
FindFace Enterprise Server SDK

Release 2.6

NtechLab

Jun 15, 2023

Contents:

1	Get Started	3
2	System Requirements	5
2.1	General Requirements	5
2.2	Video Face Detection	7
2.3	FindFace Web User Interface	7
3	Choose Deployment Architecture	9
4	Deploy FindFace Server	13
4.1	Prerequisite Software	13
4.2	Install FindFace Server	15
4.3	Create Authentication Token	29
4.4	Test Requests	29
5	Video Face Detection	39
5.1	About Video Face Detection	39
5.2	Configure and Start Video Face Detection	41
5.3	Configuration Parameters	44
5.4	Render Detection Results	51
6	Increase Performance	53
6.1	Fast Index	53
7	FindFace Web User Interface	55
8	Advanced Features	63
8.1	Gender, Age and Emotions Recognition	63
8.2	Dynamic Person Creation	65
8.3	'Friend or Foe' Identification	72
8.4	Bulk Face Enrollment	74
8.5	Extraction API	77
8.6	Shard Galleries Statistics	83
8.7	Direct API Requests to Tarantool	85
8.8	Hacks for <code>tntapi</code>	90
9	Integration	93
9.1	REST API	94

9.2	Plugins	122
10	Maintenance and Troubleshooting	137
10.1	Troubleshoot Licensing and NTLS	137
10.2	Analyze Log Files	139
10.3	Migrate to Different Detector or Model	140
10.4	Update to The Latest Version	144
10.5	Remove Instance	145
10.6	Troubleshoot Uploads	146
10.7	Automatic Tarantool Recovery	147
11	Appendix	149
11.1	Neural Network Models	149
	Index	151

FindFace Enterprise Server SDK provides professional face recognition services based on neural networks. Implement these services to your ecosystem to take full advantage of them.

Features:

- Fast and robust face detection and database enrollment. Possibility of enrolling faces in bulk.
- Intelligent video face detection and analytics.
- Fast and accurate face identification and verification based on neural networks.
- Gender, age and emotions recognition.
- Dynamic person creation and ‘friend or foe’ identification.
- Almost infinite scalability due to integration with Tarantool.
- Truly RESTful API available in an embedded user friendly framework.
- Plugin support.
- Possibility of formatting API-responses.
- Highly intuitive web user interface.
- Network or on-premise licensing.

Tip: To read a release changelog, execute:

```
$ dpkg-parsechangelog -l /usr/share/doc/findface-repo/changelog.Debian.gz --all
```

FindFace Enterprise Server SDK will be of interest to independent software vendors (ISVs), system integrators, enterprise IT customers, and original equipment manufacturers (OEMs). It can be harnessed in numerous areas, such as retail, banking, social networking, entertainment, sports, event management, dating services, video surveillance, public safety, homeland security, etc.

Being integrated into specific solutions or Android applications, FindFace Enterprise Server SDK empowers businesses to accomplish such goals as biometric identification and access control, customer analytics, customer offer tailoring, offline retargeting, managing white and black lists, sorting and optimizing media libraries, borrower scoring, fraud prevention, employee productivity control, and many more.

This guide is intended for developers and system integration engineers who are going to integrate face recognition services into their systems. Prior to deploying a development environment, explore the [9 steps to face recognition](#). This will give you a general idea of the deployment process.

Let's get started!

CHAPTER 1

Get Started

Here you can see a typical biometric system based on FindFace Enterprise Server SDK:

FindFace Enterprise Server SDK consists of the **Biometric Data Analysis and Recognition Server** (**FindFace Server** or **Server** hereinafter) and, optionally, the video face detector and other *additional components*.

The FindFace Server functioning is provided by interaction of the following components:

Component	Description
findface-facenapi	Python daemon which runs HTTP API and provides the system functioning. It interfaces with MongoDB and extraction-api and tarantool@FindFace daemons.
tntapi (tarantool@FindFace as a shard)	Daemon which enables interaction with the face descriptors index (Tarantool database).
findface-extraction-api	Daemon which detects a face and extracts a feature vector (based on neural networks). Requires the packages with <i>models</i> <findface-data>.deb . Can also provide <i>advanced functions</i> .
MongoDB	Database which stores faces metadata, galleries details, settings, etc.
findface-upload	Nginx web server used to receive images using WebDAV.
NTLS	Local license server which interfaces with NtechLab Global License Server (for network licensing) or a USB dongle (for on-premise licensing) and passes a license to licensable components.

Follow the **9 steps** below to start delivering face recognition services to your customers:

1. *Choose deployment architecture*
2. *Prepare hardware*
3. *Install prerequisites*
4. *Install FindFace Server*
5. *Create a token and test the system work*

6. *Configure video face detection*
7. Increase performance by setting up *fast index*
8. *Add advanced features*
9. *Finalize the system with coding*

System Requirements

In this chapter:

- *General Requirements*
 - *Hosts*
 - *Supported Images*
- *Video Face Detection*
 - *Video Face Detector Host*
 - *Supported Video File Formats and Codecs*
- *FindFace Web User Interface*

2.1 General Requirements

2.1.1 Hosts

Prior to installing FindFace Enterprise Server SDK, ensure that the host(s) meet the following minimum requirements:

Note: Standalone installation of FindFace Enterprise Server SDK is recommended when the number of faces in the database **does not** exceed some 1,000,000. Otherwise you should install Findface Enterprise Server SDK in a cluster environment and configure *fast index* search. See *Choose Deployment Architecture* for details.

Re-requirement	Description
CPU	x86-64 CPU (Intel), >2.0 Ghz, >2 cores. The CPU AVX support is required for operation of all the components, except findface-upload.
RAM	RAM consumption depends on the number of faces in your dataset. Use the benchmark results below to calculate the memory size you need. Note that if there are 2 or more galleries with facens, you have to multiply the given MongoDB and Tarantool RAM consumption by the relevant number of galleries. As a rule, 10,000,000 faces require 20Gb RAM for Tarantool. MongoDB does not need much RAM as it uses HDD as RAM when needed.
HDD	10,000,000 faces require ~20x[number of snapshots for each shard] GB for Tarantool (by default 20x3=60 GB) and 24 GB for MongoDB. To store all uploaded images via findface-upload: size of all uploaded images + 10%
Operating system	Ubuntu 16.04 LTS (only x64)
Virtual machine support	VMware

Here you can see the FindFace Enterprise Server SDK memory usage benchmark results. Use these data to calculate the RAM size you need.

Note: Memory usage may slightly vary from test to test.

Note: Depending on your needs, [adjust](#) the Tarantool maximum memory usage at `/etc/tarantool/instances.enabled/FindFace.lua`.

Note: Starting with version 2.6, the `nnapi` component is deprecated, and `extraction-api` is used as a default facen extractor (as well as a default face detector).

The testing setup is the following:

- Facen [model](#): `apricot_320`
- Models for [gender, age and emotions recognition](#) (GAE in the table): `fr_1_gender0`, `fr_1_age0`, `emotion_1`
- Models used in [extraction-api](#): `apricot_320`, `fr_1_gender0`, `fr_1_age0`, `emotion_1`
- MongoDB, Tarantool: facens are stored in one gallery. If there are 2 or more galleries with facens, multiply the given RAM amount by the relevant number of galleries.

Number of faces	RAM consumption by components, MB				
	Mon-goDB	Taran-tool	nnapi	nnapi + GAE	extraction-api
0 (own needs)	~70	~77	~265	~1000	~1GB (1 Core)/~7GB (8 Cores) (up to 10,5 under load)
50,000	~181	~189	~400	~1400	
100,000	~294	~263	~400	~1400	
500,000	~1190	~1013	~400	~1400	
1,000,000	~2310	~1943	~400	~1400	

2.1.2 Supported Images

FindFace Enterprise Server SDK supports the following image formats:

- JPEG,
- PNG,
- WebP.

The maximum image size is 10 MB. The minimum distance between pupils is 40 px.

2.2 Video Face Detection

2.2.1 Video Face Detector Host

A host for the *video face detection* component must meet the following requirements (given that a video stream is 1 x 720p (1280×720) at 25FPS playback speed):

Note: Requirements depend on motion activity and the number of faces in video, the video face detector settings and FindFace Enterprise Server SDK overall load. To select an optimal configuration, contact our experts by info@ntechlab.com.

Requirement	Description
CPU	INTEL Core i5 6400 (2 physical core CPU). AVX support required.
RAM	4 GB in the real-time mode.
Operating system	Ubuntu 16.04 LTS (only x64).

2.2.2 Supported Video File Formats and Codecs

The `fkvideo_detector` component supports all video file formats and codecs that can be decoded by `FFmpeg`.

2.3 FindFace Web User Interface

To process video in the FindFace Enterprise Server SDK *web user interface*, its host should meet the same requirements as for the *video face detector*.

Choose Deployment Architecture

FindFace Enterprise Server SDK is delivered in the following distributable packages:

- A package with components **<findface-repo> .deb**.
- Several packages with neural network models **<findface-data> .deb**. Each model is delivered in a separate package.

Depending on your system characteristics and performance requirements, there are 2 FindFace Enterprise Server SDK deployments:

De- ploy- ment	Recommendation
Stan- dalone	You can deploy FindFace Enterprise Server SDK and neural network models on a single host (standalone) if the number of faces in the database does not exceed some 1,000,000. This variant makes it easier to start deployment and cater to basic requirements of your applications. Standalone deployment can be done <i>step-by-step</i> , from a <i>developer-friendly installer</i> , or as a fully pre-configured <i>virtual machine image</i> .
Clus- ter	If the number of faces in the database does exceed 1,000,000, deploy FindFace Enterprise Server SDK in a cluster environment and configure fast index search. In this case, FindFace Enterprise Server SDK components will be distributed across several hosts. This is a medium and large deployment which can be scaled almost infinitely. It will also suit professional high load projects with severe requirements to performance. Cluster deployment can be only done <i>step-by-step</i> .

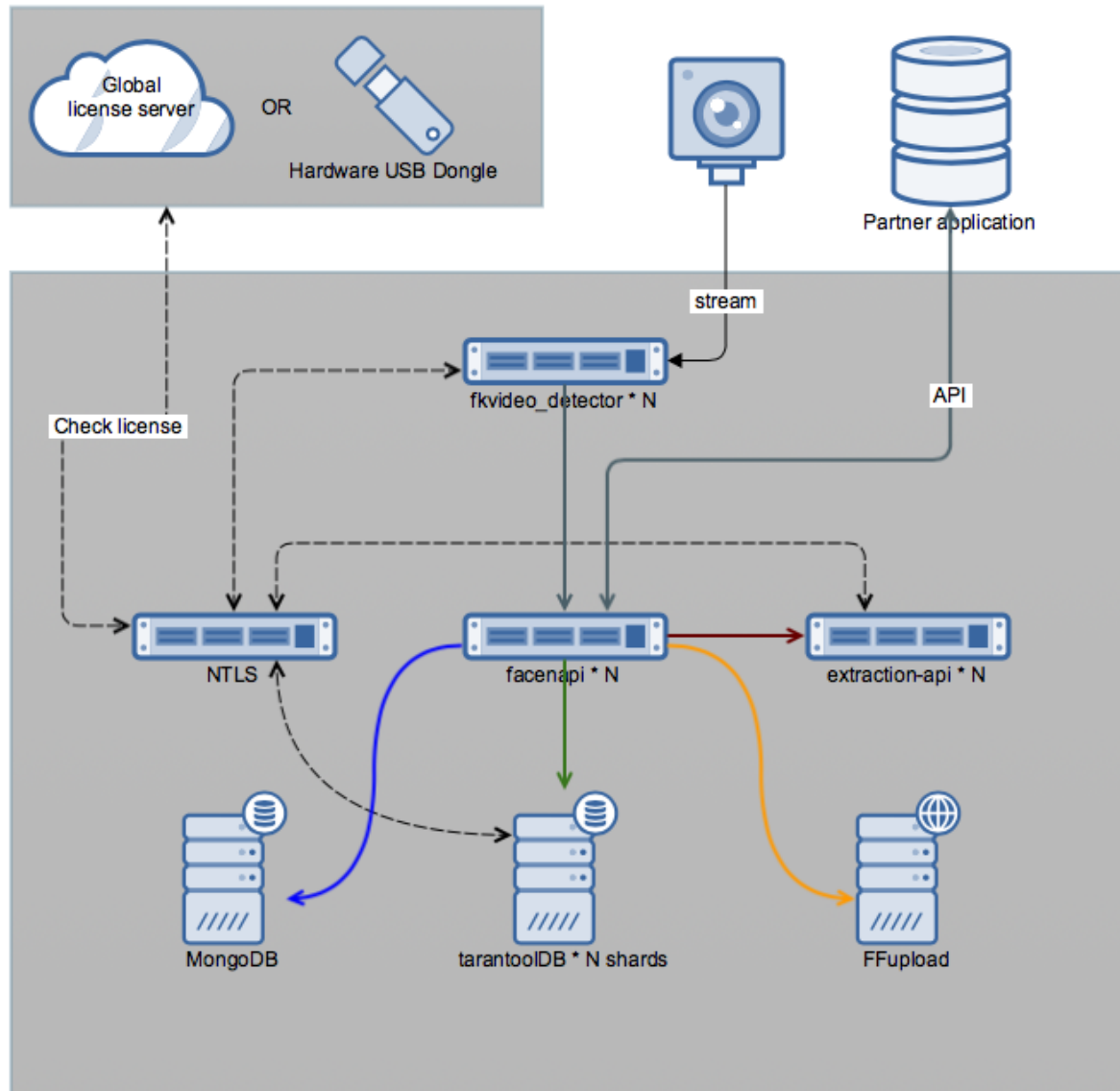
The FindFace Enterprise Server SDK basic configuration (FindFace Server) includes the following components:

Component	Description
findface-facenapi	Python daemon which runs HTTP API and provides the system functioning. It interfaces with MongoDB and extraction-api and tarantool@FindFace daemons.
tntapi (tarantool@FindFace as a shard)	Daemon which enables interaction with the face descriptors index (Tarantool database).
findface-extraction-api	Daemon which detects a face and extracts a feature vector (based on neural networks). Requires the packages with <i>models</i> <findface-data>.deb. Can also provide <i>advanced functions</i> .
MongoDB	Database which stores faces metadata, galleries details, settings, etc.
findface-upload	Nginx web server used to receive images using WebDAV.
NTLS	Local license server which interfaces with NtechLab Global License Server (for network licensing) or a USB dongle (for on-premise licensing) and passes a license to licensable components.

In addition to FindFace Server, you can also harness advanced features provided by the following components from the <findface-repo>.deb package:

Component	Description
fkvideo-detect	The video face detection component <i>fkvideo_detector</i> extracts faces from a RTSP camera stream or a video file on-the-fly and sends them via REST API to findface-facenapi for further processing. Licensable.
findface-extraction-api	By default, the <i>findface-extraction-api</i> component is used as an extractor of the face feature vector. The component also provides several <i>advanced features</i> , for example, flexible configuration of the API response format. Use this feature to extract various face data, including the bounding box coordinates, normalized face, gender, age, and emotions. Implementing this feature to your system can remarkably broaden the scope of analytic tasks it is capable of fulfilling. Licensable.
findface-mass-enroll	The <i>findface-mass-enroll</i> component allows for enrolling faces to findface-facenapi from images in bulk.
findface-ui	A <i>web user interface</i> which generally duplicates the functionality available via REST API. To be installed on the findface-facenapi host.
findface-tarantool-build-index	The <i>findface-tarantool-build-index</i> component creates a fast index for galleries with the number of faces over 1,000,000.

A typical FindFace Enterprise Server SDK architecture is shown in the diagram below.



Deploy FindFace Server

4.1 Prerequisite Software

In order to run successfully, FindFace Server needs the following software:

Prerequisite software	Usage	Installation
MongoDB	Internal database that provides proper functioning of FindFace Server. Stores faces metadata, galleries details, settings and other internal data.	Manually, before installing the FindFace Server components
Tarantool	Stores faces vector data.	Automatically along with the tntapi component.

In this section:

- *MongoDB*
 - *Install MongoDB on Application Host*
 - *Install MongoDB on Dedicated Host*
 - *Connect to Existing MongoDB*
- *Tarantool*

4.1.1 MongoDB

Prior to installing FindFace Server, set up a database for it. You may install MongoDB either on the application host where the findface-facenapi component resides, or on a dedicated host. For the standalone architecture, we recommend

you the first option. FindFace Enterprise Server SDK is compatible with [MongoDB 3.2](#) or later.

Install MongoDB on Application Host

To install the latest stable version of MongoDB (3.4 at the moment) on the application host, do the following:

Note: To install a different version of MongoDB, please refer to that version's documentation. For example, see [version 3.2](#).

1. Import the public key used by the package management system:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv_
↪0C49F3730359A14518585931BC711F9BA15703C6
```

2. Create a list file (/etc/apt/sources.list.d/mongodb-org-3.4.list) for MongoDB:

```
echo "deb [ arch=amd64,arm64 ] http://repo.mongodb.org/apt/ubuntu xenial/mongodb-
↪org/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
```

3. Reload the local package database:

```
sudo apt-get update
```

4. Install the latest stable version of MongoDB:

```
sudo apt-get install -y mongodb-org
```

5. Start the mongod service:

```
sudo service mongod start
```

Install MongoDB on Dedicated Host

To install MongoDB on a dedicated host, do the following:

1. On the dedicated host, install MongoDB in the same manner as on the *application host*.
2. Open the MongoDB configuration file:

```
sudo vi /etc/mongod.conf
```

3. To allow for incoming connections, comment out the line `bind_ip = 127.0.0.1`. This will allow MongoDB to accept connections from any IP address. Ensure that the only access to the host is from the LAN:

```
#bind_ip = 127.0.0.1
```

4. Restart the mongod service:

```
sudo service mongod restart
```

Connect to Existing MongoDB

If you wish to establish connection to an existing MongoDB instance, specify its IP address in the *network settings*.

4.1.2 Tarantool

FindFace Enterprise Server SDK is compatible only with Tarantool 1.7.3.673.g23cc4dc-1. This version is automatically installed along with the `tntapi` component.

4.2 Install FindFace Server

Several types of installation that we offer will surely make your user experience great. Choose the most convenient one, given your architecture outline:

- The cluster deployment can only be done *step-by-step*.
- The standalone deployment can be done *step-by-step*, from a developer-friendly *console installer*, and as a fully pre-configured *virtual machine image*.

Warning: For highload projects, installation as a virtual machine is not recommended even in test mode.

4.2.1 Install Step-By-Step

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

Tip: Standalone installation can also be done from a developer-friendly *console installer* or as a fully pre-configured *virtual machine image*.

In this section:

- *Prepare Packages for Installation*
- *Licensing*
- *Install Components*
 - *Install facenapi*
 - *Install extraction-api*
 - *Install findface-upload*
 - *Install tntapi*
 - * *Install tntapi standalone*
 - * *Install tntapi cluster*
- *Configure Network*

Prepare Packages for Installation

FindFace Enterprise Server SDK can be installed from a local repository. You can receive the FindFace Enterprise Server SDK distributable packages from your NTechLab manager. To prepare the packages for installation, do the following:

Warning: The `ntech` user will be automatically created at this stage. To avoid a conflict, make sure that such a user does not already exist in the system.

1. Unpack the package with components on each designated host.

