong to the same person.

6. Use the *mobile app*.

1.5.7 Basic Maintainance

1. *Configure* automatic events cleanup.

2. Manually *purge* events from the database.

3. Regularly *backup* the database.

1.5.8 Go Further

1. Set up *webhooks* to automatically send notifications about certain events to a given URL. In this case, when such an event occurs, FindFace Security 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 certain event, invoke required behaviour on a target website, solve security tasks such as automated access control, etc. *Learn more*.

2. Harness the FindFace Security functions through *HTTP API*.

3. Check out the list of our *partner integrations*.

4. Harness *plugins* to set your own directives that determine how FindFace Security processes detected faces.

See also:

- *Camera Management*

- *Face Monitoring and Dossier Database*

- *User Management*

- *Advanced Functionality*

- *Maintenance and Troubleshooting*

1.6 Work with FindFace Security

Use the web interface to interact with FindFace Security. To open the web interface, enter its basic address in the address bar of your browser, and log in.

Note: The basic address is set during *deployment*.

Important: To log in for the first time, use the admin account created during *deployment*. To create more users, refer to *User Management*.

The web interface has a highly intuitive and handy design and provides the following functionality:

- *Camera Management*. Group cameras subject to their location. Add and configure a camera.

- *Dossier Database*. Manage dossier classification lists (watch lists). Create dossiers manually and in bulk.

- *User Management*. Manage FindFace Security users and their roles.

- *Offline Video Processing*. Offline video face identification.

- *General Preferences*. Configure the confidence threshold for face verification. Set up automatic cleanup of the event database.

- *Compare faces*. Verify that 2 given faces belong to the same person.

- Operator's Guide. *Real time face identification* in live streams. *Organize Events with Episodes*. *Search for faces* in the event list and dossier database. *Video surveillance*.

1.6.1 Camera Management

To configure video-based biometric identification, add cameras to FindFace Security, grouping them subject to their location.

Note: Privileges to create camera groups and cameras are managed in user's permissions (see *User Management*).

In this chapter:

- *Create Camera Group*
- *Add Camera*
- *Monitor Camera Operation*

Create Camera Group

To create a group of cameras, do the following:

1. Navigate to the *Preferences* tab. Click *Camera Groups*.

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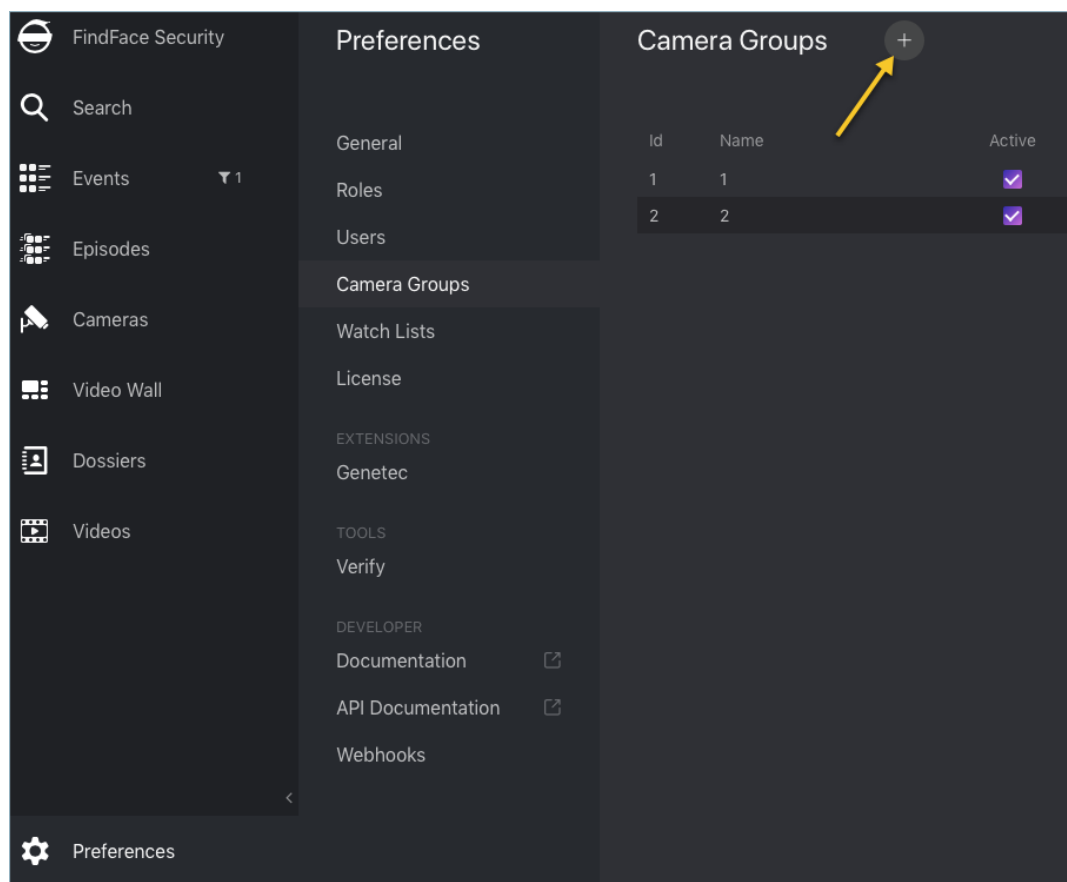
3. On the *Information* tab, specify the group name. Add a comment if needed.

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` configuration file. See *Allocate findface-video-worker to Camera Group* for details.

---

5. If you want to deduplicate events from cameras that belong to the same group, i. e. exclude coinciding events, check *Deduplicate Events* and specify the deduplication interval (interval between 2 consecutive checks for event uniqueness).

Warning: Use deduplication with extreme caution. If cameras within a group observe different scenes, some faces may be skipped. See *Deduplicate Events* for details.

---

6. Check *Active*.

7. Click *Save*.

8. On the *Permissions* tab, assign privileges on the camera group, specifying which user roles are allowed to change/view the camera group settings.
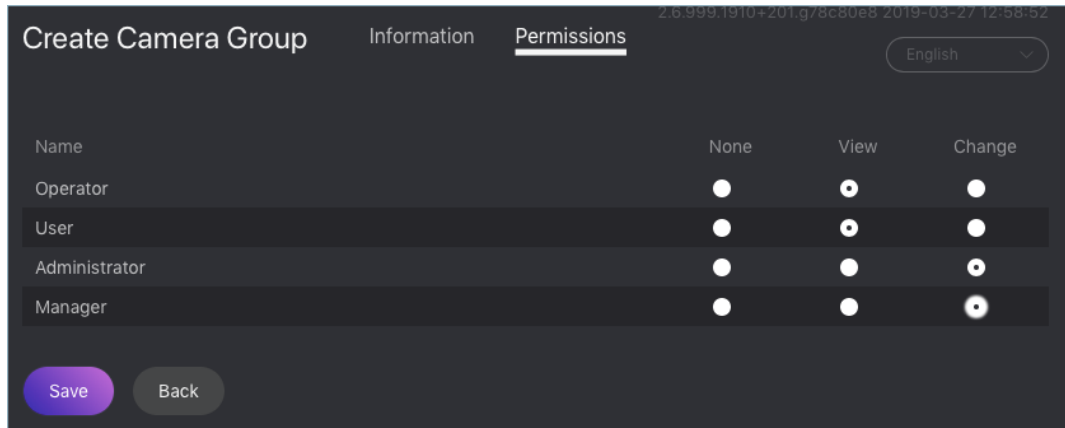
9. Click *Save*.

Add Camera

To add a camera, do the following:

1. Navigate to the *Cameras* tab.

2. Click +.
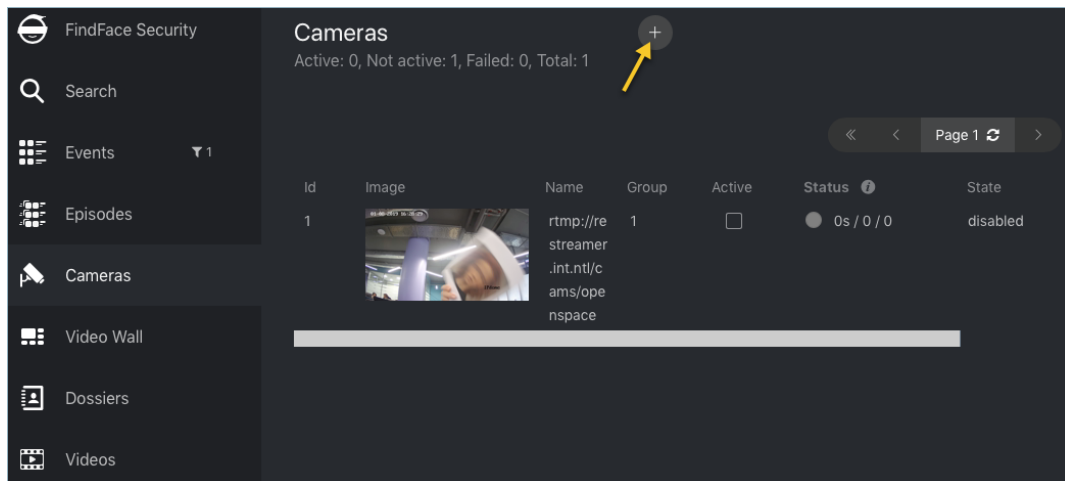
3. Specify the name of a camera and add it to a group. If necessary, add a comment.

4. Specify the camera URL or path to the video file, for example, `file:///data/some.mp4`.

5. Check *Active*.

6. To configure video processing, click *Parameters* and make adjustments:

   - *Minimum face snapshot quality* (`filter_min_quality`): Minimum quality of a face snapshot to post. To be fitted empirically: negatives values around 0 = high quality faces, -1 = good quality, -2 = satisfactory quality, -5 = inverted faces and large face angles, face recognition may be inefficient.

   - *Minimum face size* (`filter_min_face_size`): Minimum face size in pixels to post. If 0, the filter is off.

   - *Maximum face size* (`filter_max_face_size`): Maximum face size in pixels in post.

   - *Compression quality* (`jpeg_quality`): Full frame compression quality.

- *FFMPEG options* (`ffmpeg_params`): FFMPEG options for a video stream in the key-value format ["rtsp_transpotr=tcp", "ss=00:20:00"].

- *Offline mode* (`overall_only`): Offline mode. Enable posting one snapshot of the best quality for each face.

- *Time interval* (`realtime_post_interval`): Time interval in seconds (integer or decimal) within which the face tracker picks up the best snapshot in realtime mode.

- *Post best snapshot* (`realtime_post_every_interval`): If true, post the best snapshot obtained within each Time interval (realtime_post_interval) in realtime mode. If false, post the best snapshot only if its quality has improved comparing to the previously posted snapshot.

- *Posting timeout* (`router_timeout_ms`): Timeout in milliseconds for posting faces.

- *Retrieve timestamps from stream* (`use_stream_timestamp`): If true, retrieve and post timestamps from a video stream. If false, post the actual date and time.

- *Add to timestamps* (`start_stream_timestamp`): Add the specified number of seconds to timestamps from a stream.

- *Play speed limit* (`play_speed`): If less than zero, the speed is not limited. In other cases, the stream is read with the given `play_speed`. Not applicable for live streams.

- *Region of Tracking* (`ROT`): Enable detecting and tracking faces only inside a clipping rectangle. Use this option to reduce the video face detector load.

- *Region of Interest* (`ROI`): Enable posting faces detected only inside a region of interest.

Tip: To specify ROT/ROI, use the visual wizard. First, create a camera without ROT/ROI. Then open it for editing and click *Parameters*. You will see the visual wizard appear.

If necessary, specify optional parameters for video processing. Click *Advanced Parameters*.

- *Force input format* (`ffmpeg_format`): Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.

- *Verify SSL* (`router_verify_ssl`): If true, enable verification of the server SSL certificate when the face tracker posts faces to the server over https. If false, a self-signed certificate can be accepted.

- *Minimum motion intensity* (`imotion_threshold`): Minimum motion intensity to be detected by the motion detector.

7. Click *Save*.

Note: Each created camera is associated with a so-called job, a video processing task that contains configuration settings and stream data and is assigned to `findface-video-worker`. This task can be restarted (see *Monitor Camera Operation*).

Monitor Camera Operation

To monitor the operation of cameras, navigate to the *Cameras* tab.
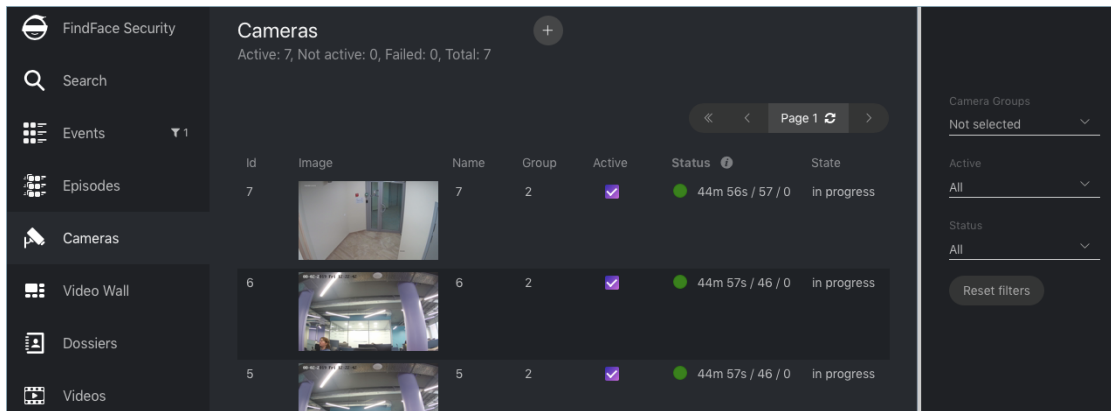
Camera statuses:

- 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 a face.

- Red: the video stream cannot be processed.

- Grey: camera disabled.

For each camera, you will be provided with the following statistics: current session duration/ the number of successfully posted faces/ the number of faces processed with errors after the last job restart.

Note: Each created camera is associated with a so called job, a video processing task that contains configuration settings and stream data and is assigned to `findface-video-worker`. This task can be restarted.

To restart a job, click in the *Action* column. In this case, the number of errors will be reset to `0`.

With a large number of cameras in the system, use the following filters:

- *Camera groups*,

- *Active*,

- *Status*.

See also:

- *Allocate findface-video-worker to Camera Group*

- *Deduplicate Events*

1.6.2 Face Monitoring and Dossier Database

This chapter is all about monitoring detected faces and creating the dossier database. Each dossier has to contain one or several photos of a person and belong to a certain classification list (watch list), black or white in the simplest case. You can create several watch lists, subject to a person status or hazard level.

Tip: To create dossiers in bulk, use the *batch photo upload* functionality.

In this section:

- *Monitoring Unmatched Faces*
- *Create Watch List*
- *Create Dossier Manually*
- *Batch Photo Upload*
- *Filter Dossiers by Watch List*

Monitoring Unmatched Faces

FindFace Security features one pre-configured watch list that is used for monitoring only unmatched faces. This watch list cannot be removed from the system. To edit its settings or deactivate it, navigate to the *Preferences* tab. Click *Watch Lists* and then click *Unmatched* in the table.

Note: To view only unmatched faces in the event list, select *Unmatched* in the *Watch lists* filter on the *Events* tab (refer to *Real-time Face Identification Events* for details).
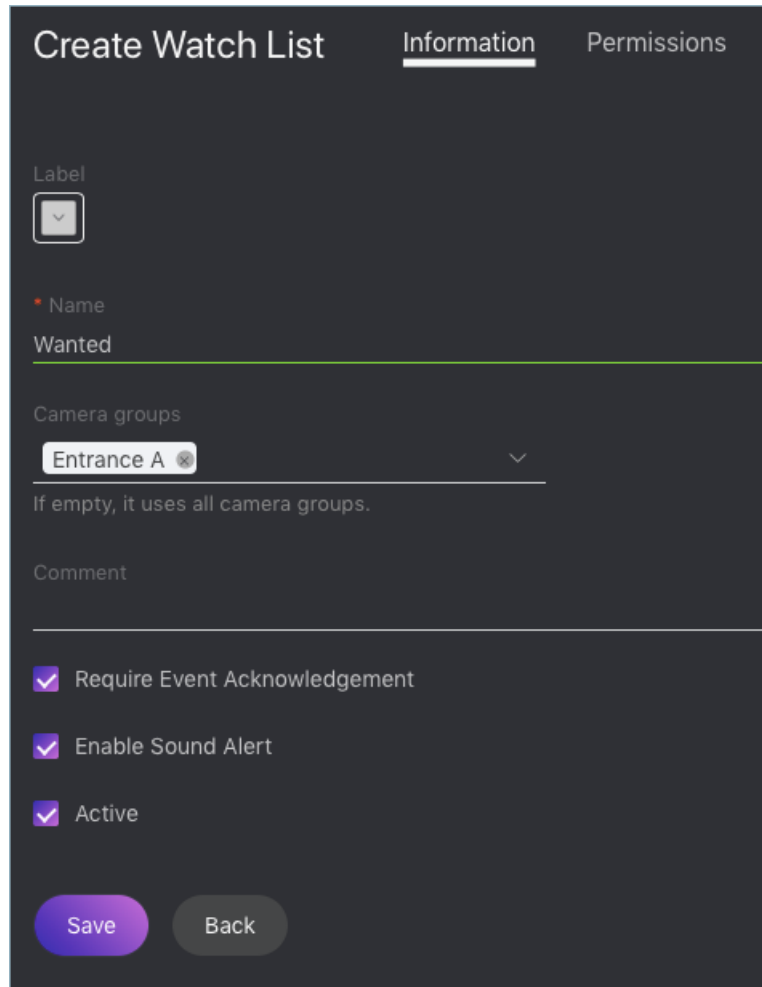
Create Watch List

To create a custom watch list, do the following:

1. Navigate to the *Preferences* tab. Click *Watch Lists*.

2. Click +.
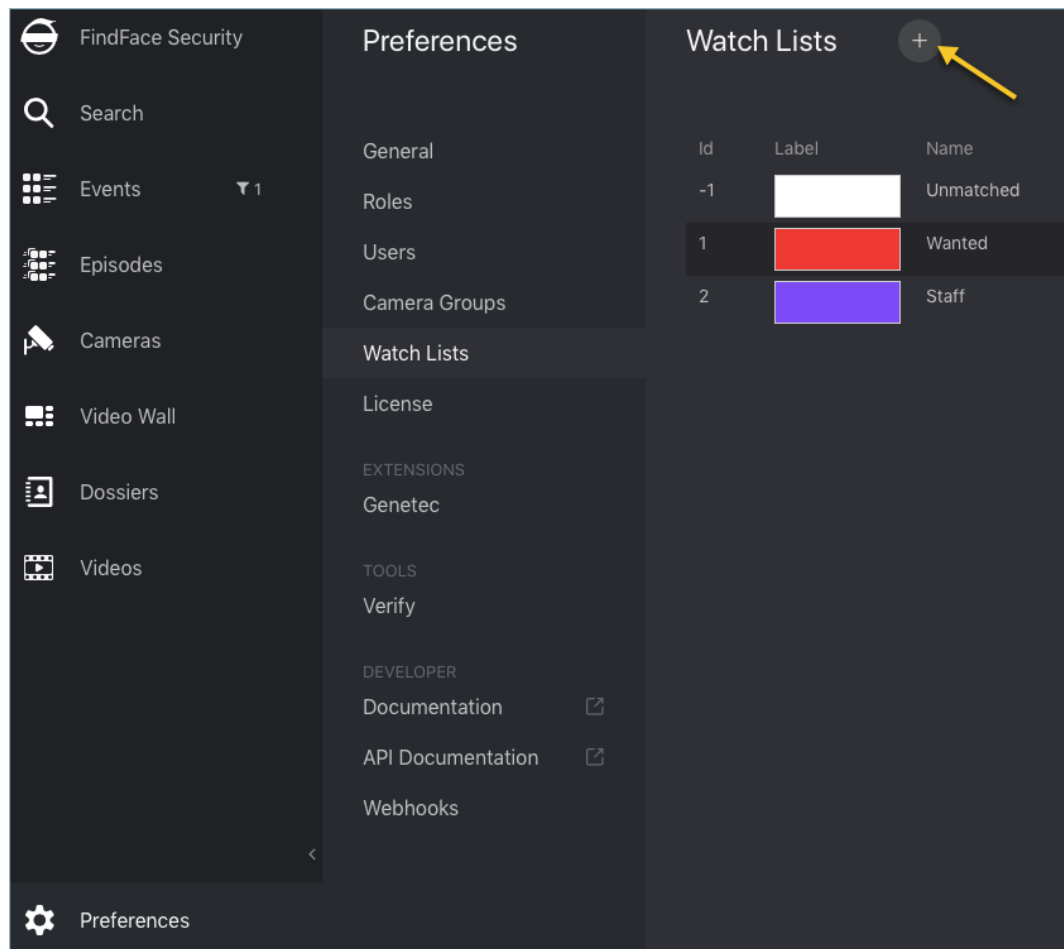
3. From the *Label* palette, select a color which will be shown in notifications for this list. Keep in mind that the right color makes for quicker response of security and hospitality managers.

4. Specify the watch list name. Add a comment if needed.

5. Select a camera group(s) which will be used to monitor the watch list. If no groups specified, the watch list will be monitored by all active cameras in the system.

6. Check *Require acknowledgment* if it is mandatory that events associated with the list be manually acknowledged.

7. Check *Enable sound alert* to turn on sound notifications for the list if needed.

8. Check *Active*.

9. Click *Save*.

10. On the *Permissions* tab, assign privileges on the watch list, specifying which user roles are allowed to change/view the watch list settings.
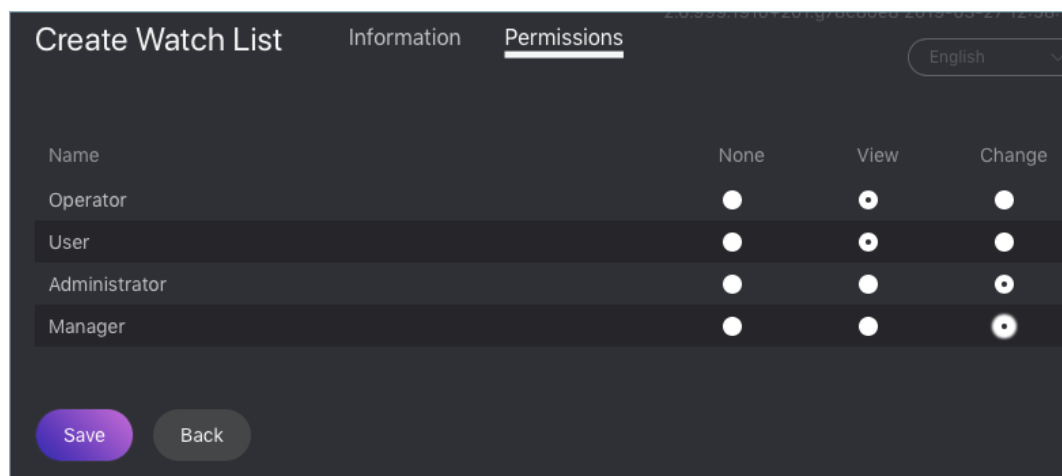
11. Click *Save*.

Create Dossier Manually

To create a dossier manually, do the following:
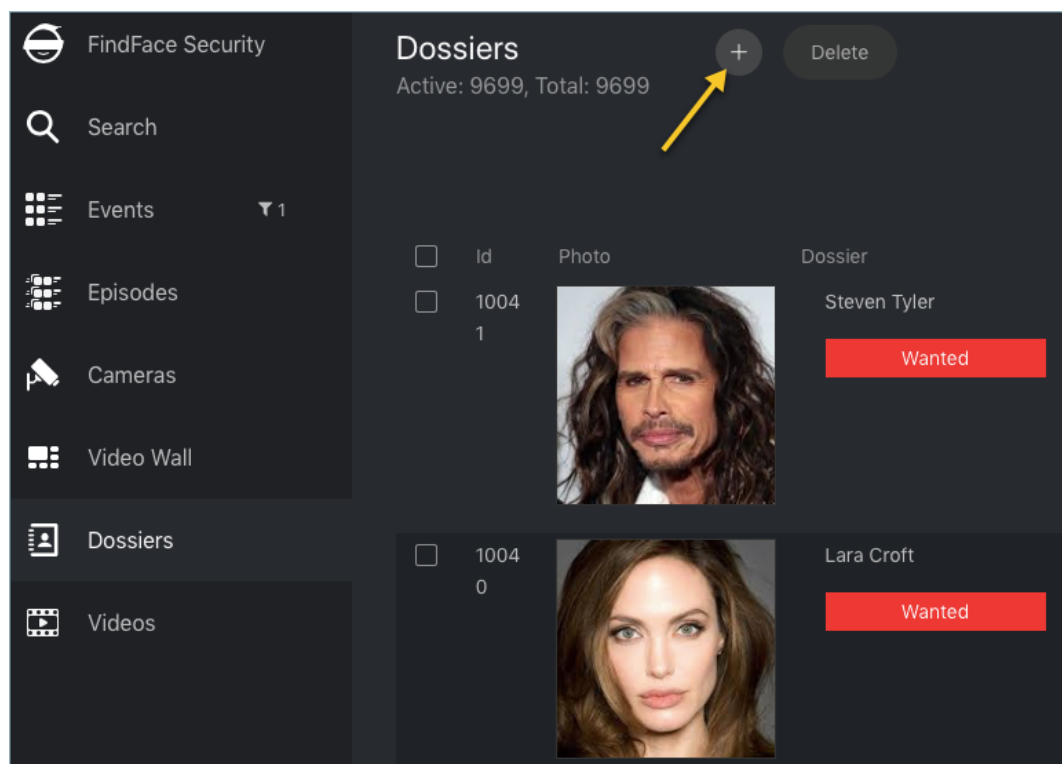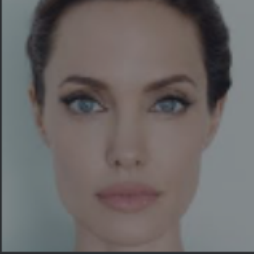
1. Navigate to the *Dossiers* tab.

2. Click +.

3. Attach a photo and specify the name of a person. If necessary, add a comment.

Important: A face in the photo must be of high quality, i.e. close to a frontal position. Distance between pupils: 60 px. Supported formats: WEBP, JPG, BMP, PNG. Photos that do not meet the requirements will be rejected with a detailed error description.

---

4. From the *Watch lists* drop-down menu, select a classification list (or several lists, one by one) for the dossier.

5. Check *Active*. If a dossier is inactive, it is excluded from the real time face identification.

6. Click *Save*.

Batch Photo Upload

To create dossiers in bulk, use the batch photo upload. Do the following:

Tip: If you need to upload a large number of photos (more than 10,000), use *Console Bulk Photo Upload*.

Important: Faces in photos must be of high quality, i.e. close to a frontal position. Distance between pupils: 60 px. Supported formats: WEBP, JPG, BMP, PNG. Photos that do not meet the requirements will be rejected with a detailed error description.

---

1. Navigate to the *Dossiers* tab.
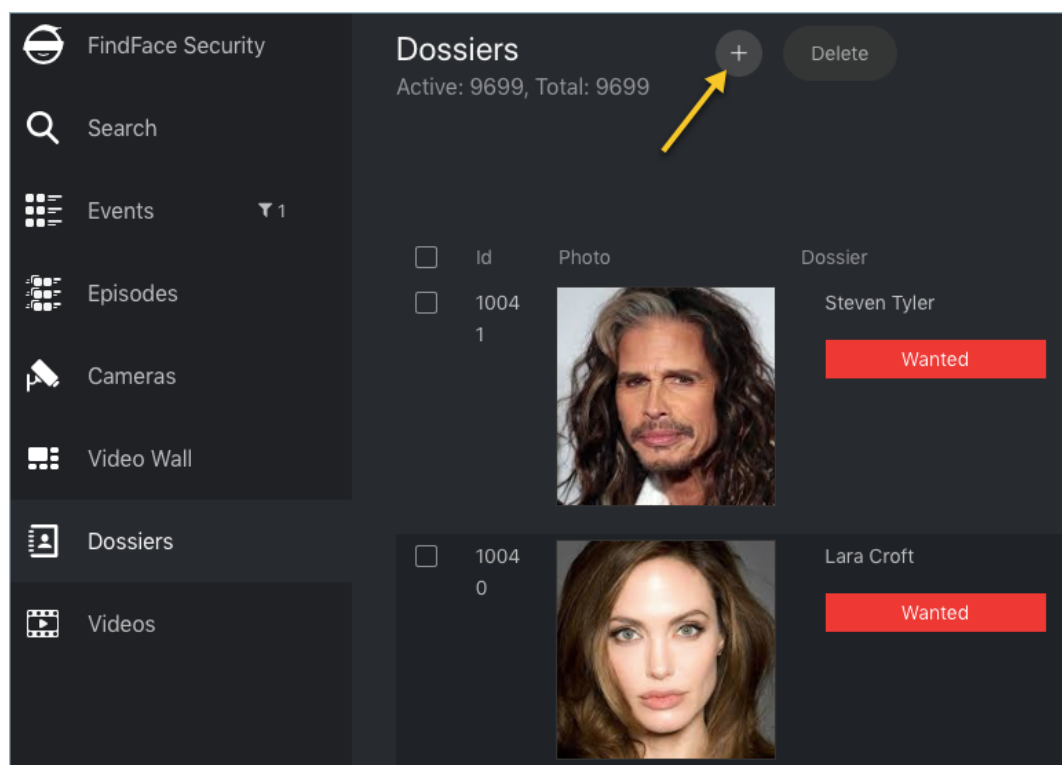
2. Click +.

3. Click *Batch Dossier Upload*.
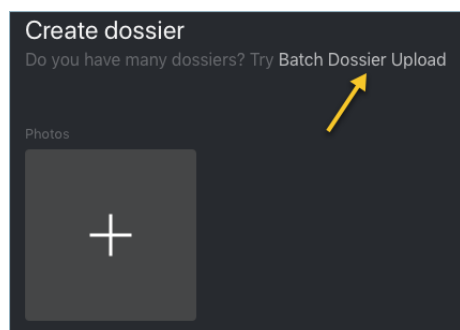
4. Select multiple image files, or a folder.

5. You can use image file names as a basis for names and/or comments in dossiers to be created. Select the necessary option(s). Then configure the automatic name/comment generation rule by appending a custom prefix and/or postfix to the file name.
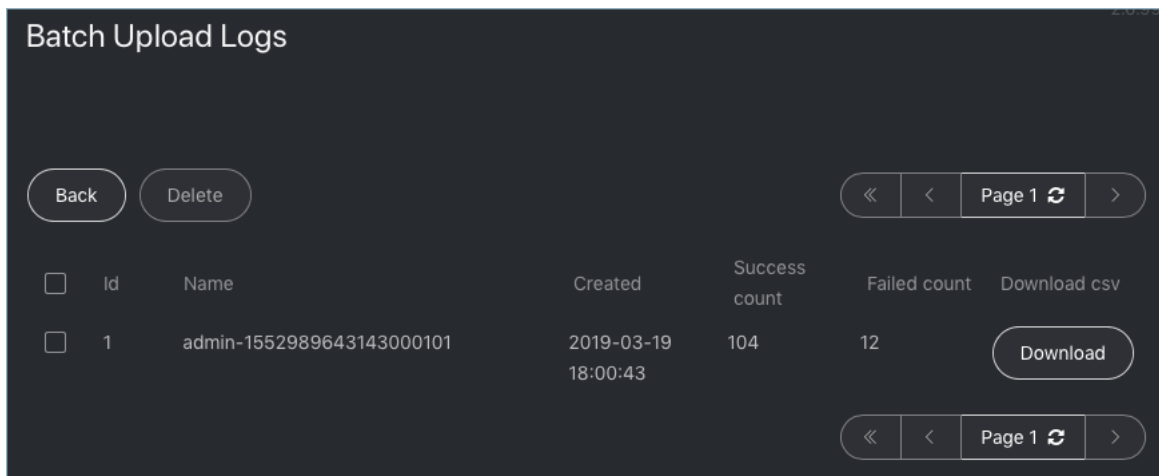
Tip: To avoid merging the 3 words into one, use underscore or another symbol in the prefix and postfix.