```
sudo dpkg -i <findface-repo>.deb
```

2. Add a signature key on each designated host.

```
sudo apt-key add /var/findface-repo/public.key
sudo apt-get update
```

3. Unpack the packages with *models* (face, gender, age, and emotions). In the cluster environment, models are installed only on the `extraction-api` hosts.

```
sudo dpkg -i findface-data*
```

Licensing

You receive a license file from your NTechLab manager along with the FindFace Enterprise Server SDK distributable packages. If you opt for on-premise licensing, we will also send you a Guardant USB dongle. The licensing scheme for FindFace Enterprise Server SDK is shown on the diagram below.

To provide the FindFace Enterprise Server SDK licensing, follow the steps below:

1. Install and configure the local license server **NTLS**.
2. If the licensable components (`extraction-api`, `tntapi`, `fkvideo_detector`) are installed on remote hosts, specify the NTLS host IP address in their configuration files.

To install and configure NTLS, do the following:

1. Install the NTLS component:

```
sudo apt-get update
sudo apt-get install ntls
```

Tip: In the NTLS configuration file, you can change the license folder and specify your proxy server IP address if necessary. You can also change the NTLS web interface remote access settings. To open the NTLS configuration file, execute:

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

If necessary, change the license folder in the `license-dir` parameter. By default, license files are stored at `/ntech/license`:

```
license-dir = /ntech/license
```

If necessary, uncomment the `proxy` line and specify your proxy server IP address:

```
proxy = http://192.168.1.1:12345
```

By default, you can access the NTLS web interface from any remote host (`ui = 0.0.0.0:3185`). To indicate that accessing the NTLS web interface must originate from a specific IP address, edit the `ui` parameter:

```
ui = 127.0.0.1:3185
```

2. Enable the NTLS service autostart and launch the service:

```
sudo systemctl enable ntlis && sudo systemctl start ntlis
```

3. Upload the license file via the NTLS web interface `http://<NTLS_IP_address>:3185/#/`. You can also use the NTLS web interface to consult your license information, and upgrade or extend the license.

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

Important: If the licensable components (`extraction-api`, `tntapi`, `fkvideo_detector`) are to be installed on remote hosts, keep in mind that you have to specify the IP address of the NTLS host in their configuration files after installation.

See also:

[Troubleshoot Licensing and NTLS](#)

Install Components

Now that you have prepared the FindFace Enterprise Server SDK packages and provided the licensing, install the Server components on designated host(s) according to your architecture outline.

Install facenapi

Install and configure the `findface-facenapi` component as follows:

1. Install the component.

```
sudo apt-get update
sudo apt-get install python3-facenapi
```

2. If MongoDB is installed on a remote host, specify its IP address in the `findface-facenapi` configuration file.

```
sudo vi /etc/findface-facenapi.ini

mongo_host = '192.168.113.1'
```

3. Check if the component is runnable. To do so, invoke the `findface-facenapi` application by executing the command below. As the application is invoked, hold 1 minute, and if no errors display, hit `Ctrl+C`.

If MongoDB is installed on the same host, execute:

```
findface-facenapi
```

If MongoDB is installed on a remote host, execute:

```
sudo findface-facenapi --config=/etc/findface-facenapi.ini
```

4. Check if the findface-facenapi service autostart at system startup is disabled.

```
systemctl list-unit-files | grep findface-facenapi
```

5. Enable the service autostart and launch the service.

```
sudo systemctl enable findface-facenapi.service && sudo service findface-facenapi_↵  
↵start
```

6. Make sure that the service is up and running. The command will return a service description, a status (should be Active), path and running time.

```
sudo service findface-facenapi status
```

Tip: You can view the findface-facenapi *logs* by executing:

```
sudo tail -f /var/log/syslog | grep facenapi
```

Install extraction-api

Install and configure the findface-extraction-api component as follows:

Note: The extraction-api component requires the packages with *models* **<findface-data>.deb**. Make sure they have been installed.

1. Install the component.

```
sudo apt-get update  
sudo apt-get install findface-extraction-api
```

2. Open the findface-extraction-api.ini configuration file.

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

3. If *NTLS* is remote, specify its IP address.

```
license_ntls_server: 192.168.113.2:3133
```

4. The model_instances parameter indicates how many extraction-api instances are used. Specify the number of instances that you purchased. The default value (0) means that this number is equal to the number of CPU cores.

Note: This parameter severely affects RAM consumption.

```
model_instances: 2
```

5. Enable the Extraction API service autostart and launch the service.

```
sudo systemctl enable findface-extraction-api && sudo systemctl start findface-
↪extraction-api
```

6. Make sure that the service is up and running. The command will return a service description, a status (should be Active), path and running time.

```
sudo service findface-extraction-api status
```

Tip: You can view the extraction-api *logs* by executing:

```
sudo tail -f /var/log/syslog | grep extraction-api
```

Install findface-upload

To store all original images which have been processed by Server, as well as such Server artifacts as face thumbnails and normalized images, install and configure the `findface-upload` component.

Tip: Skip this procedure if you do not want to store these data on the FindFace Enterprise Server SDK host. In this case, only face features vectors (facens) will be stored (in the MongoDB and Tarantool databases).

Do the following:

1. Install the component:

```
sudo apt-get update
sudo apt-get install findface-upload
```

2. By default the original images, thumbnails and normalized images are stored at `/var/lib/ffupload/uploads/`. You can view this folder content at `http://127.0.0.1:3333/uploads/` in your browser. Make sure that this address is available (you will see `Forbidden` if so).

```
curl -I http://127.0.0.1:3333/uploads/
##HTTP/1.1 403 Forbidden
```

Important: You will have to specify this address when *configuring network*.

See also:

Troubleshoot Uploads

Install tntapi

The `tntapi` component connects the Tarantool database and the `facenapi` component, transferring search results from the database to `facenapi` for further processing. To increase search speed, multiple `tntapi` 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. In the case of standalone deployment, you need only one shard (created by default), while in a cluster environment the number of shards has to be calculated depending on several parameters (see below).

Install tntapi standalone

Install and configure the `tntapi` component as follows:

1. Install `tntapi`. Tarantool will be installed automatically along with it.

```
sudo apt-get update
sudo apt-get install findface-tarantool-server
```

2. Disable the Tarantool example service autostart and stop the service.

```
sudo systemctl disable tarantool@example && sudo systemctl stop tarantool@example
```

3. For a small-scale project, the `tntapi` shard created by default (`tarantool@FindFace`) would suffice as 1 shard can handle up to 10,000,000 faces. Configuration settings of the default shard are defined in the file `/etc/tarantool/instances.enabled/FindFace.lua`. We strongly recommend you not to add or edit anything in this file, except the maximum memory usage (`memtx_memory`), the NTLS IP address required for the `tntapi` licensing, and the remote access setting. The maximum memory usage should be set in bytes, depending on the number of faces the shard handles, at the rate roughly 1280 byte per face.

Open the configuration file:

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

Edit the value due to the number of faces the shard handles. The value `1.2*1024*1024*1024` corresponds to 1,000,000 faces:

```
memtx_memory = 1.2 * 1024 * 1024 * 1024,
```

Specify the NTLS IP address if NTLS is remote:

```
FindFace.start("127.0.0.1", 8001, {license_ntls_server="192.168.113.2:3133"})
```

With standalone deployment, you can access Tarantool by default only locally (`127.0.0.1`). If you want to access Tarantool from a remote host, either specify the remote host IP address in the `FindFace.start` section, or change `127.0.0.1` to `0.0.0.0` there to allow access to Tarantool from any IP address. In the case-study below, you allow access only from `192.168.113.10`:

```
FindFace.start("192.168.113.10", 8001, {license_ntls_server="192.168.113.2:3133"})
```

Now you allow access from any IP address:

```
FindFace.start("0.0.0.0", 8001, {license_ntls_server="192.168.113.2:3133"})
```

4. Configure the `tntapi` shard to autostart and start the shard.

```
sudo systemctl enable tarantool@FindFace && sudo systemctl start_
↪tarantool@FindFace
```

5. Retrieve the shard status. The command will return a service description, a status (should be Active), path and running time.

```
sudo systemctl status tarantool@FindFace
```

Tip: You can view the `tntapi` *logs* by executing:


```
sudo tail -f /var/log/tarantool/FindFace.log
```

6. The `tntapi.json` file containing the `tntapi` shard parameters is automatically installed along with `tntapi` into the `/etc/` folder.

Important: You will have to uncomment the path to `tntapi.json` when *configuring network*.

Install tntapi cluster

Install and configure the `tntapi` component as follows:

1. Install `tntapi` on designated hosts. Tarantool will be installed automatically along it.

```
sudo apt-get update
sudo apt-get install findface-tarantool-server
```

2. Create `tntapi` shards on each `tntapi` host. To learn how to shard, let's consider a case-study of a cluster environment containing 4 database hosts. We'll create 4 shards on each.

Important: When creating shards in large installations, observe the following rules:

1. 1 `tntapi` shard can handle approximately 10,000,000 faces.
2. The number of shards on a single host should not exceed the number of its physical processor cores minus 1.

3. Disable the Tarantool example service autostart and stop the service. Do so for all the 4 hosts.

```
sudo systemctl disable tarantool@example && sudo systemctl stop tarantool@example
```

4. Disable the shard created by default. Do so for all the 4 hosts.

```
sudo systemctl disable tarantool@FindFace
```

5. Write a bash script `shard.sh` that will automatically create configuration files for all shards on a particular host. Do so for the 4 hosts. Use the following script as a base for your own code. The exemplary script creates 4 shards listening to the ports: `tntapi 33001..33004` and `http 8001..8004`.

Important: The script below creates configuration files based on the default configuration settings stored in the file `/etc/tarantool/instances.enabled/FindFace.lua`. We strongly recommend you not to add or edit anything in the default file, except the maximum memory usage (`memtx_memory`) and the NTLS IP address required for the `tntapi` licensing. The maximum memory usage should be set in bytes for each shard, depending on the number of faces a shard handles, at the rate roughly 1280 byte per face.

Open the configuration file:

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

Edit the value due the number of faces a shard handles. The value `1.2*1024*1024*1024` corresponds to 1,000,000 faces:

```
memtx_memory = 1.2*1024*1024*1024,
```

Specify the NTLS IP address if NTLS is remote:

```
FindFace.start("127.0.0.1", 8001, {license_ntls_server="192.168.113.2:3133"})
```

```
#!/bin/sh
set -e

for I in `seq 1 4`; do
    TNT_PORT=$((33000+$I)) &&
    HTTP_PORT=$((8000+$I)) &&
    sed "
        s#/opt/ntech/var/lib/tarantool/default#/opt/ntech/var/lib/
        ↪tarantool/shard_${I}#g;
        s/listen = .*$/listen = '127.0.0.1:$TNT_PORT',/;
        s/\"127.0.0.1\", 8001,/\"0.0.0.0\", $HTTP_PORT,/;
        " /etc/tarantool/instances.enabled/FindFace.lua > /etc/tarantool/instances.
        ↪enabled/FindFace_shard_${I}.lua;

    mkdir -p /opt/ntech/var/lib/tarantool/shard_${I}/snapshots
    mkdir -p /opt/ntech/var/lib/tarantool/shard_${I}/xlogs
    mkdir -p /opt/ntech/var/lib/tarantool/shard_${I}/index
    chown -R tarantool:tarantool /opt/ntech/var/lib/tarantool/shard_${I}
    echo "Shard #${I} initied"
done;
```

Tip: Download the exemplary script.

6. Run the script from the home directory.

```
sudo sh ~/shard.sh
```

7. Check the configuration files created.

```
ls /etc/tarantool/instances.enabled/

##example.lua FindFace.lua FindFace_shard_1.lua FindFace_shard_2.lua FindFace_
↪shard_3.lua FindFace_shard_4.lua
```

8. Launch all the 4 shards. Do so on each host.

```
for I in `seq 1 4`; do sudo systemctl enable tarantool@FindFace_shard_${I}; done;
for I in `seq 1 4`; do sudo systemctl start tarantool@FindFace_shard_${I}; done;
```

9. Retrieve the shards status.

```
sudo systemctl status tarantool@FindFace*
```

You should get the following output:

```
tarantool@FindFace_shard_3.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset:
↪enabled)
```

(continues on next page)

(continued from previous page)

```
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
tarantool@FindFace_shard_2.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset:
↳enabled)
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
tarantool@FindFace_shard_1.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset:
↳enabled)
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
tarantool@FindFace_shard_4.service - Tarantool Database Server
Loaded: loaded (/lib/systemd/system/tarantool@.service; disabled; vendor preset:
↳enabled)
Active: active (running) since Tue 2017-01-10 16:22:07 MSK; 32s ago
...
```

Tip: You can view the `tntapi logs` by executing:

```
sudo tail -f /var/log/tarantool/FindFace_shard_{1,2,3,4}.log
```

10. On the `findface-facenapi` host, create a file `tntapi_cluster.json` containing the addresses and ports of all the shards. Distribute available shards evenly over ~1024 cells in one line. Click [here](#) to see the file for 4 hosts with 4 shards on each.

Tip: You can create `tntapi_cluster.json` as follows:

1. Create a file that lists all the shards, each shard with a new line (click [here](#) to view the example).

```
sudo vi s.txt
```

2. Run the script below (click [here](#) to view the script). As a result, a new file `tntapi_cluster.json` will be created and contain a list of all shards distributed evenly over 1024 cells.

```
cat s.txt | perl -lane 'push(@s,$_); END{$m=1024; $t=scalar @s;for(
↳$i=0;$i<$m;$i++){ $k=int ($i*$t/$m); push(@r,"\".$s[$k].\"") }
↳print "[".join(", ",@r)."]"; }' > tntapi_cluster.json
```

11. Move `tntapi_cluster.json` to the directory `/etc/`.

Important: You will have to uncomment and specify the path to `tntapi_cluster.json` when *configuring network*.

Configure Network

After you install the FindFace Server components, configure their interaction with each other. Do the following:

1. Open the `findface-facenapi.ini` configuration file:

```
sudo vi /etc/findface-facenapi.ini
```

2. Uncomment and/or edit the settings to align with your network specifications, substituting the suggested values with actual location:

```
ffupload_url = 'http://127.0.0.1:3333'  
mongo_host = '127.0.0.1'  
extraction_api_url = 'http://127.0.0.1:18088'  
tntapi_servers_file = '/etc/tntapi.json'
```

Warning: The `findface-facenapi.ini` content must be correct Python code.

Note: Do not specify `ffupload_url` if the `findface-upload` component is not installed.

3. By default, if one or several `tntapi` shards are out of service during face identification, **`findface-facenapi`** returns an error. If necessary, uncomment the `tntapi_ignore_search_error` parameter and assign it `True`. In this case `findface-facenapi` will use available `tntapi` shards to obtain face identification results, indicating the number of available servers vs the total number of servers in the response:

```
tntapi_ignore_search_errors = True
```

4. Restart all the FindFace Enterprise Server SDK services and `nginx` (for `findface-upload`) on the relevant host(s).

```
sudo service 'findface*' restart  
sudo service nginx restart
```

5. Check the services status. The command will return the services description, status (should be Active), path and running time.

```
sudo service 'findface*' status  
sudo service nginx status
```

4.2.2 Install from Console Installer

To install FindFace Enterprise Server SDK in a standalone configuration, you can use a developer-friendly console installer.

Warning: The installer cannot be used to update FindFace Enterprise Server SDK from version 2.3 or earlier.

See also:

- [Install Step-By-Step](#)
- [Install as Pre-Configured Virtual Machine](#)

Do the following:

1. Download the installer file `<findface-server-xxx>.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-server-xxx>.run
```

4. Execute the `.run` file.

Warning: The `ntech` user will be automatically created at this stage. To avoid a conflict, make sure that such a user does not already exist in the system.

```
sudo ./<findface-server-xxx>.run
```

The installer will perform several automated checks to ensure that the host meets the system requirements. After that, the FindFace Enterprise Server SDK components will be automatically installed, configured and/or started in the following configuration:

Component	Details
findface-facenapi	Installed and started with enabled and configured <i>dynamic person creation</i> and <i>“friend or foe” identification</i> .
findface-server-tarantool (tntapi)	Installed and started with the number of tntapi shards: $N = \min(\text{cores}, \text{RAM} / 2\text{Gb}) / 2$
findface-tarantool-build-index	Installed. Before use, consult the <i>component documentation</i> .
ffupload	Installed and started.
fkvideo_detector	Only installed. Use the command line or FindFace Web UI to manually start it. Before use, consult the <i>component documentation</i> .
findface-extraction-api	Installed and started as a face detector and facen extractor. Consult the <i>component documentation</i> for advanced features.
NTLS	Installed and started.
FindFace Web UI	Installed and started.
findface-mass-enroll	Only installed. Use the command line to work with it. Before use, consult the <i>component documentation</i> .
nginx	Installed and started.
MongoDB	Installed and started.
Tarantool Database	Installed and started.
jq	Installed. Used to pretty-print API responses from FindFace Server.

5. Once the installation is complete, the following output will be shown in the console:

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

```
#####
#           Installation is complete           #
#####
- upload your license to http://172.16.213.249:3185/
  login:      admin
  password:   fZh9-zZDX
- user interface: http://172.16.213.249:8000/
- token for UI:  fZh9-zZDX
- documentation: http://172.16.213.249:8000/v1/docs/v1/overview.html
Should you forget your password, recover it by executing
```

(continues on next page)

(continued from previous page)

```
findface-facenapi.token
user@ubuntu:~$
```

6. Upload the FindFace Enterprise Server SDK license file via the NTLS web interface `http://<Host_IP_address>:3185/#/`. To access the NTLS web interface, use the credentials provided in the console.

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.

4.2.3 Install as Pre-Configured Virtual Machine

You can deploy FindFace Enterprise Server SDK as a fully pre-configured ready-to-use virtual machine image that you can run inside a virtualization environment in any operating system. This type of installation is the simplest and requires minimum skills.

Important: This type of installation is suitable only for the *standalone deployment*.

Warning: For highload projects, installation as a virtual machine is not recommended even in test mode.

See also:

- [Install Step-By-Step](#)
- [Install from Console Installer](#)

Important: We officially support only **VMware** as a virtualization environment. Install it prior to proceeding with this instruction.

Tip: Contact your NtechLab manager by info@ntechlab.com to request the virtual machine image. You will be provided with files `ffserver-*.ovf` and `disk-*.vmdk` (discrete or in an archive).