---

6. From the *Watch lists* drop-down menu, select a classification list for the dossiers.

7. Use the *Parallel Upload* option to specify the number of photo upload streams. The more streams you use, the

faster it takes to complete the upload, however it requires more resources as well.

8. From the *Group Photo* drop-down menu, select the system behavior upon detecting several faces in a photo: reject the photo, or upload the biggest face.

9. Click *Start* to launch the photo upload.
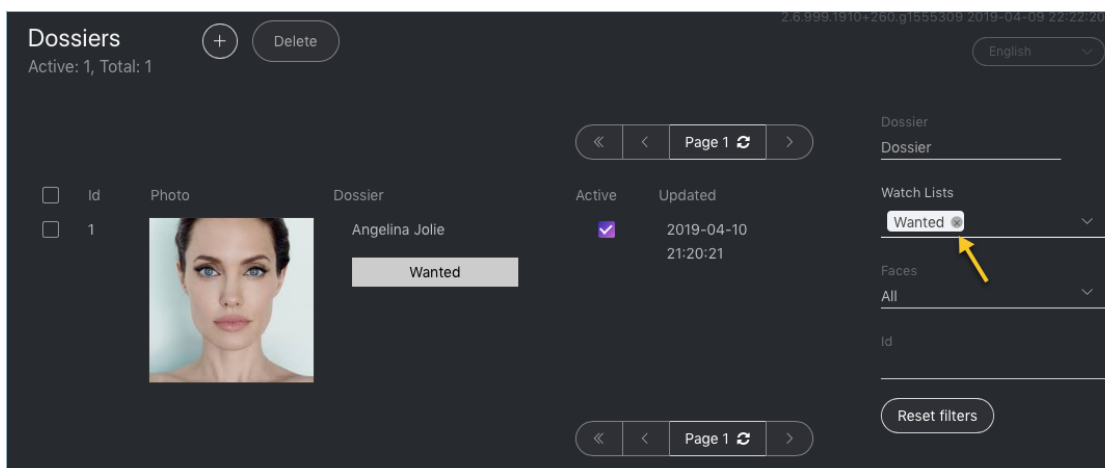
Important: To view the batch photo upload log, click *Logs*. You can then download the log in the `.csv` format if needed.
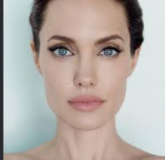
Filter Dossiers by Watch List

You can find all dossiers created in FindFace Security on the *Dossiers* tab. Use the *Watch lists* filter to filter dossiers by list.

1.6.3 User Management

In this chapter:

- *Predefined Roles*
- *Create Custom Role*
- *Primary and Additional User Privileges*
- *Create User*
- *Deactivate or Delete User*

Predefined Roles

FindFace Security provides the following predefined roles:

- Administrator has rights to *manage cameras*, events, FindFace Security users, the *dossier database*, and full access to all other functions.

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

- Operator can *create dossiers manually*, receive and acknowledge events, and search for faces on the event list. The other data is available read-only. The *batch dossier creation* is unavailable.

- User has a right to receive and acknowledge events, and to search for faces on the event list. The other data is available read-only.

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

Create Custom Role

To create a custom role, do the following:

1. Navigate to the *Preferences* tab. Click *Roles*.

2. Click +.
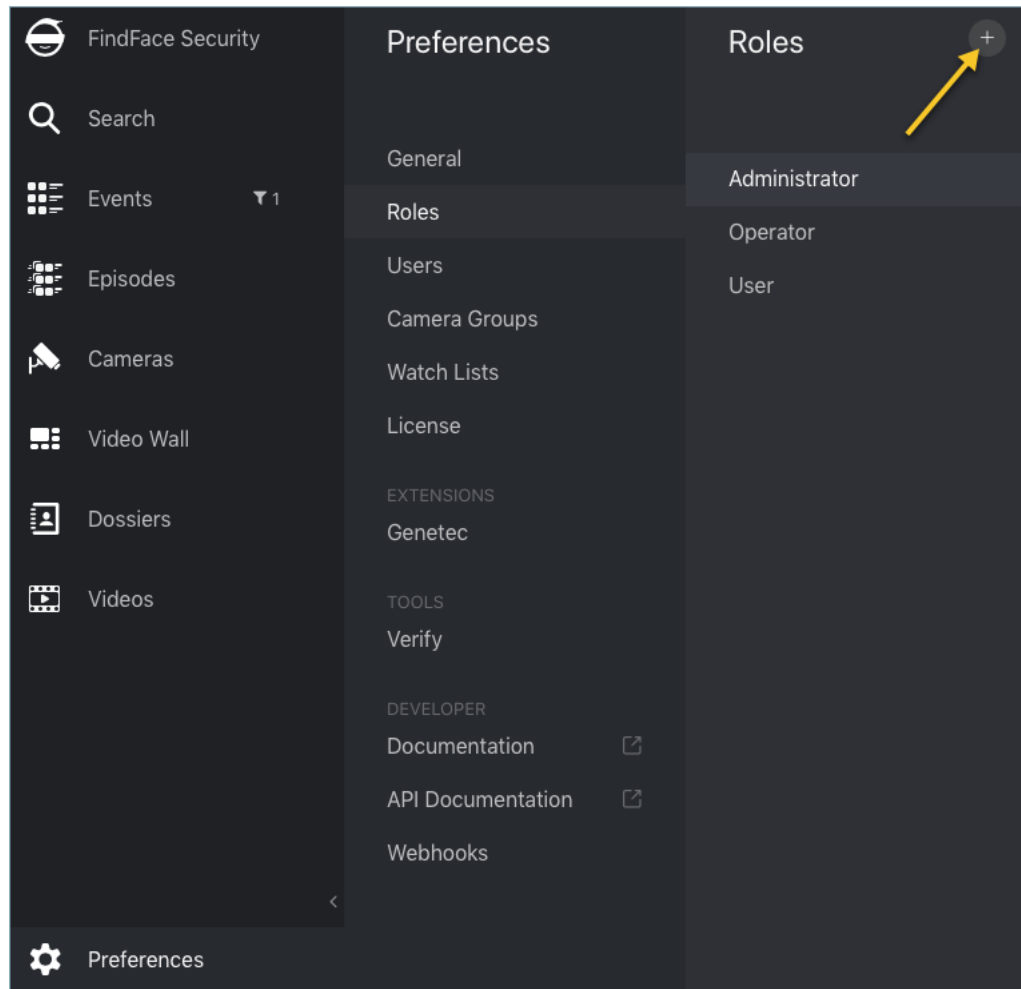
3. On the *Information* tab, specify the role name.
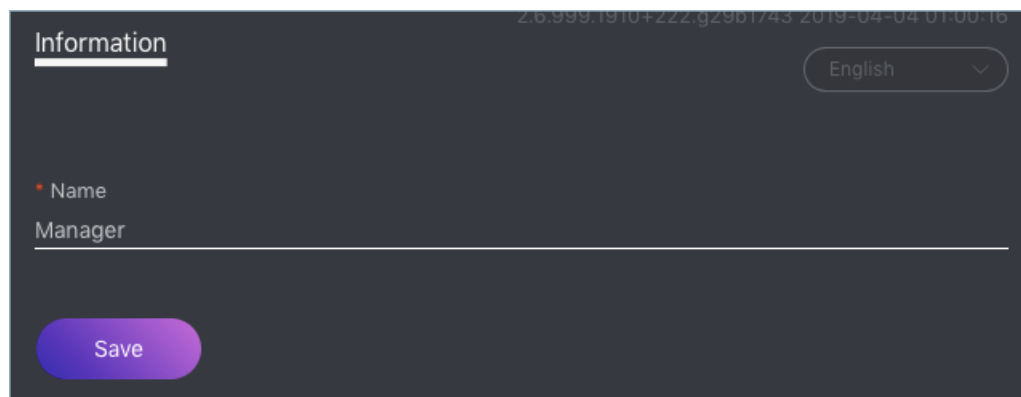
4. Click *Save*. You will see additional tabs appear next to the *Information* tab. You can use these tabs to assign the role privileges for specific watch lists (the *Watch Lists* tab) and camera groups (*Camera Groups*), as well as for entire system functions and entities (*Permissions*).

Note: For example, if you set `None` for a certain camera group on the *Camera Groups* tab, users with this role won't be able to work with **this** very group of cameras. Setting `None` for `cameragroup` on the *Permissions*

---

tab will prevent users from viewing and working with **all** camera groups.

Note: The right for an event consists of the rights for a corresponding camera and watch list. To see unmatched events, you only need the rights for a camera.

The full list of the FindFace Security entities is as follows:

- `dossierlist`: *watch list*
- `dossier`: *dossier*
- `dossierface`: *photo in a dossier*
- `cameragroup`: *camera group*
- `camera`: *camera*
- `listevent`: *event list*
- `eventepisode`: *episodes*
- `uploadlist`: list of photos in *batch upload*
- `upload`: item (photo) in batch photo upload
- `user`: *user*
- `group`: *user role*
- `hook`: *webhook*
- `videosource`: *face identification in offline video*

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

- `configure_genetec`: configuration of *Genetec integration*
- `configure_ntls`: configuration of the `findface-ntls` *license server*
- `batchupload_dossier`: *batch photo upload*
- `view_runtimesetting`: viewing the FindFace Security *general preferences*
- `change_runtimesetting`: changing the FindFace Security general preferences

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.

The different between a primary and additional roles is the following. If a user is assigned a certain primary role, this role will be **automatically** granted the `Change` permissions for all objects newly created by this user (cameras, watch lists, dossiers, etc.). This doesn't happen when you assign an additional role. For example, if a user is assigned a primary role `Manager`, all users with the `Manager` role will be able to change the objects newly created by this user. On the contrary, if you assign `Manager` as an additional role, other users with the `Manager` role will need a relevant permission to change the objects created by the user.

See also:

Create User

Create User

To create a user, do the following:

1. Navigate to the *Preferences* tab. Click *Users*.

2. Click +.
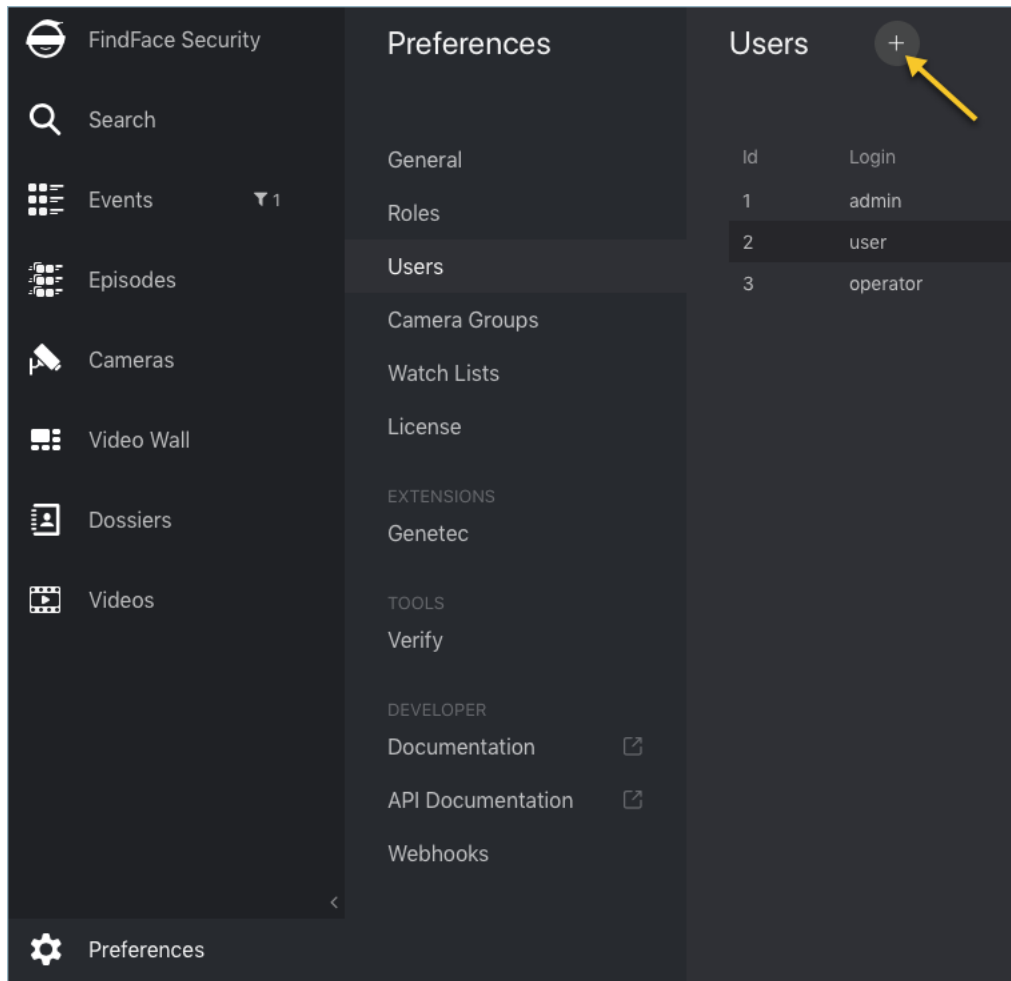
3. Specify such user data as name, login and password. If necessary, add a comment.

4. From the *Roles* drop-down menu, select one or several user roles. Set one of them as the *Primary role*.

5. Check *Active*.

6. Click *Create*.

Deactivate or Delete User

In order to deactivate a user, simply uncheck *Active* on the user list (*Preferences -> Users*).

To delete a user from FindFace Security, click on the user login on the list. Click *Delete*.

1.6.4 Face Identification in Offline Videos

Besides real-time face identification, FindFace Security allows for offline video processing. This functionality has a wide range of possible applications, among which the most common case is face detection and recognition in archived videos.

To identify faces in an offline video, do the following:

1. Create a *camera group* with basic settings.

2. Assign this camera group to all *watch lists* that you want to monitor when processing the video.

3. Create a video in FindFace Security by uploading it from a file or online storage/cloud. To do so, navigate to the *Videos* tab.

4. Click +.
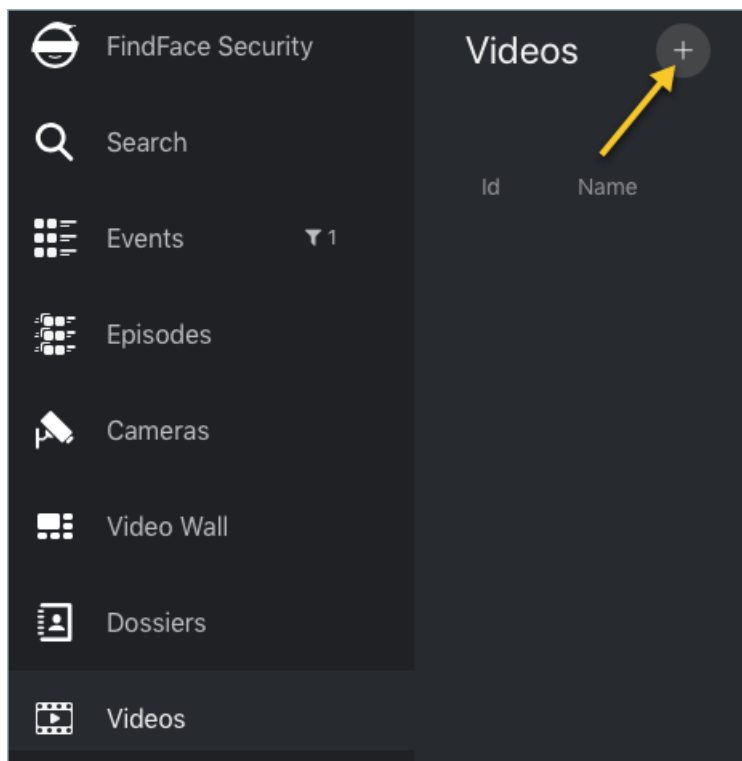
5. Specify the video name.
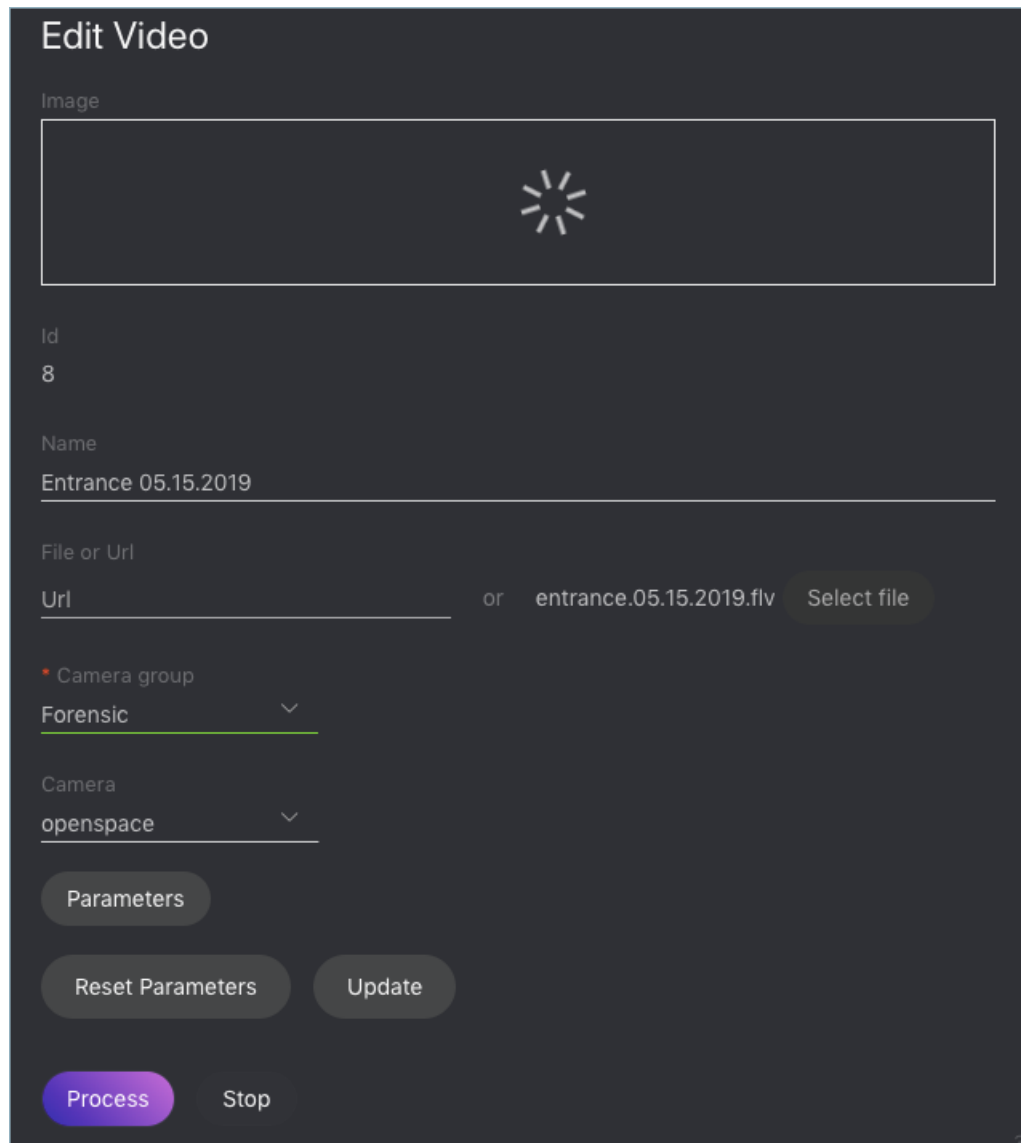
6. Specify the video URL in an online storage, or select a video file.

7. Select the camera group that you have just created.

8. (Optional) Select a camera to which you want to attribute the face recognition events found in the video.

9. (Optional) Specify parameters of video processing in the same manner as you do when configuring a *camera*.

10. Click *Save* to upload the video.

11. Once the video uploaded, click *Process* to start face identification. To view face identification events, navigate to the *Events* tab and filter the list of events by the camera group associated with the video.

1.6.5 General Preferences

To configure the confidence threshold for face verification and automatic events/episodes cleanup, navigate to the *Preferences* tab. Click *General*. After you are finished, click *Update*.
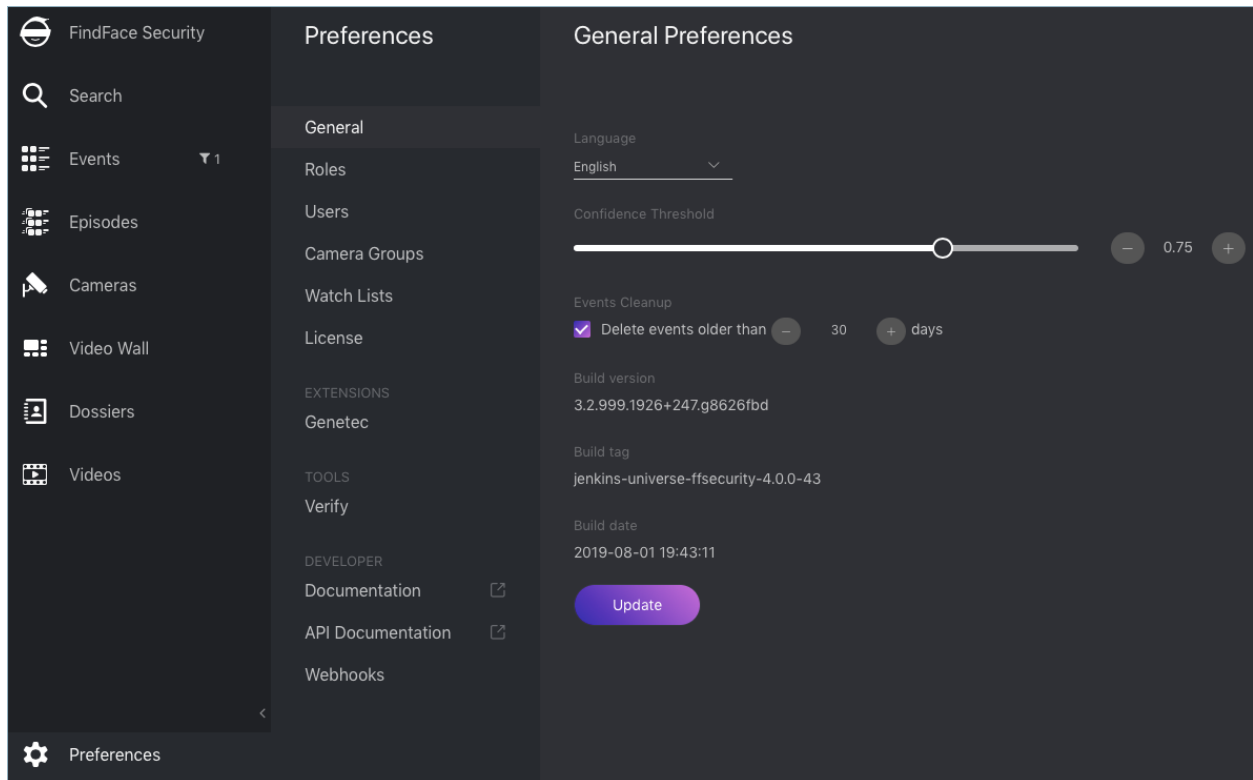
In this section:

- *Confidence Threshold*
- *Automatic Events Cleanup*

Confidence Threshold

FindFace Security verifies that a detected face and some face from the dossiers belong to the same person (i. e. the faces match), based on the pre-defined similarity threshold. The default threshold is set to 0.75 which can be considered as optimum. If necessary, you can change the threshold.

Note: The higher is the threshold, the less are chances that a wrong person will be positively verified, however, some valid photos may also fail verification.

Automatic Events Cleanup

Use the same tab to schedule purging old events and related episodes from the database regularly.

1.6.6 Compare Faces

FindFace Security allows you to compare 2 faces. Do the following:

1. Navigate to the *Preferences* tab. Click *Verify*.
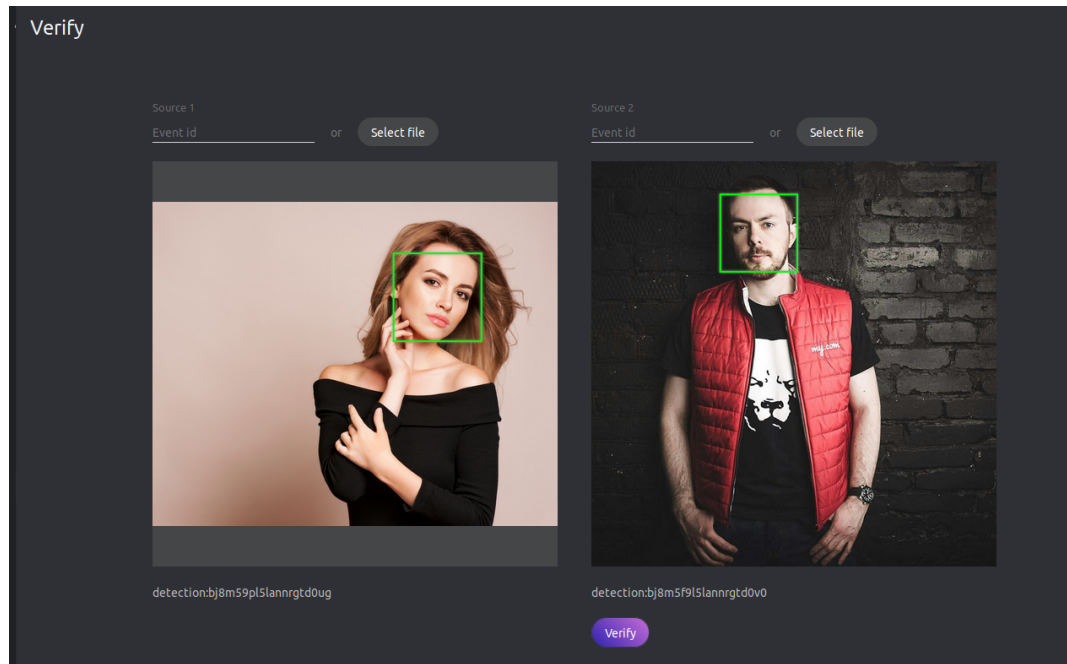
2. Specify the IDs of events that feature the faces you want to compare, and/or upload photos with the faces.

Tip: You can find event IDs on the *Events* tab.

---

3. Click *Verify*. You will see the probability of the faces belonging to the same person appear.
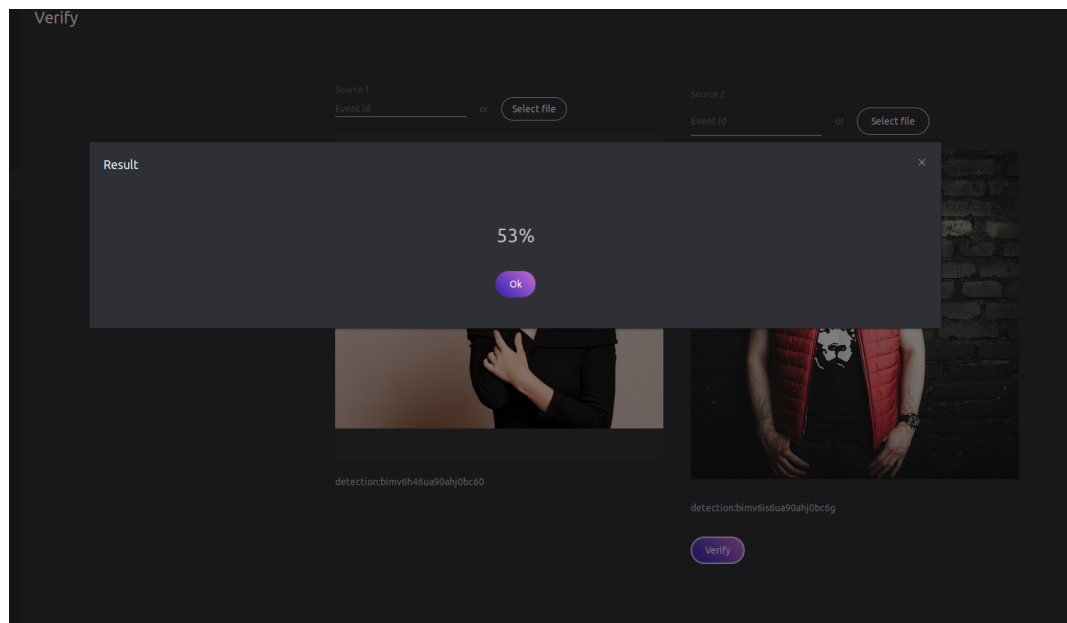
1.7 Advanced Functionality

1.7.1 Configure Episodes

In this section:

- *About Episodes*
- *Episode Settings*
- *Grant Rights for Episodes*

About Episodes

An episode is a set of identification events that feature faces of the same person, detected within a certain period of time.

There are two types of episodes:

- LIVE: an episode is currently active, with more events to be possibly added.

- Closed: an episode is closed, no events can be added.

Episode Settings

To configure the episodes, use the `findface-security` configuration file. You need to add the following parameters into the `FFSECURITY` section:

- `EPISODE_SEARCH_INTERVAL`: The period of time preceding an event, within which the system searches the biometric database for events with similar faces. If no such an event is found, the system creates a new episode. Otherwise, it picks up the most relevant event from a LIVE episode after sorting out the 100 most recent similar faces.

  > **Note:** The threshold similarity in episodes is the same as for face verification. See *General Preferences*.

- `EPISODE_MAX_DURATION`: The maximum episode duration in seconds. After this time, an episode automatically closes.

- `EPISODE_EVENT_TIMEOUT`: The maximum time in seconds since the last event has been added to an episode. After this time, an episode automatically closes.

```
sudo vi /etc/ffsecurity/config.py

...

FFSECURITY = {
    ...
    'EPISODE_SEARCH_INTERVAL': 60,
    'EPISODE_MAX_DURATION': 300,
    'EPISODE_EVENT_TIMEOUT': 30,
    ...
```

(continues on next page)

```
}

...
```

See also:

To see episodes work, navigate to the *Episodes* tab. See *Organize Events with Episodes* for details.

Grant Rights for Episodes

A user receives a notification of a new episode if they have rights for the first event. Viewing new events in the episode also requires proper rights.

The right for an event consists of the rights for a corresponding camera and watch list.

Note: To see unmatched events, you only need the rights for a camera.

To manage rights of a role for the entire `Episode` entity, open permissions for this role and adjust the `eventepisode` permission.

Tip: See *User Management*.

1.7.2 Allocate `findface-video-worker` to Camera Group

In distributed architectures, 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, simply allocate the local `findface-video-worker` to the camera group.

Do the following:

1. Navigate to the *Preferences* tab. Click *Camera Groups*.

2. Open the camera group settings.

3. In the *Labels*, create or select one or several allocation labels. Save changes.

4. Open the `findface-video-worker` configuration file and specify the allocation labels in the following format: `label_name=true` (label `terminal_1` in the example below).

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

labels = terminal_1=true
```

5. Restart `findface-video-worker`.

```
sudo systemctl restart findface-video-worker-cpu.service
sudo systemctl restart findface-video-worker-gpu.service
```

Note: If a camera is assigned an allocation label, its video stream can be processed by a `findface-video-worker` instance with the same label, as well as by all unlabeled `findface-video-worker` instances.

Warning: If a labeled camera is processed by an unlabeled `findface-video-worker` instance and a free similar-labeled instance appears, the camera won't automatically switch to the latter. To switch the camera, restart the similar-labeled `findface-video-worker` instance.

1.7.3 Console Bulk Photo Upload

To bulk-upload photos to the dossier database, you can use the **findface-security-uploader** utility from the FindFace Security package (in addition to the web interface upload functionality). Use this utility when you need to upload a large number of photos (more than 10,000).

Tip: To view the **findface-security-uploader** help, execute:

```
findface-security-uploader --help
```

Do the following:

1. Write the list of photos and metastrings to a CSV or TSV file.

   Important: The file used as a metadata source must have the following format: `path to photo | metastring`.

   To prepare a TSV file, use either a `script` or the `find` command.

   Note: Both the script and the command in the examples below create the `images.tsv` file. Each image in the list will be associated with a metastring coinciding with the image file name in the format `path to photo | metastring`.

   To build a TSV file listing photos from a specified directory (`/home/user/25_celeb/` in the example below), run the following command:

```
python3 tsv_builder.py /home/user/25_celeb/
```

The `find` usage example:

```
find photos/ -type f -iname '*g' | while read x; do y="${x%.*}"; printf "%s\t%s\n
↪" "$x" "${y##*/}"; done
```

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

```
findface-security-uploader add images.tsv
```

The `add` options:

- `--format`: input file format, `tsv` by default,

- `--delimiter`: field delimiter, by default `"\t"` for TSV, and `","` for CSV.

Note: A job file represents a sqlite database which can be opened on the **sqlite3** console.

---

3. Process the job file by using `run`.

```
findface-security-uploader run --dossier-lists 2 --api http://127.0.0.1:80 --user
↪admin --password password
```

The `run` options:

- `--parallel`: the number of photo upload threads, 10 by default. The more threads you use, the faster the bulk upload is completed, however it requires more resources too.

- `--api`: `findface-security` API URL, `http://127.0.0.1:80/` by default.

- `--user`: login.

- `--password`: password.

- `--dossier-lists`: comma-separated list of the watch lists id's.

- `--failed`: should an error occur during the job file processing, correct the mistake and try again with this option.

1.7.4 Deduplicate Events

In this section:

- *How It Works*
- *Enable Deduplication*

If observation scenes of cameras within one group overlap, consider to enable Deduplication. This feature allows you to exclude coinciding facial recognition events among cameras belonging to the same group.

Warning: Use deduplication with extreme caution, as if cameras within a group observe different scenes, some faces may be skipped.

How It Works

The deduplication algorithm's infographics are shown in the diagram below:

1. If the video face detector is working in the offline mode without deduplication, the server receives one best face snapshot per camera. We recommend to use this mode if cameras in the same group observe different scenes.

2. If the video face detector is working in the online mode without deduplication, the server receives several images from each camera of a group. This mode is the most storage intensive. In the case of large number of visitors, security operators may also experience difficulties dealing with a large number of identical face recognition events.

3. With enabled deduplication, the server receives only one face snapshot per group, the best one in the current tracking session whatever the video face detector mode. Use deduplication only if the observation scenes of cameras within a group overlap.

Enable Deduplication

To enable event deduplication, do the following:

1. Navigate to the *Preferences* tab. Click *Camera Groups*.

2. Open the camera group settings.

3. Check *Deduplicate Events* and specify the deduplication interval in seconds (interval between 2 consecutive checks for event uniqueness).

1.7.5 Real-time Face Liveness Detection

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

To spot fake faces and prevent photo attacks, use the integrated 2D anti-spoofing system that distinguishes a live face from a face image. Due to the analysis of not one, but a number of frames, the algorithm captures any changes in a facial expression and skin texture. This ensures that it is a live person in front of a camera and eliminates the possibility of fraud using images on paper or mobile device screens.

The liveness detector estimates a face liveness with a certain level of confidence and returns the confidence score along with a binary result `real`/`fake`, depending on the pre-defined liveness threshold.

In this section:

- *Enable Face Liveness Detector*
- *Configure Liveness Threshold*
- *Face Liveness in Web Interface*

Enable Face Liveness Detector

To enable the face liveness detector, do the following:

1. Open the `/etc/findface-video-worker-gpu.ini` (`/etc/findface-video-worker-cpu.ini`) configuration file. In the `liveness` –> `fnk` parameter, specify the path to the face liveness detector model as shown below.

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

[liveness]
#----------------------------
## path to liveness fnk
fnk = /usr/share/findface-data/models/faceattr/liveness.v3.gpu.fnk
```

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

[liveness]
#----------------------------
## path to liveness fnk
fnk = /usr/share/findface-data/models/faceattr/liveness.v3.cpu.fnk
```

2. Restart `findface-video-worker`.

```
sudo systemctl restart findface-video-worker-gpu
sudo systemctl restart findface-video-worker-cpu
```

Configure Liveness Threshold

If necessary, you can adjust the liveness `threshold` in the `/etc/ffsecurity/config.py` configuration file. The liveness detector will estimate a face liveness with a certain level of confidence. Depending on the threshold value,

---

it will return a binary result `real` or `fake`.

Note: The default value is optimal. Before changing the threshold, we recommend you to seek advice from our experts by support@ntechlab.com.

```
sudo vi /etc/ffsecurity/config.py

'LIVENESS_THRESHOLD': 0.75,
```

Face Liveness in Web Interface

Once the face liveness detector configured, you will see liveness estimation for each event.

Note: The liveness score is `null` when the liveness detector is unable to estimate the face liveness in the provided image.

Use the *Liveness* filter to display only real or only fake faces in the event list.

1.7.6 Face Features Recognition

Subject to your needs, you can enable automatic recognition of such face features as gender, age, emotions, glasses, and/or beard. This functionality can be activated on both GPU- and CPU-accelerated video face detectors.

In this section:

- *Enable Face Features Recognition*
- *Display Features Recognition Results in Events*
- *Face Features in Events*

Enable Face Features Recognition

Important: This step will enable face features recognition via HTTP API.

To enable automatic recognition of face features, open the `/etc/findface-extraction-api` configuration file and enable relevant recognition models: gender, age, emotions, glasses3, and/or beard. 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.

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

models:
  age: faceattr/age.v1.cpu.fnk
  emotions: faceattr/emotions.v1.cpu.fnk
  face: face/grapefruit_480.cpu.fnk
  gender: faceattr/gender.v2.cpu.fnk
  beard: faceattr/beard.v0.cpu.fnk
  glasses3: faceattr/glasses3.v0.cpu.fnk
```

The following models are available:

Note: You can find face features recognition models at `/usr/share/findface-data/models/faceattr/`.

```
ls /usr/share/findface-data/models/faceattr/
age.v1.cpu.fnk  age.v1.gpu.fnk  beard.v0.cpu.fnk  beard.v0.gpu.fnk  emotions.v1.cpu.
↪fnk  emotions.v1.gpu.fnk  gender.v2.cpu.fnk  gender.v2.gpu.fnk  glasses3.v0.cpu.fnk␣
↪  glasses3.v0.gpu.fnk  liveness.v3.gpu.fnk
```

Face feature	Acceleration	Configuration file parameter
face (biometry)	CPU	`face: face/grapefruit_480.cpu.fnk        face: face/grapefruit_160.cpu.fnk`
	GPU	`face: face/grapefruit_480.gpu.fnk        face: face/grapefruit_160.gpu.fnk`
age	CPU	`age: faceattr/age.v1.cpu.fnk`
	GPU	`age: faceattr/age.v1.gpu.fnk`
gender	CPU	`gender: faceattr/gender.v2.cpu.fnk`
	GPU	`gender: faceattr/gender.v2.gpu.fnk`
emotions	CPU	`emotions: faceattr/emotions.v1.cpu.fnk`
	GPU	`emotions: faceattr/emotions.v1.gpu.fnk`
glasses3	CPU	`glasses3: faceattr/glasses3.v0.cpu.fnk`
	GPU	`glasses3: faceattr/glasses3.v0.gpu.fnk`
beard	CPU	`beard: faceattr/beard.v0.cpu.fnk`
	GPU	`beard: faceattr/beard.v0.gpu.fnk`

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

```
models:
  gender: ""
  age: ""
  emotions: ""
```

Restart `findface-extraction-api`.

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

Once the models are enabled, be sure to *configure* the web interface to display the recognition results.

Display Features Recognition Results in Events

To display the face features recognition results in the event list, add the following line into the `FFSECURITY` section: 'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'], subject to the list of enabled models.

Warning: This line must be placed between `SF_API_ADDRESS` and `LIVENESS_THRESHOLD` as shown in the example.

```
sudo vi /etc/ffsecurity/config.py


...
FFSECURITY = {
...
'SF_API_ADDRESS': 'http://127.0.0.1:18411',
'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'],
'LIVENESS_THRESHOLD': 0.75,
'BEARD_THRESHOLD': 0.7,
 }
```

Restart `findface-security`.

```
sudo systemctl restart findface-security
```

Face Features in Events

Once the face features recognition configured, you will see the recognition result for each found face in the following format:

Face feature	Result format	Example
Age	Feature: `age`: number of years	age: 33
Gender	Result: `male/female` (feature: `gender`): algorithm confidence in result	female (gender): 0.95
Emotions	Result: `angry/disgust/fear/happy/sad/surprise` (feature: `emotions`): algorithm confidence in result	happy (emotions): 0.99
Glasses	Result: `eye/sun/none` (feature: `glasses`): algorithm confidence in result	none (glasses): 0.87
Beard	Result: `beard/none` (feature: `beard`): algorithm confidence in result	none (beard): 0.91

Filter events by face features when needed.

1.7.7 Multiple Video Cards Usage

Should you have several video cards installed on a physical server, you can create additional `findface-extraction-api-gpu` or `findface-video-worker-gpu` instances and distribute them across the video cards, one instance per card.

In this section:

- *Distribute `findface-extraction-api-gpu` Instances Across Several Video Cards*
- *Allocate `findface-video-worker-gpu` to Additional Video Card*

Distribute `findface-extraction-api-gpu` Instances Across Several Video Cards

To distribute `findface-extraction-api-gpu` instances across several video cards, do the following:

1. Stop the initial `findface-extraction-api-gpu` service.

```
sudo service findface-extraction-api stop
```

2. Create several copies of the `findface-extraction-api-gpu` configuration file, subject to how many video cards you are going to use for biometric samples extraction. Append the appropriate GPU device numbers to the new configuration files names as shown in the example below (GPU devices #0 and #6).

```
/etc/findface-extraction-api@0.ini
/etc/findface-extraction-api@6.ini
```

3. Open the new configuration files. Specify the GPU device numbers and adjust the listening ports.

```
sudo vi /etc/findface-extraction-api@0.ini

listen: 127.0.0.1:18666
...
extractors:
  gpu_device: 0
```

```
sudo vi /etc/findface-extraction-api@6.ini

listen: 127.0.0.1:18667
...
extractors:
  gpu_device: 6
```

4. Start the new services.

```
sudo service findface-extraction-api@0 start
sudo service findface-extraction-api@6 start
```

Allocate `findface-video-worker-gpu` to Additional Video Card

To create an additional `findface-video-worker-gpu` instance and allocate it to a different video card, do the following:

1. Display the status of the `findface-video-worker-gpu` primary service by executing:

```
sudo systemctl status findface-video-worker-gpu.service
```

2. Find the full path to the service in the line `Loaded: loaded (/lib/systemd/system/findface-video-worker-gpu.service; enabled; vendor preset: enabled.` It is `findface-video-worker-gpu.service` in our example (name may vary). Create a copy of the service under a new name.

```
sudo cp /lib/systemd/system/findface-video-worker-gpu.service /lib/systemd/system/
→findface-video-worker-gpu2.service`
```

3. In the same manner, create a copy of the primary service configuration file under a new name.

```
sudo cp /etc/findface-video-worker-gpu.ini /etc/findface-video-worker-gpu2.ini
```

4. Open the just created configuration file and actualize the GPU device number to use.

```
sudo vim /etc/findface-video-worker-gpu2.ini

## cuda device number
device_number = 1
```

5. Open the new service and specify the just created configuration file.

```
sudo vim /lib/systemd/system/findface-video-worker-gpu2.service

ExecStart=/usr/bin/findface-video-worker-gpu --config /etc/findface-video-worker-
→gpu2.ini
```

6. Reload the `systemd` daemon to apply the changes.

```
sudo systemctl daemon-reload
```

7. Enable the new service autostart.

```
sudo systemctl enable findface-video-worker-gpu2.service

Created symlink from /etc/systemd/system/multi-user.target.wants/findface-video-
→worker-gpu2.service to /lib/systemd/system/findface-video-worker-gpu2.service
```

8. Launch the new service.

```
sudo systemctl start findface-video-worker-gpu2.service
```

9. Check the both `findface-video-worker-gpu` services status.

```
sudo systemctl status findface-video-worker-* | grep -i 'Active:' -B 3

 findface-video-worker-gpu2.service - findface-video-worker-gpu daemon
  Loaded: loaded (/lib/systemd/system/findface-video-worker-gpu2.service; enabled;
→ vendor preset: enabled)
  Active: active (running) since Thu 2019-07-18 10:32:02 MSK; 1min 11s ago

...

  findface-video-worker-gpu.service - findface-video-worker-gpu daemon
```

(continues on next page)

```
  Loaded: loaded (/lib/systemd/system/findface-video-worker-gpu.service; enabled;␣
→vendor preset: enabled)
  Active: active (running) since Mon 2019-07-15 15:18:33 MSK; 2 days ago
```

1.7.8 Direct API Requests to Tarantool

You can use HTTP API to extract data directly from the Tarantool Database.

In this section:

- *General Information*
- *Add Face*
- *Remove Face*
- *Face Search*
- *Edit Face Metadata and Feature Vector*
- *List Galleries*
- *Get Gallery Info*
- *Create Gallery*
- *Remove Gallery*

General Information

API requests to Tarantool are to be sent to `http://<tarantool_host_ip:port>`.

Tip: The port for API requests can be found in the `FindFace.start` section of the Tarantool configuration file:

```
cat /etc/tarantool/instances.enabled/FindFace.lua

##8001:
FindFace.start("127.0.0.1", 8001)
```

Note: In the case of the standalone deployment, you can access Tarantool by default only locally (127.0.0.1). If you want to access Tarantool remotely, *alter* the Tarantool configuration file.

API requests to Tarantool may contain the following parameters in path segments:

- `:ver`: API version (v2 at the moment).
- `:name`: gallery name.

Tip: To list gallery names on a shard, type in the following command in the address bar of your browser:

```
http://<tarantool_host_ip:shard_port>/stat/list/1/99
```

The same command on the console is as such:

```
curl <tarantool_host_ip:shard_port>/stat/list/1/99 \| jq
```

You can also list gallery names by using a direct request to Tarantool:

```
echo 'box.space.galleries:select()' | tarantoolctl connect <tarantool_host_
↪ip:shard_port>
```

Note that if there is a large number of shards in the system, chances are that a randomly taken shard does not contain all the existing galleries. In this case, just list galleries on several shards.

Add Face

```
POST /:ver/faces/add/:name
```

Parameters in body:

JSON-encoded array of faces with the following fields:

- `"id"`: face id in the gallery, uint64_t,

- `"facen"`: raw feature vector, base64,

- `"meta"`: face metadata, dictionary.

Returns:

- HTTP 200 and empty body on success.

- HTTP 404 if a gallery with the given name doesn't exist.

- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8001/v2/faces/add/testgal' --data '
[
  {
    "id": 9223372036854776000,
    "facen": "qgI3vZRv/z...NpO9MdHavW1WuT0=",
    "meta": {
"cam_id": "223900",
"person_name": "Mary Ostin",

    }
  }
]
```

Response

```
HTTP/1.1 200 Ok
Content-length: 1234
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

Remove Face

```
POST /v2/faces/delete/:name
```

Parameters in body:

JSON-encoded array of face ids to be removed

Returns:

- HTTP 200 and empty body on success.

- HTTP 404 if a face with the given id is not found in the gallery.

- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8001/v2/faces/delete/testgal' --data '[1, 4, 922, 3]'
```

Response

```
HTTP/1.1 200 Ok
Content-length: 111
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

Face Search

```
POST /v2/faces/search/:name
```

Parameters in body:

JSON-encoded search request with the following fields:

- `limit`: maximum number of faces in the response.

- `sort`: sorting order. Pass one of the following values: `id`: increasing order by id, `-id`: decreasing order by id, `-score`: decreasing order by face similarity (only if you search for faces with similar feature vectors).

- `filter` (filters):

    - `facen`: (optional) search for faces with similar feature vectors. Pass a dictionary with the following fields: `data`: raw feature vector, base64; `score`: range of similarity between faces [threshold similarity; `1`], where `1` is 100% match.

    - `id` and `meta/<meta_key>`: search by face id and metastring content. To set this filter, use the following operators:

        * `range`: range of values, only for numbers.

        * `set`: id or metastring must contain at least one value from a given set, for numbers and strings.

        * `subset`: id or metastring must include all values from a given subset, for numbers and strings.

        * `like`: by analogy with `like` in SQL requests: only 'aa%', '%aa', and '%aa%' are supported. Only for strings and set[string]. In the case of set[string], the filter will return result if at least one value meets the filter condition.

        * `ilike`: by analogy with `like` but case-insensitive, only for strings and set[string].

Returns:

- JSON-encoded array with faces on success. The value in the `X-search-stat` header indicates whether the fast index was used for the search: `with_index` or `without_index`.

Note: Fast index is not used in API v2.

- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8001/v2/testgal/search' --data '
{
        "limit": 2,
    "sort": {
                "score": -1
        },
        "filter": {
                "facen": {
                        "data": "qgI3vZRv/z0BQTk9rcirOyZrNpO9MdHavW1WuT0=",
                        "score": [0.75, 1]
                },
                "id": {
                        "range": [9223372036854000000, 9223372036854999000]
                },
                "meta": {
                        "person_id": {
                                "range": [444, 999]
                        },
                        "cam_id": {
                                "set": ["12767", "8632", "23989"]
                        }
                }
```

(continues on next page)

```
        }
}'
```

Response

```
HTTP/1.1 200 Ok
Content-length: 1234
X-search-stat: without_index
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{
  "results": [
    {
      "facen": " qgI3vZRv/z0BQTk9rcirOyZrNpO9MdHavW1WuT0=",
      "meta": {
        "timestamp": 0,
        "photo_hash": "",
        "person_id": 777,
        "cam_id": "8632"
      },
      "score": 0.9964,
      "id": 9223372036854776000
    }
  ]
}
```

Edit Face Metadata and Feature Vector

```
POST /v2/faces/update/:name
```

Parameters in body:

JSON-encoded array with faces with the following fields:

- `"id"`: face id, uint64_t.

- `"facen"`: (optional) new feature vector, base64. If omitted or passed as `null`, the relevant field in the database won't be updated.

- `"meta"`: dictionary with metadata to be updated. If some metastring is omitted or passed as `null`, the relevant field in the database won't be updated.

Returns:

- HTTP 200 and dictionary with all face parameters, including not updated, on success.

- HTTP 404 and error description if a face with the given id doesn't exist.

- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8001/v2/faces/update/sandbox' --data '[{"id":1,"facen
→":null,"meta":{"m:timestamp":1848}}]'
```

Response

```
HTTP/1.1 200 Ok
Content-length: 151
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"meta":{"m:timestamp":1848,"normalized_id":"1_b9pkrf00mjt6h1vmq1kg.png","m:cam_id":
→"a9f7a973-f07e-469d-a3bd-41ddd510b26f","feat":"{\"score\":0.123}"}, "id":1, ... }
```

List Galleries

```
POST /v2/galleries/list
```

Returns:

JSON-encoded array with galleries with the following fields: `name`: gallery name, `faces`: number of faces in a gallery.

Example

Request

```
curl -D - -s -X POST http://localhost:8001/v2/galleries/list
```

Response

```
HTTP/1.1 200 Ok
Content-length: 42
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{
  "results": [
    {
      "name": "testgal",
      "faces": 2
    }
  ]
}
```

Get Gallery Info

```
POST /v2/galleries/get/:name
```

Returns:

- HTTP 200 and dictionary with gallery parameters on success.

- HTTP 404 and error description if a gallery with the given name doesn't exist.

- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -s -X POST http://localhost:8001/v2/galleries/get/testgal
```

```
HTTP/1.1 200 Ok
Content-length: 11
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"faces":2}
```

Create Gallery

```
POST /v2/galleries/add/:name
```

Returns:

- HTTP 200 and empty body on success.

- with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -X POST -s 'http://localhost:8001/v2/galleries/add/123'
```

Response

```
HTTP/1.1 409 Conflict
Content-length: 57
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"error":{"message":"gallery already exists","code":409}}
```

Remove Gallery

```
POST /v2/galleries/delete/:name
```

Returns:

- HTTP 200 and empty on success.

- with a status other than 200 and error description in the body on failure.

Example

Request

```
curl -D - -X POST -s 'http://localhost:8001/v2/galleries/delete/123'
```

Response

```
HTTP/1.1 204 No content
Content-length: 0
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

1.8 Maintenance and Troubleshooting

1.8.1 Update FindFace Security to 4.x

In this section:

- *Update from 1.x to 4.x*
- *Update from 2.x to 4.x*

Update from 1.x to 4.x

To update FindFace Security from 1.x to 4.x, do the following:

1. Stop the services:

   ```
   sudo systemctl stop findface-extraction*
   sudo systemctl stop findface-security*
   sudo systemctl stop findface-videomanager*
   ```

2. Install a new version according to your architecture outline, following instructions in *Deploy FindFace Security*.

3. Migrate data drom PostgreSQL to Tarantool.

Important: Before you proceed, make sure that the size of free disk space is equal or larger than the occupied space.

```
sudo findface-security tnt_migrate
```

Note: To purge PostgreSQL after migration is completed, execute the command with the option `--purge-sql`. All old data will be LOST.

```
sudo findface-security tnt_migrate --purge-sql
```

Note: It is absolutely data-safe to interrupt the migration process and resume it later.

Update from 2.x to 4.x

To update FindFace Security from 2.x to 4.x, do the following:

1. Open the `findface-security` configuration file. Save the values of the following parameters for later use: `EXTERNAL_ADDRESS`, `SECRET_KEY`, `VIDEO_DETECTOR_TOKEN`, `ROUTER_URL`.

```
sudo vi /etc/ffsecurity/config.py

EXTERNAL_ADDRESS = "http://172.20.77.58"

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

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

2. Stop the `findface-security` service.

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

3. Create a backup of the Tarantool-based biometric database in any directory of your choice, for example, `/tmp/dump`.

Tip: See *Backup and Restore Data Storages* for details.

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

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

---

5. Install the services from the repository, following your architecture outline.

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

6. Open the `findface-security` configuration file and paste the saved `EXTERNAL_ADDRESS`, `SECRET_KEY`, `VIDEO_DETECTOR_TOKEN`, and `ROUTER_URL` into it.

```
sudo vi /etc/ffsecurity/config.py
```

7. Modify the Tarantool database structure by applying the `tnt_schema.lua` file from the new version.

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

8. Remove the Tarantool database (default database or shards).

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

sudo rm -f /opt/ntech/var/lib/tarantool/shard-001/{index,snapshots,xlogs}/*
...
sudo rm -f /opt/ntech/var/lib/tarantool/shard-00N/{index,snapshots,xlogs}/*
```

9. Restart the Tarantool database.

```
systemctl restart tarantool*
```

10. Restore the Tarantool database from the backup.

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

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

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

12. Restart the `findface-security` service.

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

1.8.2 Backup and Restore Data Storages

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

- Tarantool-based biometric database that stores biometric samples (feature vectors) and face identification events.

- Main system database based on PostgreSQL, that stores internal system data, dossiers, user accounts, and camera settings.

- Directory `/var/lib/ffsecurity/uploads` that stores uploaded dossier photos, video files, and such event artifacts as full frames, face thumbnails, and normalized face images.

- Directory `/var/lib/ffupload/` that stores only such event artifacts as face thumbnails.

In this section:

- *Biometric Database Backup and Restore*
  - *Utilities*
  - *Backup Database*
  - *Restore Database*
- *Main Database Backup*
- *Artifacts Backup*

Biometric Database Backup and Restore

There are 3 galleries in the Tarantool-based biometric database:

- `ffsec_dossier_face`: biometric samples extracted from dossier photos.
- `ffsec_events`: biometric samples extracted from faces detected in the video.
- `ffsec_monitoring`: biometrics samples from the active dossiers under watch.

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

To avoid data loss, we recommend you to create a biometric database backup at least once a week. Overall, the frequency of backups depends on the number of dossiers and face recognition events, as well as available disk space.

Be sure to backup the database before *migrating* your system to another biometric model.

Utilities

To backup and restore the FindFace Security biometric database, the following utilities are needed:

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

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

Backup Database

To backup the biometric database, use the `findface-storage-api-dump` utility as follows:

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

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

---

1. On the server with `findface-sf-api`, create a directory to store the backup files.

2. From this directory, launch the `findface-storage-api-dump` utility by executing:

```
sudo findface-storage-api-dump -config /etc/findface-sf-api.ini
```

The utility will backup at once all the galleries into the files with corresponding names `ffsec_dossier_face.json`, `ffsec_events.json`, `ffsec_monitoring.json`, and save them into the directory. These files contain all the data needed to restore the entire database.

Restore Database

To restore the biometric database from the backup, do the following:

1. Use HTTP API to create initial galleries in the database: `ffsec_dossier_face`, `ffsec_events`, `ffsec_monitoring`.

Tip: See *HTTP API*.