The virtual machine image has the following software pre-installed:

- Ubuntu Server 16.04 LTS x64 with no graphical user interface
- FindFace Enterprise Server SDK in the following configuration:

Component	Details
findface-facenapi	Installed and started with enabled and configured <i>dynamic person creation</i> and <i>“friend or foe” identification</i> .
findface-server-tarantool (tntapi)	Installed and started (1 shard). Sharding may be required.
findface-tarantool-build-index	Installed. Before use, consult the <i>component documentation</i> .
ffupload	Installed and started.
fkvideo_detector	Only installed. Use the command line or FindFace Web UI to manually start it. Before use, consult the <i>component documentation</i> .
findface-extraction-api	Installed and started as a face detector and facen extractor. Consult the <i>component documentation</i> for advanced features.
NTLS	Installed and started.
FindFace Web UI	Installed and started.
findface-mass-enroll	Only installed. Use the command line to work with it. Before use, consult the <i>component documentation</i> .
nginx	Installed and started.
MongoDB	Installed and started.
Tarantool Database	Installed and started.
jq	Installed. Used to pretty-print API responses from FindFace Server.

To deploy FindFace Enterprise Server SDK as a virtual machine, do the following:

1. Put the `ffserver-*.ovf` and `disk-*.vmdk` virtual machine files into the same directory.
2. Start the virtualization environment. Click *Open a Virtual Machine* and select the `ffserver-*.ovf` file. If prompted, convert the file to a VMware format. This may take a while.
3. After the virtual machine is imported into the virtualization environment, navigate to the virtual machine hardware settings: *Edit virtual machine settings* → *Hardware*.

Tip: Refer to the VMware [official documentation](#).

- Choose the [network connection type](#), given the host networking.
- By default, the virtual machine hardware is already configured in the way that ensures optimal performance in most medium-load systems. Make sure it meets your project requirements as well. If you are going to simultaneously process several video streams, or maintain a large dataset, you may need to allocate additional resources to the virtual machine RAM and increase the number of CPU cores. Be sure to save the settings.

Important: You may also need to set up `tntapi` sharding later on the virtual machine console.

4. Power on the virtual machine. Wait until Ubuntu is finished starting.
5. To log in, enter the following credentials: login `user`, password `ntechlab`.
6. Find out the primary network interface IP address of the virtual machine (192.168.112.144 in the case study).

```
ifconfig
```

(continues on next page)

(continued from previous page)

```

ens33 Link encap:Ethernet HWaddr 00:0c:29:8f:db:d5
inet addr:192.168.112.144 Bcast:192.168.112.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe8f:dbd5/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:37751 errors:0 dropped:0 overruns:0 frame:0
TX packets:36205 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:5621377 (5.6 MB) TX bytes:39193951 (39.1 MB)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:152521 errors:0 dropped:0 overruns:0 frame:0
TX packets:152521 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:24549909 (24.5 MB) TX bytes:24549909 (24.5 MB)

```

7. Assign the primary network interface IP address to the `ffupload_url` parameter in the `findface-facenapi` configuration file.

```

sudo vi /etc/findface-facenapi.ini

ffupload_url = 'http://192.168.112.144:3333'

```

Warning: The `findface-facenapi.ini` content must be correct Python code.

8. Restart all the FindFace Enterprise Server SDK services.

```

sudo service 'findface*' restart

```

9. Make the virtual machine IP address static. To do so, 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.

Important: Be sure to edit the `etc/network/interfaces` file with extreme care. Please refer to the [Ubuntu guide on networking](#) before proceeding.

```

sudo vi /etc/network/interfaces

# The primary network interface
iface eth0 inet static
address 192.168.112.144
netmask 255.255.255.0
gateway 192.168.112.254
dns-nameservers 192.168.112.254

```

10. Restart networking.

```

sudo service networking restart

```

11. Upload the FindFace Enterprise Server SDK license file via the local license server web interface at `http://<IP_address>:3185/#/` (`http://192.168.112.144:3185/#/` in our example).

12. Create an *authentication token*. Use it to access the *FindFace Web Interface* at `http://<IP_address>:8000/`.

4.3 Create Authentication Token

Once the FindFace Server components installed, create a token in the long or short format, depending on your preference. Either format, this token will be valid to authenticate your FindFace Enterprise Server SDK instance in API requests.

To create a long token, execute:

```
findface-facenapi.token
##0123456789_abcdefghijklmnopqrstuvw
```

To create a short token, execute:

```
findface-facenapi.token --short
##A0B1-C2D3
```

If MongoDB is installed on a remote host, you have to indicate the path to the `findface-facenapi.ini` configuration file in the token generation command.

```
sudo findface-facenapi.token --config=/etc/findface-facenapi.ini
```

Note: Use the authentication token to access the FindFace Web User Interface at `http://<facenapi_ip>:8000/#/`. See *FindFace Web User Interface* for details.

4.4 Test Requests

Before you can proceed with development and deliver face recognition services to your customers, make sure that the FindFace Server components are working. To do so, run the test requests below, minding the sequence. To pretty-print responses, we recommend you to use `jq`.

Note: The request messages here are provided only for reference. To create valid requests out of the examples below, replace the token in the messages with the *actual* one.

Tip: To proceed with development, find more code samples (in C#, PHP, Java and Python) on our [GitHub](#).

In this section:

- [How to Pretty-Print Responses](#)
- [List Galleries](#)

- *Create New Gallery*
- *Detect Face in Image*
- *Add Face to Gallery*
- *Compare Face with Those from Gallery*
- *Compare Two Faces*
- *List Faces from Galleries*
- *Recognize Gender, Age and Emotions*

4.4.1 How to Pretty-Print Responses

Use **jq** to parse JSON data in responses. To install **jq**, execute:

```
sudo apt-get install jq
```

Note: Since **jq** approximates integers larger than 2^{53} (e.g., for `"id":12107867323949968228`, the output is `"id": 12107867323949967000`, etc.), you may want to use **json_pp** instead.

4.4.2 List Galleries

This request returns the name of the only gallery existing at the present moment. It is the `default` gallery. Relevant REST API method: */galleries GET*.

Request

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywY1lFoR2QkHXi-" http://localhost:8000/  
↪v0/galleries | jq
```

Response

```
{  
  "results": [  
    "default"  
  ]  
}
```

4.4.3 Create New Gallery

This request creates a new gallery `testgal`. Relevant REST API method: */galleries/new POST*.

Request

```
curl -H "Authorization: Token t3WGNhZbYaE_GFyQaywY1lFoR2QkHXi-" -X POST http://
↳localhost:8000/v0/galleries/testgal | jq
```

Response

```
{
  "name": "testgal"
}
```

4.4.4 Detect Face in Image

This request detects faces in a sample image on the Internet and returns coordinates of the rectangle around a detected face (*bbox*). Relevant REST API method: */detect POST*.

Request

```
curl -H "Authorization: Token t3WGNhZbYaE_GFyQaywY1lFoR2QkHXi-" -F "photo=http://
↳static.findface.pro/sample.jpg" http://localhost:8000/v0/detect | jq
```

Response

```
{
  "faces": [
    {
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ],
  "orientation": 1
}
```

4.4.5 Add Face to Gallery

This request processes the same sample image as in the previous request, detects a face and adds the detected face to the default gallery with a unique meta tag. Relevant REST API method: */face POST*.

Request

```
curl -H "Authorization: Token t3WGNhZbYaE_GFyQaywY1lFoR2QkHXi-" -F "photo=http://
↳static.findface.pro/sample.jpg" -F "meta=Sam Berry" http://localhost:8000/v0/face |
↳jq
```

Response

```
{
  "results": [
    {
      "friend": false,
      "galleries": [
        "default"
      ],
      "id": 3827229391220303,
      "meta": "Sam Berry",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/1495011480937809.
↪ jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950114809306293.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/149501148093593.jpeg
↪ ",
      "timestamp": "2017-05-17T08:58:00.930572",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ]
}
```

The following request also adds a face to a gallery but this time the face is extracted from a local image, and the gallery is custom ('testgal').

Request

```
curl -H "Authorization: Token t3WGNhZbYaE_GFyQaywY1lFoR2QkHXi-" -F "photo=@sample.jpg
↪ " -F "meta=sample" -F "galleries=testgal" http://localhost:8000/v0/face | jq
```

Response

```
{
  "results": [
    {
      "friend": false,
      "galleries": [
        "default",
        "testgal"
      ],
      "id": 3827229578000564,
      "meta": "sample",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/14950115538997407.
↪ jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950115538939695.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950115538985784.
↪ jpeg",
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    "timestamp": "2017-05-17T08:59:13.893921",
    "x1": 595,
    "x2": 812,
    "y1": 127,
    "y2": 344
  }
]
}

```

4.4.6 Compare Face with Those from Gallery

The following 2 requests process an image on the Internet (#1) and a local image (#2), detect a face and compare it with those from the default gallery. Return data of most similar faces and their similarity index. Relevant REST API method: */identify POST*.

Request #1

```

curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywY1lFoR2QkHXi-" -F "photo=http://
↪static.findface.pro/sample2.jpg" http://localhost:8000/v0/identify | jq

```

Response

```

{
  "results": {
    "[515, 121, 821, 427]": [
      {
        "confidence": 0.9373,
        "face": {
          "age": 26.0483455657959,
          "emotions": [
            "neutral",
            "sad"
          ],
          "friend": false,
          "galleries": [
            "default"
          ],
          "gender": "female",
          "id": 3827062458772442,
          "meta": "Sam Berry",
          "normalized": "http://192.168.113.88:3333/uploads//20170516/
↪1494946272949371.jpeg",
          "person_id": 5,
          "photo": "http://192.168.113.88:3333/uploads//20170516/14949462729435823.
↪jpeg",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.88:3333/uploads//20170516/
↪14949462729480093.jpeg",
          "timestamp": "2017-05-16T14:51:12.943000",
          "x1": 595,
          "x2": 812,

```

(continues on next page)

(continued from previous page)

```
        "y1": 127,
        "y2": 344
    }
}
]
```

Request #2

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywYllFoR2QkHXi-" -F "photo=@Pictures/
↪sample.jpg" http://localhost:8000/v0/identify | jq
```

Response

```
{
  "results": {
    "[595, 127, 812, 344]": [
      {
        "confidence": 0.9999,
        "face": {
          "age": 26.0483455657959,
          "emotions": [
            "neutral",
            "sad"
          ],
          "friend": false,
          "galleries": [
            "default"
          ],
          "gender": "female",
          "id": 3827062458772442,
          "meta": "Sam Berry",
          "normalized": "http://192.168.113.88:3333/uploads//20170516/
↪1494946272949371.jpeg",
          "person_id": 5,
          "photo": "http://192.168.113.88:3333/uploads//20170516/14949462729435823.
↪jpeg",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.88:3333/uploads//20170516/
↪14949462729480093.jpeg",
          "timestamp": "2017-05-16T14:51:12.943000",
          "x1": 595,
          "x2": 812,
          "y1": 127,
          "y2": 344
        }
      ]
    ]
  }
}
```

4.4.7 Compare Two Faces

This request compares a face in a local image and that on the Internet. Relevant REST API method: */verify POST*.

Request

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywY1lFoR2QkHXi-" -F "photo1=@Pictures/sample.jpg" -F "photo2=http://static.findface.pro/sample2.jpg" http://localhost:8000/v0/verify | jq
```

Response

```
{
  "results": [
    {
      "bbox1": {
        "x1": 595,
        "x2": 812,
        "y1": 127,
        "y2": 344
      },
      "bbox2": {
        "x1": 515,
        "x2": 821,
        "y1": 121,
        "y2": 427
      },
      "confidence": 0.9373794198036194,
      "verified": true
    }
  ],
  "verified": true
}
```

4.4.8 List Faces from Galleries

The following requests return the list of all faces stored in galleries, both default and custom (#1), and only custom (#2). Relevant REST API method: */faces GET*.

Request #1

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywY1lFoR2QkHXi-" http://localhost:8000/v0/faces | jq
```

Response

```
{
  "next_page": "/v0/faces?max_id=3827058103081960",
  "prev_page": null,
}
```

(continues on next page)

(continued from previous page)

```

"results": [
  {
    "friend": false,
    "galleries": [
      "default",
      "testgal"
    ],
    "id": 3827229578000564,
    "meta": "sample",
    "normalized": "http://192.168.113.88:3333/uploads//20170517/14950115538997407.
↪ jpeg",
    "person_id": 5,
    "photo": "http://192.168.113.88:3333/uploads//20170517/14950115538939695.jpeg",
    "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
    "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950115538985784.
↪ jpeg",
    "timestamp": "2017-05-17T08:59:13.893000",
    "x1": 595,
    "x2": 812,
    "y1": 127,
    "y2": 344
  },
  {
    "friend": false,
    "galleries": [
      "default"
    ],
    "id": 3827229391220303,
    "meta": "Sam Berry",
    "normalized": "http://192.168.113.88:3333/uploads//20170517/1495011480937809.
↪ jpeg",
    "person_id": 5,
    "photo": "http://192.168.113.88:3333/uploads//20170517/14950114809306293.jpeg",
    "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
    "thumbnail": "http://192.168.113.88:3333/uploads//20170517/149501148093593.jpeg
↪ ",
    "timestamp": "2017-05-17T08:58:00.930000",
    "x1": 595,
    "x2": 812,
    "y1": 127,
    "y2": 344
  },
  {
    "age": 26.0483455657959,
    "emotions": [
      "neutral",
      "sad"
    ],
    "friend": false,
    "galleries": [
      "default"
    ],
    "gender": "female",
    "id": 3827227793957831,
    "meta": "Sam Berry",
    "normalized": "http://192.168.113.88:3333/uploads//20170517/14950108570078573.
↪ jpeg",

```

(continues on next page)

(continued from previous page)

```

    "person_id": 5,
    "photo": "http://192.168.113.88:3333/uploads//20170517/14950108570022256.jpeg",
    "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
    "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950108570066717.
↪ jpeg",
    "timestamp": "2017-05-17T08:47:37.002000",
    "x1": 595,
    "x2": 812,
    "y1": 127,
    "y2": 344
  }
]
}

```

Request #2

```

curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywY1lFoR2QkHXi-" http://localhost:8000/
↪ v0/faces/gallery/testgal | jq

```

Response

```

{
  "next_page": "/v0/faces/gallery/testgal?max_id=3827059994026334",
  "prev_page": null,
  "results": [
    {
      "friend": false,
      "galleries": [
        "default",
        "testgal"
      ],
      "id": 3827229578000564,
      "meta": "sample",
      "normalized": "http://192.168.113.88:3333/uploads//20170517/14950115538997407.
↪ jpeg",
      "person_id": 5,
      "photo": "http://192.168.113.88:3333/uploads//20170517/14950115538939695.jpeg",
      "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
      "thumbnail": "http://192.168.113.88:3333/uploads//20170517/14950115538985784.
↪ jpeg",
      "timestamp": "2017-05-17T08:59:13.893000",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    },
    {
      "galleries": [
        "default",
        "testgal"
      ],
      "id": 3827059994026334,
      "meta": "sample",

```

(continues on next page)

(continued from previous page)

```
"normalized": "http://127.0.0.1:3333/uploads//20170516/14949453101653092.jpeg",
"photo": "http://127.0.0.1:3333/uploads//20170516/14949453101581762.jpeg",
"photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
"thumbnail": "http://127.0.0.1:3333/uploads//20170516/14949453101640306.jpeg",
"timestamp": "2017-05-16T14:35:10.158000",
"x1": 595,
"x2": 812,
"y1": 127,
"y2": 344
}
]
}
```

4.4.9 Recognize Gender, Age and Emotions

This request detects faces in a sample image on the internet and returns coordinates of the rectangle around a detected face (bbox) along with gender, age and emotions information. Relevant REST API method: */detect POST*. API version: v1.

Note: First, you need to *configure* gender, age and emotions recognition.

Request

```
curl -H "Authorization: Token t3WGNhZbyaE_GFyQaywY1lFoR2QkHXi-" -F 'photo=https://
↪static.findface.pro/sample2.jpg' -F 'gender=true' -F 'emotions=true' -F 'age=true'
↪http://localhost:8000/v1/detect | jq
```

Response

```
{
  "faces": [
    {
      "age": 29.057680130004883,
      "emotions": [
        "neutral",
        "happy"
      ],
      "gender": "female",
      "x1": 515,
      "x2": 821,
      "y1": 121,
      "y2": 427
    }
  ],
  "orientation": 1
}
```

5.1 About Video Face Detection

To add video face detection to your FindFace Server Enterprise SDK instance, you need the **fkvideo_detector** component. This component extracts faces from video and posts them to FindFace Server over API for further processing. It can work with both live streams and files, and supports all video file formats and codecs that can be decoded by FFmpeg.

In this section:

- *Installation*
- *How It Works*
 - *Motion Detection and Face Tracking*
 - *Best Face Search*
 - * *Real-Time Mode*
 - * *Offline Mode*
- *Configuration and Usage*
- *Video Stream Management*

5.1.1 Installation

Install `fkvideo_detector` from the `<findface-repo>.deb` package on one of the FindFace Server hosts or on a separate host:

Tip: Click [here](#) for the package preparation instruction.

```
sudo apt-get update
sudo apt-get install fkvideo-detector
```

5.1.2 How It Works

Motion Detection and Face Tracking

When processing video, `fkvideo_detector` consequently uses the following algorithms:

- **Motion detection.** This algorithm is aimed to reduce system resources consumption. Only when the motion detector recognizes motion of certain intensity in video that the face tracker can be triggered.
- **Face tracking.** The face tracker tracks, detects and captures faces from video, and posts them to FindFace Server. It can simultaneously process several faces.

Tip: Configure the maximum number of processed faces in the `fkvideo_detector` *configuration file*.

Each captured face is posted as a snapshot and a bbox in a request `/face` or `/identify`, depending on the *configuration settings*. If there are several active trackers, the face tracker sends the same number of requests with a unique snapshot and bbox in each.

Best Face Search

When tracking a face, the face tracker searches for its best snapshot before posting it to FindFace Server.

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

- Real-time
- Offline

Real-Time Mode

The real-time mode allows posting a face immediately after it appears in a 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 Server.
- 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 Server.
 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.

Offline Mode

The offline mode is less storage intensive than the real-time one as it allows posting only one snapshot per face, 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 Server.

5.1.3 Configuration and Usage

To configure `fkvideo_detector`, you can specify its options in any of the following ways:

- As command line arguments upon starting `fkvideo_detector`.

```
fkvideo_detector [options]
```

- As parameters in the `fkvideo_detector` configuration file.

Warning: The default `fkvideo_detector` configuration file is `/etc/fkvideo.ini`. Avoid editing `/etc/fkvideo.ini`, especially if `fkvideo_detector` and *FindFace Web UI* are running on the same host, as FindFace Web UI also uses this configuration file. Instead, make a copy of this file, edit the copy and specify it in the option `-c` when starting `fkvideo_detector`.

```
sudo cp /etc/fkvideo.ini /etc/fkvideo_example.ini

fkvideo_detector -c /etc/fkvideo_example.ini
```

See *Configuration Parameters* for the full option list.