```
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/add/ffsec_dossier_face'
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/add/ffsec_events'
curl -D - -X POST -s 'http://localhost:8101/v2/galleries/add/ffsec_monitoring'
```

2. From the directory with the backup files, launch the `findface-storage-api-restore` utility for each gallery at a time:

```
sudo findface-storage-api-restore -config /etc/findface-sf-api.ini < ffsec_
↪dossier_face.json
sudo findface-storage-api-restore -config /etc/findface-sf-api.ini < ffsec_events.
↪json
sudo findface-storage-api-restore -config /etc/findface-sf-api.ini < ffsec_
↪monitoring.json
```

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

See also:

- *Backup Options*
- *Restore Options*

Main Database Backup

To backup the PostgreSQL database, execute:

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

Artifacts Backup

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

- `/var/lib/ffsecurity/uploads`

- `/var/lib/ffupload/`

To backup the artifacts, execute:

```
tar -cvzf var_lib_ffsecurity_uploads.tar.gz /var/lib/ffsecurity/uploads
tar -cvzf var_lib_ffupload.tar.gz /var/lib/ffupload/
```

1.8.3 Migrate to Different Facen Model

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

Important: Before the migration, sure to *backup* the database.

Sometimes you have to migrate your face biometric data (facens) to another facen model. This usually happens when you decide to update to the latest version of the product.

To migrate to a different facen model, use the `findface-sf-api-migrate` utility. To pass migration settings, launch it with the `-config` option and provide a configuration file shown in the example below.

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

Example of the configuration file:

```
extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 0s
  extraction-api: http://127.0.0.1:18666
storage-api-from: # current location of the gallery
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  max-idle-conns-per-host: 20
  shards:
    - master: http://127.0.0.1:8001/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://127.0.0.1:8002/v2/
      slave: ""
workers_num: 3
faces_limit: 100
extraction_batch_size: 8
```

(continues on next page)

```
normalized_storage:
  type: webdav
  enabled: True
  webdav:
    upload-url: http://127.0.0.1:3333/uploads/
  s3:
    endpoint: 172.20.77.75:9000
    bucket-name: sf-api-normalized
    access-key: W0G6EQT6MC3BZC8136DW
    secret-access-key: XnottrdxRFp70wfEGdkvKgkzKZ3mEa2Y9bYmob4I
    secure: False
    region: ""
    operation-timeout: 10
    public-url: 123
```

Parameter		Description
`extraction-api` `extraction-api`	`->`	`findface-extraction-api` with a new facen model in its configuration file.
`storage-api-from`		Previous facen storage
`storage-api-to`		Storage for re-generated facens
`normalized_storage` `upload-url`	`->`	Storage of normalized face images.

1.8.4 Modify Biometric Database Structure

Sometimes it may be necessary to apply a new structural schema to your Tarantool-based biometric 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*

- *Structure Modification*

About Database Structure

In FindFace Security, the database structure is set via the `/etc/ffsecurity/tnt_schema.lua` file.

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

- id: field id;

- name: field name, must be the same as the name of a relevant face parameter;

- field_type: data type;

- default: field default value. If a default value exceeds '1e14 – 1', use a string data type to specify it, for example, `"123123.."` instead of `123123...`

You can find the default `tnt_schema.lua` file below:

```
scheme = {
    -- internal.normalized_id:
    {
        default = '',
        field_type = 'string',
        id = 1,
        name = 'normalized_id',
    },
    -- internal.feat:
    {
        default = '',
        field_type = 'string',
        id = 2,
        name = 'feat',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged:
    {
        default = 0,
        field_type = 'unsigned',
        id = 3,
        name = 'm:acknowledged',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged_by:
    {
        default = 0,
        field_type = 'unsigned',
        id = 4,
        name = 'm:acknowledged_by',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged_date:
    {
        default = 0,
        field_type = 'unsigned',
        id = 5,
     name = 'm:acknowledged_date',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:acknowledged_
→reaction:
    {
        default = '',
        field_type = 'string',
        id = 6,
        name = 'm:acknowledged_reaction',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:camera:
    {
        default = 0,
        field_type = 'unsigned',
        id = 7,
        name = 'm:camera',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:camera_group:
    {
        default = 0,
        field_type = 'unsigned',
        id = 8,
        name = 'm:camera_group',
```

(continues on next page)

```
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:confidence:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 9,
        name = 'm:confidence',
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:created_date:
    {
        default = 0,
        field_type = 'unsigned',
        id = 10,
        name = 'm:created_date',
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:episode:
    {
        default = 0,
        field_type = 'unsigned',
        id = 11,
        name = 'm:episode',
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_age:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 12,
        name = 'm:f_age',
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_beard_class:
    {
        default = '',
        field_type = 'string',
        id = 13,
        name = 'm:f_beard_class',
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_beard_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 14,
        name = 'm:f_beard_score',
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_emotions_class:
    {
        default = '',
        field_type = 'string',
        id = 15,
        name = 'm:f_emotions_class',
    },
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_emotions_score:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 16,
        name = 'm:f_emotions_score',
    },
```

```
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_gender_class:
{
    default = '',
    field_type = 'string',
    id = 17,
    name = 'm:f_gender_class',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_gender_score:
{
    default = "10000000000000000000",
    field_type = 'unsigned',
    id = 18,
    name = 'm:f_gender_score',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_glasses_class:
{
    default = '',
    field_type = 'string',
    id = 19,
    name = 'm:f_glasses_class',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_glasses_score:
{
    default = "10000000000000000000",
    field_type = 'unsigned',
    id = 20,
    name = 'm:f_glasses_score',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_liveness_class:
{
    default = '',
    field_type = 'string',
    id = 21,
    name = 'm:f_liveness_class',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_liveness_score:
{
    default = "10000000000000000000",
    field_type = 'unsigned',
    id = 22,
    name = 'm:f_liveness_score',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_race_class:
{
    default = '',
    field_type = 'string',
    id = 23,
    name = 'm:f_race_class',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:f_race_score:
{
    default = "10000000000000000000",
    field_type = 'unsigned',
    id = 24,
    name = 'm:f_race_score',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:thumbnail:
```

```
{
    default = '',
    field_type = 'string',
    id = 25,
    name = 'm:thumbnail',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame:
{
    default = '',
    field_type = 'string',
    id = 26,
    name = 'm:frame',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_bottom:
{
    default = 0,
    field_type = 'unsigned',
    id = 27,
    name = 'm:frame_coords_bottom',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_left:
{
    default = 0,
    field_type = 'unsigned',
    id = 28,
    name = 'm:frame_coords_left',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_right:
{
    default = 0,
    field_type = 'unsigned',
    id = 29,
    name = 'm:frame_coords_right',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:frame_coords_top:
{
    default = 0,
    field_type = 'unsigned',
    id = 30,
    name = 'm:frame_coords_top',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:pk:
{
    default = 0,
    field_type = 'unsigned',
    id = 31,
    name = 'm:pk',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched:
{
    default = 0,
    field_type = 'unsigned',
    id = 32,
    name = 'm:matched',
},
-- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched_dossier:
{
```

```
        default = 0,
        field_type = 'unsigned',
        id = 33,
        name = 'm:matched_dossier',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched_face:
    {
        default = 0,
        field_type = 'unsigned',
        id = 34,
        name = 'm:matched_face',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:matched_lists:
    {
        default = {},
        field_type = 'set[unsigned]',
        id = 35,
        name = 'm:matched_lists',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:normalized_photo:
    {
        default = '',
        field_type = 'string',
        id = 36,
        name = 'm:normalized_photo',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:quality:
    {
        default = "10000000000000000000",
        field_type = 'unsigned',
        id = 37,
        name = 'm:quality',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:scores:
    {
        default = '',
        field_type = 'string',
        id = 38,
        name = 'm:scores',
    },
    -- <class 'ffsecurity.entities_tnt.event.models.ListEvent'>.m:video_source:
    {
        default = 0,
        field_type = 'unsigned',
        id = 39,
        name = 'm:video_source',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:dossier:
    {
        default = 0,
        field_type = 'unsigned',
        id = 40,
        name = 'm:dossier',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:modified_
→date:
    {
```

```
        default = 0,
        field_type = 'unsigned',
        id = 41,
        name = 'm:modified_date',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:source_
↪photo:
    {
        default = '',
        field_type = 'string',
        id = 42,
        name = 'm:source_photo',
    },
    -- <class 'ffsecurity.entities_tnt.dossier_face.models.DossierFace'>.m:source_
↪photo_name:
    {
        default = '',
        field_type = 'string',
        id = 43,
        name = 'm:source_photo_name',
    },
}
-- Fields referenced by multiple models: m:frame_coords_left, m:pk, m:frame_coords_
↪top, m:created_date, m:frame_coords_right, m:thumbnail, m:frame_coords_bottom
```

Structure Modification

To modify the database structure, do the following:

1. Stop the `findface-security` service.

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

2. Create a backup of the Tarantool-based biometric database in any directory of your choice, for example, `/tmp/dump`.

   Tip: See *Backup and Restore Data Storages* for details.

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

3. Prepare the `tnt_schema.lua` file containing the new database structure.

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

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

5. Open the Tarantool configuration file. Make sure that there is a line `dofile("/etc/ffsecurity/tnt_schema.lua")` before the `FindFace.start` section and `meta_scheme=meta_scheme` is defined in the `FindFace.start` parameters.

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

(continued from previous page)

```
dofile("/etc/ffsecurity/tnt_schema.lua")

FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    facen_size=480,
    meta_scheme = meta_scheme
})
```

6. Remove the Tarantool database (default database or shards).

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

sudo rm -f /opt/ntech/var/lib/tarantool/shard-001/{index,snapshots,xlogs}/*
...
sudo rm -f /opt/ntech/var/lib/tarantool/shard-00N/{index,snapshots,xlogs}/*
```

7. Restore the Tarantool database from the backup.

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

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

8. Restart the `findface-security` service.

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

1.8.5 Remove FindFace Security Instance

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

Do the following:

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

2. From this directory, make the script executable.

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

3. Run the script.

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

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

1.8.6 Checking Component Status

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

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

1.8.7 Logs

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

Component	Command to view log
`findface-extraction-api`	sudo tail -f /var/log/syslog	grep extraction-api
`findface-sf-api`	sudo tail -f /var/log/syslog	grep sf-api
`findface-tarantool-server`	sudo tail -f /var/log/tarantool/FindFace.log
`findface-video-manager`	sudo tail -f /var/log/syslog	grep video-manager
`findface-video-worker`	sudo tail -f /var/log/syslog	grep video-worker
`findface-security`	sudo tail -f /var/log/syslog	grep findface-security
`findface-ntls`	sudo tail -f /var/log/syslog	grep ntls
`findface-security`	sudo tail -f /var/log/syslog	grep security
`etcd`	sudo tail -f /var/log/syslog	grep etcd

You can also consult audit log for each component. To do so, use the `journalctl -u <component>` command, for example:

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

Important: In order to enable saving audit logs to your hard drive, uncomment and edit the `Storage` parameter in the `/etc/systemd/journald.conf` file:

```
sudo vi /etc/systemd/journald.conf
...
[Journal]
Storage=persistent
```

If necessary, uncomment and edit the `SystemMaxUse` parameter as well. This parameter determines the maximum volume of log files on your hard drive (10% by default).

```
SystemMaxUse=15
```

To view the FindFace Security audit logs, execute the following command:

```
journalctl -o verbose SYSLOG_IDENTIFIER=ffsecurity
```

When interpreting audit logs, first of all pay attention on the following parameters:

- `REQUEST_USER`: user who made the changes;

- `REQUEST_PATH`: URL of the request;

- `REQUEST_DATA`: detailed information of the request.

In the log below, the `admin` user creates a dossier `id=1879`:

```
Fr 2017-12-22 17:53:32.436258 MSK [s=0b5566699751426983e13241301205e9;i=e26015;
↪b=907c34cc1fde4398af63bb575587d9ba;m=246f620c449;t=560eefaf59bc5;x=ed60a136c8fc6362]
    PRIORITY=6
    _UID=123
    _GID=130
    _CAP_EFFECTIVE=0
    _BOOT_ID=907c34cc1fde4398af63bb575587d9ba
    _MACHINE_ID=a3eea61c03e041ef8e64d5c72f5fce40
    _HOSTNAME=ntechadmin
    SYSLOG_IDENTIFIER=ffsecurity
    THREAD_NAME=MainThread
    _TRANSPORT=journal
    _PID=6579
    _COMM=findface-securi
    _EXE=/opt/ffsecurity/bin/python3
    _CMDLINE=/opt/ffsecurity/bin/python /opt/ffsecurity/bin/findface-security runworker
    _SYSTEMD_CGROUP=/system.slice/system-findface\x2dsecurity\x2dworker.slice/findface-
↪security-worker@4.service
    _SYSTEMD_UNIT=findface-security-worker@4.service
    _SYSTEMD_SLICE=system-findface\x2dsecurity\x2dworker.slice
    CODE_FILE=/opt/ffsecurity/lib/python3.5/site-packages/ffsecurity/mixins.py
    CODE_LINE=94
    CODE_FUNC=finalize_response
    REQUEST_USER=admin
    LOGGER=ffsecurity.audit
    MESSAGE=N8Be05il POST /dossier-faces/ 201 by admin
    REQUEST_DATA={"dossier": "'1879'", "source_photo": "<InMemoryUploadedFile:␣
↪14927016033292449.jpeg (image/jpeg)>"}
    REQUEST_PATH=/dossier-faces/
    REQUEST_ID=N8Be05il
    _SOURCE_REALTIME_TIMESTAMP=1513954412436258
```

In the next log, the list of faces is requested for the dossier `id=1879`:

```
Fr 2017-12-22 17:53:32.475467 MSK [s=0b5566699751426983e13241301205e9;i=e26016;
↪b=907c34cc1fde4398af63bb575587d9ba;m=246f6215d82;t=560eefaf634fe;x=b1374a144a46b5cd]
    PRIORITY=6
    _UID=123
    _GID=130
    _CAP_EFFECTIVE=0
    _BOOT_ID=907c34cc1fde4398af63bb575587d9ba
    _MACHINE_ID=a3eea61c03e041ef8e64d5c72f5fce40
```

(continues on next page)

```
_HOSTNAME=ntechadmin
SYSLOG_IDENTIFIER=ffsecurity
THREAD_NAME=MainThread
_TRANSPORT=journal
_COMM=findface-securi
_EXE=/opt/ffsecurity/bin/python3
_CMDLINE=/opt/ffsecurity/bin/python /opt/ffsecurity/bin/findface-security runworker
_SYSTEMD_SLICE=system-findface\x2dsecurity\x2dworker.slice
_PID=6588
_SYSTEMD_CGROUP=/system.slice/system-findface\x2dsecurity\x2dworker.slice/findface-
→security-worker@2.service
_SYSTEMD_UNIT=findface-security-worker@2.service
CODE_FILE=/opt/ffsecurity/lib/python3.5/site-packages/ffsecurity/mixins.py
CODE_LINE=94
CODE_FUNC=finalize_response
REQUEST_USER=admin
REQUEST_DATA={}
LOGGER=ffsecurity.audit
MESSAGE=Dee7Qvy4 GET /dossier-faces/?dossier=1879&limit=1000 200 by admin
REQUEST_ID=Dee7Qvy4
REQUEST_PATH=/dossier-faces/?dossier=1879&limit=1000
_SOURCE_REALTIME_TIMESTAMP=1513954412475467
```

1.8.8 Licensing

View and Update License

To view your current licensing information or upload a new license file, navigate to *Preferences -> License*.
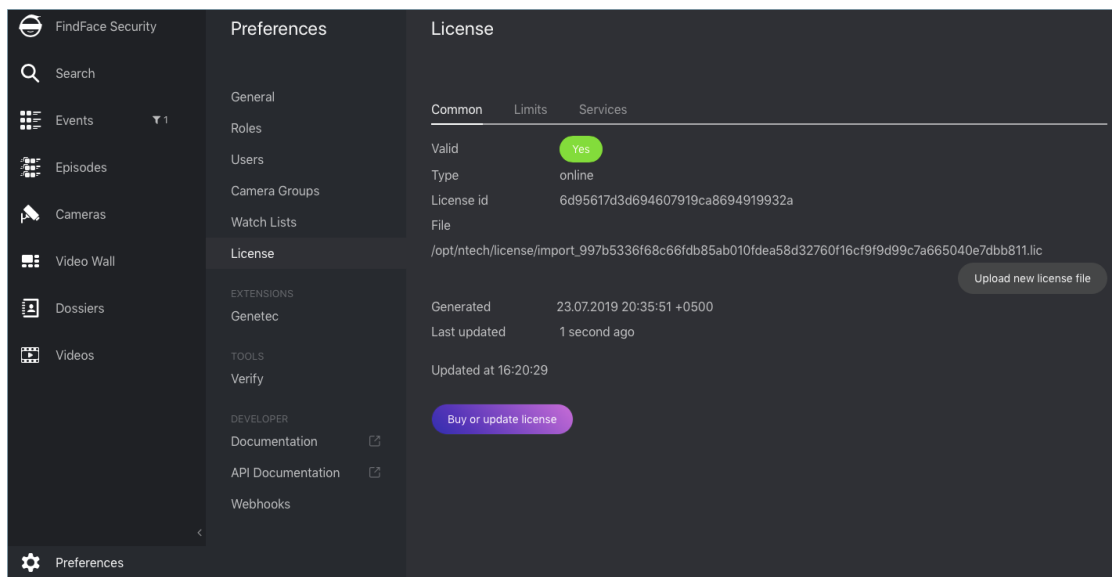
Troubleshoot Licensing and `findface-ntls`

When troubleshooting licensing and `findface-ntls` (see *Licensing Principles*), 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.

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

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

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

Interpret the `last_updated` value as follows:

- [0, 5] — everything is alright.

- (5, 30] — there may be some problems with connection, or with the local drive where the license file is stored.

- (30; 120] — almost certainly something bad happened.

- (120; ∞) — the licensing source response has been timed out. Take action.

- `"valid": false`: connection with the licensing source was never established.

```
curl http://localhost:3185/license.json -s | jq
{
  "name": "NTLS",
  "time": 1565186356,
  "type": "online",
  "license_id": "61063ce4b86945e1b70c3bdbedea453b",
  "generated": 1514467939,
  "last_updated": 5,
  "valid": {
    "value": true,
    "description": ""
  },
  "source": "/opt/ntech/license/import_
→b68d7b7ec9a7310d18832035318cff0c9ddf11e3a9ab0ae962fbe48645e196d1.lic",
  "limits": [
    {
      "type": "time",
      "name": "end",
      "value": 1609161621
    },
    {
      "type": "number",
      "name": "faces",
      "value": 9007199254740991,
      "current": 0
    },
    {
      "type": "number",
      "name": "cameras",
      "value": 4294967295,
```

```
    "current": 0
  },
  {
    "type": "number",
    "name": "extraction_api",
    "value": 256,
    "current": 0
  },
  {
    "type": "boolean",
    "name": "gender",
    "value": true
  },
  {
    "type": "boolean",
    "name": "age",
    "value": true
  },
  {
    "type": "boolean",
    "name": "emotions",
    "value": true
  },
  {
    "type": "boolean",
    "name": "fast-index",
    "value": true
  },
  {
    "type": "boolean",
    "name": "sec-genetec",
    "value": false
  },
  {
    "type": "boolean",
    "name": "countries",
    "value": false
  },
  {
    "type": "boolean",
    "name": "beard",
    "value": false
  },
  {
    "type": "boolean",
    "name": "race",
    "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"
    }
  ]
}
```

1.8.9 Automatic Tarantool Recovery

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

To enable automatic database recovery, do the following:

1. Open the Tarantool configuration file.

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

2. Uncomment `force_recovery = true`.

```
box.cfg{

    force_recovery = true,

}
```

1.8.10 Manually Purge Old Events and Episodes from Database

Tip: To schedule automatic events and episodes cleanup, see *Automatic Events Cleanup*.

To manually remove old events and related episodes from the FindFace Security database, use the `cleanup_events` utility.

To invoke the `cleanup_events` help message, execute:

```
sudo findface-security cleanup_events --help
usage: findface-security cleanup_events [-h] [--version] [-v {0,1,2,3}]
                                        [--settings SETTINGS]
                                        [--pythonpath PYTHONPATH]
                                        [--traceback] [--no-color]
                                        --age AGE
```

```
Delete old events

optional arguments:
 -h, --help            show this help message and exit
 --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.
 --age AGE             Minimum age in days of events to clean up
```

In order to remove events and episodes older than a given number of days, use the `--age` option. For example, to remove events and episodes older than 5 days, execute:

```
sudo findface-security cleanup_events --age 5
```

1.9 Appendices

1.9.1 Enable Data Encryption

To ensure data security, it is recommended to enable SSL encryption. Do the following:

Important: We do not recommend to use a self-signed certificate.

1. Under the nginx configuration directory, create a directory that will be used to hold all of the SSL data:

```
sudo mkdir /etc/nginx/ssl
```

2. Create the SSL key and certificate files:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/
→my-example-domain.com.key -out /etc/nginx/ssl/my-example-domain.com.crt
```

You will be asked a few questions about your server in order to embed the information correctly in the certificate. Fill out the prompts appropriately. The most important line is the one that requests the `Common Name`. You need to enter the domain name or public IP address that you want to be associated with your server. Both of the files you created (`my-example-domain.com.key` and `my-example-domain.com.crt`) will be placed in the `/etc/nginx/ssl` directory.

3. Configure nginx to use SSL. Open the nginx configuration file. Copy the code from the example below into the file.

```
sudo vi /etc/nginx/nginx.conf

upstream ffsecurity {
        server 127.0.0.1:8002;
}

# redirect from http to https version of the site
server {
        listen 80;
        server_name domain.ru www.domain.ru;
        rewrite ^(.*) https://domain.ru$1 permanent;
        access_log off;
}

server {
        listen 443 ssl;

        ssl_certificate     /etc/nginx/ssl/domain.crt;
        ssl_certificate_key /etc/nginx/ssl/domain.key;

        root /var/lib/ffsecurity;

        autoindex off;

        server_name domain.ru;

        location @ffsec {
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-For $remote_addr;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_pass http://ffsecurity;
        }

         location /static/ {
        }
        location /uploads/ {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GET';
                add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-
→Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range,Authorization
→';
                add_header 'Access-Control-Expose-Headers' 'Content-Length,
→Content-Range';
                add_header 'Access-Control-Max-Age' 2592000;
        }
        location /ui-static/ {
                alias /usr/share/ffsecurity-ui/ui-static/;
        }
        location /doc/ {
                alias /opt/ffsecurity/doc/;
        }
        location / {
                try_files $uri $uri/ @ffsec;
                client_max_body_size 100m;
```

(continues on next page)

```
                alias /usr/share/ffsecurity-ui/;

        }
}
```

4. Restart nginx.

```
sudo service nginx restart
```

5. Edit the `findface-security` configuration file. In the `EXTERNAL_ADDRESS` and `ROUTER_URL` parameters, substitute the `http://` prefix with `https://`.

```
sudo vi /etc/ffsecurity/config.py

...
EXTERNAL_ADDRESS="https://my-example-domain.com"
...
ROUTER_URL="https://IP_address"
```

6. If there are running `findface-video-worker` services in the system, you need to either recreate cameras in the web interface, or change the `router_url` parameter in relevant video processing jobs, substituting the `http://` prefix with `https://`. This can be done with the following command:

```
curl -s localhost:18810/jobs | jq -r '.[]["id"]' | xargs -I {} curl -X PATCH -d '{
↪"router_url": "https://domain.ru/video-detector/frame"}' http://localhost:18810/
↪job/{}
```

1.9.2 Components in Depth

findface-extraction-api

The `findface-extraction-api` service uses neural networks to detect a face in an image, extract face biometric data (feature vector), and recognize gender, age, emotions, and other features.

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

- Gets original images with faces and normalized face images.

- Returns the coordinates of the face bounding box, and (optionally) feature vector, gender, age and emotions data, should these data be requested by `findface-sf-api`.

Functionality:

- face detection in an original image (with return of the bbox coordinates),

- face normalization,

- feature vector extraction from a normalized image,

- face feature recognition (gender, age, emotions, beard, glasses3, etc.).

The `findface-extraction-api` service can be based on CPU (installed from the `findface-extraction-api` package) or GPU (installed from the `findface-extraction-api-gpu` package). For both CPU- and GPU-accelerated services, configuration is done through the `/etc/findface-extraction-api.ini` configuration file. Its content varies subject to the acceleration type.

CPU-service configuration file:

```
allow_cors: false
detector_instances: 0
dlib:
  model: /usr/share/findface-data/normalizer.dat
  options:
    adjust_threshold: 0
    upsample_times: 1
extractors:
  instances: 1
  max_batch_size: 16
  models:
    age: ""
    beard: ""
    emotions: ""
    face: face/grapefruit_480.cpu.fnk
    gender: ""
    glasses3: ""
    liveness: ""
  models_root: /usr/share/findface-data/models
fetch:
  enabled: true
  size_limit: 10485760
license_ntls_server: 127.0.0.1:3133
listen: 127.0.0.1:18666
max_dimension: 6000
nnd:
  model: /usr/share/nnd/nnd.dat
  options:
    max_face_size: .inf
    min_face_size: 30
    o_net_thresh: 0.9
    p_net_max_results: 0
    p_net_thresh: 0.5
    r_net_thresh: 0.5
    scale_factor: 0.79
  quality_estimator: true
  quality_estimator_model: /usr/share/nnd/quality_estimator_v2.dat
ticker_interval: 5000
```

GPU-service configuration file:

```
listen: 127.0.0.1:18666
dlib:
  model: /usr/share/findface-data/normalizer.dat
  options:
    adjust_threshold: 0
    upsample_times: 1
nnd:
  model: /usr/share/nnd/nnd.dat
  quality_estimator: true
  quality_estimator_model: /usr/share/nnd/quality_estimator_v2.dat
  options:
    min_face_size: 30
    max_face_size: .inf
```

```
    scale_factor: 0.7900000214576721
    p_net_thresh: 0.5
    r_net_thresh: 0.5
    o_net_thresh: 0.8999999761581421
    p_net_max_results: 0
detector_instances: 0
extractors:
  models_root: /usr/share/findface-data/models
  max_batch_size: 3
  instances: 2
  models:
    age: ""
    beard: ""
    emotions: ""
    face: face/grapefruit_480.gpu.fnk
    gender: ""
    glasses3: ""
    liveness: ""
  cache_dir:
  gpu_device: 0
license_ntls_server: 172.17.46.26:3133
fetch:
  enabled: true
  size_limit: 10485760
max_dimension: 6000
allow_cors: false
ticker_interval: 5000
prometheus:
  timing_buckets:
  - 0.001
  - 0.005
  - 0.01
  - 0.02
  - 0.03
  - 0.05
  - 0.1
  - 0.2
  - 0.3
  - 0.5
  - 0.75
  - 0.9
  - 1
  - 1.1
  - 1.3
  - 1.5
  - 1.7
  - 2
  - 3
  - 5
  - 10
  - 20
  - 30
  - 50
  resolution_buckets:
  - 10000
  - 20000
  - 40000
```

```
  - 80000
  - 100000
  - 200000
  - 400000
  - 800000
  - 1e+06
  - 2e+06
  - 3e+06
  - 4e+06
  - 5e+06
  - 6e+06
  - 8e+06
  - 1e+07
  - 1.2e+07
  - 1.5e+07
  - 1.8e+07
  - 2e+07
  - 3e+07
  - 5e+07
  - 1e+08
  faces_buckets:
  - 0
  - 1
  - 2
  - 5
  - 10
  - 20
  - 50
  - 75
  - 100
  - 200
  - 300
  - 400
  - 500
  - 600
  - 700
  - 800
  - 900
  - 1000
```

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

Parameter	Description
nnd -> quality	Enables face quality estimation. In this case, `findface-extraction-api` returns a face quality score in the `detection_score` field. Interpret the quality score further in analytics. Upright faces in frontal position are considered the best quality. They result in values around 0, mostly negative (such as -0.00067401276, for example). Inverted faces and large face angles are estimated with negative values some -5 and less.
nnd -> min_face_size	The minimum size of a face (bbox) guaranteed to be detected. The larger the value, the less resources required for face detection.
nnd -> max_face_size	The minimum size of a face (bbox) guaranteed to be detected.
license_lst_lsense	The license server IP address and port.
gpu_device	(Only for GPU) The number of the GPU device used by `findface-extraction-api-gpu`.

You will also have to enable recognition models for face features such as gender, age, emotions, glasses3, and/or beard, subject to your needs. 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.

```
models:
  age: faceattr/age.v1.cpu.fnk
  emotions: faceattr/emotions.v1.cpu.fnk
  face: face/grapefruit_480.cpu.fnk
  gender: faceattr/gender.v2.cpu.fnk
  beard: faceattr/beard.v0.cpu.fnk
  glasses3: faceattr/glasses3.v0.cpu.fnk
```

The following models are available:

Face fea-ture	Accelera-tion	Configuration file parameter
face (biome-try)	CPU	`face:    face/grapefruit_480.cpu.fnk          face:    face/grapefruit_160.cpu.fnk`
	GPU	`face:    face/grapefruit_480.gpu.fnk          face:    face/grapefruit_160.gpu.fnk`
age	CPU	`age:    faceattr/age.v1.cpu.fnk`
	GPU	`age:    faceattr/age.v1.gpu.fnk`
gender	CPU	`gender:    faceattr/gender.v2.cpu.fnk`
	GPU	`gender:    faceattr/gender.v2.gpu.fnk`
emotions	CPU	`emotions:    faceattr/emotions.v1.cpu.fnk`
	GPU	`emotions:    faceattr/emotions.v1.gpu.fnk`
glasses3	CPU	`glasses3:    faceattr/glasses3.v0.cpu.fnk`
	GPU	`glasses3:    faceattr/glasses3.v0.gpu.fnk`
beard	CPU	`beard:    faceattr/beard.v0.cpu.fnk`
	GPU	`beard:    faceattr/beard.v0.gpu.fnk`

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

```
models:
  gender: ""
  age: ""
  emotions: ""
```

findface-sf-api

The `findface-sf-api` service implements HTTP API for the FindFace core main functionality such as face detection and face recognition (the mentioned functions themselves are provided by *findface-extraction-api'*). It interfaces with the biometric database powered by Tarantool via the `findface-tarantool-server` service, as well as with `findface-extraction-api` (provides face detection and face recognition) and `findface-upload` (provides a storage for original images and FindFace core artifacts).

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

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

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

Functionality:

- HTTP API implementation (face detection and face recognition methods, performed via `findface-extraction-api`).

- saving face data to the biometric database (performed via `findface-tarantool-server`),

- saving original images, face thumbnails and normalized face images to an NginX-powered web server (via `findface-upload`).

- provides interaction between all the FindFace core components.

The `findface-sf-api` configuration is done through the `/etc/findface-sf-api.ini` configuration file.

```
listen: 127.0.0.1:18411
extraction-api:
  timeouts:
    connect: 5s
    response_header: 30s
    overall: 35s
    idle_connection: 10s
  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
  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: ''
limits:
  url-length: 4096
  deny-networks: 127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8
  body-image-length: 33554432
  allow-return-facen: false
```

```
cache:
  type: memcache
  inmemory:
    size: 16384
  memcache:
    nodes:
    - 127.0.0.1:11211
    timeout: 100ms
  redis:
    network: tcp
    addr: localhost:6379
    password: ''
    db: 0
    timeout: 5s
normalized-storage:
  type: webdav
  enabled: true
  webdav:
    upload-url: http://127.0.0.1:3333/uploads/
    timeouts:
      connect: 5s
      response_header: 30s
      overall: 35s
      idle_connection: 10s
  s3:
    endpoint: ''
    bucket-name: ''
    access-key: ''
    secret-access-key: ''
    secure: true
    region: ''
    public-url: ''
    operation-timeout: 30
```

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

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

findface-tarantool-server

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

Tip: See Tarantool official documentation for details.

- From `findface-sf-api`, `findface-tarantool-server` receives data, such as information of detected in video faces, to write into the biometric database.

- By request from `findface-sf-api`, `findface-tarantool-server` performs database searches and returns search results.

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

Functionality:

- saving face data to the biometric database,

- database search,

- implementation of direct API requests to the database (see *Direct API Requests to Tarantool*).

The `findface-tarantool-server` configuration is done through the `/etc/tarantool/instances.enabled/<shard-*>.lua` configuration file. In a cluster environment, configuration has to be done for each shard.

```
--
-- Please, read the tarantool cfg doc:
-- https://tarantool.org/doc/reference/configuration/index.html#box-cfg-params
--

box.cfg{
    --port to listen, direct tarantool access
    --Only need for admin operations
    --THIS IS NOT PORT YOU NEED FOR facenapi/sf-api
    listen = '127.0.0.1:33001',

    --Directory to store data
    vinyl_dir = '/opt/ntech/var/lib/tarantool/shard-001',
    work_dir = '/opt/ntech/var/lib/tarantool/shard-001',
    memtx_dir = '/opt/ntech/var/lib/tarantool/shard-001/snapshots',
    wal_dir = '/opt/ntech/var/lib/tarantool/shard-001/xlogs',

    --Maximum mem usage in bytes
    memtx_memory = 200 * 1024 * 1024,

    checkpoint_interval = 3600*4,
    checkpoint_count = 3,

    --uncomment only if you know what you are doing!!! and don't forget box.snapshot()
    -- wal_mode = 'none',

    --if true, tarantool tries to continue if there is an error while reading a␣
→snapshot/xlog files:  skips invalid records, reads as much data as possible and re-
→builds the file
    -- force_recovery = true,
}

pcall(function() box.schema.user.grant('guest', 'execute,read,write', 'universe') end)

dofile("/etc/ffsecurity/tnt_schema.lua")
```

(continues on next page)

```
-- host,port to bind for http server
-- this is what you need for facenapi
FindFace = require("FindFace")
FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    facen_size=480,
    meta_scheme = meta_scheme
})
```

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

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

Default database structure is passed from `/etc/ffsecurity/tnt_schema.lua` to the `meta_scheme` parameter.

findface-upload

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

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

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

Video face detection: findface-video-manager and findface-video-worker

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

In this section:

- *Functions of* `findface-video-manager`
- *Functions of* `findface-video-worker`
- *Configure Video Face Detection*
- *Jobs*

Functions of `findface-video-manager`

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

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

- It supplies `findface-video-worker` with settings and the list of to-be-processed video streams. To do so, it issues a so-called *job*, a video processing task that contains configuration settings and stream data.

- In a distributed system, it distributes video streams (jobs) across vacant `findface-video-worker` instances.

Note: Configuration settings passed via jobs have priority over the `findface-video-manager` configuration file.

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

Functionality:

- allows for configuring video face detection parameters,

- allows for managing the list of to-be-processed video streams,

- implements video face detection management.

Functions of `findface-video-worker`

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

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

- By request, `findface-video-worker` gets a job with settings and the list of to-be-processed video streams from `findface-video-manager`.

- The `findface-video-worker` posts extracted normalized face images, along with the full frames and meta data (such as bbox, camera ID and detection time) to the `findface-facerouter` service for further processing.

Note: In FindFace Security, the `findface-facerouter` functions are performs by `findface-security`.

Functionality:

- detects faces in video,

- extracts normalized face images,

- searches for the best face snapshot,

- snapshot deduplication (only one snapshot per face detection event).

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

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

- **`Face tracking`**. The face tracker tracks, detects and captures faces in the video. It can simultaneously be working with several faces. It also searches for the best face snapshot, using an embedded neural network. After the best face snapshot is found, it is posted to `findface-facerouter`.

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

- Real-time

- Offline

Real-Time Mode

In the real-time mode, `findface-video-worker` posts a face immediately after it appears in the camera field of view.

- If `rt-perm=True`, the face tracker searches for the best face snapshot within each time period equal to `rt-delay` and posts it to `findface-facerouter`.

- If `rt-perm=False`, the face tracker searches for the best face snapshot dynamically:

  1. First, the face tracker estimates whether the quality of a face snapshot exceeds a pre-defined threshold value. If so, the snapshot is posted to `findface-facerouter`.

  2. The threshold value increases after each post. Each time the face tracker gets a higher quality snapshot of the same face, it is posted.

  3. When the face disappears from the camera field of view, the threshold value resets to default.

By default, the real-time mode is disabled (`realtime=false` in the `/etc/findface-video-manager.conf` file).

Offline Mode

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

By default, the offline mode is enabled (`overall=true` in the `/etc/findface-video-manager.conf` file).

Configure Video Face Detection

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

1. The `findface-video-manager` configuration file `/etc/findface-video-manager.conf`:

```
listen: 127.0.0.1:18810
etcd:
  endpoints: 127.0.0.1:2379
  dial_timeout: 3s
kafka:
   enabled: false
   endpoints: 127.0.0.1:9092
master:
  lease_ttl: 10
  self_url: 127.0.0.1:18811
  self_url_http: 127.0.0.1:18811
rpc:
  listen: 127.0.0.1:18811
  heart_beat_timeout: 4s
router_url: http://127.0.0.1:18820/v0/frame
exp_backoff:
  enabled: false
  min_delay: 1s
  max_delay: 1m0s
  factor: 2
  flush_interval: 2m0s
ntls:
  enabled: false
  url: http://127.0.0.1:3185/
  update_interval: 1m0s
prometheus:
  jobs_processed_duration_buckets:
  - 1
  - 30
  - 60
  - 500
  - 1800
  - 3600
  - 21600
  - .inf
job_scheduler_script: ''
stream_settings:
  ffmpeg_params: []
  md_threshold: 0.002
  md_scale: 0.3
  fd_frame_height: -1
  uc_max_time_diff: 30
  uc_max_dup: 3
  uc_max_avg_shift: 10
  det_period: 8
  realtime: false
  npersons: 4
  disable_drops: false
  tracker_threads: 4
  parse_sei: false
  image_arg: photo
  additional_headers: []
  additional_body: []
  api_timeout: 15000
  api_ssl_verify: true
  post_uniq: true
  min_score: -2
```

(continues on next page)

```
  min_d_score: -1000
  realtime_dly: 500
  realtime_post_perm: false
  rot: ''
  roi: ''
  draw_track: false
  send_track: 0
  min_face_size: 0
  max_face_size: 0
  overall: true
  only_norm: false
  max_candidates: 0
  jpeg_quality: 95
  ffmpeg_format: ''
stream_settings_gpu:
  play_speed: -1
  filter_min_quality: -2
  filter_min_face_size: 1
  filter_max_face_size: 8192
  normalized_only: false
  jpeg_quality: 95
  overall_only: true
  use_stream_timestamp: false
  ffmpeg_params: []
  router_timeout_ms: 15000
  router_verify_ssl: true
  router_headers: []
  router_body: []
  start_stream_timestamp: 0
  imotion_threshold: 0
  rot: ''
  roi: ''
  realtime_post_interval: 1
  realtime_post_every_interval: false
  ffmpeg_format: ''
  disable_drops: false
```

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

Option	Description
`router_url`	IP address and port of the `findface-facerouter` host to receive detected faces from `findface-video-worker`. In FindFace Security, `findface-facerouter` functions are performed by `findface-security`. Default value: `http://127.0.0.1:18820/v0/frame`.
`etcd -> endpoints`	IP address and port of the `etcd` service. Default value: `127.0.0.1:2379`.
`ntls -> enabled`	If true, `findface-video-manager` will send a job to `findface-video-worker` only if the total number of processed cameras does not exceed the allowed number of cameras from the license. Default value: false.
`ntls -> url`	IP address and port of the `findface-ntls` host. Default value: `http://127.0.0.1:3185/`.

You can also configure the following parameters:

Note: In the `stream_settings(-gpu)` section of the file, you will find settings common to all video streams. Settings of a particular stream, passed in a job, have priority over those in the configuration file (see *Jobs*).

CPU-option	GPU-option	Description
`additional_body`	`router_body`	Additional body fields in a request body when posting a face: ["key = value"]. Default value: body fields not specified.
`additional_headers`	`router_headers`	Additional header fields in a request when posting a face: ["key = value"]. Default value: headers not specified.
`api_ssl_verify`	`router_verify`	Enables ssl a https certificate verification when `findface-video-worker` and `findface-facerouter` (or `findface-security` in the standard FindFace Security configuration) interact over https. Default value: true. If false, a self-signed certificate can be accepted.
`api_timeout`	`router_timeout`	Timeout for a `findface-facerouter` (or `findface-security` in the standard FindFace Security configuration) response to a `findface-video-worker` API request, in milliseconds. If the timeout has expired, the system will log an error. Default value: 15000.
`disable_drops`	`disable_drops`	Enables posting all appropriate faces without drops. By default, if `findface-video-worker` does not have enough resources to process all frames with faces, it drops some of them. If this option is active, `findface-video-worker` puts odd frames on the waiting list to process them later. Default value: false.
`draw_track`	N/a	Enables drawing a face motion track in a bbox. Default value: false.
`fd_frame_height`	N/a	Video frame height for the face tracker, in pixels. Scale down in the case of high resolution cameras, or close up faces, or if the CPU load is too high, to reduce the system resources consumption. Make sure that the scaled face size exceeds the `min-face-size` value. Default value: -1 (negative values corresponds to the original size). Optimal value to reduce load: 640-720.
`ffmpeg_format`	`ffmpeg_format`	Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.
`ffmpeg_params`	`ffmpeg_params`	List of a video stream ffmpeg options with their values as a key=value array: ["rtsp_transpotr=tcp", .., "ss=00:20:00"]. Check out the FFmpeg web site for the full list of options. Default value: options not specified.
`image_arg`	N/a	Name of the argument containing a bbox with a face, in an API request. Default value: `photo`.
`jpeg_quality`	`jpeg_quality`	Quality of an original frame JPEG compression, in percents. Default value: 95%.
`max_face_size`	`filter_max_size`	Maximum size of a face in pixels. Oversized faces are not posted. Default value: 0 (filter disabled).
`md_scale`	N/a	Video frame scaling coefficient for the motion detector, relative to the original size from 0 to 1. Scale down in the case of high resolution cameras, or close up faces, or if the CPU load is too high, to reduce the system resources consumption. Make sure that the scaled face size exceeds the `min-face-size` value. Default value: 1 (original size).
`md_threshold`	`motion_threshold`	Minimum motion intensity to be detected by the motion detector. The threshold value is to be fitted empirically. Empirical units: zero and positive rational numbers. Milestones: 0 = detector disabled, 0.002 = default value, 0.05 = minimum intensity is too high to detect motion.

Continued on next page

Table 1 – continued from previous page

CPU-option	GPU-option	Description
min_score	filter_min_quality	Minimum threshold value for a face image quality. A face is posted if it has better quality. The threshold value is to be fitted empirically. Empirical units: negative rational numbers to zero. Milestones: 0 = high quality faces, -1 = good quality, -2 = satisfactory quality, -5 = face recognition maybe inefficient. Default value: -2.
min_face_size	filter_min_size	Minimum size of a face in pixels. Undersized faces are not posted. Default value: 0 (filter disabled).
min_d_score	N/a	Maximum deviation of a face from its frontal position. A face is posted if its deviation is less than this value. The deviation is to be fitted empirically. Empirical units: negative rational numbers to zero. Milestones: -3.5 = large face angles, face recognition may be inefficient, -2.5 = satisfactory deviation, -0.05 = close to the frontal position, 0 = frontal face. Default value: -1000.
npersons	N/a	Maximum number of faces simultaneously tracked by the face tracker. This parameter severely affects performance. Default value: 4.
only_norm	normalized	Enable posting only normalized face images without full frames. Default value: false.
overall	overall_only	Enables the offline mode for the best face search. Default value: true.
N/a	play_speed	If less than zero, the speed is not limited. In other cases, the stream is read with the given `play_speed`. Not applicable for live streams.
post_uniq	N/a	Enables face deduplication, i.e. posting only a certain number of faces belonging to one person, during a certain period of time. In this case, if `findface-video-worker` posts a face to `findface-facerouter` and then tracks another one within the time period `uc_max_time_diff`, and the distance between the two faces doesn't exceed `uc_max_avg_shift`, `findface-video-worker` estimates their similarity. If the faces are similar and the total number of similar faces during the `uc_max_time_diff` period does not exceed the number `uc_max_dup`, `findface-video-worker` posts the other face. Otherwise, the other face is not posted. Default value: true.
realtime	N/a	Enables the real-time mode for the best face search. Default value: false.
realtime_dly	realtime_post_first	Only for the real-time mode. If `realtime_post_perm=True`, defines the time period in milliseconds within which the face tracker picks up the best snapshot and posts it to `findface-facerouter`. If `realtime_post_perm=False`, defines the minimum time period between 2 posts of the same face with increased quality. Default value: 500.
realtime_post_perm	realtime_post_every_interval	Only for the real-time mode. Post best snapshots obtained within each `realtime_dly` time period. If false, search for the best snapshot dynamically and send snapshots in order of increasing quality. Default value: false.
roi	roi	Enable posting faces detected only inside a region of interest WxH+X+Y. Default value: region not specified.
rot	rot	Enables detecting and tracking faces only inside a clipping rectangle WxH+X+Y. You can use this option to reduce `findface-video-worker` load. Default value: rectangle not specified.
send_track	N/a	Enables posting a face motion track as array of the bbox center coordinates. As the `send_track` value, specify the number of dots in the motion track. Default value: 0 (array not posted).

Continued on next page

Table 1 – continued from previous page

CPU-option	GPU-option	Description
N/a	start_stream_timestamp	Add the specified number of seconds to timestamps from a stream.
tracker_threads	N/a	Number of tracking threads for the face tracker. This value should be less or equal to the npersons value. We recommend you to set them equal. If the number of tracking threads is less than the maximum number of tracked faces, resource consumption is reduced but so is the tracking speed. Default value: 1.
uc_max_time_diff	N/a	Only if post_uniq: true (face deduplication enabled). Maximum time period in seconds during which a number of similar faces are considered as belonging to one person. Default value: 30.
uc_max_dup	N/a	Only if post_uniq: true (face deduplication enabled). Maximum number of faces during the uc_max_time_diff period that is posted for a person. Default value: 3.
uc_max_avg_shift	N/a	Only if post_uniq: true (face deduplication enabled). Distance in pixels within which a number of similar faces are considered as belonging to one person. Default value: 10.
N/a	use_stream_timestamp	If true, retrieve and post timestamps from a video stream. If false, post the actual date and time.

1. If you opt for the CPU-accelerated package `findface-video-worker-cpu`, use the `/etc/findface-video-worker-cpu.ini` configuration file:

```
## read streams from file, do not use VideoManager
input =

## exit on first finished job, only when --input specified
exit_on_first_finished = false

## batch size
batch_size = 4

## http server port for metrics, 0=do not start server
metrics_port = 0

## resize scale, 1=do not resize
resize_scale = 1.000000

## maximum number of streams
capacity = 10

## command to obtain videomanager's grpc ip:port
mgr_cmd =

## videomanager grpc ip:port
mgr_static = 127.0.0.1:18811

## ntls server ip:port
ntls_addr = 127.0.0.1:3133

## debug: save faces to dir
save_dir =

## minimum face size
min_face_size = 60
```

```
## preinit detector for specified resolutions: "640x480;1920x1080"
resolutions =

## worker labels: "k=v;group=enter"
labels =

## use timestamps from SEI packet
use_time_from_sei = false

#-----------------------------
[streamer]
#-----------------------------
## streamer server port, 0=disabled
port = 18999

## streamer url - how to access this worker on streamer_port
url = ws://127.0.0.1:18999/stream/

#-----------------------------
[liveness]
#-----------------------------
## path to liveness fnk
fnk =

## liveness threshold
threshold = 0.945000

## liveness internal algo param
interval = 1.000000

## liveness internal algo param
stdev_cnt = 1

#-----------------------------
[send]
#-----------------------------
## posting faces threads
threads = 8

## posting faces maximum queue size
queue_limit = 256

#-----------------------------
[tracker]
#-----------------------------
## max face miss duration, sec
miss_interval = 1.000000

## overlap threshold
overlap_threshold = 0.250000

#-----------------------------
[models]
#-----------------------------
## path to detector fnk
detector = /usr/share/findface-data/models/facedet/mtcnn.cpu.fnk
```

```
## path to quality fnk
quality = /usr/share/findface-data/models/faceattr/quality.v0.cpu.fnk

## path to norm for quality fnk
norm_quality = /usr/share/findface-data/models/facenorm/ant.v2.cpu.fnk

## path to norm200 fnk, for face send
norm_200 = /usr/share/findface-data/models/facenorm/ant.v2.cpu.fnk

## path to norm_crop2x fnk, for face send
norm_crop2x = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.cpu.
↪fnk
```

If you opt for the GPU-accelerated package `findface-video-worker-gpu`, use the `/etc/findface-video-worker-gpu.ini` configuration file.

```
## cuda device number
device_number = 0

## old gpu detector models directory
models_dir = /usr/share/findface-gpudetector/models

## read streams from file, do not use VideoManager
input =

## exit on first finished job, only when --input specified
exit_on_first_finished = false

## batch size
batch_size = 8

## http server port for metrics, 0=do not start server
metrics_port =

## resize scale, 1=do not resize
resize_scale = 1.000000

## maximum number of streams
capacity = 30

## command to obtain videomanager's grpc ip:port
mgr_cmd =

## videomanager grpc ip:port
mgr_static = 127.0.0.1:18811

## ntls server ip:port
ntls_addr = 127.0.0.1:3133

## debug: save faces to dir
save_dir =

## minimum face size
min_face_size = 60
```

```
## preinit detector for specified resolutions: "640x480;1920x1080"
resolutions =

## worker labels: "k=v;group=enter"
labels =

## use timestamps from SEI packet
use_time_from_sei = false

#-----------------------------
[streamer]
#-----------------------------
## streamer server port, 0=disabled
port = 18999

## streamer url - how to access this worker on streamer_port
url = ws://172.17.46.17:18999/stream/

#-----------------------------
[liveness]
#-----------------------------
## path to liveness fnk
fnk =

## liveness threshold
threshold = 0.945000

## liveness internal algo param
interval = 1.000000

## liveness internal algo param
stdev_cnt = 1

#-----------------------------
[send]
#-----------------------------
## posting faces threads
threads = 8

## posting faces maximum queue size
queue_limit = 256

#-----------------------------
[tracker]
#-----------------------------
## max face miss duration, sec
miss_interval = 1.000000

## overlap threshold
overlap_threshold = 0.250000

#-----------------------------
[models]
#-----------------------------
## path to detector fnk
detector =
```

```
## path to quality fnk
quality =

## path to norm for quality fnk
norm_quality =

## path to norm200 fnk, for face send
norm_200 = /usr/share/findface-data/models/facenorm/ant.v2.gpu.fnk

## path to norm_crop2x fnk, for face send
norm_crop2x = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.gpu.
↪fnk

## path to cache directory
cache_dir =


#------------------------------
[video_decoder]
#------------------------------
## decode video on cpu
cpu = false
```

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

CPU	GPU	Description
ntls-addr		IP address and port of the `findface-ntls` host.
mgr-static		IP address of the `findface-video-manager` host to provide `findface-video-worker` with settings and the list of to-be-processed streams.
capacity		Maximum number of video streams to be processed by `findface-video-worker`.
mgr-exec		(Optional, instead of the `mgr-static` parameter) A script to describe dynamic IP address of the `findface-video-manager` host.
labels		Labels used to allocate a video face detector instance to a certain group of cameras. See *Allocate findface-video-worker to Camera Group*.
N/a	fnk	Path to the face *liveness* detector.
input		Process streams from file, ignoring stream data from `findface-video-manager`.
exit_on_first_finish		(Only if input is specified) Exit on the first finished job.
resize_scale		Rescale video frames with the given coefficient.
save_dir		(For debug) Save detected faces to the given directory.
min_face_size		Minimum face size to be detected.
resolutions		Preinitialize the `findface-video-worker` for specific resolutions to speed up its performance.
N/a	device_number	GPU device number to use.
N/a	models_dir	Old directory with GPU detector models. Otherwise, use the `[models]` section.
N/a	cpu	If necessary, decode video on CPU.

Jobs

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

The content of a typical job is shown in the example below.

```
curl http://127.0.0.1:18810/job/1 | jq
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                Dload  Upload   Total   Spent    Left  Speed
 100  1771  100  1771    0      0    447k       0 --:--:-- --:--:-- --:--:--  576k
{
  "id": "1",
  "enabled": true,
  "stream_url": "rtmp://restreamer.int.ntl/cams/openspace",
  "labels": {},
  "router_url": "http://172.17.46.13/video-detector/frame",
  "single_pass": false,
  "stream_settings": {
    "ffmpeg_params": [],
    "md_threshold": 0.002,
    "md_scale": 0.3,
    "fd_frame_height": -1,
    "uc_max_time_diff": 30,
    "uc_max_dup": 3,
    "uc_max_avg_shift": 10,
    "det_period": 8,
    "realtime": false,
    "npersons": 4,
    "disable_drops": false,
    "tracker_threads": 4,
    "parse_sei": false,
    "image_arg": "photo",
    "additional_headers": [
      "Authorization=Token b612396adc3a6dd71b82b5fe333a0a30"
    ],
    "additional_body": [],
    "api_timeout": 15000,
    "api_ssl_verify": true,
    "post_uniq": true,
    "min_score": -2,
    "min_d_score": -1000,
    "realtime_dly": 500,
    "realtime_post_perm": false,
    "rot": "",
    "roi": "",
    "draw_track": false,
    "send_track": 0,
    "min_face_size": 0,
    "max_face_size": 0,
    "overall": true,
    "only_norm": false,
    "max_candidates": 0,
    "jpeg_quality": 95,
    "ffmpeg_format": ""
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": -2,
    "filter_min_face_size": 1,
    "filter_max_face_size": 8192,
    "normalized_only": false,
    "jpeg_quality": 95,
```

```json
    "overall_only": false,
    "use_stream_timestamp": false,
    "ffmpeg_params": [],
    "router_timeout_ms": 15000,
    "router_verify_ssl": true,
    "router_headers": [
      "Authorization=Token b612396adc3a6dd71b82b5fe333a0a30"
    ],
    "router_body": [],
    "start_stream_timestamp": 0,
    "imotion_threshold": 0,
    "rot": "",
    "roi": "",
    "realtime_post_interval": 1,
    "realtime_post_every_interval": false,
    "ffmpeg_format": "",
    "disable_drops": true
  },
  "status": "INPROGRESS",
  "status_msg": "",
  "statistic": {
    "processed_duration": 14879,
    "faces_posted": 777,
    "faces_failed": 3,
    "faces_not_posted": 1206,
    "processing_fps": 18.816668,
    "frames_dropped": 0,
    "frames_processed": 0,
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "job_starts": 56
  },
  "restream_url": "",
  "worker_id": "ffsec40_213ab8c0ed5d954e",
  "version": "bl068taaa7tcafrfsmq0"
}
```

Each job has the following parameters:

- `id`: job id.

- `enabled`: active status.

- `stream_url`: URL/address of video stream/file to process.

- `labels`: tag(s) that will be used by the `findface-facerouter` component (`findface-security` in the standard FindFace Security configuration) to find processing directives for faces detected in this stream.

- `single_pass`: if true, disable restarting video processing upon error (by default, false).

- `router_url`: IP address and port of the `findface-facerouter` component (`findface-security` in the standard FindFace Security configuration) to receive detected faces from the `findface-video-worker` component for processing.

- `stream_settings`, `stream_settings_gpu`: video stream settings that duplicate *those* in the `findface-video-manager` configuration file (while having priority over them).

- `status`: job status.

- `status_msg`: additional job status info.

- `statistic`: job progress statistics (progress duration, number of posted and not posted faces, processing fps, the number of processed and dropped frames, job start time, etc.).

- `worker_id`: id of the `findface-video-worker` instance executing the job.

findface-ntls

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

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

- USB dongle if you opt for the on-premise licensing.

Use the main web interface to manage `findface-ntls`:

- view the list of purchased features,

- view license limitations,

- upload a license file,

- view the list of currently active components.

The following components are licensable:

- `findface-tarantool-server`,

- `findface-extraction-api`,

- `findface-video-manager`,

- `findface-video-worker`.

Important: After connection between `findface-ntls` and a licensable component, or between `findface-ntls` and the global license server is broken, you will have 6 hours to restore it before the licensable components will be automatically stopped.

The `findface-ntls` configuration is done through a configuration file `/etc/findface-ntls.cfg`.

```
# Listen address of NTLS server where services will connect to.
# The format is IP:PORT
# Use 0.0.0.0:PORT to listen on all interfaces
# This parameter is mandatory and may occur multiple times
# if you need to listen on several specific interfaces or ports.
listen = 127.0.0.1:3133


# Directory with license files.
# NTLS use most recently generated one.
# Note: "recentness" of a license file is detected not by
#       mtime/ctime but from its internal structure.
#
# This parameter is mandatory and must occur exactly once.
license-dir = /opt/ntech/license

# You can specify proxy which NTLS will use to access
# global license server. The syntax is the same that is used by curl.
# Proxy is optional
#proxy = http://192.168.1.1:12345
```

```
# This is bind address for NTLS web-interface.
# Note: there're no authorization or access restriction mechanisms
#       in NTLS UI. If you need one, consider using nginx as proxy
#       with .htaccess / ip-based ACLs.
# This parameter may be specified multiple times.
ui = 127.0.0.1:3185
```

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

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

findface-security

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

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

The `findface-security` configuration is done through the `/etc/ffsecurity/config.py` configuration file.

```
sudo vi /etc/ffsecurity/config.py

MEDIA_ROOT = "/var/lib/ffsecurity/uploads"
STATIC_ROOT = "/var/lib/ffsecurity/static"

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

EXTERNAL_ADDRESS = "http://172.20.77.58"

DEBUG = False

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'ffsecurity',
    }
}
```

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

FFSECURITY = {
    'VIDEO_DETECTOR_TOKEN': '381b0f4a20495227d04185ab02f5085f',
    'CONFIDENCE_THRESHOLD': 0.75,
    'MINIMUM_DOSSIER_QUALITY': -2,
    'IGNORE_UNMATCHED': False,
    'EXTRACTION_API': 'http://127.0.0.1:18666/',
    'VIDEO_MANAGER_ADDRESS': 'http://127.0.0.1:18810',
    'EVENTS_MAX_AGE': 30,
    'NTLS_HTTP_URL': 'http://127.0.0.1:3185',
    'ROUTER_URL': 'http://172.20.77.58',
    'MONITORING_UPDATE_INTERVAL': 60,
    'SF_API_ADDRESS': 'http://127.0.0.1:18411',
    'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'],
    'LIVENESS_THRESHOLD': 0.945,
    'BEARD_THRESHOLD': 0.7,
}

ASGI_THREADS = 16

UVICORN_SETTINGS = {
    'workers': 4,
    'host': 'localhost',
    'port': 8002,
 }

FFSECURITY_UI_CONFIG = {
    "event": {
        "features": {
            "f_gender_class": ["male", "female"],
            "age": {
                "f_age_gte": "",
                "f_age_lte": ""
            },
            "f_emotions_class": ["angry", "disgust", "fear", "happy", "sad",
→"surprise"],
            "f_glasses_class": ["none", "eye", "sun"],
            "f_beard_class": ["none", "beard"],
            "f_liveness_class": ["real", "fake"],
        }
    }
}

# integration plugins
INSTALLED_APPS.append('ffsecurity_genetec')  # remove or comment out this␣
→line to disable genetec integration
```

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

Parameter	Description
EXTERNAL_ADDRESS	External IP address or URL that will be used to access the FindFace Security web interface.
VIDEO_DETECTOR_TOKEN	To authorize the video face detection module, come up with a token and specify it here.
VIDEO_MANAGER_ADDRESS	IP address of the `findface-video-manager` host.
NTLS_HTTP	IP address of the `findface-ntls` host.
ROUTER_URL	IP address of the `findface-security` host that will receive detected faces from the `findface-video-worker` instance(s). Specify either external or internal IP address, subject to the network through which `findface-video-worker` interacts with `findface-security`.
SF_API_ADDRESS	IP address of the `findface-sf-api` host.
IGNORE_UNMATCHED	Disable logging events for faces which have no match in the dossiers (negative verification result). Set true if the system has to process a large number of faces.
CONFIDENCE_THRESHOLD	Face similarity threshold for verification
MINIMUM_DOSSIER_QUALITY	Minimum quality of a face in a dossier photo. Photos containing faces of worse quality will be rejected when uploading to a dossier. Upright faces in frontal position are considered the best quality. They result in values around 0, mostly negative (such as -0.00067401276, for example). Inverted faces and large face angles are estimated with negative values some -5 and less. By default, `'MINIMUM_DOSSIER_QUALITY'`: `-2` which is the average quality.
EVENTS_FEATURES	If you enabled recognition models in the `findface-extraction-api` configuration file, list them here.
LIVENESS_THRESHOLD	The liveness detector will estimate a face liveness with a certain level of confidence. Depending on the confidence threshold, it will return a binary result `real` or `fake`.
BEARD_THRESHOLD	The presence of a beard on a face is determined with a certain level of confidence. Depending on the confidence threshold, the system returns a binary result `none` or `beard`.
EPISODE...	(Add manually for Episodes) The period of time preceding an event, within which the system searches the biometric database for events with similar faces. If no such an event is found, the system creates a new episode. Otherwise, it picks up the most relevant event from a LIVE episode after sorting out the 100 most recent similar faces. See *Configure Episodes*.
EPISODE...	(Add manually for Episodes) The maximum episode duration in seconds. After this time, an episode automatically closes.
EPISODE...	(Add manually for Episodes) The maximum time in seconds since the last event has been added to an episode. After this time, an episode automatically closes.

> **Warning:** The `FFSECURITY` section must end with the `EVENTS_FEATURES`/ `LIVENESS_THRESHOLD`/ `BEARD_THRESHOLD` parameters which have to be given in this very order.
>
> ```
> ...
> 'SF_API_ADDRESS': 'http://127.0.0.1:18411',
> 'EVENTS_FEATURES': ['gender', 'age', 'emotions', 'beard', 'glasses'],
> 'LIVENESS_THRESHOLD': 0.945,
> 'BEARD_THRESHOLD': 0.7,
> ```

findface-facerouter

> **Important:** The `findface-facerouter` is not included in the FindFace Security standard configuration. Use it for integration if necessary. See *Plugins*.

The `findface-facerouter` service sets processing directives for faces detected in video. The directives are set

through custom plugins.

The `findface-facerouter` service accepts a face bbox and normalized image along with the original image and other data (for example, the detection date and time) from the `findface-video-worker` service. In general, `findface-facerouter` allows you to apply arbitrary face processing directives, including directly sending faces to a partner application. In the basic configuration, `findface-facerouter` is pre-configured to redirect faces to `findface-sf-api` for further processing, but you will still have to set processing directives by creating a plugin.

Functionality:

- sets processing directives for faces detected in video,

- redirects faces detected in video to `findface-sf-api` or other service (including a third-party application) for further processing.

The `findface-facerouter` configuration is done through a configuration file `/etc/findface-facerouter.py`.

```
# main.py options:


# debug                       = False
## debug - debug mode
# detector                    = ''
## detector - Detector to use if client fails to provide normalized face
## (nnd).Use "nnd" if you need to detect faces in such requests. Empty value
## rejects requests without face0.
# host                        = ''
## host - host to listen
# port                        = 18820
## port - port to listen
# sfapi_url                   = 'http://localhost:18411'
## sfapi_url - SF-API URL
# version                     = False
## version - print version


# plugin_dir.py options:


# plugin_dir                  = ''
## plugin_dir - Plugin directory for plugin_source='dir'


# abstract_define.py options:


# plugin_source               = 'dir'
## plugin_source - Plugin source (dir)


# log.py options:


# log_file_max_size           = 100000000
## log_file_max_size - max size of log files before rollover
# log_file_num_backups        = 10
## log_file_num_backups - number of log files to keep
# log_file_prefix             = None
## log_file_prefix - Path prefix for log files. Note that if you are running
## multiple tornado processes, log_file_prefix must be different for each of
## them (e.g. include the port number)
# log_rotate_interval         = 1
## log_rotate_interval - The interval value of timed rotating
# log_rotate_mode             = 'size'
## log_rotate_mode - The mode of rotating files(time or size)
```

```
# log_rotate_when                = 'midnight'
## log_rotate_when - specify the type of TimedRotatingFileHandler interval other
## options:('S', 'M', 'H', 'D', 'W0'-'W6')
# log_to_stderr                  = None
## log_to_stderr - Send log output to stderr (colorized if possible). By default
## use stderr if --log_file_prefix is not set and no other logging is
## configured.
# logging                        = 'info'
## logging - Set the Python log level. If 'none', tornado won't touch the
## logging configuration.
```

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

Parameter	Description
`sfapi_url`	IP address and port of the `findface-sf-api` host.
`plugin_dir`	List of directories with plugins to define face processing directives.