5.1.4 Video Stream Management

You can specify video streams to be processed by `fkvideo_detector` as follows:

- A single stream can be specified directly by using the `--camid` and `--source` options when configuring `fkvideo_detector`.
- A list of streams has first to be posted to FindFace Server by applying the */camera POST* method to each stream. When posting, all streams in the list have to be assigned a common user-defined string, so called `detector`. This string should then be specified as the `--detector-name` option when configuring `fkvideo_detector`. In this case, `fkvideo_detector` will retrieve the list of streams from FindFace Server, based on their `detector-name`, and begin to process each stream individually. It will also be periodically updating the list of cameras from FindFace Server with a polling interval defined by the `reload-timeout` parameter.

5.2 Configure and Start Video Face Detection

This section will guide you through the `fkvideo_detector` deployment process. Follow the steps below minding the sequence.

Note: The `fkvideo_detector` component has to be *installed*.

In this section:

- *Specify Video Streams*
- *Start Component as Application*
- *Start Component as Service*

5.2.1 Specify Video Streams

To specify video streams for face detection, do the following:

1. Make a copy of the configuration file `/etc/fkvideo.ini`. Open the new file for editing.

```
sudo cp /etc/fkvideo.ini /etc/fk_local_config.ini
sudo vi /etc/fk_local_config.ini
```

2. If you have only one camera, you can add it in the new configuration file.

```
[General]
; Host settings
api-host=127.0.0.1
; Put your token here
api-token=RczGgVEMizRlnjHHQegNH_g9mwGl6-A1
api-port=8000

; Camera params
; If params doesn't set detector ask cameras list from server by key
; Key for receiving cameras list
;detector-name=detec1
; Camera ID
camid=local
; Stream path
; Example: rtsp:// - network stream; /dev/video0 - webcam; file@FPS:PATH - file_
↔with configurable FPS
source=rtsp://admin:qwert1234@192.168.104.199:554/Streaming/Channels/1
; Maximum cameras
detectors-max=20

; Motion detector scale coefficient for best performance
scale=0.3

; In realtime mode detector posts many frames wih increasing quality
; Else it sends only best frame
realtime=1

; URL that will receive frames
request-url=/v1/face/
; You can add custom head and body params to HTML POST request
head=
body-mf_selector=all,meta=User Meta
;

; Address of ntls server
license-ntls-server=127.0.0.1:3133
```

Tip: You can find an example of the configuration file [here](#).

3. If you have more than one camera, use the Server to store all your cameras. Add your camera to server by POST request `v1/camera`. For example, add camera to `detector=detec1`:

Request

```
curl -H 'Authorization: Token 1234567890qwertyuiop' -F "detector=detec1" -F
↪"url=rtsp://user:pass@192.168.1.1:554/Streaming/Channels/1" -F "meta=test"
↪http://localhost:8000/v1/camera
```

Response

```
{"detector": "detec1", "id": "0e663c00-b945-4676-bb0e-032c1dcf353a", "meta": "test"
↪, "url": "rtsp:// user:pass@192.168.1.1:554/Streaming/Channels/1"}
```

Now edit your configuration file. For example, detector will connect to server, and get all cameras with `detector=detec1`

```
[General]
; Host settings
api-host=127.0.0.1
; Put your token here
api-token=RczGgVEMizRlnjHHQegNH_g9mwGl6-A1
api-port=8000

; Camera params
; If params doesn't set detector ask cameras list from server by key
; Key for receiving cameras list
detector-name=detec1
; Camera ID
; camid=
; Stream path
; Example: rtsp:// - network stream; /dev/video0 - webcam; file@FPS:PATH - file
↪with configurable FPS
; source=
; Maximum cameras
detectors-max=20

; Motion detector scale coefficient for best performance
scale=0.3

; In realtime mode detector posts many frames wih increasing quality
; Else it sends only best frame
realtime=1

; URL that will receive frames
request-url=/v1/face/
; You can add custom head and body params to HTML POST request
head=
body-mf_selector=all,,meta=UserMeta
;
```

(continues on next page)

(continued from previous page)

```
; Address of ntls server  
license-ntls-server=127.0.0.1:3133
```

Tip: You can find an example of the configuration file [here](#).

5.2.2 Start Component as Application

To start `fkvideo_detector` as an application, execute:

```
fkvideo_detector -c /etc/fk_local_config.ini
```

Use this method for testing purposes.

5.2.3 Start Component as Service

To run the face detection component as a service, do the following:

1. Execute the following command:

```
sudo service fkvideo_detector@fk_local_config start
```

2. Check service status. The command will return a service description, a status should be active (running).

```
sudo service fkvideo_detector@fk_local_config status
```

Note: You can get the list of your cameras by the following request:

```
curl -H 'Authorization: Token 1234567890qwertyuiop' http://localhost:8000/v1/  
↪camera | jq
```

5.3 Configuration Parameters

To configure `fkvideo_detector`, you can specify its options in any of the following ways:

- As command line arguments upon starting `fkvideo_detector`.

```
fkvideo_detector [options]
```

- As parameters in the `fkvideo_detector` configuration file.

Warning: The default `fkvideo_detector` configuration file is `/etc/fkvideo.ini`. Avoid editing `/etc/fkvideo.ini`, especially if `fkvideo_detector` and *FindFace Web UI* are running on the same host, as *FindFace Web UI* also uses this configuration file. Instead, make a copy of this file, edit the copy and specify it in the option `-c` when starting `fkvideo_detector`.

```
sudo cp /etc/fkvideo.ini /etc/fkvideo_example.ini  
  
fkvideo_detector -c /etc/fkvideo_example.ini
```




In this section:

- *Command Line Arguments*
- *Configuration File Format*

5.3.1 Command Line Arguments

Usage:

```
fkvideo_detector [options]
```

Allowed options:

Warning: The following parameters are mandatory: `api-url`, `api-token`, `--license-ntls-server`.

Option	Description	Argument	Example
-c [--config] arg	Invoke fkvideo_detector with a given configuration file (.ini). The command line parameters and those in the configuration file have the same names and meaning, but if a parameter is set either way, the command line value has priority.	Path to the .ini configuration file. If you specify the file name alone (without the full path), fkvideo_detector will search for the file in the fkvideo_detector working directory. The default fkvideo_detector configuration file is /etc/fkvideo.ini. If fkvideo_detector and <i>Find-Face Web UI</i> are running on the same host, avoid editing /etc/fkvideo.ini as it is also used by FindFace Web UI. Instead, make a copy of this file, edit the copy and specify it in the option -c when starting fkvideo_detector.	\$ fkvideo_detector -c /etc/fkvideo_example.ini

Continued on next page

Table 1 – continued from previous page

Option	Description	Argument	Example
-S [--source] arg	Define a video stream as the relevant camera address (see also --camid). If a video stream is not specified, <code>fkvideo_detector</code> requests the <i>list of cameras</i> from FindFace Server with a polling interval defined by the <code>reload-timeout</code> parameter.	Camera address: <code>rtsp://... - network stream, /dev/video0 - webcam, file@FPS:PATH - file with configurable FPS.</code>	--source <code>rtsp://192.168.120.55:500</code>
--camid arg	Define a video stream as the relevant camera id (see also --source). If a video stream is not specified, <code>fkvideo_detector</code> requests the <i>list of cameras</i> from FindFace Server with a polling interval defined by the <code>reload-timeout</code> parameter.	Camera id.	--camid <code>b28a898b-6334</code>
--source-params arg	Define ffmpeg options for a video stream.	List of ffmpeg options with their values.	--source-params <code>rtsp_transport=tcp, rtsp_flags=prefer, timeout=-1</code>
--md-threshold arg	Define the minimum motion intensity to be detected by the motion detector. The threshold value is to be fitted empirically.	Motion intensity in 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.	--md-threshold <code>0.003</code>
--scale arg	Define a video frame scaling coefficient for the motion detector. 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 <code>min-face-size</code> value.	Video frame scaling coefficient.	--scale <code>0.3</code>
--rescale-height arg	Define a video frame height for the face tracker. 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 <code>min-face-size</code> value.	Video frame height in pixels.	--rescale-height <code>480</code>
--uc-max-time-diff arg	Define the maximum time period during which a number of similar faces are considered as belonging to one person.	Maximum time period in seconds.	--uc-max-time-diff <code>1</code>
--uc-max-dup arg	Define the maximum number of faces during the <code>uc-max-time-diff</code> period that is posted for a person.	Maximum number of faces.	--uc-max-dup <code>3</code>
--uc-max-avg-shift arg	Define the distance within which a number of similar faces are considered as belonging to one person.	Distance in pixels.	--uc-max-avg-shift <code>10</code>

Continued on next page

Table 1 – continued from previous page

Option	Description	Argument	Example
-r [--realtime] [=arg(=1)]	Enable the <i>real-time</i> mode of fkvideo_detector.	Mode of fkvideo_detector: 1 = real-time, 0 = off-line. -r and -r 1 are equal.	-r or -r 1, -r 0
--realtime1844 [=arg(=1)]	Simultaneously use the real-time and off-line modes of fkvideo_detector. In this case, fkvideo_detector will be sending bounding boxes to FindFace Server under 2 different labels.	Boolean: 1 = use both modes of fkvideo_detector, 0 = use the mode defined by the -r parameter. --realtime1844 and --realtime1844 1 are equal.	--realtime1844 or --realtime1844 1, --realtime1844 0
--start-ts=arg	By default, faces are posted to FindFace Server with a current time stamp. To change the time stamp, specify a different start time of a video stream processing. In the case of a video file processing, you have to specify the start date and time of a video file.	Start date and time of a video file, or a video stream processing.	--start-ts='2016-01-20 12:34:56'
--max-persons arg	Define the maximum number of faces simultaneously tracked by the face tracker. This parameter severely affects performance.	Maximum number of simultaneously tracked faces.	--max-persons 4
--disable-drops [=arg(=1)]	Enable posting all appropriate faces without drops. By default, if fkvideo_detector does not have enough resources to process all frames with faces, it drops some of them. If this option is active, fkvideo_detector puts odd frames on the waiting list to process them later.	Boolean: 1 = drops are disabled, 0 = drops are enabled. --disable-drops and --disable-drops 1 are equal.	--disable-drops
--tracker-threads arg	Define the number of tracking threads for the face tracker. This value should be less or equal to the max-persons 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.	Number of tracking threads	--tracker-threads 4
--sink-url arg	Only if fkvideo_detector processes 1 camera defined in the configuration file or in command line arguments. Defines the nginx video server IP address for the output video stream (it is there further redirected to <i>FindFace Web UI</i>).	Nginx video server IP address.	--sink-url 192.168.15.1:3222
--sink-res arg	Define the output video stream resolution.	Resolution WH	--sink-res 1280x720
--rotate arg	Rotate input video in a clockwise direction.	Rotation angle in degrees from 0° to 360°.	--rotate 90
--request-url arg	Define the request fkvideo_detector sends to FindFace Server when posting a face.	/v0/face/ or /v0/identify/.	--request-url /v0/identify

Continued on next page

Table 1 – continued from previous page

Option	Description	Argument	Example
--img-arg arg	Define the name of the argument containing a bbox with a face, in an API request.	Argument name (photo by default).	--img-arg picture
--headers arg	Create an additional header field in a POST request when posting a face.	Additional header field in a POST request.	--headers xxx = yyy --headers kkk = ppp
--body arg	Create additional body fields in the request body when posting a face.	Additional body field(s). You can specify several values for each field, separated by a comma.	--body gal- leries='testgal1,testgal2' --body gen- der=true --body age=true --body emo- tions=true --body meta=video.mp4
--bbox-scale	Define a bbox scaling coefficient.	Bbox scaling coefficient (1 by default).	--bbox-scale 1.3
--post-uniq arg	Enable posting only a certain number of faces belonging to one person, during a certain period of time. In this case, if fkvideo_detector posts a face to FindFace Server and then tracks another one within the time period <code>uc-max-time-diff</code> , and the distance between the two faces doesn't exceed <code>uc-max-avg-shift</code> , fkvideo_detector estimates their similarity. If the faces are similar and the total number of similar faces during the <code>uc-max-time-diff</code> period does not exceed the number <code>uc-max-dup</code> , fkvideo_detector posts the other face. Otherwise, the other face is not posted.	Boolean: 1 = only a certain number of faces belonging to one person are posted, 0 = all captured faces are posted.	--post-uniq 1
--min-score arg	Define the 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.	Minimum threshold value for the face quality in empirical units (negative rational numbers to zero). Milestones: 0 = high quality faces, -1 = good quality, -2 = satisfactory quality, -5 = face recognition maybe inefficient. The default value is -7.	--min-score -1.5

Continued on next page

Table 1 – continued from previous page

Option	Description	Argument	Example
--min-dir-score arg	Define the 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.	Maximum deviation of a face from its frontal position in 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. The default value is -1000.	--min-dir-score -1
--rt-delay arg	Only for the real-time mode. If <code>rt-perm=True</code> , defines the time period within which the face tracker picks up the best snapshot and posts it to FindFace Server. If <code>rt-perm=False</code> , defines the minimum time period between 2 posts of the same face with increased quality.	Time period in milliseconds.	--rt-delay 100
--rt-perm arg	Only for the realtime mode. Post best snapshots obtained within each <code>rt-delay</code> time period.	Boolean: 1 = post best snapshots obtained within each <code>rt-delay</code> time period, 0 = search for the best snapshot dynamically and send snapshots in order of increasing quality.	--rt-perm 1
--rot arg	Enable detecting and tracking faces only inside a clipping rectangle. You can use this option to reduce <code>fkvideo_detector</code> load.	Clipping rectangle: <code>WxH+X+Y</code> (see the specification of X geometry).	--rot 150x123+300+155
--roi arg	Enable posting faces detected only inside a region of interest.	Region of interest: <code>WxH+X+Y</code> (see the specification of X geometry).	--roi 123x122+159+220
--draw-track [=arg(=1)]	Enable drawing a face motion track in a bbox.	Boolean: 1 = enable drawing a track, 0 = disable drawing a track. <code>--draw-track</code> and <code>--draw-track 1</code> are equal.	--draw-track
--send-track [=arg(=1)]	Enable posting a face motion track as array of the bbox center coordinates, in a request.	Boolean: 1 = enable posting a track coordinates, 0 = disable posting a track coordinates. <code>--send-track</code> and <code>--send-track 1</code> are equal.	--send-track
--min-face-size arg	Define the minimum size of a face. Undersized faces are not posted.	Minimum size of a bbox minor side in pixels.	--min-face-size 50
--max-face-size arg	Define the maximum size of a face. Oversized faces are not posted.	Maximum size of a bbox major side in pixels.	--max-face-size 120
--only-norm arg	Enable posting only normalized face images without full frames.	Boolean: 1 = only normalized faces are posted to FindFace Server, 0 = full frames and normalized faces are posted.	--only-norm 1

Continued on next page

Table 1 – continued from previous page

Option	Description	Argument	Example
--license-ntls-server arg	Mandatory. Define the IP address and port of <i>NTLS</i> . Edit only if NTLS is remote.	NTLS IP address:port	--license-ntls-server 192.168.10.1:3133
--api-url arg	Mandatory. Define the host fkvideo_detector sends requests to (FindFace Server in particular).	IP address:port.	--api-host 127.0.0.1:8000
--api-token arg	Mandatory. Define the authentication token for FindFace Server.	<i>Authentication token.</i>	--api-token c9FsRNDAt
--to-verify-cert arg	Need to verify a https certificate.	Boolean: 1 = a https certificate verification is necessary, 0 = a self-signed certificate can be accepted.	--to-verify-cert 1
-n [--detector-name] arg	Apply fkvideo_detector to a given list of cameras.	Unique video detector identifier (hostname by default) which corresponds to a particular list of cameras stored on FindFace Server.	--detector-name detec1
-d [--detectors-max] arg	Define the maximum number of video streams to be processed by fkvideo_detector.	Maximum number of video streams simultaneously processed by fkvideo_detector (5 by default).	--detectors-max 7
-t [--reload-timeout] arg	Define the interval between 2 consecutive requests fkvideo_detector sends to FindFace Server to update the list of cameras.	Interval in seconds between 2 consecutive camera list updates (15 s by default).	-t 20
--camera-url arg	Define the request fkvideo_detector sends to FindFace Server to obtain the list of cameras.	/v0/camera (default) or /v1/camera.	--camera-url /v1/camera
--req-timeout arg	Define a timeout for a FindFace Server response to a fkvideo_detector API request.	API response timeout in seconds (3 s by default).	--req-timeout 2
--single-pass [=arg(=1)]	Disable periodical updates of the list of cameras. Use this option if fkvideo_detector should process a video file. In this case, fkvideo_detector will request the list of cameras only once.	Boolean: 1 = updates are disabled, 0 = updates are enabled. --single-pass and --single-pass 1 are equal.	--single-pass 0

Continued on next page

Table 1 – continued from previous page

Option	Description	Argument	Example
--remote-config [=arg(=1)]	Get the list of cameras from FindFace Server.	Boolean: 1 = the list of cameras is obtained from FindFace Server (default), 0 = fkvideo_detector processes a video stream defined in the --source and --camid parameters. The --source and --camid parameters have priority: if a video stream is specified, the --remote-config option is automatically set to 0. --remote-config and --remote-config 1 are equal.	--remote-config
-h [--help]	Produce the fkvideo_detector help message.	–	–

5.3.2 Configuration File Format

```
[General]
| long-arg=option ; long-arg from command line arguments
| ...

| api-url=127.0.0.1:8000
| api-token=1MDkdLnO6FF-PLfwmMuypTsM
| license-ntls-server=192.168.10.1:3133
| source-params=rtsp_transport=tcp,rtsp_flags=prefer,timeout=-1
| body=galleries=testgall\,testgal2,gender=true,age=true,emotions=true,meta=video.mp4
| start-ts = 2013-01-22 12:34:56
```

Note: You can specify several values for an additional body field (body), separated by a comma. In the configuration file, a comma between values of the same field must be preceded by a backslash (\,) to avoid a parsing error.

5.4 Render Detection Results

The fkvideo_detector component does not process FindFace Server responses to face identification and camera operation API requests. You should write your own proxy script that will manage communication between fkvideo_detector and FindFace Server and redirect API responses to an application that can process and render them. A typical rendering topology is shown on the diagram below:

When writing the proxy script, hold to the following logic:

1. A request from fkvideo_detector transparently goes to FindFace Server in the following format:

```
curl -X POST -H 'Authorization: Token ntech' -F "gender=true" -F "emotions=true" -
↪F "age=true" -F "cam_id=1b19a189-26b9-42e5-8cd8-6cabde79dc7e" -F
↪"timestamp=2017-08-25T13:09:54" -F "bbox=[[620,380,1383,1143]]" -F
↪"photo=@15036665986531599.jpeg" -F "face0=@15036665986766284_norm.png" -F
↪'detectorParams={"score": -0.000911839, "direction_score": -0.568228}' http://
↪192.168.104.184:8000/v1/face
```

(continues on next page)

(continued from previous page)

-
2. As FindFace Server replies to `fkvideo_detector`, your proxy script should redirect the response to your application for further processing.