1.9.3 Installation File

FindFace Security installation configuration is automatically saved to a file `/tmp/<findface-installer-*>.json`. You can edit this file and use it to install FindFace Security on other hosts without having to answer the installation questions again.

Tip: See *Deploy from Console Installer* to learn more about the FindFace Security installer.

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

Here is an example of the installation file:

```
{
  "findface-security.config": {
    "EXTERNAL_ADDRESS": "http://172.20.77.17"
  },
  "product": "security",
  "ext_ip.bind": "0.0.0.0",
  "findface-ntls.config": {
    "NTLS_LISTEN": "127.0.0.1:3133",
    "NTLS_LISTEN_UI": "127.0.0.1:3185",
    "NTLS_LICENSE_DIR": "/opt/ntech/license"
  },
  "components": [
    "findface-data",
    "memcached",
    "etcd",
    "redis",
    "postgresql",
    "findface-ntls",
    "findface-extraction-api",
    "findface-sf-api",
    "findface-upload",
```

```
    "findface-video-manager",
    "findface-video-worker",
    "findface-security",
    "findface-tarantool-server"
  ],
  "memcached.config": {
    "max_memory": 1024,
    "listen_host": "127.0.0.1",
    "item_size": 16
  },
  "findface-video-manager.config": {
    "listen": "127.0.0.1:18810",
    "master": {
      "self_url_http": "127.0.0.1:18811",
      "self_url": "127.0.0.1:18811"
    },
    "rpc": {
      "listen": "127.0.0.1:18811"
    },
    "ntls": {
      "url": "http://127.0.0.1:3185/",
      "enabled": false
    }
  },
  "findface-video-worker.variant": "cpu",
  "findface-extraction-api.variant": "cpu",
  "ignore_lowmem": true,
  "findface-video-worker.config": {
    "FKVD_WRK_CAP": "10",
    "FKVD_MGR_ADDR": "127.0.0.1:18811",
    "FKVD_NTLS_ADDR": "127.0.0.1:3133"
  },
  "findface-extraction-api.config": {
    "listen": "127.0.0.1:18666",
    "extractors": {
      "instances": 1,
      "models": {
        "gender": "",
        "face": "face/elderberry_576.cpu.fnk",
        "age": "",
        "emotions": ""
      }
    },
    "nnd": {
      "quality_estimator": true
    },
    "license_ntls_server": "127.0.0.1:3133"
  },
  "ext_ip.advertised": "172.20.77.17",
  "findface-tarantool-server.config": {
    "shard-002": {
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_LISTEN": "127.0.0.1:33002",
      "TNT_FF_LISTEN_IP": "127.0.0.1",
      "TNT_EXTRA_LUA": "\ndofile(\"/etc/ffsecurity/tnt_schema.lua\")\n",
      "TNT_FF_NTLS": "127.0.0.1:3133",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-002",
```

```
      "TNT_FF_LISTEN_PORT": "8102"
    },
    "shard-001": {
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_LISTEN": "127.0.0.1:33001",
      "TNT_FF_LISTEN_IP": "127.0.0.1",
      "TNT_EXTRA_LUA": "\ndofile(\"/etc/ffsecurity/tnt_schema.lua\")\n",
      "TNT_FF_NTLS": "127.0.0.1:3133",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-001",
      "TNT_FF_LISTEN_PORT": "8101"
    }
  },
  "tnt_instances": 2,
  "inter_ip.bind": "127.0.0.1",
  "type": "stand-alone",
  "findface-sf-api.config": {
    "listen": "127.0.0.1:18411",
    "extraction-api": {
      "extraction-api": "http://127.0.0.1:18666"
    },
    "storage-api": {
      "shards": [
        {
          "master": "http://127.0.0.1:8101/v2/",
          "slave": ""
        },
        {
          "master": "http://127.0.0.1:8102/v2/",
          "slave": ""
        }
      ]
    }
  },
  "findface-facerouter.config": {
    "plugin_source": "dir",
    "port": "18820",
    "plugin_dir": "/etc/findface-facerouter-plugins",
    "sfapi_url": "http://127.0.0.1:18411",
    "host": "127.0.0.1"
  },
  "inter_ip.advertised": "127.0.0.1"
}
```

1.9.4 Neural Network Models

Here you can see a summary for neural network models created by our Lab and used in FindFace Security:

Note: The CPU and GPU benchmark setup is the following:

- CPU - Intel® Core™ i7-5930K CPU @ 3.50GHz × 12

- GPU - GeForce GTX 1080

Important: The default face biometrics model upon a clean install is `grapefruit_480`.

Warning: Strictly not recommended to use `face/elderberry_160` for work.

Model	CPU, FPS	GPU, FPS	Type
face/elderberry_160	14.99	204.98	Face biometrics
face/elderberry_576.r2	2.07	71.14	
face/grapefruit_160	Benchmark test required		
face/grapefruit_480	Benchmark test required		
faceattr/age.v1	14.99	529.35	Age recognition
faceattr/beard.v0	15.03	532.05	Beard recognition
faceattr/emotions.v1	10.99	235.59	Emotions recognition
faceattr/gender.v2	15.01	523.22	Gender recognition
faceattr/glasses3.v0	15.01	529.64	Glasses recognition

1.9.5 FindFace Security Data Storages

In this section:

- *List of Storages*
- *Biometric Database Galleries*

List of Storages

FindFace Security uses the following data storages:

- Tarantool-based biometric database that stores biometric samples (feature vectors) and face identification events.

- Main system database based on PostgreSQL, that stores internal system data, dossiers, user accounts, and camera settings.

- Directory `/var/lib/ffsecurity/uploads` that stores uploaded dossier photos, video files, and such event artifacts as full frames, face thumbnails, and normalized face images.

- Directory `/var/lib/ffupload/` that stores only such event artifacts as face thumbnails.

Biometric Database Galleries

There are 3 galleries in the Tarantool-based biometric database:

- `ffsec_dossier_face`: biometric samples extracted from dossier photos.

- `ffsec_events`: biometric samples extracted from faces detected in the video.

- `ffsec_monitoring`: biometrics samples from the active dossiers under watch.

1.9.6 Backup Options

To backup the biometric database, you need the `findface-storage-api-dump` utility. It can be launched with the following options:

Note: You can find the detailed information on the `findface-storage-api-dump` usage in *Backup and Restore Data Storages*.

```
findface-storage-api-dump --help

Usage of findface-storage-api-dump:
  -cache string
        Cache type: inmemory, redis or memcache (default "memcache")
  -cache-inmemory-size int
        Maximum number of items in ARC cache (default 16384)
  -cache-memcache-nodes value
        Comma-separated list of memcache shards (default 127.0.0.1:11211)
  -cache-memcache-timeout duration
        Specifies read/write timeout (default 100ms)
  -cache-redis-addr string
        Host:Port address (default "localhost:6379")
  -cache-redis-db int
        Database to be selected after connecting to the server.
  -cache-redis-network string
        Network type, either tcp or unix (default "tcp")
  -cache-redis-password string
        Optional password. Must match the password specified in the requirepass␣
↪server configuration option.
  -cache-redis-timeout duration
        Specifies dial/read/write timeout (default 5s)
  -config string
        Path to config file
  -config-template
        Output config template and exit
  -extraction-api-extraction-api string
        Extraction API address (default "http://127.0.0.1:18666")
  -extraction-api-timeouts-connect duration
        extraction-api-timeouts-connect (default 5s)
  -extraction-api-timeouts-idle-connection duration
        extraction-api-timeouts-idle-connection (default 10s)
  -extraction-api-timeouts-overall duration
        extraction-api-timeouts-overall (default 35s)
  -extraction-api-timeouts-response-header duration
        extraction-api-timeouts-response-header (default 30s)
  -limits-allow-return-facen
        Allow returning raw feature vectors to detect responses if ?return_facen=true
  -limits-body-image-length int
        Maximum length of image supplied in request body (default 33554432)
  -limits-deny-networks string
        Comma-separated list of subnets that are not allowed to fetch from (default
↪"127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8")
  -limits-url-length int
        Maximum supported url length in bytes (default 4096)
  -listen string
        IP:port to listen on (default ":18411")
  -normalized-storage-enabled
```

(continues on next page)

```
        Enables normalize saving (default true)
  -normalized-storage-s3-access-key string
        Access key for the object storage
  -normalized-storage-s3-bucket-name string
        S3 storage bucket name
  -normalized-storage-s3-endpoint string
        S3 compatible object storage endpoint
  -normalized-storage-s3-operation-timeout int
        Storage operations (Get,Put,Delete) timeout in seconds (default 30)
  -normalized-storage-s3-public-url string
        Storage public url
  -normalized-storage-s3-region string
        Storage region
  -normalized-storage-s3-secret-access-key string
        Secret key for the object storage
  -normalized-storage-s3-secure
        If 'true' API requests will be secure (HTTPS), and insecure (HTTP) otherwise␣
↪(default true)
  -normalized-storage-webdav-timeouts-connect duration
        normalized-storage-webdav-timeouts-connect (default 5s)
  -normalized-storage-webdav-timeouts-idle-connection duration
        normalized-storage-webdav-timeouts-idle-connection (default 10s)
  -normalized-storage-webdav-timeouts-overall duration
        normalized-storage-webdav-timeouts-overall (default 35s)
  -normalized-storage-webdav-timeouts-response-header duration
        normalized-storage-webdav-timeouts-response-header (default 30s)
  -normalized-storage-webdav-upload-url string
        webdav storage for normalized, disable normalized if empty string (default
↪"http://127.0.0.1:3333/uploads/")
  -normalized_storage string
        Normalized storage type: webdav, s3 (default "webdav")
  -storage-api-max-idle-conns-per-host int
        storage-api-max-idle-conns-per-host (default 20)
  -storage-api-timeouts-connect duration
        storage-api-timeouts-connect (default 5s)
  -storage-api-timeouts-idle-connection duration
        storage-api-timeouts-idle-connection (default 10s)
  -storage-api-timeouts-overall duration
        storage-api-timeouts-overall (default 35s)
  -storage-api-timeouts-response-header duration
        storage-api-timeouts-response-header (default 30s)
```

1.9.7 Restore Options

To restore the biometric database from a backup, you need the `findface-storage-api-restore` utility. It can be launched with the following options:

Note: You can find the detailed information on the `findface-storage-api-restore` usage in *Backup and Restore Data Storages*.

```
findface-storage-api-restore --help
Usage of findface-storage-api-restore:
  -cache string
```

```
        Cache type: inmemory, redis or memcache (default "memcache")
  -cache-inmemory-size int
        Maximum number of items in ARC cache (default 16384)
  -cache-memcache-nodes value
        Comma-separated list of memcache shards (default 127.0.0.1:11211)
  -cache-memcache-timeout duration
        Specifies read/write timeout (default 100ms)
  -cache-redis-addr string
        Host:Port address (default "localhost:6379")
  -cache-redis-db int
        Database to be selected after connecting to the server.
  -cache-redis-network string
        Network type, either tcp or unix (default "tcp")
  -cache-redis-password string
        Optional password. Must match the password specified in the requirepass␣
→server configuration option.
  -cache-redis-timeout duration
        Specifies dial/read/write timeout (default 5s)
  -config string
        Path to config file
  -config-template
        Output config template and exit
  -extraction-api-extraction-api string
        Extraction API address (default "http://127.0.0.1:18666")
  -extraction-api-timeouts-connect duration
        extraction-api-timeouts-connect (default 5s)
  -extraction-api-timeouts-idle-connection duration
        extraction-api-timeouts-idle-connection (default 10s)
  -extraction-api-timeouts-overall duration
        extraction-api-timeouts-overall (default 35s)
  -extraction-api-timeouts-response-header duration
        extraction-api-timeouts-response-header (default 30s)
  -limits-allow-return-facen
        Allow returning raw feature vectors to detect responses if ?return_facen=true
  -limits-body-image-length int
        Maximum length of image supplied in request body (default 33554432)
  -limits-deny-networks string
        Comma-separated list of subnets that are not allowed to fetch from (default
→"127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8")
  -limits-url-length int
        Maximum supported url length in bytes (default 4096)
  -listen string
        IP:port to listen on (default ":18411")
  -normalized-storage-enabled
        Enables normalize saving (default true)
  -normalized-storage-s3-access-key string
        Access key for the object storage
  -normalized-storage-s3-bucket-name string
        S3 storage bucket name
  -normalized-storage-s3-endpoint string
        S3 compatible object storage endpoint
  -normalized-storage-s3-operation-timeout int
        Storage operations (Get,Put,Delete) timeout in seconds (default 30)
  -normalized-storage-s3-public-url string
        Storage public url
  -normalized-storage-s3-region string
        Storage region
```

```
 -normalized-storage-s3-secret-access-key string
       Secret key for the object storage
 -normalized-storage-s3-secure
       If 'true' API requests will be secure (HTTPS), and insecure (HTTP) otherwise␣
↪(default true)
 -normalized-storage-webdav-timeouts-connect duration
       normalized-storage-webdav-timeouts-connect (default 5s)
 -normalized-storage-webdav-timeouts-idle-connection duration
       normalized-storage-webdav-timeouts-idle-connection (default 10s)
 -normalized-storage-webdav-timeouts-overall duration
       normalized-storage-webdav-timeouts-overall (default 35s)
 -normalized-storage-webdav-timeouts-response-header duration
       normalized-storage-webdav-timeouts-response-header (default 30s)
 -normalized-storage-webdav-upload-url string
       webdav storage for normalized, disable normalized if empty string (default
↪"http://127.0.0.1:3333/uploads/")
 -normalized_storage string
       Normalized storage type: webdav, s3 (default "webdav")
 -storage-api-max-idle-conns-per-host int
       storage-api-max-idle-conns-per-host (default 20)
 -storage-api-timeouts-connect duration
       storage-api-timeouts-connect (default 5s)
 -storage-api-timeouts-idle-connection duration
       storage-api-timeouts-idle-connection (default 10s)
 -storage-api-timeouts-overall duration
       storage-api-timeouts-overall (default 35s)
 -storage-api-timeouts-response-header duration
       storage-api-timeouts-response-header (default 30s)
```

Operator's Guide

2.1 Web Interface

Use the web interface to interact with FindFace Security. To open the web interface, enter its address in the address bar of your browser, and log in.

Note: Request credentials from administrator.

The web interface has a highly intuitive and handy design and provides the following functionality:

- *Search Databases*.
- *Real-time Face Identification Events*.
- *Dossier* (only for users with operator privileges).
- *Video Wall*.

2.2 Search Databases

FindFace Security allows you to search for faces in the following databases:

- Database of detected faces (the *Events* tab).
- Dossier database (the *Dossiers*). Contains face reference images.

To find a face in a database, navigate to the *Search* tab.

In this chapter:

- *Search for Faces in Event List*

- *Search for Faces in Dossier List*

2.2.1 Search for Faces in Event List

FindFace Security allows you to search the database of detected faces.

Note: You can access this database by navigating to the event list (the *Events* tab).

To find a face, do the following:

1. Navigate to the *Search* tab.
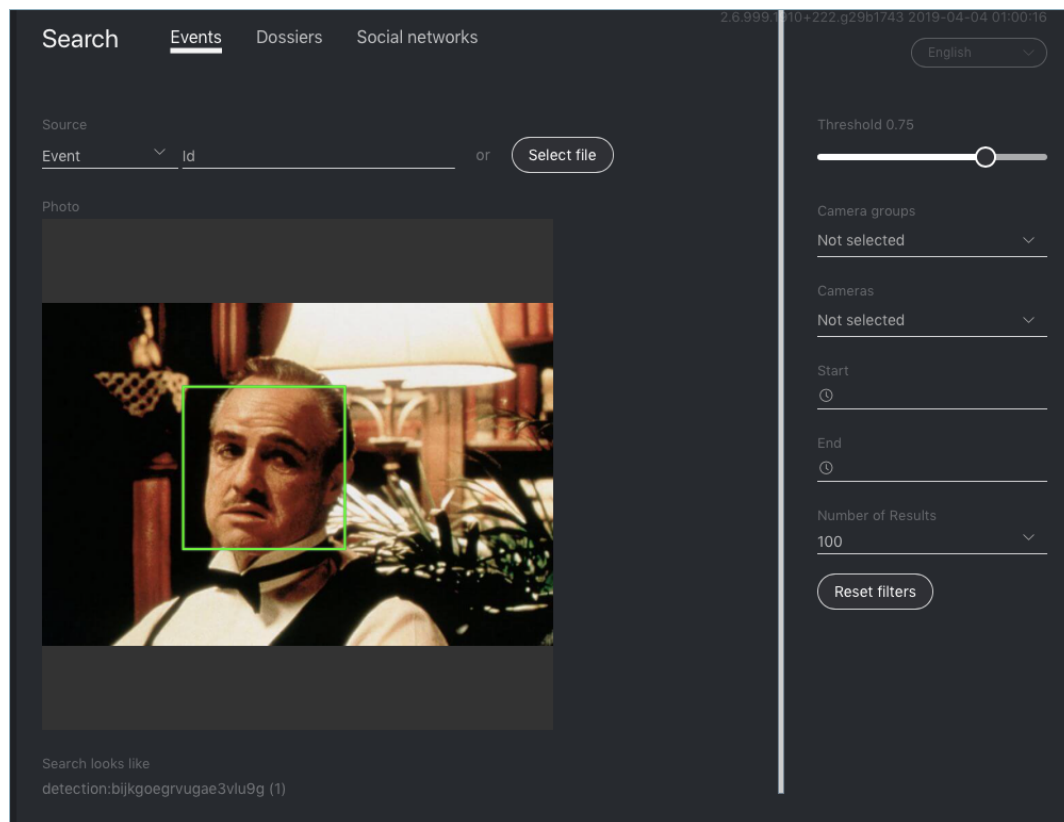
2. Specify a database to search: *Events*.

3. Upload a photo. It will be displayed in the *Photo* area. If there are multiple faces in the image, select the one you want.

Note: Instead of a photo, you can specify the ID of an event that features the face you want to find.

---

4. By default, the system searches for faces using the identification threshold 0.75. If necessary, set your own value using the *Threshold* filter.

5. (Optional) Specify a group of cameras, camera and a time period within which the event occurred.

6. Specify the maximum number of dossiers in the search results.

7. Click *Search*. You will see the search results appear below. For each face found, the matching confidence level is provided.

2.2.2 Search for Faces in Dossier List

FindFace Security allows you to search the database of dossiers containing face reference images.

To find a face, do the following:

1. Navigate to the *Search* tab.
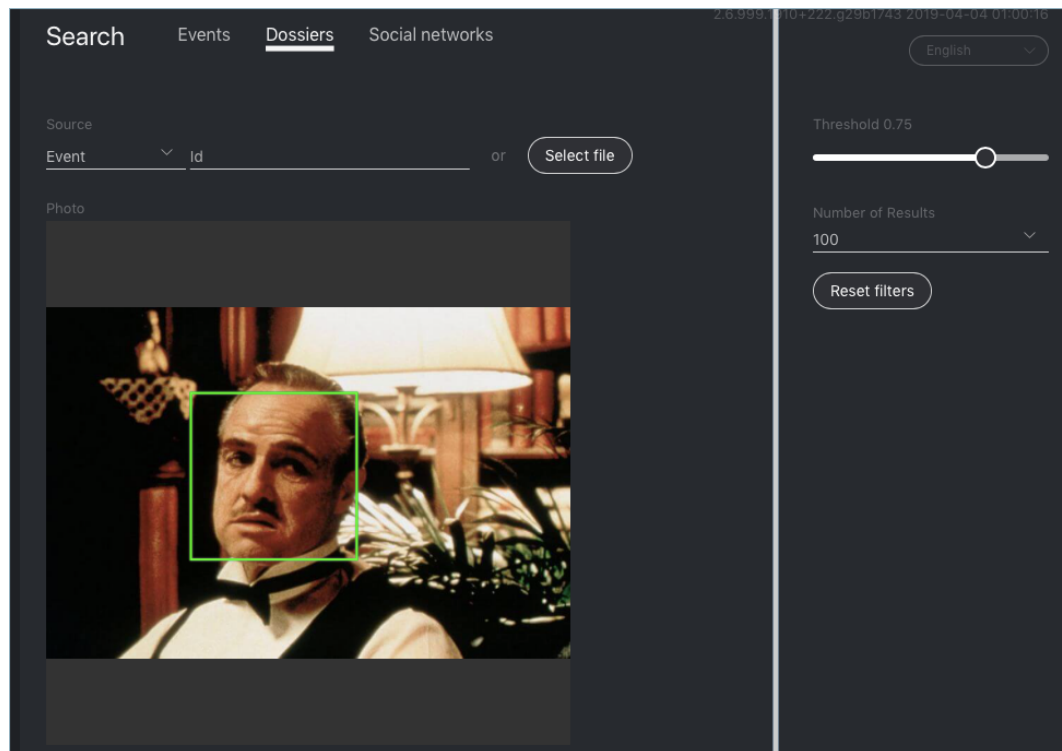
2. Specify a database to search: *Dossiers*.

3. Upload a photo. It will be displayed in the *Photo* area. If there are multiple faces in the image, select the one you want.

Note: Instead of a photo, you can specify the ID of an event that features the face you want to find.

---

4. By default, the system searches for faces using the identification threshold 0.75. If necessary, set your own value using the *Threshold* filter.

5. Specify the maximum number of dossiers in the search results.

6. Click *Search*. You will see the search results appear below. For each face found, the matching confidence level is provided.

2.3 Real-time Face Identification Events

To monitor the real-time face identification in live videos, use the *Events* and *Episodes* tabs. Besides monitoring, both tabs allow you to access the history of identification events. This section is all about the *Events*.

Tip: Take your security up a notch with *Organize Events with Episodes*.

Tip: Search for faces through the event database and dossier database on the *Search* tab.

Tip: To perform the face identification in archived videos, see *Face Identification in Offline Videos*.

In this chapter:

- *View Identification Events*
- *Face Liveness and Face Features Recognition*
- *Event Ticket. Acknowledging Event*
- *Event Ticket. Face Search*

2.3.1 View Identification Events

Once a face detected, you will see a notification on the event list.
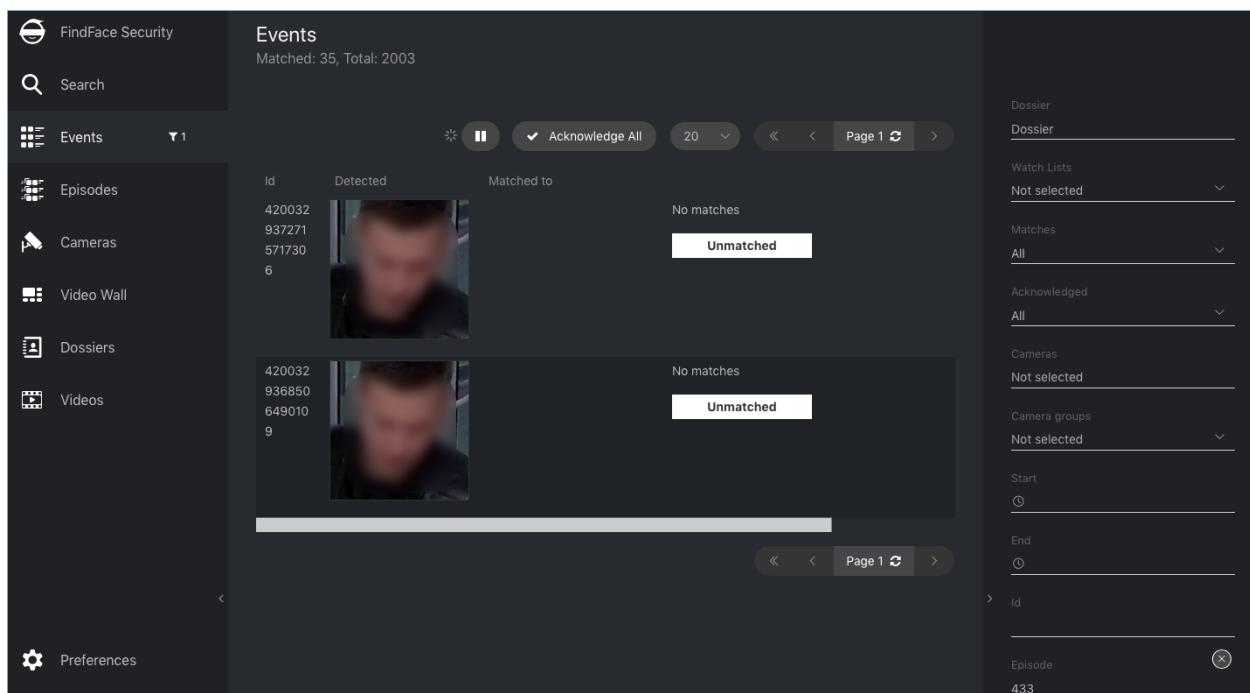
A notification can feature different pieces of information, depending on whether a detected face has a match in the database:

- Match not found: a normalized face image, detection date and time, the name of a camera group.

- Match found: a normalized face image, the photo from a dossier, the name of a person, similarity between faces, the comment from a dossier, the name of a dossier list, detection date and time, the name of a camera group.

Note: You can configure the system in such a way that you will get notifications only for the faces with a match.

Important: In order to pause the notifications thread, click above the list of events.

When working with events, the following default filters may come in handy:

- *Dossier*: display events only for a selected dossier.

- *Watch lists*: display events only for a selected dossier category (watch list).

  Note: To view only unmatched faces on the event list, select *Unmatched* in this filter.

- *Matches*: display events only with/without matches, or all events.

- *Acknowledged*: display only acknowledged/unacknowledged events, or all events.

- *Cameras*: display only events from a selected camera.

- *Camera groups*: display only events from a selected group of cameras.

- *Start*, *End*: display only events that occurred within a certain time period.

- *id*: display an event with a given ID.

- *Episode*: display events from the episode with a given ID.

2.3.2 Face Liveness and Face Features Recognition

Depending on the system settings, you can see an estimation of face liveness and/or a result of such face features recognition as gender, age, emotions, glasses, and/or beard.

The face liveness detector automatically spots fake faces and prevents photo attacks by distinguishing a live face from a face image.

Note: The liveness score can be `null`. It is so when the liveness detector is disabled or unable to estimate the face liveness in the provided image.
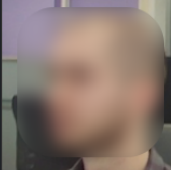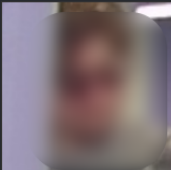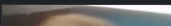
The face feature recognition result is in the following format:

Face feature	Result format	Example
Age	Feature: `age`: number of years	age: 33
Gender	Result: `male/female` (feature: `gender`): algorithm confidence in result	female (gender): 0.95
Emotions	Result: `angry/disgust/fear/happy/sad/surprise/neutral` (feature: `emotions`): algorithm confidence in result	happy (emotions): 0.99
Glasses	Result: `eye/sun/none` (feature: `glasses`): algorithm confidence in result	none (glasses): 0.87
Beard	Result: `beard/none` (feature: `beard`): algorithm confidence in result	none (beard): 0.91

Filter events by face features and liveness when needed.

2.3.3 Event Ticket. Acknowledging Event

In order to navigate to an event ticket from the list of events, click on the face recognition result in a notification (*No matches* or the name of a matching person).

An event ticket contains the same data as a relevant *notification*. It also allows for acknowledging the event. To do so, click *Not accepted* to change the event acknowledgment status. Click *Save*.
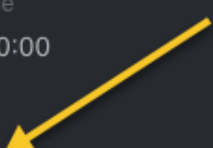
Tip: If a detected face has a match in the dossiers, you can navigate into a relevant one by clicking on the person's name in the event ticket.
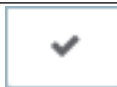
Tip: In order to acknowledge all the events, click above the list of events.

Note: Event acknowledgment can be automated for selected watch lists.

2.3.4 Event Ticket. Face Search

FindFace Security allows you to search detected faces through the list of events and dossier database. To navigate from an event ticket to the search tab, click *Events* or *Dossiers* respectively.
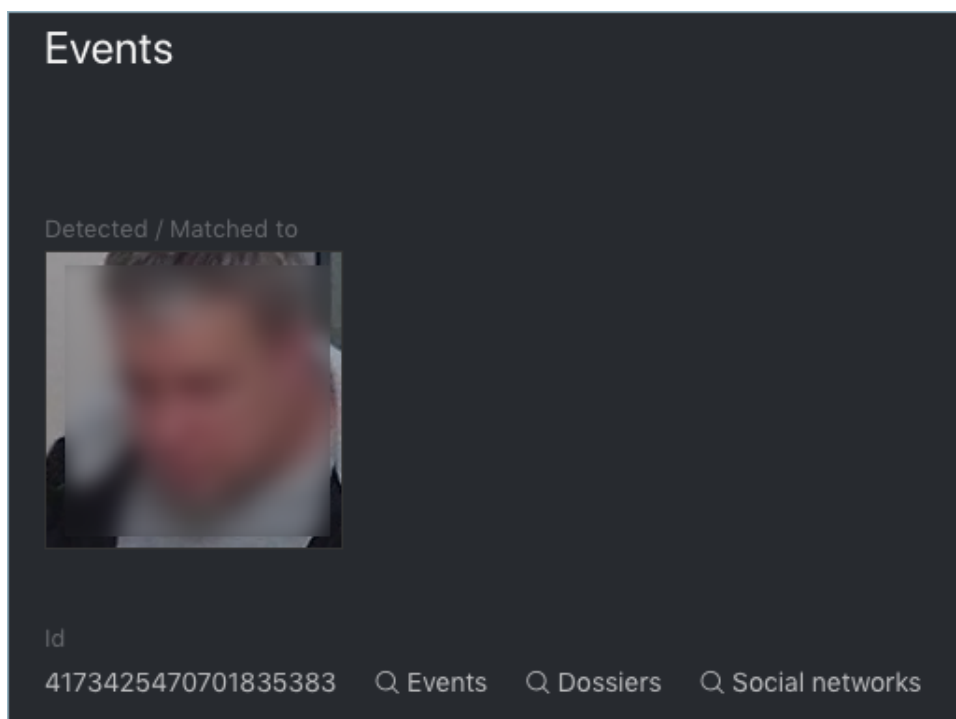
See also:

- *Search Databases*.

2.4 Organize Events with Episodes

To monitor the real-time face identification in live videos, use the *Events* and *Episodes* tabs. Besides monitoring, both tabs allow you to access the history of identification events. This section is all about the *Episodes*.

See also:

- *Real-time Face Identification Events*

- *Configure Episodes*

An episode is a set of identification events that feature faces of the same person, detected within a certain period of time. As events on the *Events* tab show up in an arbitrary order, a large number of miscellaneous events can make the

---

work difficult and unproductive. With the episodes, the system uses AI to organize incoming events based on the faces similarity and detection time. This allows for easy processing of diverse events, even in large numbers.

Tip: Search for faces through the event database and dossier database on the *Search* tab.

Tip: To perform the face identification in archived videos, see *Face Identification in Offline Videos*.

In this chapter:

- *View Identification Episodes*
- *Event and Episode Acknowledging*
- *Filter Events by Episode ID*

2.4.1 View Identification Episodes

There are two types of episodes:

- LIVE: an episode is currently active, with more events to be possibly added.
- Closed: an episode is closed, no events can be added.

You can find the list of episodes with filters and statistics on the *Episodes* tab. Once a face is detected, it is either added to an existing LIVE episode, or used as a starting point of a new episode. Each episode is assigned an identifier which can be later used to filter events and episodes.
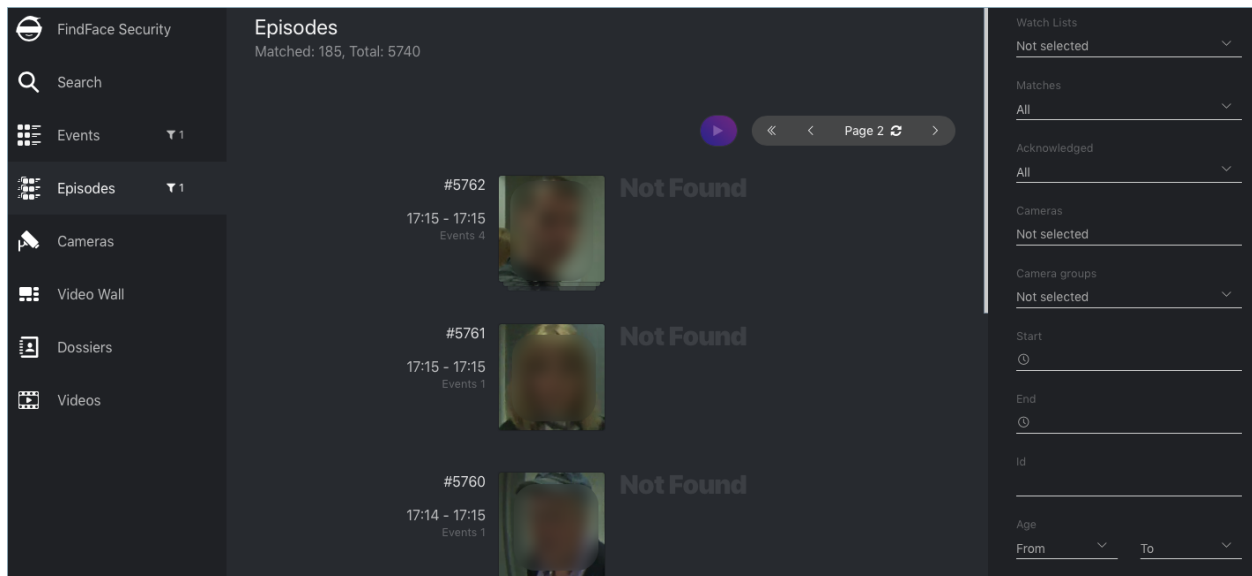
When working with episodes, the following default filters may come in handy:

- *Dossier*: display episodes only for a selected dossier.

- *Watch lists*: display episodes only for a selected dossier category (watch list).

Note: To view only unmatched faces on the episode list, select *Unmatched* in this filter.

- *Matches*: display episodes only with/without matches, or all episodes.
- *Acknowledged*: display only acknowledged/unacknowledged episodes, or all episodes.
- *Cameras*: display only episodes from a selected camera.
- *Camera groups*: display only episodes from a selected group of cameras.
- *Start*, *End*: display only episodes that occurred within a certain time period.
- *id*: display an episode with a given ID.

You can also filter episodes by face liveness and face features (if applicable).

To view the events added to an episode, click it on the list. You will be redirected to the *Events* tab with the corresponding episode ID set in the *Episode* filter:

Work with the *Events* tab as described in *Real-time Face Identification Events*.

2.4.2 Event and Episode Acknowledging

To acknowledge an entire episode, click *Not accepted* for this episode on the list. As a result, all events in the episode will be automatically acknowledged, including those that are yet-to-appear (in the case of a LIVE episode).
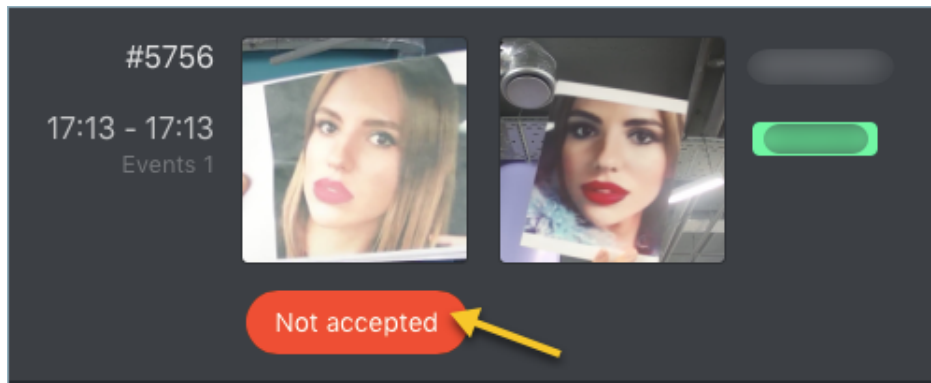
An episode is also automatically acknowledged after acknowledging all its events one by one.

2.4.3 Filter Events by Episode ID

To display events by episode ID, either use the *id* filter on the *Episodes* tab or the *Episode ID* filter on the *Events* tab.

2.5 Dossier

The dossier database contains dossiers on the unwanted persons and VIP guests. A dossier has to contain one or several photos of a person and belong to a certain classification list (watch list).

In this section:

- *Create Dossier*
- *View Dossier*

2.5.1 Create Dossier

To create a dossier, do the following:
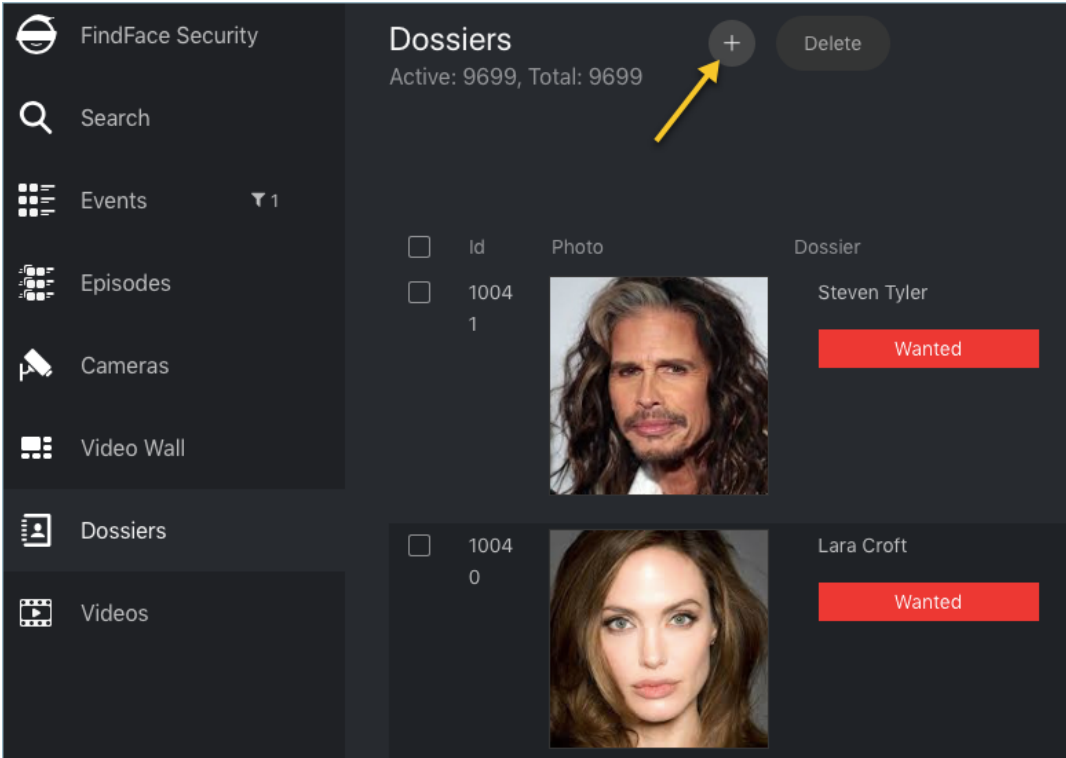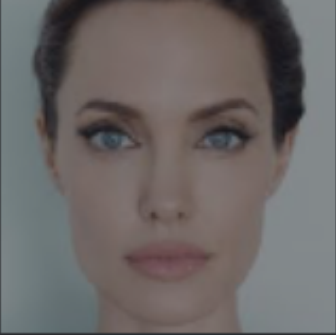
1. Navigate to the *Dossiers* tab.

2. Click +.

3. Attach a photo and specify the name of a person. If necessary, add a comment.

Important: A face in the photo must be of high quality, i.e. close to a frontal position. Photos that do not meet the requirement will be rejected with a detailed error description.

---

4. From the *Watch lists* drop-down menu, select a classification list (or several lists, one by one) for the dossier.

Note: If you cannot find an appropriate watch list for the dossier, *create* a new one, or ask an administrator to do so.

---

5. Check *Active*. If a dossier is inactive, it is excluded from the *real time face identification*.

6. Click *Save*.

2.5.2 View Dossier

You can find all dossiers created in FindFace Security on the *Dossiers* tab. Use the *Watch lists* filter to filter dossiers by list.

2.6 Video Wall

FindFace Security allows basic video surveillance. The video image from cameras and/or video files can be displayed on the Video Wall.

To display video on the Video Wall, do the following:

1. Navigate to the *Video Wall* tab.

2. Select one of the 4 predefined Video Wall layouts.
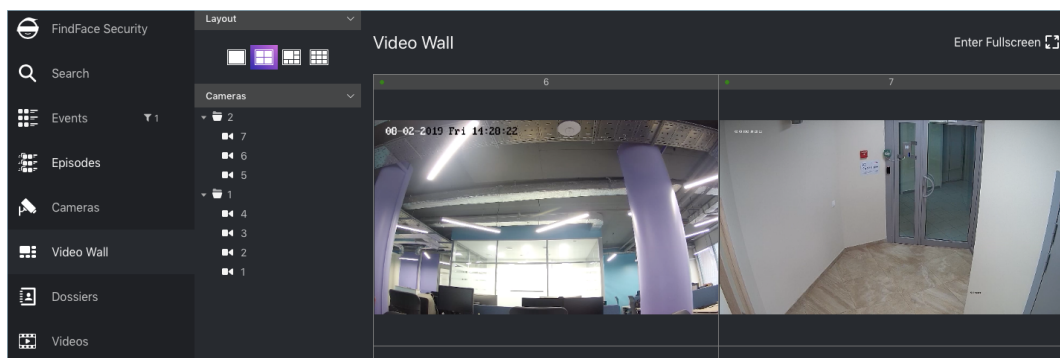
3. Drag-n-drop cameras of your choice to the Video Wall.

2.7 Mobile App

To interact with FindFace Security on the go, use the mobile app. The FindFace Security app is available on request for Android.

In the app, specify your login and password, as well as the FindFace Security URL address, and log in.

The mobile app has a highly intuitive and handy design and provides the following functionality:

- Search for faces in the event list and dossier database.

- Real time face identification in live streams and video files

---

Important: To receive push notifications of events in the mobile version, open a relevant watch list settings in the full version, and check *Require Event Acknoledgment* and *Enable Sound Alert*.
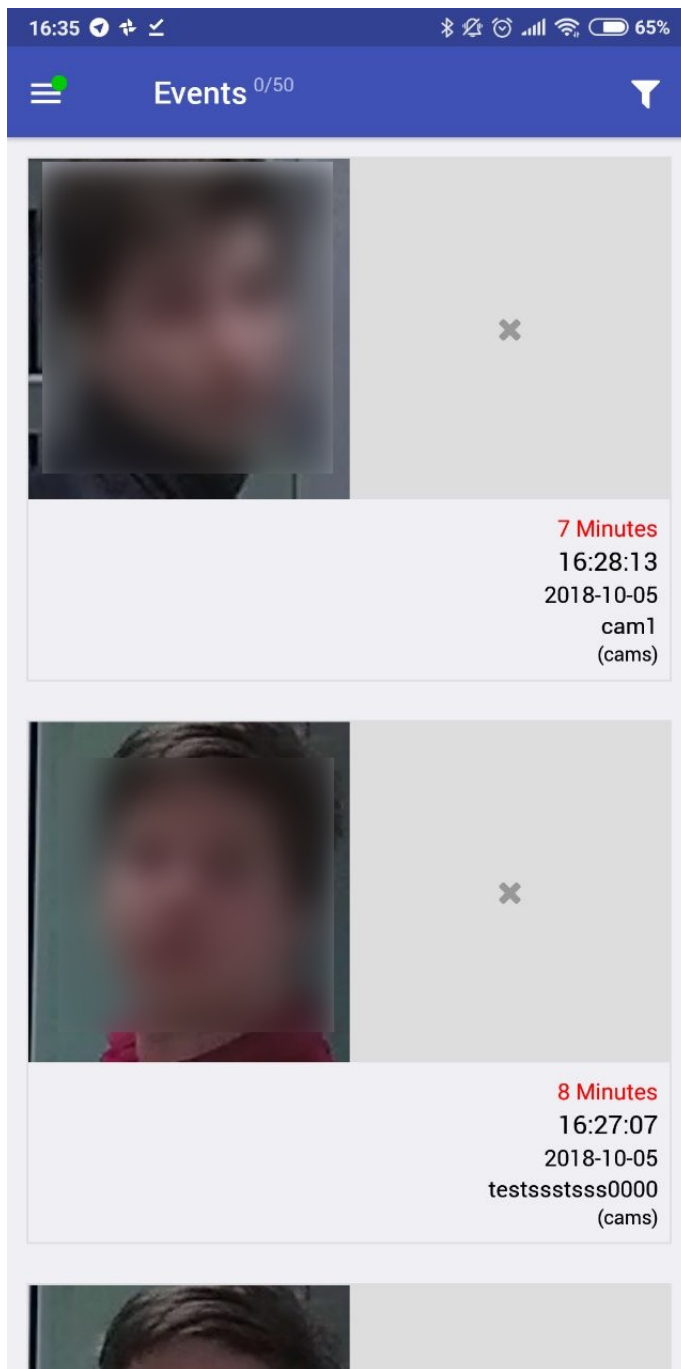
- View and create a dossier on a person.

Working with the mobile app is similar to the full version.

Important: To access *Settings*, you need to enter a PIN code, `1234` by default.

Integrations

This chapter is all about integration with FindFace Security. Integrate your system through HTTP API, webhooks, and plugins, or check out our turnkey partner integrations.

3.1 HTTP API

Detailed interactive documentation on the FindFace Security HTTP API is available after installation at `http://<ffsecurity_ip:port>/api-docs`. Learn and try it out.

Tip: You can also find it by navigating to *Preferences -> Documentation* in the web interface.

3.2 Webhooks

You can set up FindFace Security to automatically send notifications about certain events to a given URL. To do so, create and configure a webhook. In this case, when such an event occurs, FindFace Security 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 certain event, invoke required behaviour on a target website, solve security tasks such as automated access control, etc.

In this section:

- *Configure Webhook*
- *Webhook in Action*

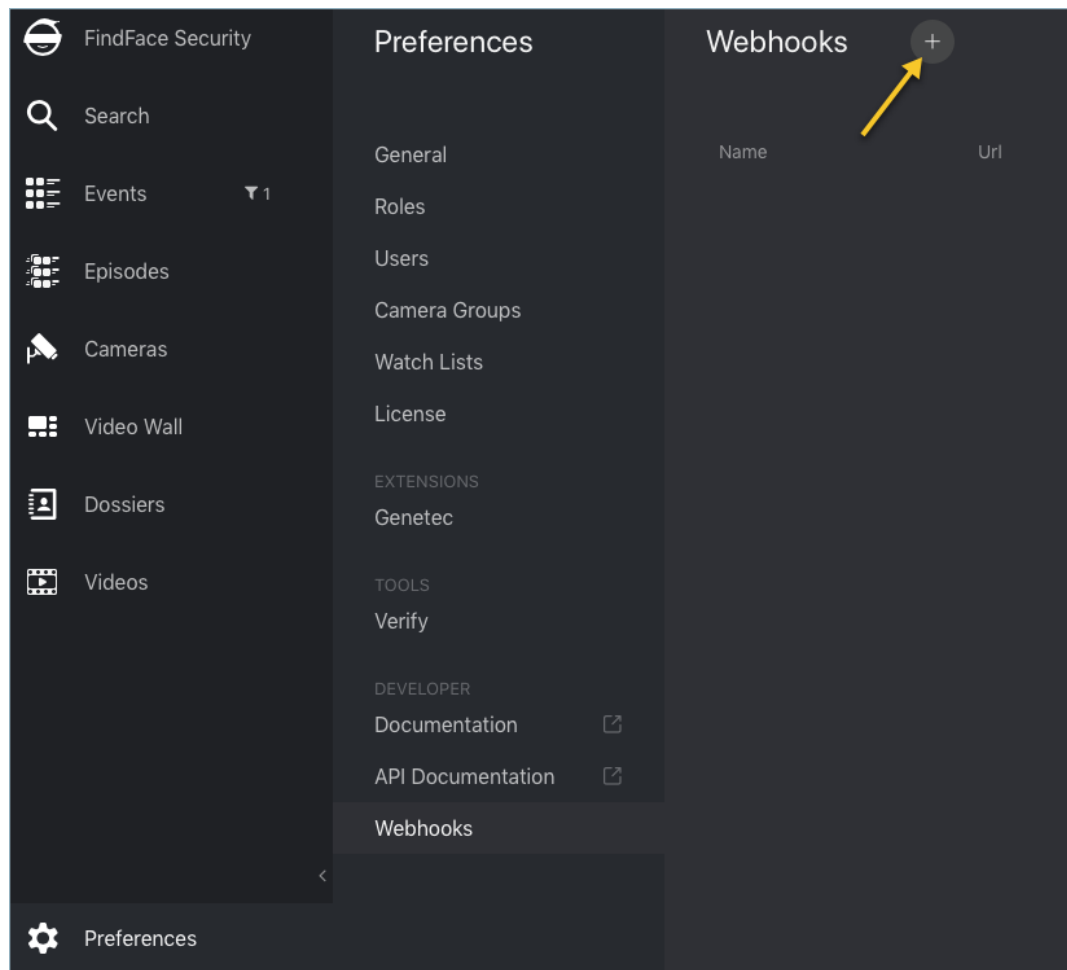### 3.2.1 Configure Webhook

Important: You need Administrator privileges to create a webhook.

Note: To use the webhooks, be sure that at least one of the following parameters is specified in `/etc/ffsecurity/config.py`: `SERVICE_EXTERNAL_ADDRESS` or `EXTERNAL_ADDRESS`.

To create and configure a webhook, do the following:

1. Navigate to the *Preferences* tab. Click *Webhooks*.

2. Click +.

3. Specify the webhook title.

4. Specify URL to automatically send notifications to.

5. FindFace Security will be automatically sending notifications on events which match given filters. You can filter events by the following event parameters:

   - `camera_group_in`: camera group id, number.

   - `matched_dossier_in`: matched dossier id, number.

   - `matched`: event `matched` status (`true` or `false`), boolean.

   - `camera_in`: camera id, number.

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 notifications about all matched events, pass only curly braces without any enclosed filters:

```
{}
```

Note: You can specify several values for each filter (except `matched`). In this case, the web hook will be triggered once one of the values from this filter has been matched. In the example below, you will get an event from the camera group `1` or `3` if a matched dossier is `12` or `25`.

```
{
    "camera_group_in": [1, 3],
    "matched_dossier_in": [12,25]
}
```

6. Check *Active*.

7. Click *Save*.

3.2.2 Webhook in Action

Try out a webhook by capturing event notifications with a simple web server in Python:

```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)
```

If no filters are configured for a webhook, this web server will be getting notifications about each event that occurs in the system. The notifications have the following format:

```
======== Running on http://0.0.0.0:8888 ========
(Press CTRL+C to quit)
[{'acknowledged': True,
  'acknowledged_by': None,
  'acknowledged_date': '2019-04-09T12:29:23Z',
  'acknowledged_reaction': None,
  'camera': 2,
  'confidence': 0.9098,
  'created_date': '2019-04-09T12:29:23Z',
  'face': 'http://172.20.77.17/uploads/2019/04/09/event/122955_face_aT3ZZh.jpg',
```

(continues on next page)

```
'features': {'age': None,
             'beard': None,
             'emotions': None,
             'gender': None,
             'glasses': None,
             'liveness': None},
'frame': 'http://172.20.77.17/uploads/2019/04/09/event/122955_image_3msdHH.jpg',
'frame_coords_bottom': 981,
'frame_coords_left': 1630,
'frame_coords_right': 1911,
'frame_coords_top': 701,
'id': '4173669353687265180',
'looks_like_confidence': None,
'matched': True,
'matched_dossier': 1,
'matched_face': '4173665826982243136',
'matched_lists': [1],
'normalized_photo': 'http://172.20.77.17/uploads/2019/04/09/event/122955_face0_
→E638aW.png',
'quality': -0.000158,
'scores': {'direction_score': -2.62964,
           'frame_no': 800,
           'score': -0.000158435,
           'tracking_duration': 34000}}]
```

To view the webhook pulling status, execute:

```
sudo journalctl -u  findface-security.service | grep webhook
```

Success:

```
`Apr 09 16:02:28 ubuntu ffsecurity[1524]: INFO     [-] hook 1 was pulled on http://172.
→20.77.70:8888`
```

Failure:

```
`Apr 09 15:59:02 ubuntu ffsecurity[1524]: INFO     [-] While working on hook 1␣
→Exception occured: Cannot connect to host 172.20.77.70:8888 ssl:False [Connection␣
→refused]`
```

3.3 Partner Integrations

3.3.1 Genetec Security Center

FindFace Security integration with Genetec Security Center allows you to expand the capabilities of your Genetec-based security system with face recognition functionality.

Configure Integration

Integration with Genetec Security Center is implemented via the `findface-genetec` plugin. By default, the plugin is enabled, and the FindFace Security *Preferences* features the *Genetec* tab.

Note: If it is not so, open the `findface-security` configuration file, and check whether it features the enabled line `INSTALLED_APPS.append('ffsecurity_genetec')`. Also, make sure that at least one of the following parameters is specified: `SERVICE_EXTERNAL_ADDRESS` or `EXTERNAL_ADDRESS`.

```
sudo vi /etc/ffsecurity/config.py


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


...
FFSECURITY_UI_CONFIG = {
    "event": {
        "features": {
            "f_gender_class": ["male", "female"],
            "age": {
                "f_age_gte": "",
                "f_age_lte": ""
            },
            "f_emotions_class": ["angry", "disgust", "fear", "happy", "sad", "surprise
↪"],
            "f_glasses_class": ["none", "eye", "sun"],
            "f_beard_class": ["none", "beard"],
            "f_liveness_class": ["real", "fake"],
        }
    }
}


# integration plugins
INSTALLED_APPS.append('ffsecurity_genetec') # remove or comment out this line to
↪disable
```


Before getting started with the integration on the FindFace Security side, deploy the Genetec Web SDK and Media Gateway packages, and create an `Alarm` entity that will be triggered in Genetec Security Center when a face recognition event occurs in FindFace Security.

In this chapter:

- *Configure Genetec Web SDK and Media Gateway*
- *Create Alarm in Genetec Security Center*
- *Configure Endpoints in FindFace Security*
- *Import Cameras from Genetec Security Center*
- *Create Watch Lists and Dossiers in FindFace Security*

Configure Genetec Web SDK and Media Gateway

To enable and configure Web SDK, use Genetec Config Tool. For details, refer to *Security Center Administrator Guide -> Chapter 52: Role Types -> Web-based SDK configuration tabs*.
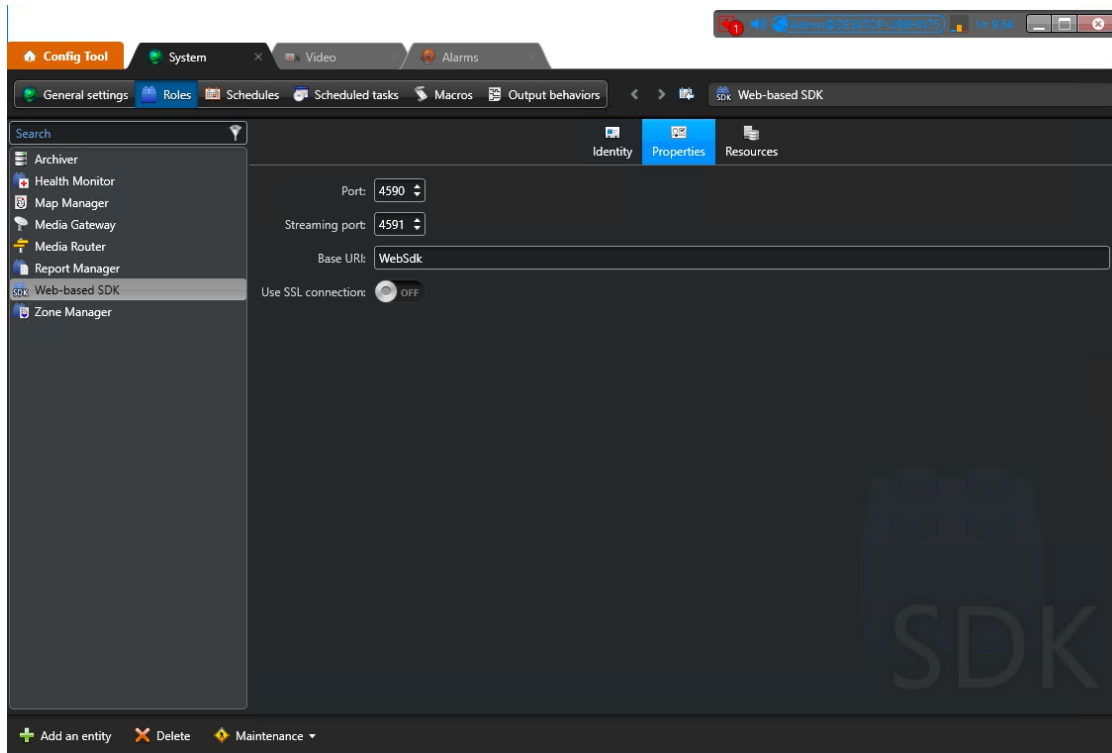

When enabling and configuring Media Getaway in Genetec Config Tool, refer to *Security Center Administrator Guide -> Chapter 24: Video Deployment*.
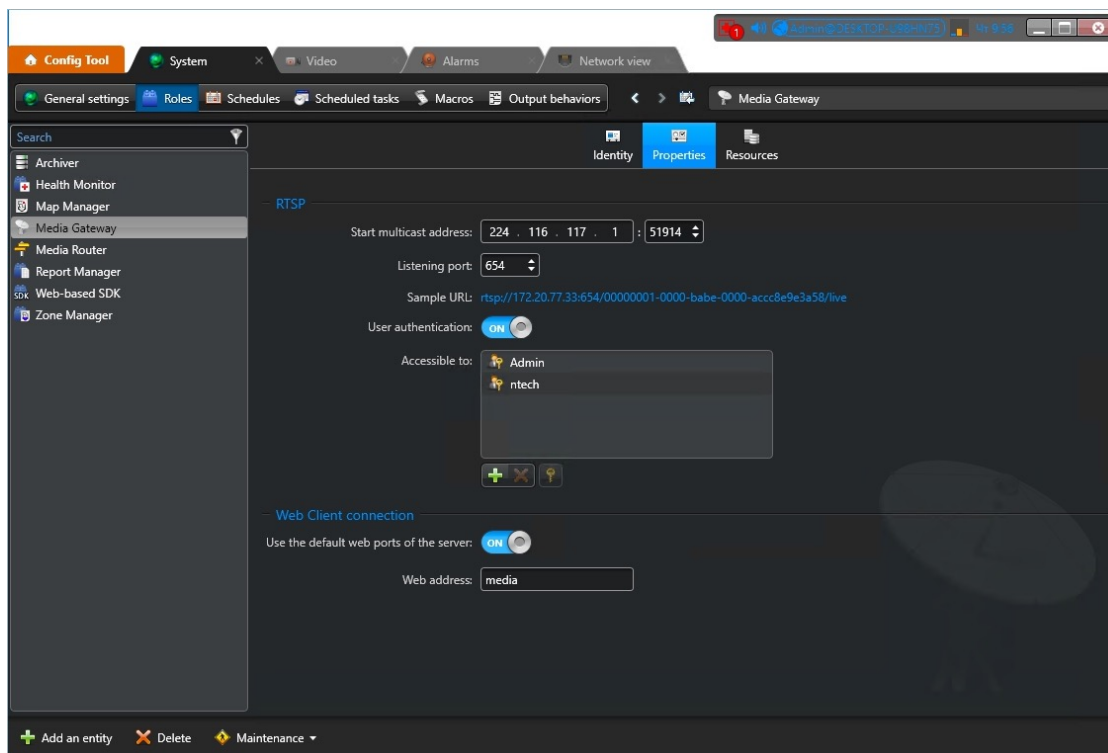
Important: Make sure that the firewall is configured so that the ports for the WebSDK and Media Gateway are open.