Note: FindFace Server responses to requests sent directly or by `fkvideo_detector` are same. They may contain a link to a face thumbnail and other data which can be parsed in your application.

6.1 Fast Index

For galleries with the number of faces over 1,000,000, we recommend you to speed up search by using a fast index. To prepare the fast index, you will need the `findface-tarantool-build-index` utility from your distribution package. This utility is independent of the `tntapi` component and can be installed either on a localhost or on a remote host with access to Tarantool.

To prepare the fast index, do the following:

1. Install the `findface-tarantool-build-index` utility.

```
sudo apt-get install findface-tarantool-build-index
```

2. Create the fast index for your gallery (`testgal` in the case-study). First, connect to the Tarantool console.

Note: You have to repeat the fast index creation on each `tntapi` shard.

```
tarantoolctl connect 127.0.0.1:33001
```

3. Run `prepare_preindex`. Each element of the gallery will be moved from the linear space to preindex:

```
127.0.0.1:33001> FindFace.Gallery.new("testgal"):prepare_preindex()  
---  
...
```

4. Prepare a file for generating the index:

```
127.0.0.1:33001> FindFace.Gallery.new("testgal"):save_preindex("/tmp/preindex.bin  
↪")  
---  
...
```

5. Launch index generation with the `findface-build-index` utility (see `--help` for additional options). Depending on the number of elements, this process can take up to several hours and can be done on a separate, more powerful machine (for huge galleries we recommend `c4.8xlarge` amazon, for example `spot-instance`).

```
sudo findface-build-index --input /tmp/preindex.bin --facen_size 320 --out /opt/
↪ntech/var/lib/tarantool/default/index/testgal.idx

0% 10 20 30 40 50 60 70 80 90 100%
|----|----|----|----|----|----|----|----|----|
*****
0% 10 20 30 40 50 60 70 80 90 100%
|----|----|----|----|----|----|----|----|----|
*****

[Benchmark] create_index took 29.994ms
Index saved at /opt/ntech/var/lib/tarantool/default/index/testgal.idx
```

6. Delete the `preindex.bin` file.

```
sudo rm /tmp/preindex.bin
```

7. Enable the fast index for the gallery.

Note: If Tarantool works as a *replica set*, copy the index file (`.idx`) from the master instance to the same path on the replica before enabling the fast index for the master instance (`:use_index`).

Tip: Do not forget to remove obsolete index files on the replica in order to avoid unnecessary index transitions, should the master instance and replica be heavily out of sync.

```
127.0.0.1:33001> FindFace.Gallery.new("testgal"):preindex_to_index()
---
...
127.0.0.1:33001> FindFace.Gallery.new("testgal"):use_index("/opt/ntech/var/lib/
↪tarantool/default/index/testgal.idx")
---
...
```

8. Search through the gallery should now be significantly faster. Information about the index remains in the service space, so when you restart Tarantool, index will also be uploaded.

Warning: Do not move the index file to another location!

FindFace Web User Interface

FindFace Enterprise Server SDK is equipped with a web user interface which generally duplicates the functionality available via REST API.

To install the web interface, execute on the `findface-facenapi` host:

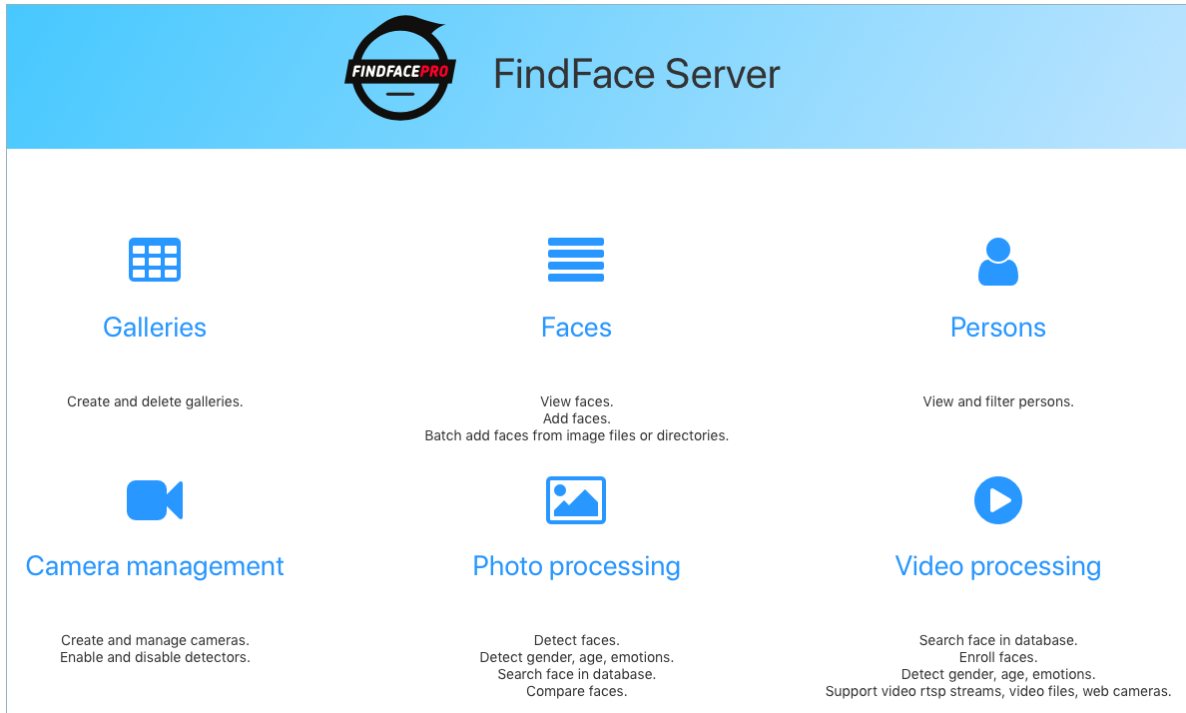
Note: First, install `nginx` if you do not already have it. You can do this as such:

```
sudo apt-get install nginx
```

```
sudo apt-get install findface-ui
```

To open the web interface, do the following:

1. In the address bar of your browser, enter `http://<facenapi_ip>:8000/#/`.
2. To log in, specify the *authentication token* for your FindFace Enterprise Server SDK instance. The web interface home page will appear.



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

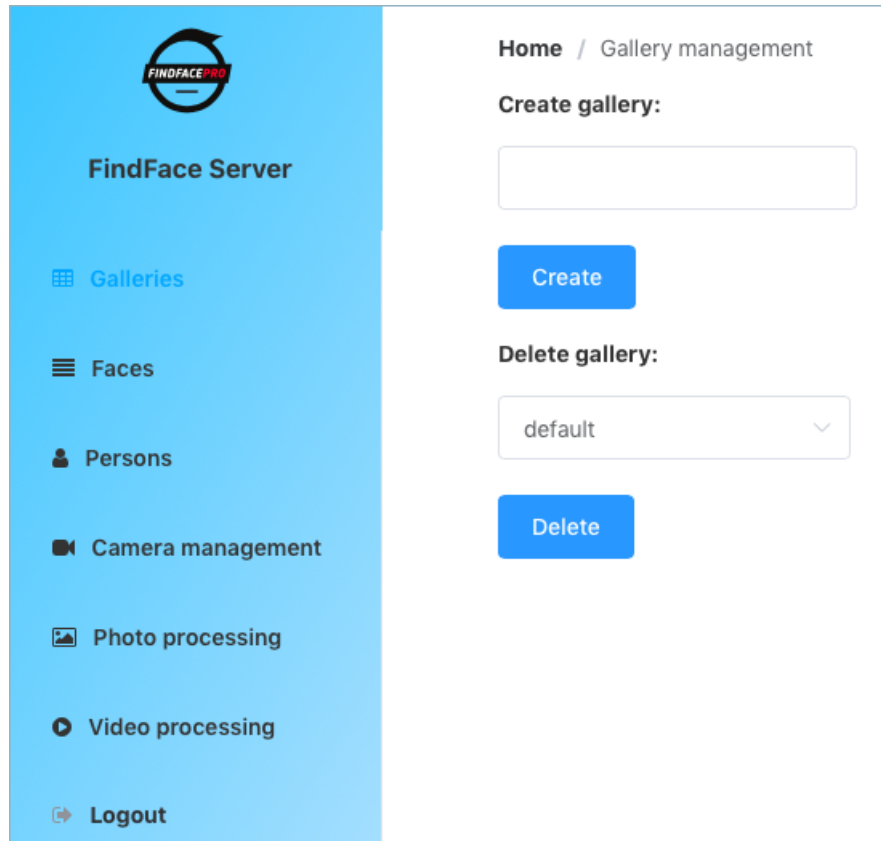
Note: To work with gender, age and emotions recognition (GAE) in the web interface, you need to *configure* it in the settings.

Note: Working with photos requires configured *findface-upload*.

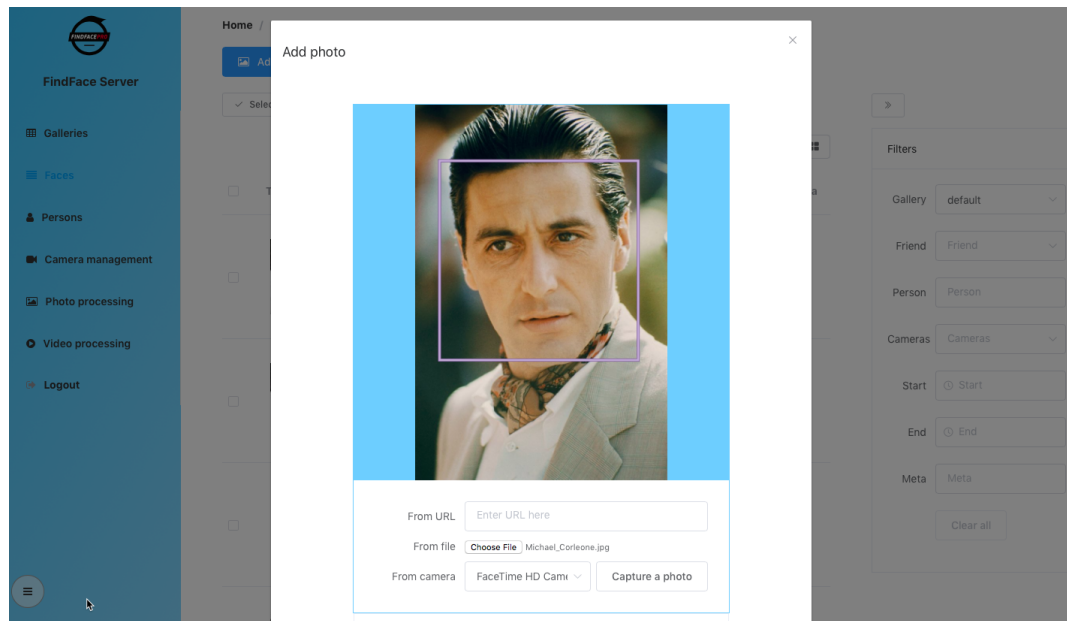
Note: Working with persons requires configured *dynamic person creation*.

Note: To allow the web interface to run Flash in **Chrome**, add its IP address to the relevant list: *Settings* → *Advanced* → *Content settings* → *Flash* → *Allow* → *Add a site* `http://<facenapi_ip>:8000/#/`. Restart **Chrome**.

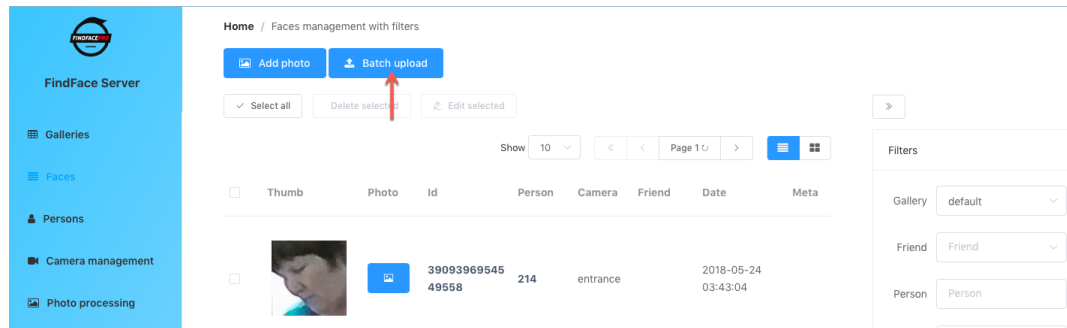
- *Galleries*. Create and delete galleries here.



- *Faces*. In this section, you can view, add and delete faces from the galleries.

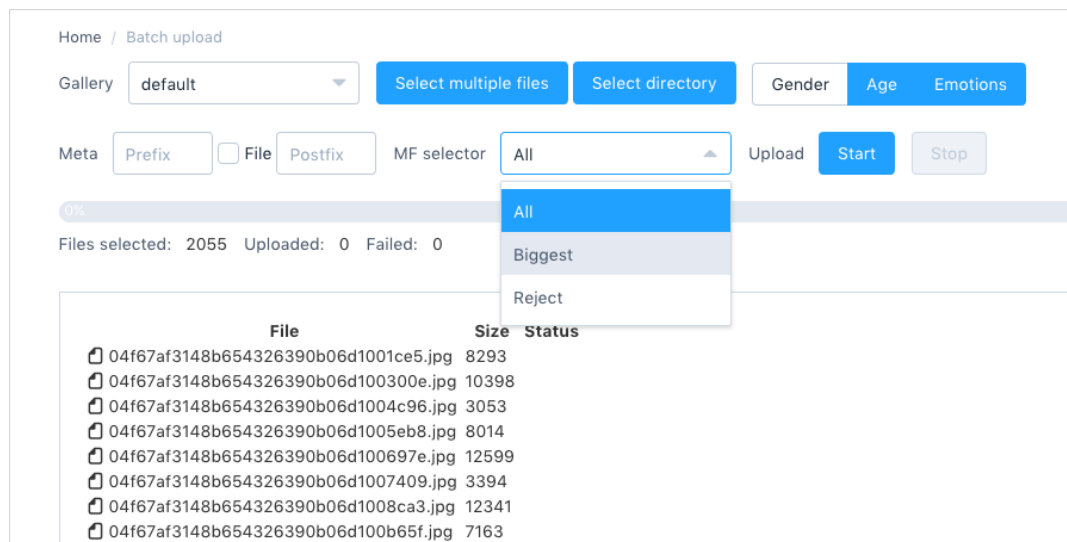


Use the *Batch upload* option to upload image files in bulk.



Tip: You may also want to use its *console alternative*.

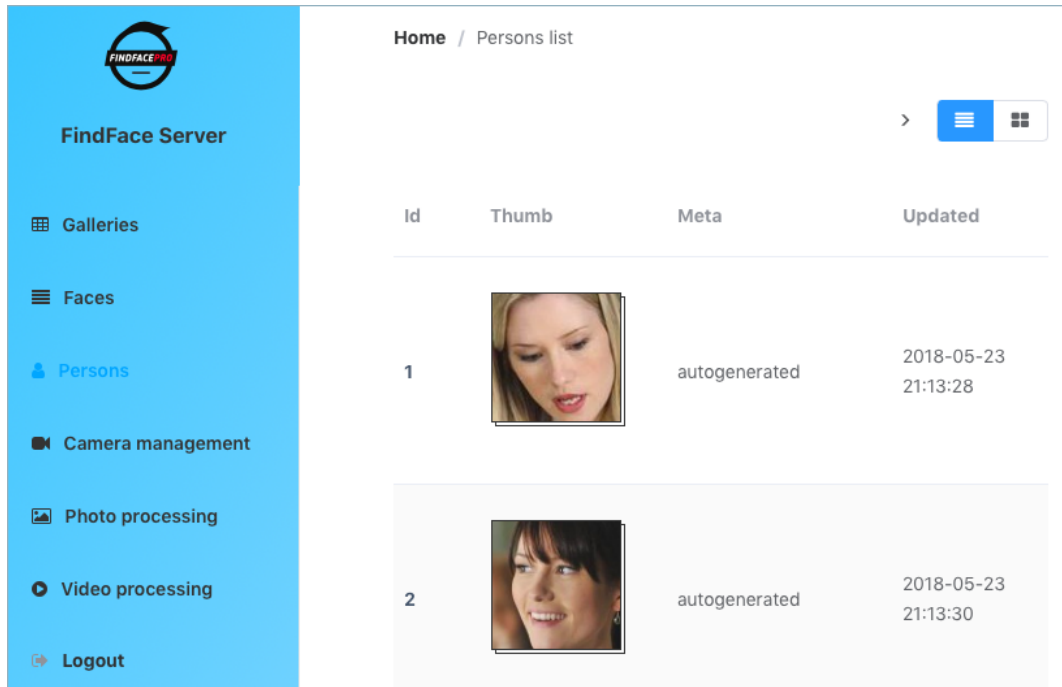
Select multiple files or a directory, and then configure the automatic meta description for the enrolled faces. Use *MF selector* to specify behavior in case if multiple faces are detected in an image: enroll all faces, only the biggest one, or reject enrollment.



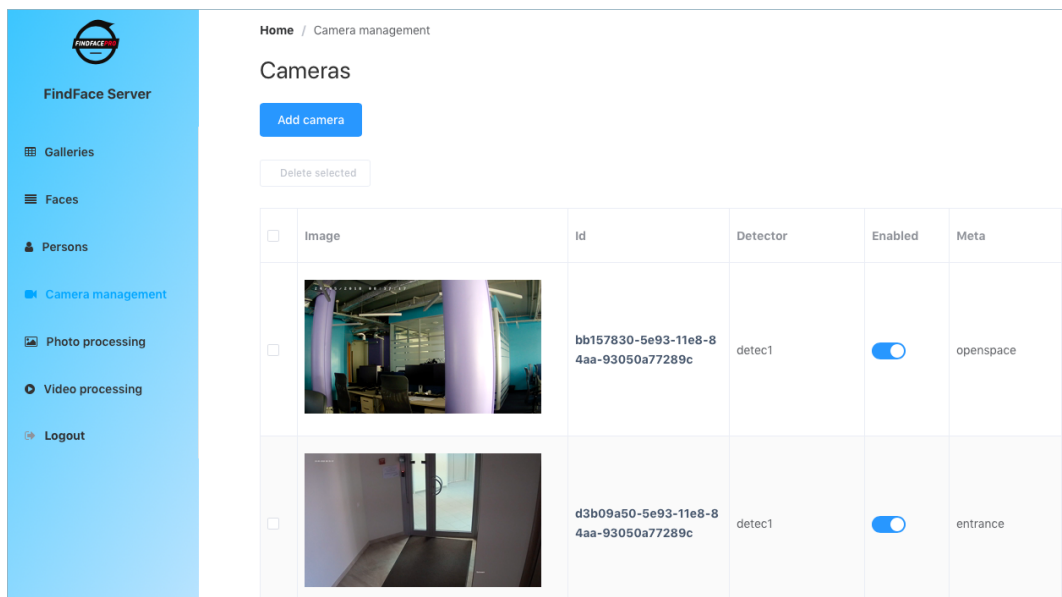
Tip: You can configure the automatic face meta by appending a custom prefix and/or postfix to the image file name. To avoid merging the 3 words into one, use underscore or another symbol in the prefix and postfix.

Tip: To select photos in the *icons* mode, click on them as you hold down the CTRL key.

- *Persons*. View and filter persons here.



- **Camera management.** Add cameras to your system. When adding a camera, you can specify ROT to enable detecting and tracking faces only inside a clipping rectangle (region of tracking), and/or ROI to enable posting faces detected only inside a region of interest.



Tip: The ROT option is used to reduce the video face detector load.

Tip: To specify ROT and ROI, use the visual wizard. First, create a camera without ROT and ROI, then open it for editing and click *Change ROI/ROT with image*.

Update camera

*

Id

5e4bca30-626e-11e8-80ab-df2a903be7da

Meta

entrance

*

Url

http://172.17.45.87/hls/entrance.m3u8

*

Detector

default

Advanced

Change ROI/ROT with image

Region of tracking (ROT):

X

Y

Width

Height

Region of interest (ROI):

X

Y

Width

Height

Update camera

- *Photo processing.* Select this section to detect faces in static images, recognize gender, age and emotions, search a face in the database (identification), and compare two faces (verification).

FindFace Server

Galleries

Faces

Persons

Camera management

Photo processing

Video processing

Logout

Home / Photo processing / Gae

Compare

Search

GAE

Select features:

Gender

Age

Emotions

Detect features

Gender

male

Age

40.245613

Emotions

["neutral", "surprise"]

From URL

Enter URL here

From file

Choose File

Michael_Corleone.jpg

From camera

FaceTime HD Cam


Capture a photo

- *Video processing.* Here you can work with video streams from rtsp and web cameras, and video files. Detect, enroll (add to a gallery) and identify faces in video with gender, age and emotions recognition. Generate enrollment and face identification reports in HTML by clicking on the *Save demo report* button.

60


Chapter 7. FindFace Web User Interface

Home / Video processing




Events 3/100000Image sizeSource10 /page

2017-11-01 15:23:31 | found




97%




Gender: male
Age: 32.58
Emotions: neutral, happy

Known: 0.97
Id: 3864419606482186

2017-11-01 15:23:27 | found




95%




Gender: male
Age: 43.46
Emotions: neutral, happy

Known: 0.95
Id: 3864419598335087

2017-11-01 15:23:09 | found



97%



Gender: male
Age: 34.74
Emotions: neutral, happy

Known: 0.97
Id: 3864419670722472

> Detector running with PID 61393

Storage settings

StorageMemoryBrowser DB

Skip 'notfound' events

Saved events 3

Storage limit 100000

* Older events will be deleted when exceed the limits

** Limits is not guaranteed

Clear storagesSave demo report

FindFace Demo Report. Part 1 of 1. Events 3 of 3.

2017-11-01 15:23:09 | found



97%



Gender: male
Age: 34.74
Emotions: neutral, happy

Known: 0.97
Id: 3864419670722472

2017-11-01 15:23:27 | found



95%



Gender: male
Age: 43.46
Emotions: neutral, happy

Known: 0.95
Id: 3864419598335087

2017-11-01 15:23:31 | found



97%



Gender: male
Age: 32.58
Emotions: neutral, happy

Known: 0.97
Id: 3864419606482186

Note: The video processing functionality in the web interface is great for tests. In production mode, use *fkvideo_detector*.

8.1 Gender, Age and Emotions Recognition

In this section:

- *Enable Gender, Age and Emotions Recognition*
- *API Requests for Gender, Age and Emotions Recognition*

8.1.1 Enable Gender, Age and Emotions Recognition

Note: Gender, age and emotions recognition uses around 2 GB of RAM in addition to the FindFace Server *general requirements*.

To enable gender, age and emotions recognition, uncomment and edit the line `gae = False` in the `findface-facenapi` configuration file. Restart `findface-facenapi`.

Warning: The `findface-facenapi.ini` content must be correct Python code.

```
sudo vi /etc/findface-facenapi.ini

    → gae = True

sudo service findface-facenapi restart
```

8.1.2 API Requests for Gender, Age and Emotions Recognition

An exemplary API request for recognizing gender, age and emotions of a person, and the corresponding response are shown below.

Request #1

```
POST /v1/detect/ HTTP/1.1
Host: 192.168.113.76:8000
Connection:close
Authorization: Token BpdNA6eaU1N9bPhXVSK1r92_SF0ODPOU
Content-Type: application/json
Content-Length: 108

{
  "photo": "https://static.findface.pro/sample.jpg",
  "emotions": true,
  "gender": true,
  "age": true
}
```

Response

```
HTTP/1.1 200 OK
Date: Thu, 06 Apr 2017 12:38:40 GMT
Server: TornadoServer/4.4.2
Content-Length: 120
Content-Type: application/json; charset=UTF-8

{
  "faces": [
    {
      "age": 26,
      "emotions": [
        "neutral",
        "sad"
      ],
      "gender": "female",
      "x1": 595,
      "x2": 812,
      "y1": 127,
      "y2": 344
    }
  ]
}
```

To add a face to the database with its gender, age and emotions information, send a POST request to **v1/face**.

Request #2

```
POST /v1/face/ HTTP/1.1
Host: 127.0.0.1
```

(continues on next page)

(continued from previous page)

```

Authorization: Token e93437ccd57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "meta": "Jane Berry",
  "photo": "http://static.findface.pro/sample.jpg",
  "galleries": ["gall", "nicepl"],
  "emotions": true,
  "gender": true,
  "age": true
}

```

Response

```

HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 06:04:02 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: [length]

{
  "results": [
    {
      "galleries": ["default", "gall", "nicepl"],
      "id": 2334,
      "meta": "Jane Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:11:29.425339",
      "age": 26,
      "emotions": [
        "neutral",
        "sad"
      ],
      "gender": "female",
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    }
  ]
}

```

8.2 Dynamic Person Creation

You can tailor FindFace Enterprise Server SDK to work in video surveillance and video analytics systems. To do so, harness the Dynamic Person Creation feature.

In this section:

- *How it works*
- *Configure Dynamic Person Creation*
- *REST API Sequence for Person Dataset Analysis*
 - *Method /face POST*
 - *Method /history/search POST*
- *Other API Methods to Work with Persons*
 - *Add and change person_id*
 - *Retrieve person history*
 - *List persons*

8.2.1 How it works

- An image containing a face (for example, extracted from a RTSP video stream by the video face detector) is sent via a `/face POST` request to FindFace Server for identification.
- When identifying a person, the system uses a face property `person_id`. For each person in the database, the value of this property should be unique.
- FindFace Server takes the new face and searches for the most similar one in the database (the so-called reference face). If similarity between the faces is equal or exceeds the threshold specified in the `findface-facenapi.ini` configuration file (`person_identify_threshold`), the new face is added to the database and assigned the same `person_id` value as the reference face. It also inherits some other reference face properties. This means that the new face has been identified as belonging to an existing person.
- If similarity between the faces does not exceed the given threshold, the system considers the new face as unidentified. In this case, the new face is added to the database as belonging to a new person, with a new `person_id`.
- In either case, FindFace Server returns a response containing the `person_id` value assigned to the added face. This value can be then used in analysis.

Warning: On account of the very logic of dynamic person creation, resembling faces of different people can be identified as belonging to one person and get the same `person_id`. To avoid this situation, we recommend you to periodically inspect the person database and manually resolve each identity conflict.

8.2.2 Configure Dynamic Person Creation

By default, dynamic person creation is disabled. This means that all newly added faces are not assigned the `person_id` property and the system does not discern persons.

To enable dynamic person creation, do the following:

1. Open the `findface-facenapi.ini` configuration file for editing.

```
sudo vi /etc/findface-facenapi.ini
```

2. Edit the settings.

Warning: The `findface-facenapi.ini` content must be correct Python code.

Uncomment and edit the line `person_identify = False`. This will enable dynamic person creation.

```
→ person_identify = True
```

By default, dynamic person creation is performed independently for each camera. To merge person identification results across all cameras, uncomment and edit the line `person_identify_global = False`. This option works well only in small-scale systems with less than 5 cameras. Otherwise, leave it deactivated.

```
→ person_identify_global = True
```

Uncomment and set the threshold for person identification between 0 and 1.

```
→ person_identify_threshold = 0.75
```

3. Restart the service.

```
sudo service findface-facenapi restart
```

8.2.3 REST API Sequence for Person Dataset Analysis

There are many ways to harness the dynamic person creation feature in analytics. A typical REST API sequence to identify a person and then work with their data is the following:

#	Method	Description
1	/face POST	Add a face to the database and receive a JSON representation of it, including the <code>person_id</code> value.
2	/history/search POST	Retrieve all events from the history of cameras, related to the person whose <code>person_id</code> was received in the /face POST response.

Method /face POST

Request

```
POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "photo": "http://static.findface.pro/sample.jpg"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": {
    "[595, 127, 812, 344]": [
      {
        "confidence": 1,
        "face": {
          "friend": false,
          "galleries": [
            "default"
          ],
          "id": 2,
          "meta": "Jack Smith",
          "normalized": "http://192.168.113.76:3333/uploads/20170418/1492509569217098.
↪ jpeg",
          "person_id": 2,
          "photo": "http://192.168.113.76:3333/uploads/20170418/14925095692111893.jpeg
↪ ",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.76:3333/uploads/20170418/14925095692159095.
↪ jpeg",
          "timestamp": "2017-04-18T09:59:29.211000",
          "x1": 595,
          "x2": 812,
          "y1": 127,
          "y2": 344
        }
      ]
    }
  ]
}
```

Method /history/search POST

Request

```
POST /v0/history/search HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "person_id": 2,
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
```

(continues on next page)

(continued from previous page)

```

Content-Type:  application/json
Content-Length: [length]
{
  "next_page": "/v0/history/search?max_id=4",
  "results": [
    {
      "friend": false,
      "meta": "Jack Smith",
      "photo_hash": "9fda49f2444f93c33ad8aa914e20e53b",
      "cam_id": "12345678123456781234567812345678",
      "person_id": 2,
      "timesamp": "2016-10-11T14:36:27.450000",
      "photo": "http://192.168.113.76:3333/uploads/20170418/149250956922566.jpeg",
      "id": 20146,
      "y1": 77,
      "x1": 285,
      "x2": 552,
      "y2": 345
    },
    {
      "friend": false,
      "meta": "Jack Smith",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "cam_id": "12345678123456781234567812345678",
      "person_id": 2,
      "timesamp": "2016-10-12T12:57:07.509000",
      "photo": "http://192.168.113.76:3333/uploads/20170418/14925095692111596.jpeg",
      "id": 20147,
      "x1": 236,
      "y1": 345,
      "x2": 311,
      "y2": 419
    }
  ]
}

```

8.2.4 Other API Methods to Work with Persons

Add and change `person_id`

To add or change the `person_id` value for a particular face, use the method `PUT /face/id/<face_id>`.

Warning: Since the `person_id` property is assigned only to newly added faces, old faces in the database are excluded from the person identification process. Use the method `PUT /face/id/<face_id>` to solve the problem.

Request

```

PUT /v0/face/id/5/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e

```

(continues on next page)

(continued from previous page)

```
Content-Type: application/json
Content-Length: [length]
```

```
{
  "person_id": "4"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "id": 5,
  "meta": "Jane Richardson",
  "person_id": "4",
  "photo": "http://static.findface.pro/sample2.jpg",
  "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
  "timestamp": "2016-06-13T11:06:42.075754",
  "x1": 225,
  "x2": 307,
  "y1": 345,
  "y2": 428
}
```

Retrieve person history

To retrieve all events from the history of cameras, related to the person with a given `person_id`, you can use the method `GET /person/history/id/<person_id>` (equally with `/history/search POST`).

Request

```
GET v0/person/history/id/2001 HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "cam_ids": [1, 25, 26, 27],
  "start": "2016-06-13T11:00:00.000000",
  "end": "2016-06-14T11:00:00.000000"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```

Content-Length: [length]
{
  "results":
  [
    {
      "person_id": 2001,
      "face_id": 240344,
      "cam_id": 25,
      "meta": "Sam Berry",
      "screenshot": "https://static.findface.pro/57726179d6946f02f3763824/
→dc7ac54590729669ca869a18d92cd05e_thumb.j
pg",
      "timestamp": "2016-06-13T11:06:42.075754",
    },
    {
      "person_id": 2001,
      "face_id": 240422,
      "cam_id": 25,
      "meta": "Sam Berry",
      "screenshot": "https://static.findface.pro/57726179
d6946f02f3763824/dc7ac54590729669ca869a18d92cd05e_thumb.j
pg",
      "timestamp": "2016-06-13T11:08:44.073452",
    }
  ]
}

```

List persons

To get the list of all existing persons, use the method GET /persons.

Request

```

GET /v0/persons HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e

```

Response

```

HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    {
      "id": 2,
      "meta": ""
    }
  ]
}

```

8.3 ‘Friend or Foe’ Identification

As you configure *Dynamic Person Creation*, you can also enable ‘friend or foe’ identification in order to further enhance your video analytics.

In this section:

- *About Friends and Foes*
- *Enable ‘Friend or Foe’ Identification*
- *‘Friend or Foe’ Identification in REST API*

8.3.1 About Friends and Foes

The ‘friend or foe’ identification system of FindFace Enterprise Server SDK can positively identify only friends, not foes. A friend is a person whose face has been captured a certain number of days by the same camera during a certain period of time. In all other cases, a person is just considered to be ‘not a friend’.

8.3.2 Enable ‘Friend or Foe’ Identification

To enable ‘friend or foe’ identification, do the following:

1. Configure and tryout *dynamic person creation*.
2. Open the `findface-facenapi.ini` configuration file for editing.

```
sudo vi /etc/findface-facenapi.ini
```

3. Edit the settings.

Warning: The `findface-facenapi.ini` content must be correct Python code.

A friend is a person that has been seen a certain number of days by the same camera during an interval `[now() - $interval ; now()]`. Uncomment and edit the number of days a person has to be seen to befriend your system.

```
→ friend_count = 5
```

Interval in seconds during which a person has to be seen a certain number of days (1 week by default):

```
→ friend_interval = (3600*24*7)
```

4. Restart the service.

```
sudo service findface-facenapi restart
```

8.3.3 'Friend or Foe' Identification in REST API

The example below demonstrates a POST /face request and the corresponding response containing the 'friend' parameter ("friend": true or "friend": false).

Request

```
POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccd5e66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "photo": "http://static.findface.pro/sample.jpg"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": {
    "[595, 127, 812, 344]": [
      {
        "confidence": 1,
        "face": {
          "friend": true,
          "galleries": [
            "default"
          ],
          "id": 2,
          "meta": "Jack Smith",
          "normalized": "http://192.168.113.76:3333/uploads/20170418/1492509569217098.
↪ jpeg",
          "person_id": 2,
          "photo": "http://192.168.113.76:3333/uploads/20170418/14925095692111893.jpeg
↪ ",
          "photo_hash": "53477c4a72f52c6efc951d9c7ece42bc",
          "thumbnail": "http://192.168.113.76:3333/uploads/20170418/14925095692159095.
↪ jpeg",
          "timestamp": "2017-04-18T09:59:29.211000",
          "x1": 595,
          "x2": 812,
          "y1": 127,
          "y2": 344
        }
      ]
    ]
  }
}
```

8.4 Bulk Face Enrollment

The Bulk Face Enrollment feature allows for enrolling faces to findface-facenapi from images in bulk.

In this section:

- *General Information*
- *Example*

8.4.1 General Information

You can bulk-enroll faces in one of the following ways:

- from images in a current directory,
- from images in a given subdirectory,
- from images from all subdirectories.

To install the Bulk Face Enrollment component, execute:

```
sudo apt-get install findface-mass-enroll
```

To display the component help message, execute:

```
findface-mass-enroll --help
```

```
## $ findface-mass-enroll --help
Usage: findface-mass-enroll [OPTIONS] COMMAND [ARGS]...

Options:
  --job PATH  Job file (default: ffmassenroll.job)
  --help      Show this message and exit.

Commands:
  prepare  Prepare upload job
  print    Print contents of job file as JSON
  run      Run upload job

$ findface-mass-enroll prepare --help
Usage: findface-mass-enroll prepare [OPTIONS] [IMAGES]...

This subcommand is used to prepare one or more job files for subsequent
runs.

Examples:

Enrolling all *.jpg files in current directory with meta 'Phillip J. Fry':

$ ls
photo1.jpg photo2.jpg photo3.jpg
$ findface-mass-enroll prepare --meta-const='Phillip J. Fry' '*.jpg'
```

(continues on next page)

(continued from previous page)

```

Enrolling all JPEGs and PNGs from a subdirectory with meta from accompanying TXT
↳files:

$ ls subdir
photo1.jpg photo1.txt photo2.png photo2.txt photo3.jpeg photo3.txt
$ findface-mass-enroll prepare --meta-companion='txt' 'subdir/*.jpg' 'subdir/*.png'
↳'subdir/*.jpeg'

```

Enrolling JPEGs from all subdirectories with meta from CSV file:

```

$ cat meta.csv
"Phillip J. Fry","dir1/photo1.jpg"
"Phillip J. Fry","dir1/photo2.jpg"
"Phillip J. Fry","dir1/photo3.jpg"
"Turanga Leela","dir2/photo1.jpg"
"Turanga Leela","dir2/photo2.jpg"
"Turanga Leela","dir2/photo3.jpg"
$ ls -R
.:
meta.csv

./dir1:
photo1.jpg photo2.jpg photo3.jpg

./dir2:
photo1.jpg photo2.jpg photo3.jpg
$ findface-mass-enroll prepare --meta-csv=meta.csv '**/*.jpg' '**/*.jpeg'

```

Options:

```

--meta-const TEXT      Shared metadata string
--meta-companion TEXT  Extension of metadata files accompanying the images
                        (e.g. txt)
--meta-csv PATH        Name of the CSV file containing metadata
--meta-filename        Use file name (without extension) as metadata string
--split INTEGER        Split job file into N parts (default: don't split)
--help                 Show this message and exit.

```

```

## $ findface-mass-enroll print --help
Usage: findface-mass-enroll print [OPTIONS]

```

Print contents of job file as JSON

Options:

```

--failed  Show only failed images
--help    Show this message and exit.

```

```

## $ findface-mass-enroll run --help
Usage: findface-mass-enroll run [OPTIONS]

```

Run upload job

Options:

```

--parallel INTEGER  Number of enroll threads (default: 10)
--api TEXT          API url (default: http://127.0.0.1:8000/)
                    [required]
--token TEXT        API token [required]

```

(continues on next page)

(continued from previous page)

```

--gallery TEXT           Enroll faces into specified gallery
                        (default: default)
--failed                 Include failed images
--mf-selector [all|biggest|reject]
                        mf_selector (biggest,all,reject)
--gender                 Extract gender
--age                   Extract age
--emotions               Extract emotions
--stats-interval INTEGER Output stats after every STATS_INTERVAL
                        seconds (default: 1)
--help                   Show this message and exit.