Create Alarm in Genetec Security Center

Create and configure a new `Alarm` entity in Genetec Config Tool. Refer to *Security Center Administrator Guide -> Chapter 48: Alarms -> Creating Alarms* for details.
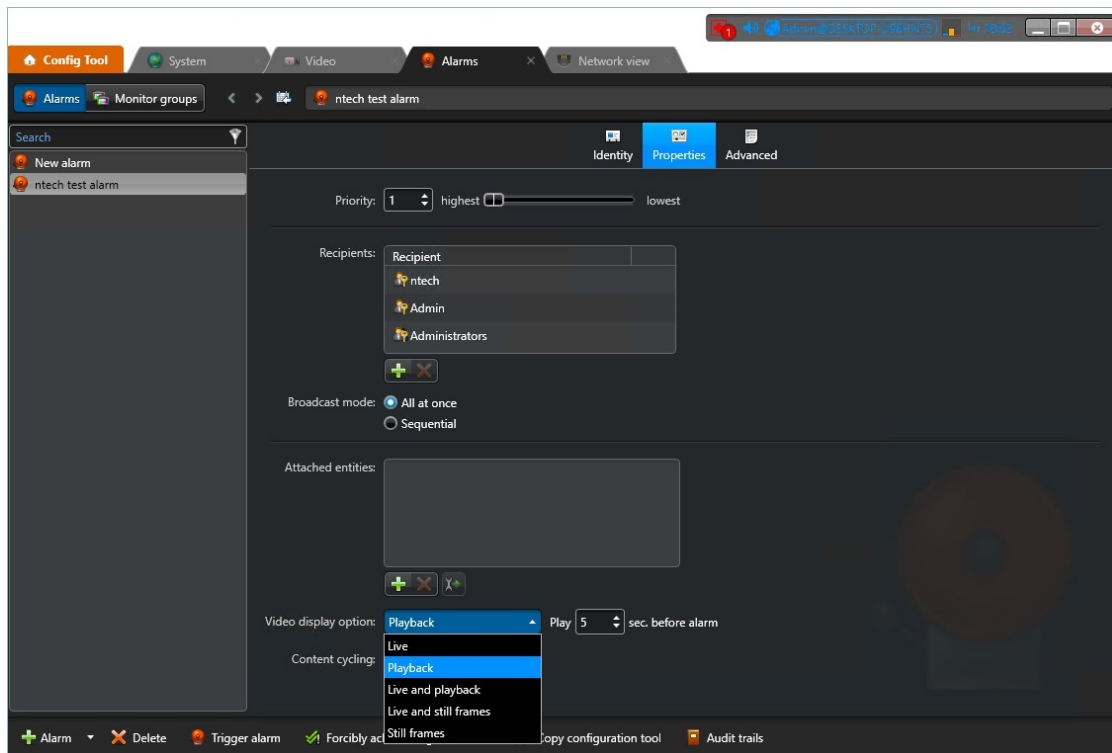
Tip: On the *Properties* tab, select the *Video display option* that suits your needs the best. Available options are *Live*, *Playback*, etc.

Tip: To enable alarm procedures and auto rotation of video right within the alarm pop-up window, enable *Content cycling*.

When configuring the integration in FindFace Security, you will have to enter the alarm logical id that is specified on the *Identity* tab.
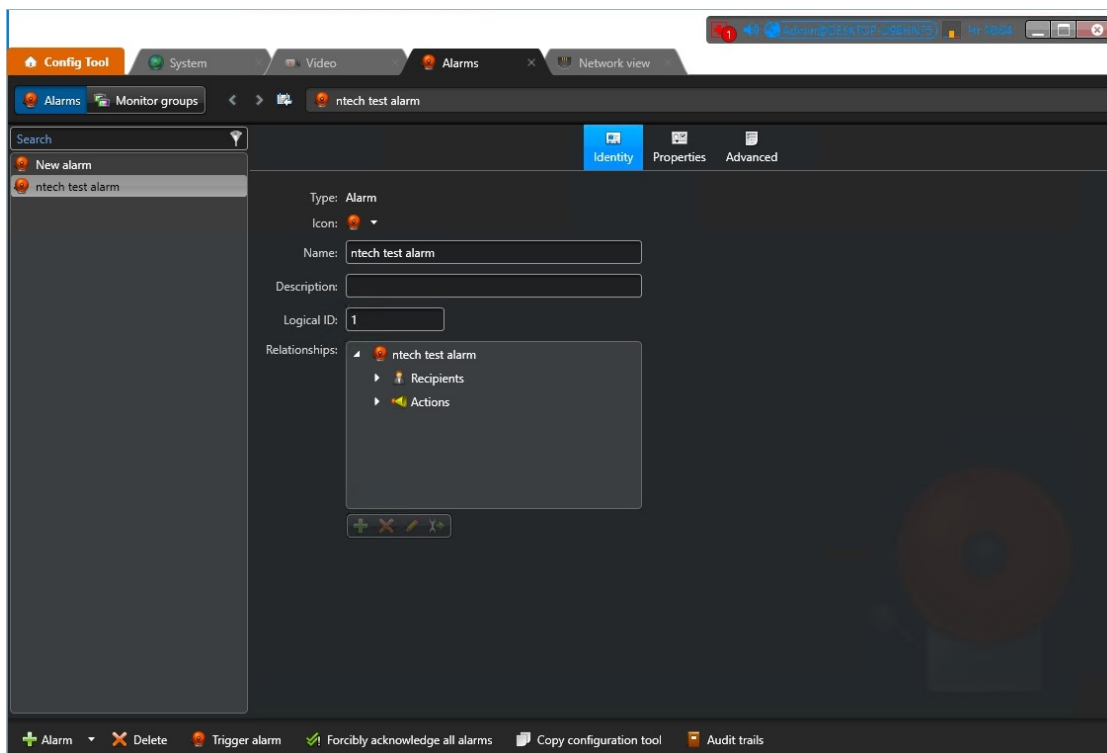
Configure Endpoints in FindFace Security

To establish connection between FindFace Security and Genetec Security Center, do the following:

1. Navigate to the *Preferences* tab. Click *Genetec*.

2. In the *Server* and *Media* sections, specify *settings* of the Web SDK and Media Gateway endpoints.

Important: The ports for the WebSDK and Media Gateway need to be open.

---

3. In the *Ids* section, specify the *logical id* of the `Alarm` entity that will be triggered in Genetec Security Center when a face recognition event occurs in FindFace Security.

4. Click *Save*. If the connection to Genetec Security Center is successfully established, you will see the *State* change to *Configured*.

Import Cameras from Genetec Security Center

Once the connection to Genetec Security Center is established, import cameras. To do so, click *Cameras* on the *Genetec* tab. Click *Import*.

This action will create a *group of cameras* `Genetec` listing all the cameras from Genetec Security Center.

To view this list of cameras, navigate to the *Cameras* tab on the FindFace Security navigation bar. If you want to exclude a camera from face recognition, simply deactivate it in the list.

Create Watch Lists and Dossiers in FindFace Security

After you have configured the endpoints and camera settings, finish the integration by creating a *dossier database*. Notifications about face recognition events will be automatically sent to Genetec Security Center. See *Notifications in Genetec Security Center*.

Notifications in Genetec Security Center

Each face recognition event from a Genetec camera, that has a match with a dossier, triggers a relevant alarm in Genetec Security Center. Every alarm triggered by FindFace Security is associated with a relevant camera (source of the face recognition event) so you can instantly watch live or playback video within the `Alarm Monitoring` task in Genetec Security Desk. FindFace Security also utilizes `Alarm Procedures` to provide a user with additional content related to the alarm, such as:

- face detected in video

- matching face from the dossier database

- person's name and comment from the dossier

- matching confidence

- watch list's name

- full frame
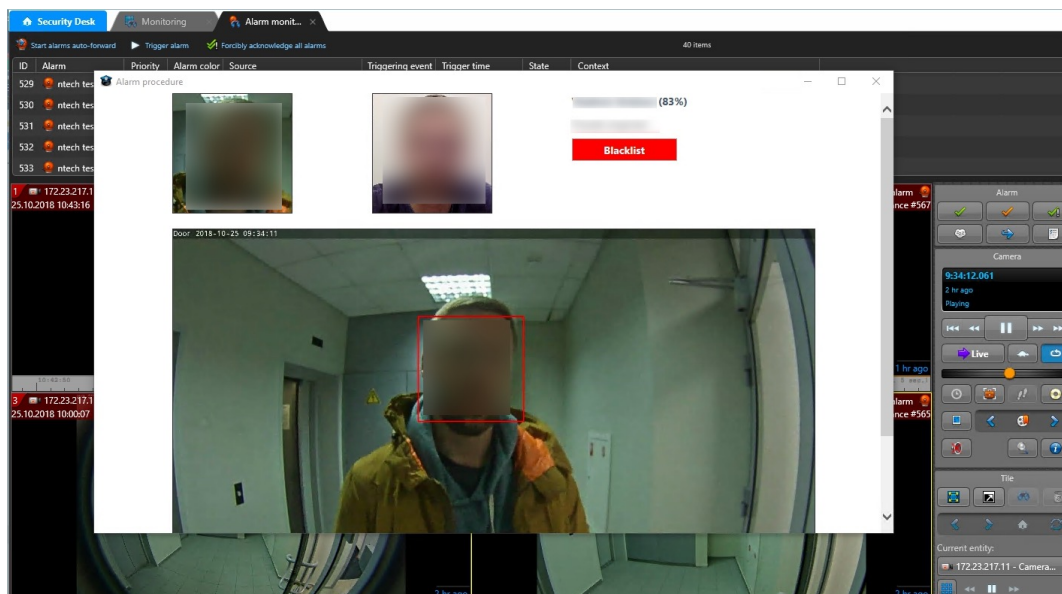
After you receive a face recognition alarm, process it as you usually do with other alarms in Genetec Security Center.

3.3.2 Axxon Next

FindFace Security integration with Axxon Next allows you to detect and identify faces in video streams from an Axxon-based security system.

Important: One FindFace Security instance supports interaction with only one Axxon Next server.

Integration with Axxon Next is implemented via the `ffsecurity_axxon` plugin.

To configure the FindFace Security integration with Axxon Next in Ubuntu, do the following:

1. Activate the plugin by appending the `INSTALLED_APPS.append('ffsecurity_axxon')` line to the `/etc/ffsecurity/config.py` configuration file.

```
sudo vi /etc/ffsecurity/config.py

...

# integration plugins
INSTALLED_APPS.append('ffsecurity_axxon') # remove or comment out this line to
↪disable
```

2. Add the `FFSECURITY->AXXON` section to the configuration file. Fill it out as shown in the example below. In the api parameter, specify the IP address of the Axxon Next server that will provide FindFace Security with Axxon API and HLS-archive streams. In the `rtsp` parameter, specify the common segment of Axxon video stream addresses.

```
 FFSECURITY = {
'AXXON': {
    'api': 'http://user:password@example.com/',
    'rtsp': 'rtsp://user:password@example.com:554/',
        }
}
```

3. (Optional). If facial recognition events are required to contain video from Axxon Next, edit the `FFSECURITY_UI_CONFIG` section as shown in the example below.

```
FFSECURITY_UI_CONFIG = {
 'dossier': {
    'video': True,
 }
}
```

4. Create representations of Axxon Next cameras in FindFace Security (see *Camera Management*). A camera representation URL must be specified in the format `axxon:<friendlyNameLong>`, where `friendlyNameLong` is a camera name on the Axxon Next server. Find out this name in the Axxon user interface, or via Axxon API by executing:

```
curl http://user:password@127.0.0.1/video-origins/

{
   "OLOLOE-DEV/DeviceIpint.vhod_1/SourceEndpoint.video:0:0" : {
      "friendlyNameLong" : "vhod_1.Vhod_1",
      "friendlyNameShort" : "Vhod_1",
      "origin" : "OLOLOE-DEV/DeviceIpint.vhod_1/SourceEndpoint.video:0:0",
      "state" : "signal_restored"
   }
}
```

For the camera from the example above, URL must be specified as `axxon:vhod_1.Vhod_1`.

The configuration is now finished. If the integration is properly configured, FindFace Security will be detecting and identifying faces in Axxon Next video streams, and facial recognition events will be featuring video clips from Axxon Next (upon relevant settings).

3.4 Plugins

In the course of configuring the system, you can set your own directives that determine how the system processes a face after it has been detected in the video. To do so, write a Python plugin(s).

Plugins are enabled through the `findface-facerouter` configuration file. They allow you to configure video face detection outcome individually for each use case.

3.4.1 Deploy `findface-facerouter` in FindFace Security

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

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

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

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

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

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

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

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

```
sudo vi /etc/ffsecurity/config.py

...
FFSECURITY = {
    'ROUTER_URL': 'http://172.20.77.58:18820/v0/frame?',
```

5. Open the `/etc/findface-video-manager.conf` configuration file. In the `router_url` parameter, specify the IP address and port of the `findface-facerouter` host to receive detected faces from `findface-video-worker`.

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

...
router_url: http://127.0.0.1:18820/v0/frame
```

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

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

7. Restart the `findface-security` service.

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

3.4.2 Configure `findface-facerouter` to Use Plugins

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

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

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

1. Put a plugin into a directory of your choice. You can distribute a set of plugins across several directories.

2. Open the `findface-facerouter` configuration file.

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

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

---

3. Uncomment the `plugins_dirs` parameter and specify the comma-separated list of plugin directories.

```
plugins_dirs                       = '/etc/findface/plugins/video, /etc/findface/
↪plugins/html'
```

4. Save the configuration file.

3.4.3 Basics

In this section:

- *Plugin Architecture*
- *The `preprocess` method*
- *The `process` method*
- *The `shutdown` method*

Plugin Architecture

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

The `activate` function has the following parameters:

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

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

1. `request.ctx.sfapi`: a set up `ntech.sfapi_client.Client` instance that can be invoked directly to process the result of video face detection (for example, to create a new gallery, add a face to a gallery, etc.).

2. `plugins`: OrderedDict with all the plugins as (`key`: plugin name, `value`: the result returned by the `activate` function).

3. `idgen`: id generator that can be invoked as `ctx.idgen()`.

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

1. `preprocess`,

2. `process`,

3. `shutdown` (optional).

The `preprocess` method

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

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

> **preprocess**(*self, request: FrHTTPRequest, labels: typing.Mapping[str, str]*) → typing.Tuple[str]
>
> > **Parameters**
> >
> > - **FrHTTPRequest** (*tornado.httpserver.HTTPRequest*) – a HTTP API request that includes an extra argument `params`
> >
> > - **labels** (*dictionary*) – a custom set of a frame labels, which are initially specified in a job parameters for `findface-video-worker` and then assigned to the frame
> >
> > The `params` argument of FrHTTPRequest includes the following fields:
> >
> > **Parameters**
> >
> > - **photo** (*bytes*) – JPEG video frame featuring a detected face
> >
> > - **face0** (*bytes*) – normalized face image
> >
> > - **bbox** (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – coordinates of the face region in the video frame
> >
> > - **cam_id** (*string*) – camera id
> >
> > - **timestamp** (*datetime.datetime*) – video frame timestamp
> >
> > - **detectorParams** (*dictionary*) – debug information from the video face detector

- **bs_type** (*string*) – best face search mode. Available options: overall (the `findface-video-worker` posts only one snapshot per track, but of the highest quality.), realtime (the `findface-video-worker` posts the best snapshot within each of consecutive time intervals).

- **labels** (*dictionary*) – (duplicates params.labels) a custom set of a frame labels, which are specified in a job parameters for `findface-video-worker` and then assigned to the frame

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

The `process` method

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

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

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

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

The `shutdown` method

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

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

> **shutdown**(*self*)

3.4.4 Classes and Methods

In this section:

- *Basic Classes*
- *Object Classes*
- *Face Detection and Gallery Management*
- *Filters for Database Search*
- *Display Error Messages*

Basic Classes

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

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

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

> **Parameters**
>
> - **FrHTTPRequest** (`tornado.httpserver.HTTPRequest`) – a HTTP API request that includes an extra argument `params`
>
> - **labels** (`dictionary`) – a custom set of a frame labels from `request.params`
>
> **Returns** one or several strings `'facen'`, `'gender'`, `'age'`, `'emotions'`
>
> **Return type** tuple

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

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

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

Accepts the detected face features.

> **Parameters**
>
> - **request** (`tornado.httpserver.HTTPRequest`) – a HTTP API request from `findface-video-worker`
>
> - **photo** (*bytes*) – JPEG video frame featuring a detected face, from `request.params`
>
> - **bbox** (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – coordinates of the face region in the video frame, from `request.params`
>
> - **event_id** (*uint64*) – id of the face automatically set by `findface –facerouter` upon receiving it from `findface-video-worker`. Can be used as a face custom identifier in the biometric database.

- **detection** (`objects.DetectFace`) – detection result received from `findface-sf-api`, that contains requested face features such as faces, gender, age and emotions.

> **Returns** n/a

> **Return type** n/a

shutdown (*self*)

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

> **Parameters** n/a

> **Returns** n/a

Object Classes

class `objects.`**BBox**

Represents coordinates of the rectangle around a face.

class `objects.`**DetectFace**

Represents a detection result with the following fields:

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

class `objects.`**DetectResponse**

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

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

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

Represents a custom face identifier object in the gallery.

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

class `objects.`**Face**

Represents a result of database search by biometric sample

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

class `objects.`**ListResponse**

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

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

Face Detection and Gallery Management

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

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

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

Detects a face and returns the result of detection.

Parameters

- **url** (*URL*) – image URL if you pass an image that is publicly accessible on the internet

- **image** (*bytes*) – PNG/JPG/WEBP image file is you pass an image as a file

- **facen** (*boolean*) – extract a biometric sample from the detected face. To save the detection result in memcached pass facen=True

- **gender** (*boolean*) – extract and return information about gender

- **age** (*boolean*) – extract and return information about age

- **emotions** (*boolean*) – extract and return information about emotions

- **return_facen** (*boolean*) – return facen in the method result

- **autorotate** (*boolean*) – automatically rotate the image in 4 different orientations to detect faces in each of them. Overlapping detections with IOU > 0.5 will be merged

- **detector** (*boolean*) – nnd or normalized. The normalized detector is used to process normalized images, for example, those which are received from fkvideo_worker.

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

Returns Detection result

Return type DetectorResponse object.

gallery (*self*, *name*)

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

Parameters name (*string*) – gallery name

Returns a gallery object

Return type sfapi_client.Gallery

list_galleries(self, timeout=None):

Returns the list of galleries.

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

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

Return type list of GalleryListItem

class ntech.sfapi_client.gallery.**Gallery**

Provides methods to work with galleries and faces.

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

Returns a list-like object with faces from the gallery, that match the given filters. The returned list-like

object has an additional property `next_page` which can be used as a value for the `page` parameter in next requests.

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

add(*self, new_id: typing.Union[int, typing.Callable], source: typing.Union[DetectFace, Face, str], *, meta: typing.Dict[str, typing.Union[int, str, typing.List[str]]] = None, regenerate_attempts=None, timeout=None*) → Face
    Creates a face in the gallery.

> **Parameters**
>
> - **new_id** (`integer or callable`) – custom face identifier (Face ID) in the database gallery. May be a (async) callable which returns the id. To generate id, you can use the `ctx.idgen()` function delivered with the context.
>
> - **source** (`sfapi_client.DetectFace, sfapi_client.Face, sfapi_client.FaceId, or string`) – face source: create a face using another face in the database or a detection result as a source.
>
> - **meta** (`dictionary`) – face metadata. Keys must be strings and values must be either ints, strings or lists of strings. Metadata keys and types must be previously specified in the storage (`findface-tarantool-server`) configuration files.
>
> - **regenerate_attempts** – number of attempts to regenerate a unique Face ID with the `ctx.idgen()` function if new_id is callable
>
> - **timeout** (`number`) – FindFace core response timeout, in seconds (if `none`, the default value is used)
>
> **Returns** representation of the newly created face
>
> **Return type** Face object

delete(*self, face: typing.Union[Face, int], timeout=None*) → None
    Removes a face from the gallery.

> **Parameters**

- **face** (*sfapi_client.Face, sfapi_client.FaceId or id in integer*) – face to be removed

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

  Returns None

get (*self, face: typing.Union[Face, int], timeout=None*) → Face
   Retrieves a face from the gallery.

   Parameters

- **face** (*sfapi_client.Face, sfapi_client.FaceId or id in integer*) – face to be retrieved

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

  Returns representation of the face

  Return type Face object

create (*self, timeout=None*) → None
   Creates a gallery in `findface-sf-api` as a `sfapi_client.Gallery` object. Being a proxy object, `sfapi_client.Gallery` doesn't require a gallery to be existing on the server.

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

   Returns None

drop (*self, timeout=None*) → None:
   Removes a gallery from `findface-sf-api`.

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

   Returns None

update (*self, face: typing.Union[Face, str], *, meta: typing.Dict[str, typing.Union[int, str, typing.List[str]]] = None, timeout=None*) → Face
   Update face meta data in the gallery.

   Parameters

- **face** (*sfapi_client.Face, sfapi_client.FaceId or id in integer*) – face to be updated

- **meta** (*dictionary*) – face meta data to be updated. Keys must be strings and values must be either ints, strings or lists of strings. If a meta string is not passed or passed as null, it won't be updated in the database.

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

  Returns representation of the updated face

  Return type Face object

Filters for Database Search

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

---

serialize(*self*)

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

>> **Returns** filter names and filter values

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

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

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

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

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

>>> **Parameters value**(*integer*) – id value

>>> **Returns** filter name (LTE) and its value.

>>> **Return type** object of `Filter` class.

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

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

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

>>> **Parameters value**(*integer*) – id value

>>> **Returns** filter name (GTE) and its value.

>>> **Return type** object of `Filter` class.

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

> **classmethod oneof**(*cls, *value: typing.Union[int]*) → Filter

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

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

>>> **Returns** filter name (IN) and its value.

>>> **Return type** object of `Filter` class.

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

class ntech.sfapi_client.filters.**Meta**

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

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

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

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

>>> **Returns** filter name (LTE) and its value.

>>> **Return type** object of `Filter` class.

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

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

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

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

>>> **Returns** filter name (GTE) and its value.

>>> **Return type** object of `Filter` class.

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

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

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

> **Parameters value** (`list of strings or integers`) – list of metastring values
>
> **Returns** filter name (IN) and its value.
>
> **Return type** object of `Filter` class.

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

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

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

> **Parameters value** (`list of strings or integers`) – list of metastring values
>
> **Returns** filter name (SUBSET) and its value.
>
> **Return type** object of `Filter` class.

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

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

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

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

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

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

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

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

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

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

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

Several Filters Usage Example

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

Display Error Messages

class sfapi_client.**SFApiRemoteError**
> This error message appears if the error occurred for a reason other than a network failure.

The error body always includes at least two fields:

- code is a short string in CAPS_AND_UNDERSCORES, usable for automatic decoding.

- reason is a human-readable description of the error and should not be interpreted automatically.

Common Error Codes

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

class sfapi_client.**SFApiMalformedResponseError**
> This error message appears if the error occurred due to a network failure, or if Client was unable to read an API response from findface-sf-api.

3.4.5 Example

The following example illustrates the basics of writing a plugin, as well as the use of classes and methods. This plugin requests face features from findface-sf-api and then sends a request to <FFSEC_URL>/video-detector/process to create an event with the data obtained from findface-sf-api.

You can find this plugin at /opt/ffsecurity/fr_plugin/ffsec_fr_plugin.py. Embed it as described *here* and try it out.

Important: Make sure that the FFSEC_URL variable contains the actual IP address and port of the findface-security host.

```python
import datetime
import logging
import aiohttp
from dateutil.tz import tzutc
from facerouter.plugin import Plugin
```

(continues on next page)

```python
from ntech import sfapi_client
from ntech.asyncio_utils import wrap_future
from ntech.asyncio_utils.noop_cookie import NoopCookieJar
from ntech.tornado_utils import asyncio_to_tornado
# change this if your ffsecurity is located on another host or listens on a non-
→default port
FFSEC_URL = 'http://127.0.0.1:8002'
logger = logging.getLogger(__name__)
class FFSecurityPlugin(Plugin):
    def __init__(self, ctx, ffsec_url):
        super().__init__(ctx)
        self.ffsec_url = ffsec_url.rstrip('/')
        self.session = aiohttp.ClientSession(cookie_jar=NoopCookieJar())
        self.future_wrapper = asyncio_to_tornado
    def deactivate(self, *args):
        self.session.close()
    def request_headers(self, request):
        return {
            "Authorization": request.headers['Authorization'],
            'X-Request-ID': request.request_id,
        }
    @wrap_futures
    async def preprocess(self, request, labels):
        # somewhat hacky way to pass data between preprocess and process:
        request.ffsec_reception_timestamp = datetime.datetime.now(tzutc())
        headers = self.request_headers(request)
        async with self.session.post(self.ffsec_url + '/video-detector/preprocess',_
→headers=headers) as resp:
            resp.raise_for_status()
            resp_json = await resp.json()
            logger.debug("request_id=%r preprocess: ffsecurity response: %r", request.
→request_id, resp_json)
            plugin_wants = resp_json['plugin_wants']
            request.ffsec_plugin_wants = plugin_wants
            logger.info("request_id=%r preprocess: ffsecurity requested features: %r",
→ request.request_id, plugin_wants)
        return plugin_wants
    @wrap_futures
    async def process(self, request, photo, bbox, event_id, detection: sfapi_client.
→DetectFace):
        headers = self.request_headers(request)
        with aiohttp.MultipartWriter('form-data') as mpwriter:
            part = aiohttp.payload.BytesPayload(request.params.photo)
            part.set_content_disposition('form-data', name='photo', filename='photo.
→jpg')
            mpwriter.append(part)
            part = aiohttp.payload.BytesPayload(b'')
            part.set_content_disposition('form-data', name='face0', filename='norm.png
→')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload(request.params.detectorParams)
            part.set_content_disposition('form-data', name='detectorParams')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload([list(bbox)])
            part.set_content_disposition('form-data', name='bbox')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.params.cam_id)
```

```python
            part.set_content_disposition('form-data', name='cam_id')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.params.timestamp.isoformat())
            part.set_content_disposition('form-data', name='timestamp')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.ffsec_reception_timestamp.
→isoformat())
            part.set_content_disposition('form-data', name='reception_timestamp')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload(request.ffsec_plugin_wants)
            part.set_content_disposition('form-data', name='plugin_wants')
            mpwriter.append(part)
            if request.params.bs_type is not None:
                part = aiohttp.payload.StringPayload(request.params.bs_type)
                part.set_content_disposition('form-data', name='bs_type')
                mpwriter.append(part)
            part = aiohttp.payload.JsonPayload({
                'id': getattr(detection, 'id', None),
                'features': detection.features,
                'bbox': detection.bbox._asdict(),
                'facen': getattr(detection, 'facen', None),
                'attributes': detection.attributes,
            })
            part.set_content_disposition('form-data', name='detection')
            mpwriter.append(part)
            async with self.session.post(
                    self.ffsec_url + '/video-detector/process',
                    data=mpwriter,
                    headers=headers
            ) as resp:
                await resp.read()
                resp.raise_for_status()
        logger.info("request_id=%r process: ffsecurity accepted event", request.
→request_id)
async def activate(app, ctx, plugin_name, plugin_source):
    plugin = FFSecurityPlugin(ctx=ctx, ffsec_url=FFSEC_URL)
    return plugin
import datetime
import logging
import aiohttp
from dateutil.tz import tzutc
from facerouter.plugin import Plugin
from ntech import sfapi_client
from ntech.asyncio_utils import wrap_futures
from ntech.asyncio_utils.noop_cookie import NoopCookieJar
from ntech.tornado_utils import asyncio_to_tornado
# change this if your ffsecurity is located on another host or listens on a non-
→default port
FFSEC_URL = 'http://127.0.0.1:8002'
logger = logging.getLogger(__name__)
class FFSecurityPlugin(Plugin):
    def __init__(self, ctx, ffsec_url):
        super().__init__(ctx)
        self.ffsec_url = ffsec_url.rstrip('/')
        self.session = aiohttp.ClientSession(cookie_jar=NoopCookieJar())
        self.future_wrapper = asyncio_to_tornado
    def deactivate(self, *args):
```

```python
        self.session.close()
    def request_headers(self, request):
        return {
            "Authorization": request.headers['Authorization'],
            'X-Request-ID': request.request_id,
        }
    @wrap_futures
    async def preprocess(self, request, labels):
        # somewhat hacky way to pass data between preprocess and process:
        request.ffsec_reception_timestamp = datetime.datetime.now(tzutc())
        headers = self.request_headers(request)
        async with self.session.post(self.ffsec_url + '/video-detector/preprocess',␣
→headers=headers) as resp:
            resp.raise_for_status()
            resp_json = await resp.json()
            logger.debug("request_id=%r preprocess: ffsecurity response: %r", request.
→request_id, resp_json)
            plugin_wants = resp_json['plugin_wants']
            request.ffsec_plugin_wants = plugin_wants
            logger.info("request_id=%r preprocess: ffsecurity requested features: %r",
→ request.request_id, plugin_wants)
        return plugin_wants
    @wrap_futures
    async def process(self, request, photo, bbox, event_id, detection: sfapi_client.
→DetectFace):
        headers = self.request_headers(request)
        with aiohttp.MultipartWriter('form-data') as mpwriter:
            part = aiohttp.payload.BytesPayload(request.params.photo)
            part.set_content_disposition('form-data', name='photo', filename='photo.
→jpg')
            mpwriter.append(part)
            part = aiohttp.payload.BytesPayload(b'')
            part.set_content_disposition('form-data', name='face0', filename='norm.png
→')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload(request.params.detectorParams)
            part.set_content_disposition('form-data', name='detectorParams')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload([list(bbox)])
            part.set_content_disposition('form-data', name='bbox')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.params.cam_id)
            part.set_content_disposition('form-data', name='cam_id')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.params.timestamp.isoformat())
            part.set_content_disposition('form-data', name='timestamp')
            mpwriter.append(part)
            part = aiohttp.payload.StringPayload(request.ffsec_reception_timestamp.
→isoformat())
            part.set_content_disposition('form-data', name='reception_timestamp')
            mpwriter.append(part)
            part = aiohttp.payload.JsonPayload(request.ffsec_plugin_wants)
            part.set_content_disposition('form-data', name='plugin_wants')
            mpwriter.append(part)
            if request.params.bs_type is not None:
                part = aiohttp.payload.StringPayload(request.params.bs_type)
                part.set_content_disposition('form-data', name='bs_type')
```

```
                mpwriter.append(part)
            part = aiohttp.payload.JsonPayload({
                'id': getattr(detection, 'id', None),
                'features': detection.features,
                'bbox': detection.bbox._asdict(),
                'facen': getattr(detection, 'facen', None),
                'attributes': detection.attributes,
            })
            part.set_content_disposition('form-data', name='detection')
            mpwriter.append(part)
            async with self.session.post(
                    self.ffsec_url + '/video-detector/process',
                    data=mpwriter,
                    headers=headers
            ) as resp:
                await resp.read()
                resp.raise_for_status()
        logger.info("request_id=%r process: ffsecurity accepted event", request.
→request_id)
async def activate(app, ctx, plugin_name, plugin_source):
    plugin = FFSecurityPlugin(ctx=ctx, ffsec_url=FFSEC_URL)
    return plugin
```

Python Module Index

f

n

o

Index