```

To harness the feature, do the following:

1. Prepare a job file containing the list of images with metadata (prepare). If all images share the same metastring, you can specify it right in the command line when preparing the job file (`--meta-const`). If each image has a unique metastring, map metastrings to images in a CSV file (`--meta-csv`).

Note: The CSV file used as a metadata source should have the following format: metastring | image. If some images are not listed in the CSV file, their metastrings will be empty.

Tip: To write the list of images to a CSV file, you can use the command below. Each image in the list will be associated with a metastring coinciding with the image full path (in the format metastring | image).

```
find /home/user/sample | grep -E 'jpg|png' | awk '{print $0,""$0}' > list.csv
```

2. If necessary, display the job file content (print).
3. Enroll faces to findface-facenapi for further processing (run).

Note: Should an error occur during the job file processing, correct the mistake and try again with the option `--failed` (see examples below).

8.4.2 Example

Enroll faces from all .jpg files in a /home/user/images/ directory with a shared metastring Phillip J. Fry:

To display the list of images in a directory, execute:

```
ls /home/user/images/
photo1.jpg photo2.jpg photo3.jpg ...
```

Prepare a job file:

```
findface-mass-enroll prepare --meta-const='Phillip J. Fry' '/home/user/images/*'

Looking for images matching '*.jpg'
2055 files prepared for upload
2055 files in job file samplejob
```


Run the job file:

```
findface-mass-enroll run --token 'RczGgVEMizR1njHHQegNH_g9mwGl6-A1' --api http://127.
↳0.0.1:8000/ --gender --age --emotions --mf-selector=all

[33/2055] faces processed (4 succeeded, 9 failed, 10 skipped). 2.14 rps. [00:00:17/
↳00:16:04]

----- Summary -----
↳-----

Found 2055 images in job file
Skipped 0 already processed images
Successfully processed 2000 images
Failed to process 55 images
```

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

```
findface-mass-enroll run --token 'RczGgVEMizR1njHHQegNH_g9mwGl6-A1' --api http://127.
↳0.0.1:8000/ --gender --age --emotions --mf-selector=all --failed
```

8.5 Extraction API

By default, the *extraction-api* component is used as a face detector and facen extractor. This section explains how to harness its advanced feature such as flexible configuration of the API response format. Use this feature to extract various face data, including the bounding box coordinates, normalized face, gender, age, and emotions, and facen without *findface-facenapi* as a mediator. Implementing this feature to your system can remarkably broaden the scope of analytic tasks it is capable of fulfilling.

Note: Being a *findface-facenapi* counterpart when it comes to data extraction via API, Extraction API is more resource-demanding. The component cannot fully substitute *findface-facenapi* as it doesn't allow adding faces and working with the database.

Tip: Normalized images received from Extraction API in base64 are qualified for posting to *findface-facenapi*.

Important: To use such Extraction API functions as *face quality estimation* and *auto-rotation* of original images, activate the **Face Quality** module. Please contact support for details by info@ntechlab.com.

In this section:

- *Install Extraction API*
- *API Requests*
- *API Response Format*

- *Examples*

8.5.1 Install Extraction API

To install and configure the `Extraction API` component, do the following:

Note: `Extraction API` requires the packages with *models* `<findface-data>.deb`. Make sure they have been installed.

1. Install the component.

```
sudo apt-get install findface-extraction-api
```

2. Open the `findface-extraction-api.ini` configuration file.

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

3. If *NTLS* is remote, specify its IP address.

```
license_ntls_server: 192.168.113.2:3133
```

4. Configure other parameters if needed. For example, enable or disable fetching Internet images.

```
fetch:
  enabled: true
  size_limit: 10485760
```

5. The `min_face_size` and `max_face_size` parameters do not work as filters. They rather indicate the guaranteed detection interval. Pick up their values carefully as these parameters affect performance.

```
nnd:
  min_face_size: 30
  max_face_size: .inf
```

6. The `model_instances` parameter indicates how many `extraction-api` instances are used. Specify the number of instances that you purchased. The default value (0) means that this number is equal to the number of CPU cores.

Note: This parameter severely affects RAM consumption.

```
model_instances: 2
```

7. To estimate the face quality, enable the `quality_estimator`. In this case, `extraction-api` will return the quality score in the *detection_score* parameter.

Tip: 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.

```
quality_estimator: true
```

Important: This function is available only if you have activated the **Face Quality** module. Please contact support for details by info@ntechlab.com.

8. Enable the `Extraction API` service autostart and launch the service.

```
sudo systemctl enable findface-extraction-api && sudo systemctl start findface-  
↪extraction-api
```

8.5.2 API Requests

The `Extraction API` component accepts POST requests to <http://127.0.0.1:18666/>.

There are 2 ways to format the request body:

- `application/json`: the request body contains only JSON.
- `multipart/form-data`: the request body contains a JSON part with the request itself, other body parts are used for image transfer.

The JSON part of the request body contains a set of requests:

```
{  
  "requests": [request1, request2, .., requestN]  
}
```

Each request in the set applies to a specific image or region in the image and accepts the following parameters:

- `"image"`: an uploaded image (use `multipart:part` to refer to a relevant request body part), or a publicly accessible image URL (`http:`, `https:`).
- `"roi"`: a region of interest in the image. If the region is not specified, the entire image is processed.
- `"detector"`: a face detector to apply to the image (`legacy`, `nnd` or `prenormalized`). The `prenormalized` mode accepts normalized face images and omits detecting faces. Use `nnd` if you need to estimate the face quality (`"quality_estimator": true`).
- `"need_facen"`: if `true`, the request returns a `facen` in the response.
- `"need_gender"`: returns gender.
- `"need_emotions"`: returns emotions.
- `"need_age"`: returns age.
- `"need_normalized"`: returns a normalized face image encoded in base64. The normalized image can then be posted again to the `Extraction API` component as `"prenormalized"`.
- `"auto_rotate"`: if `true`, auto-rotates an original image to 4 different orientations and returns faces detected in each orientation. Works only if `"detector": "nnd"` and `"quality_estimator": true`.

Important: This function is available only if you have activated the **Face Quality** module. Please contact support for details by info@ntechlab.com.

```
{
  "image": "http://static.findface.pro/sample.jpg",
  "roi": { "left": 0, "right": 1000, "top": 0, "bottom": 1000 },
  "detector": "nnd",
  "need_facen": true,
  "need_gender": true,
  "need_emotions": true,
  "need_age": true,
  "need_normalized": true,
  "auto_rotate": true
}
```

8.5.3 API Response Format

A typical response from the Extraction API component contains a set of responses to the requests wrapped into the main API request:

```
{
  "response": [response1, response2, .., responseN]
}
```

Each response in the set contains the following JSON data:

- "faces": a set of faces detected in the provided image or region of interest.
- "error": an error occurred during processing (if any). The error body includes the error code which can be interpreted automatically ("code") and a human-readable description ("desc").

```
{
  "faces": [face1, face2, .., faceN],
  "error": {
    "code": "IMAGE_DECODING_FAILED",
    "desc": "Failed to decode: reason"
  }
}
```

Each face in the set is provided with the following data:

- "bbox": coordinates of a bounding box with the face.
- "detection_score": either the face detection accuracy, or the face quality score (depending on whether `quality_estimator` is false or true at `/etc/findface-extraction-api.ini`). 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.
- "facen": the face feature vector.
- "gender": gender information (MALE or FEMALE) with recognition accuracy if requested.
- "age": age estimate if requested.
- "emotions": all available emotions in descending order of probability if requested.
- "normalized": a normalized face image encoded in base64 if requested.

```
{
  "bbox": { "left": 1, "right": 2, "top": 3, "bottom": 4 },
```

(continues on next page)

(continued from previous page)

```

    "detection_score": -0.0004299,
    "facen": "...",
    "gender": {
        "gender": "MALE",
        "score": "1.123"
    },
    "age": 23.59,
    "emotions": [
        { "emotion": "neutral", "score": 0.95 },
        { "emotion": "angry", "score": 0.55 },
        ...
    ],
    "normalized": "...",
}

```

8.5.4 Examples

Request #1

```

curl -X POST -F sample=@sample.jpg -F 'request={"requests":[{"image":"multipart:sample
↪","detector":"nnd", "need_gender":true, "need_normalized": true, "need_facen": true}
↪}]}' http://127.0.0.1:18666/| jq

```

Response

```

{
  "responses": [
    {
      "faces": [
        {
          "bbox": {
            "left": 595,
            "top": 127,
            "right": 812,
            "bottom": 344
          },
          "detection_score": -0.0012599,
          "facen": "qErDPTE...vd4oMr0=",
          "gender": {
            "gender": "FEMALE",
            "score": -2.6415858
          },
          "normalized": "iVBORw0KGgoAAAANSUhE...79CIbv"
        }
      ]
    }
  ]
}

```

Request #2

```
curl -X POST -F 'request={"requests": [{"need_age": true, "need_gender": true,
↪ "detector": "nnd", "roi": {"left": -2975, "top": -635, "right": 4060, "bottom": ↪
↪ 1720}, "image": "https://static.findface.pro/sample.jpg", "need_emotions": true}}]' ↪
↪ http://127.0.0.1:18666/ |jq
```

Response

```
{
  "responses": [
    {
      "faces": [
        {
          "bbox": {
            "left": 595,
            "top": 127,
            "right": 812,
            "bottom": 344
          },
          "detection_score": 0.9999999,
          "gender": {
            "gender": "FEMALE",
            "score": -2.6415858
          },
          "age": 26.048346,
          "emotions": [
            {
              "emotion": "neutral",
              "score": 0.90854686
            },
            {
              "emotion": "sad",
              "score": 0.051211596
            },
            {
              "emotion": "happy",
              "score": 0.045291856
            },
            {
              "emotion": "surprise",
              "score": -0.024765536
            },
            {
              "emotion": "fear",
              "score": -0.11788454
            },
            {
              "emotion": "angry",
              "score": -0.1723868
            },
            {
              "emotion": "disgust",
              "score": -0.35445923
            }
          ]
        }
      ]
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    }
  ]
}
]
}

```

Request #3. Auto-rotation

```

curl -s -F 'sample=@/path/to/your/photo.png' -F 'request={"requests":[{"image":
↪ "multipart:sample", "detector": "nnd", "auto_rotate": true, "need_normalized": true }
↪ ]}' http://192.168.113.79:18666/

```

Response

```

{
  "responses": [
    {
      "faces": [
        {
          "bbox": {
            "left": 96,
            "top": 99,
            "right": 196,
            "bottom": 198
          },
          "detection_score": -0.00019264,
          "normalized": "iVBORw0KGgoAAAANSUhE....quWKAAC"
        },
        {
          "bbox": {
            "left": 205,
            "top": 91,
            "right": 336,
            "bottom": 223
          },
          "detection_score": -0.00041600747,
          "normalized": "iVBORw0KGgoAAAANSUhEUgAA....ABYquWKAACAAElEQVR4nKy96XYbybIdnF"
        }
      ]
    }
  ]
}

```

8.6 Shard Galleries Statistics

You can get a shard galleries statistics and other data right in your browser. This functionality can be harnessed in monitoring systems.

Note: In the case of standalone deployment, you can access Tarantool by default only locally (127.0.0.1). If you want

to access Tarantool remotely, change the Tarantool configuration file.

In this section:

- *List Galleries*
- *Get Gallery Information*

8.6.1 List Galleries

To list all galleries on a shard, type in the address bar of your browser:

```
http://<tarantool_host_ip:shard_port>/stat/list/:start/:limit
```

:start is the number of a gallery the list starts with.
:limit is the maximum number of galleries in the list.

Example

Request

```
http://127.0.0.1:8001/stat/list/1/99  
or  
curl http://127.0.0.1:8001/stat/list/1/99 \ | jq
```

Response

```
% Total      % Received % Xferd  Average Speed   Time    Time     Time  Current  
100  6700   100  6700    0     0  45812      0 --:--:-- --:--:-- --:--:-- 45890  
{  
  "galleries": [  
    {  
      "cnt_indexed": 0,  
      "id": 1,  
      "cnt_preindex": 0,  
      "name": "591b0cdfb0d5bd7058ef0968_default",  
      "cnt_linear": 35268  
    },  
    {  
      "cnt_indexed": 0,  
      "id": 2,  
      "cnt_preindex": 0,  
      "name": "591b0cdfb0d5bd7058ef0968_lublino",  
      "cnt_linear": 1818  
    }  
  ]  
}
```

(continues on next page)

(continued from previous page)

```

    },
    {
      "cnt_indexed": 0,
      "id": 3,
      "cnt_preindex": 0,
      "name": "591b0cdfb0d5bd7058ef0968_gifs",
      "cnt_linear": 297
    }
  ],
  "total": 3
}

```

8.6.2 Get Gallery Information

To get a gallery information, type in the address bar of your browser:

```
http://<tarantool_host_ip:shard_port>/stat/info/:name
```

:name is the gallery name.

Example

Request

```
curl http://127.0.0.1:8001/stat/info/5968bda4a2a4bb6018bee2b2_cam_cam1 | jq
```

Response

```

% Total      % Received % Xferd  Average Speed   Time    Time     Time  Current
   Dload  Upload   Total   Spent    Left     Speed
100    210    100    210     0      0  17654      0 --:--:-- --:--:-- --:--:-- 19090
{
  "cnt_indexed": 0,
  "cnt_preindex_deleted": 0,
  "index_file": "none",
  "index_loaded": false,
  "cnt_preindex": 0,
  "cnt_linear": 85011,
  "cptr": 29556448,
  "id": 34,
  "name": "5968bda4a2a4bb6018bee2b2_cam_cam1",
  "cnt_indexed_deleted": 0
}

```

8.7 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*
- *Get Facen*
- *Remove Face*
- *Face Search*
- *List Faces*

8.7.1 General Information

API requests to Tarantool should 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 standalone deployment, you can access Tarantool by default only locally (127.0.0.1). If you want to access Tarantool remotely, change the Tarantool configuration file.

Each API request to Tarantool contains the following parameters:

- `:ver`: the API version (v1 at the moment).
- `:name`: the gallery name.

Tip: To list gallery names on a shard, type in the following command in the address bar of your browser (see [List Galleries](#) for details):

```
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.

- `:id`: the face id.

8.7.2 Add Face

Request

```
POST /:ver/:name/add/:id
```

Body: a raw feature vector (facen)

Returns:

- HTTP 200 and empty body if success.
- HTTP 409 if a face with the same id already exists in the gallery.
- HTTP with a status other than 200 and error description in the body if failure.

Example

```
curl -s -D - 'http://localhost:8001/v1/my_gal/add/1234' --data-binary _
↪@3827305024709134.facen

HTTP/1.1 200 Ok
Content-length: 0
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive
```

8.7.3 Get Facen

Request

```
GET /:ver/:name/get/:id
```

Returns:

- A JSON representation of the face with its id and base64 encoded facen if 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 if failure.

Tip: To convert a facen from base64 to a binary file, execute:

```
echo 'facen in base64' |base64 -d> facen
```

Example

```
curl -s -D - 'http://localhost:8001/v1/my_gal/get/1234' HTTP/1.1 200 Ok Content-
↪length: 1754 Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc) Connection:
↪keep-alive {"facen":"BFa9PWNlS7215fI98ETQvJkxML2hUFY9cF\Tu9ZjnLx\
↪uVc9EzWSPQTSr7zoysI8+4PSPiSjnr2GVlM8eFMKvfn9mjsPPjA8ZXoNvTEsSr0rJkM9MR
↪rTyEDNm8Ti\0ve4TrrlrnQA+Yc\KvJzqnbzOPSG998CKPBFpAr77kFO9BonDvK9B0buvjAq9Q7A\
↪u6awnTw0lvv80QZcVRfQAz0BdH498hf6vQKRcdY77c08mGRkvQ305DomnBM9XSqwnN54GT0ClFO9a+kWvhp7iT3uqqU9v1+\
↪YhZm7uKEU5IdduyBKRZfCfU9CCxPVJnvzt5T309NicxPD9Sar3f6sO8UmlhvrMI67wlTte880wYvUF867xg4\
↪g8aqNQu\
↪AAWD2z59C9CQCrPepF7Dy8qUa9iCczPfKv+Dy+bRo9KhyYPZfY0blxtbY7nKXLuvYFbr0g8rM86o0QPRCKOjla7rU9bd+3Pbqs
↪Guv2beTy56wg7p\hTPdxQgr0jxQQ9Ud0CPZcx\
↪LpLH0tF0C-V ML Y8l28 -HDEET3LW 0L QND 05 -H1507 0 -SD IN4W 0 -0 -0W7H -D -WH0MI F0D 0 -E -iDp
```

8.7. Direct API Requests to Tarantool

(continued from previous page)

8.7.4 Remove Face

Warning: Removing a face from Tarantool will not remove it from MongoDB.

Request

```
DELETE /:ver/:name/del/:id
```

Returns:

- HTTP 200 and empty body if 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 if failure.

Example

```
curl -s -D - -X DELETE 'http://localhost:8001/v1/my_gal/del/1234'  
  
HTTP/1.1 200 Ok  
Content-length: 0  
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)  
Connection: keep-alive
```

8.7.5 Face Search

Request

```
POST /:ver/:name/search/:limit/:threshold?linear_search
```

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

:threshold: the minimum similarity for faces in the response (from 0 to 1).

linear_search (boolean, optional): set linear_search=1 (true) to use only the linear space to search for faces. This setting has priority over the only_index setting (/etc/tarantool/instances.enabled/FindFace.lua).

body: a raw facen.

Returns:

- A JSON array with faces with the `conf` and `id` fields in the body if success. The value in the `X-search-stat` header indicates whether the fast index was used for the search: `with_index` or `without_index`.
- HTTP with a status other than 200 and error description in the body if failure.

Example

```
curl -s -D - 'http://localhost:8001/v1/my_gal/search/1/0.65?linear_search=1' --data-
↪binary @3827305024709134.facen

HTTP/1.1 200 Ok
Content-length: 22
X-search-stat: without_index
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

[{"conf":1,"id":1234}]
```

8.7.6 List Faces**Request**

```
GET /:ver/:name/list/:start_id/:count
```

:start_id: the minimum face_id in the response.

:count: the maximum number of faces in the response.

Returns:

- A JSON array with faces, and the next page URL if success. Each face is provided with its id, base64 encoded facen and the name of a Tarantool space where the face is located (linear, preindex, or indexed). The next page URL should be passed as :start_id in another API request to get the next page of results.
- HTTP with a status other than 200 and error description in the body if failure.

Example

```
curl -s -D - 'http://localhost:8001/v1/my_gal/list/0/1' HTTP/1.1 200 Ok Content-
↪length: 1795 Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc) Connection:␣
↪keep-alive {"faces":[{"id":1234,"space":"linear","facen":
↪"BFA9PWN1S7215fI98ETQvJkxML2hUFY9cF\Tu9ZjnLx\
↪uVc9EzWSPQTSr7zoysI8+4PSPISjnr2GVlM8eFMKvfn9mjsPPjA8ZXoNvTEsSr0rJkM9MR0IPINXSj3Em0s9awm5Oos5SD380a
↪jPd7QhDxUIzC+g90sPUWUDLwjK7U9cpWkPZ83rTyEDNm8Ti\0ve4TrrlrnQA+Yc\
↪KvJzqnbzOPSG998CKPBFpAr77kFO9BonDvK9B0buvjAq9Q7A\
↪u6awnTw0lvy80QZcvRFQAz0BdH498hf6vQKRcDy77c08mGRkvQ305DomnBM9XSqgvN54GT0ClFO9a+kWvhp7iT3uqqU9v1+\
↪vYhzm7uREt09ldouuyDKRr2PcIG9Uc8xPVJnvzt5T309NicxPD9SAr3f6sO8UmlhVRMI67wlTte880wYvUF8o7xg4\
↪g8aqNQ\
↪AAWD2z59C9CQrPepF7Dy8qUa9iCczPfKv+Dy+bRo9KhyYPZfY0b1xtbY7nKXLuvYFbr0g8Im660QrKcK0j1a7rU9bd+3Pbqs
↪Guv2beTy56wg7p\hTPdxQgr0jxQQ9Ud0CPZcx\
↪LtrLiU9bECQvUnvzspMVcM8b3OovURPET3JdHs9LyQUPsc9JzvWlZQ7y2ySPdN4Xb0xi9c8X7UevRqjVL0Mlpe9PoQpvFxxjD2l
↪PFikfIzc3pcqpafPxxuPb2tjSY9lA5lPR1NoT1+Uuu7G6gpu727wTwo6ii8iaH+PIlWY72D9QG+8lhAPuegx71VsFs8ajQLvOc
↪O1307D1ZMSk9IxxGvYCVfb1bE429hZf4vewikzWDbfG8wwYNPiqn4L2NV6Q9VKrvPTjwTr3dlG05jck+vZ\
↪KID1+n8Y8qpvnvOJjBj2P4+w8IJGgvROAfz1S4ve8QEouvQ5CkDu0OTI8\0v\
↪pvFrK5b3bkIO82LVBPCf2Yr0aGaU9RARUvEecJzlr8zk87U4vvC65ljz6kRS956U2PH6JMT5nfAg7KX7qPBz7Ejy60vk9\
```

(continues on next page)

8.8 Hacks for `tnapi`

In this section:

- *Additional Configuration Parameters*
- *Soft Deletion Mode*
- *Tarantool Replication*

8.8.1 Additional Configuration Parameters

To configure interaction between `findface-facenapi` and Tarantool, specify additional parameters in the 3rd argument of the `FindFace.start` section in the `tnapi` configuration file:

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

FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", additional_
↪parameter 1, ..., additional parameter N})

## Example:
FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", facen_size =
↪320, log_requests = false})
```

Additional parameters:

Parameter	Default value	Description
<code>log_requests</code>	<code>true</code>	Enable request logging (<code>/var/log/tarantool/FindFace.log</code>).
<code>facen_size</code>	<code>320</code>	Facen's size. Before editing this parameter, be sure to consult NTechLab experts.
<code>search_threads</code>	<code>1</code>	Number of threads for fast index search.
<code>replication</code>	<code>nil</code>	Only for a replica. Master instance IP address.
<code>soft_delete_mode</code>	<code>false</code>	Enable the soft deletion mode, when the faces are not removed from the fast index, but hidden in search results.

8.8.2 Soft Deletion Mode

Tarantool supports the soft deletion mode, when the faces are not removed from the fast index, but hidden in search results. We recommend you to enable this mode due to the following benefits:

- Tarantool starting time linearly depends on the number of faces removed from the `Indexed` space (fast index). If the soft deletion mode is on, the faces are not physically removed from the fast index, so face deletion doesn't affect the starting time.

- Fast index search quality also depends on the number of physically removed faces. It doesn't sink in the soft deletion mode.

To enable the soft deletion mode, edit the FindFace.start section as follows:

```
FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", soft_delete_
↪mode = true})
```

8.8.3 Tarantool Replication

Replication allows multiple Tarantool instances to work on copies of the same face database. The database copies are kept in sync because each instance can communicate its changes to all the other instances. Tarantool supports master-slave replication. You can add and delete data only by using the master instance, slave instances (aka replicas) are read-only, i.e. can be used only for searching and consulting data.

To learn how to deploy a Tarantool replica set, refer to the Tarantool [official documentation](#).

To start a created replica for the first time, do the following:

1. Start the master instance.
2. In the replica configuration file, specify the IP address and listening port of the master instance.

```
FindFace.start("127.0.0.1", 48001, {replication = "127.0.0.1:33001"})
```

3. Copy the latest snapshot (.snap) of the master instance into the memtx_dir directory of the replica.

```
--Directory to store data
memtx_dir = '/opt/ntech/var/lib/tarantool/default/snapshots'
```

4. Copy the master instance logs into the wal_dir directory of the replica.

```
--Directory to store data
wal_dir = '/opt/ntech/var/lib/tarantool/default/xlogs'
```

5. Start the replica. You can start as many replicas affiliated with the same master instance as needed.

Important: Before enabling the *fast index* for the master instance :`use_index("/path/to/<index>.idx")`, copy the index file (<index>.idx) to the same path on its replica. Then perform `use_index` on the master instance.

Tip: Delete obsolete index files on the replica in order to avoid unnecessary index transitions, should the master instance and replica be heavily out of sync.

Tip: To synchronize the master instance and replica, you can also copy the latest master snapshot to the replica.

Integration

Being integrated into specific solutions, FindFace Enterprise Server SDK can be used in such areas as biometric identification and access control, customer analytics, customer offer tailoring, offline retargeting, managing white and black lists, sorting and optimizing media libraries, borrower scoring, fraud prevention, employee productivity control, and many more.

You can integrate with FindFace Enterprise Server SDK in one of the following ways, depending on your purpose and resources:

- Via *REST API*. This option provides cross-platform multidimensional integration with your own application. Allows for maximum flexibility, though it is more time- and knowledge-intensive, as you have to write more code.

Tip: See *Render Detection Results* to learn how to use this approach to extend the video face detection functionality.

- By writing `findface-facenapi` *plugins* in Python. This option provides easier approach to the FindFace Enterprise Server SDK functionality than REST API as it involves less coding. The drawback of this integration approach is that you're bound to the Python programming language and hence need to have an experienced Python developer to write the plugin.
- Through custom HTTP API handlers implemented to the FindFace API Environment in addition to the main *REST API*.

Note: The FindFace API Environment is powered by `tornado.web.Application`. To implement a custom HTTP API handler, use the `add_handlers` method (see the `tornado.web` [official documentation](#) for details).

Tip: You can inherit from some ready-to-use HTTP API Handlers in your plugin (see *Plugins*).

9.1 REST API

9.1.1 How to Use REST API

In this section:

- *Endpoint*
- *API Version*
- *Authentication*
- *Common Object Types*
 - *Face*
 - *Bounding Box (bbox)*
- *Parameters Format*
- *How to Use Examples*
- *Confidence Thresholds*
- *Pagination*
- *Limits*
- *Error Reporting*

Endpoint

All REST API requests should be sent to `http://<facenapi_ip>:8000/v1/`.

API Version

The API version is increased every time a major change is made and allows us to avoid breaking backwards compatibility. The API version should be specified in the request path (for example, v1 in `/v1/detect/`).

The most recent version is v1 which provides such advanced functions as gender, age and emotions recognition.

Note: When starting a new project, you should always use the latest stable version of the API.

Authentication

All API methods require a simple token-based HTTP Authentication. In order to authenticate, you should put the word “Token” and your token key into the Authorization HTTP header, separated by a whitespace:

Note: To learn how to create a token, consult [Create Authentication Token](#).

```
Authorization: Token yfT8ftheVqnDLS3Q0yCiTH3E8YY_cm4p
```

All requests that fail to provide a valid authentication token will result in a HTTP 401 Unauthorized response.

Common Object Types

Face

Represents a human face. Note that it might be several faces on a single photo. Different photos of the same person as also considered to be different faces.

- "id" (number): unique identifier of the face generated by API.
- "timestamp" (string): time of face object creation as ISO8601 string.
- "photo" (string): URL of file name of a photo that had been used to create the face object.
- "photo_hash" (string): Hash of the original photo. Note that identical photos will always have the same hash, and different photos will most certainly have different hashes. Don't interpret this value and don't make assumptions about particular hash function used for hash calculation.
- "thumbnail" (string): URL of face thumbnail stored on FindFace servers.
- "x1" (number): x coordinate of the top-left corner of face's bounding box on the original photo.
- "y1" (number): y coordinate of the top-left corner of face's bounding box on the original photo.
- "x2" (number): x coordinate of the bottom-right corner of face's bounding box on the original photo.
- "y2" (number): y coordinate of the bottom-right corner of face's bounding box on the original photo.
- "meta" (string): metadata string that you can use to store any information associated with the face.
- "galleries" (string[]): array of galleries names that have this face.

Bounding Box (bbox)

Represents a rectangle on a photo. Usually used as a face's bounding box. May be specified in two ways:

- **Separated coordinates:**
 - "x1" (number): x coordinate of the top-left corner of the bounding box.
 - "y1" (number): y coordinate of the top-left corner of the bounding box.
 - "x2" (number): x coordinate of the bottom-right corner of the bounding box.
 - "y2" (number): y coordinate of the bottom-right corner of the bounding box.
- Array of coordinates [x1, y1, x2, y2]

The API methods accept both formats, but always return bbox as a JSON dictionary.

Note that in some case the coordinates might be outside photo dimensions, including negative values.

Parameters Format

There are three ways to pass parameters to the API methods:

- application/json: parameters are represented by a JSON contained in the body.

- `application/x-www-form-urlencoded`: parameters are represented by form field names and values.
- `multipart/form-data`: parameters are encoded into separate parts. This way supports uploading a photo image file in the same request.
- `query string`: parameters are appended to request URL. When passing parameters in a query string, complex structures (such as bboxes) should be encoded in JSON.

There are two ways of specifying a photo image file:

- As a publicly accessible URL.
- Included in a request as part of multipart form.

All responses are in JSON format and UTF-8 encoding.

How to Use Examples

Examples in methods descriptions illustrate possible method requests and responses. To check the examples without writing code, use the embedded API framework. To access the framework, enter in the address bar of your browser: `http://<facenapi_ip>:8000/v1/docs/v1/overview.html` for the API version /v1.

Confidence Thresholds

For some methods you need to specify a threshold for verification or identification confidence. The higher is the threshold, the less are chances that a wrong person will be positively verified or identified, however, some valid photos may also fail verification.

There are 4 pre-defined threshold levels:

- `Strict (0.7834)`: used for applications where a chance of misidentification should be minimized. This level corresponds to False Accept Rate (FAR) of $1e-5$ on our test dataset.
- `Medium (0.6616)`: balances low probability of misidentification and inability to identify a valid person. Corresponds to $1e-3$ FAR on our test dataset.
- `Low (0.5690)`: used when it's important to maximize the verification or identification rate, and misidentification does not cause severe consequences. Corresponds to $1e-1$ FAR on our test dataset.
- `None (0)`: use when you need to calculate similarity of different persons or find similar people rather than verify identity.

You can also specify your own threshold level from 0 to 1, depending on your environment and needs.

Note: If no threshold level is specified, it is set to the default value `0.75`.

Pagination

Some methods (such as `GET /faces/` and `GET /meta/`) may potentially return thousands and hundreds of thousands results. To avoid problems associated with such large amounts, we have introduced pagination.

Methods that support pagination return two more parameters in addition to a list of results:

- `prev_page`: URL to the previous page (path and query only)
- `next_page`: URL to the next page (path and query portion only)

For example, if GET `http://<facenapi_ip>:8000/v0/faces/` has returned the `next_page` value `'/v0/faces/?max_id=12345'`, you should request GET `http://<facenapi_ip>:8000/v0/faces/?max_id=12345` to get the next portion of the results.

Limits

FindFace Enterprise Server SDK imposes the following limits.

Limit	Value
Image formats	JPEG, PNG, WEBP
Maximum photo file size	10 MB
Minimal size of a face	50x50 pixels
Maximum number of detected faces on a single photo	Unlimited

Additionally, the URL provided to the API to fetch an image should be public (without authentication) and direct (without any redirects).

Error Reporting

If a method fails, it always returns a response with a HTTP code other than 200 and a JSON body containing the error description. The error body always includes at least two fields: `code` and `status`:

- `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
AUTH_FAILED	A wrong authentication token or no token has been provided.
BAD_PARAM	Some parameters are invalid. This response type has additional attributes 'param' and 'value' describing which parameter caused the error.
MALFORMED_JSON	The request body doesn't contain a valid JSON.
SERVICE_UNAVAILABLE	Your request cannot be processed because some components are experiencing an outage.

9.1.2 General Methods

In this section:

- *Method /detect POST*
- *Method /verify POST*
- *Method /identify POST*
- *Method /face POST*
- *Method /face/id/<id> GET*
- *Method /face/id/<id> PUT*

- *Method /face/id/<id> DELETE*
- *Method /face/meta/<meta> GET*
- *Method /faces GET*
- *Method /faces/gallery/<gallery> GET*
- *Method /meta GET*
- *Method /galleries GET*
- *Method /galleries/<gallery> POST*
- *Method /galleries/<gallery> DELETE*
- *Method /docs GET*
- *Method /docs/<version> GET*
- *Method /person/id/<id> GET*
- *Method /history/search POST*

Method /detect POST

This method detects faces in the provided image and recognize gender, age, and emotions, given the request parameters. You can either upload the image as multipart/form-data or provide the image URL.

Parameters:

- `photo`: an uploaded image, or a publicly accessible URL, containing the image
- `gender`: if true, return gender
- `age`: if true, return age
- `emotions`: if true, return emotions

Returns:

- A list of rectangles, containing the detected faces

Example

Request

```
POST /v1/detect/ HTTP/1.1
Host: 192.168.113.76:8000
Connection:close
Authorization: Token BpdNA6eaU1N9bPhXVSK1r92_SF0ODPOU
Content-Type: application/json
Content-Length: 108

{
  "photo": "https://static.findface.pro/sample.jpg",
  "emotions": true,
```

(continues on next page)

(continued from previous page)

```
"gender": true,  
"age": true  
}
```

Response

```
HTTP/1.1 200 OK  
Date: Thu, 06 Apr 2017 12:38:40 GMT  
Server: TornadoServer/4.4.2  
Content-Length: 120  
Content-Type: application/json; charset=UTF-8  
  
{  
  "faces": [  
    {  
      "age": 26,  
      "emotions": [  
        "neutral",  
        "sad"  
      ],  
      "gender": "female",  
      "x1": 595,  
      "x2": 812,  
      "y1": 127,  
      "y2": 344  
    }  
  ]  
}
```

Method /verify POST

This method is used to verify that two faces belong to the same person, or, alternatively, measures the similarity between the two faces. You can choose between these two modes by setting the `threshold` parameter.

In the case, when a binary decision is required, the user should pass a value for the `threshold` parameter. You can use one of the 3 *preset values*: `strict`, `medium` and `low` with the former aimed at minimizing the false accept rates and the latter being somewhat more permissive. You can also set a user-defined value.

In the case, when you need to calculate similarity of different persons or find similar people rather than verify identity, pass `none` to the `threshold` parameter.

Note: If no threshold level is specified, it is set to the default value 0.75.

Tip: Please feel free to contact us if you need to tune the threshold value for your specific use-case and/or dataset.

Parameters:

- `photo1`: the first uploaded image or an external URL

- `photo2`: the second uploaded image or an external URL
- `bbox1` [optional]: array of bounding boxes for the faces on the first photo
- `bbox2` [optional]: array of bounding boxes for the faces on the second photo
- `threshold` [optional]: one of “strict”, “medium”, “low” or “none”, or a value between 0 and 1. Default is 0.75.
- `mf_selector` [optional]: specifies behavior in a case of multiple faces on a photo; one of:
 - “reject”: return an error if more than one face was detected on any of image
 - “biggest” [default]: add the biggest face on the image
 - “all”: verify all faces, found on both images.

Note: Note that providing `bbox1` or `bbox2` argument overrides the value of this parameter.

Returns:

- binary verification result, only returned if threshold was not set to none. Each pair of faces is given it's own result. The given pair of photos is also provided with the verification result. It will be true if each face on the first photo has a match on the second.
- the coordinates of the bounding boxes with the faces on the images
- the algorithm's confidence in the decision, measured from 0 to 1

Example

Request

```
POST /v0/verify/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "photo1": "http://static.findface.pro/sample.jpg",
  "photo2": "http://static.findface.pro/sample2.jpg"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    {
```

(continues on next page)

(continued from previous page)

```

    "bbox1": {
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    },
    "bbox2": {
      "x1": 78,
      "x2": 185,
      "y1": 114,
      "y2": 222
    },
    "confidence": 0.4206026792526245,
    "verified": true
  }
],
"verified": true
}

```

Method /identify POST

This method is used to search through the face database. The method returns at most *n* faces (one by default), which are the most similar to the specified face, and the similarity is above the specified *threshold*. You can optionally specify a gallery id to check a photo only against photos in this gallery.

Parameters:

- *photo*: the uploaded image, or an external URL
- *x1*, *y1*, *x2*, *y2* [optional]: coordinates of a bounding box of the face on the photo
- *threshold* [optional]: one of “strict”, “medium”, “low” or “none”, or a value between 0 and 1. Default is 0.75.
- *n* [optional]: maximum number of closest faces to return, 1 by default
- *strict* [optional]: specifies behavior in case if one or several *tntapi* shards are out of service. This parameter takes priority over the *tntapi_ignore_search_errors* parameter from the *findface-facenapi configuration file*.
 - True: return an error if some *tntapi* shards are out of service
 - False [default]: use available *tntapi* shards to obtain face identification results, indicating the number of available servers vs the total number of servers in the *X-Live-Servers* header.
- *mf_selector* [optional]: specifies behavior in case if multiple faces are detected on the photo or inside the provided bounding box:
 - “reject”: return an error if more than one face was detected on any of image
 - “biggest” [default]: identify the biggest face on the image
 - “all”: identify all faces, found on the image.

Returns:

- A map where keys are array representations of bounding boxes of faces on provided photo and values are arrays face objects, along with match confidence, measured from 0 (lowest) to 1 (highest)

Example**Request**

```
POST /v0/identify/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "n": 10,
  "photo": "http://static.findface.pro/sample.jpg"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": {
    "[419, 236, 345, 311]": [
      {
        "confidence": 1,
        "face": {
          "galleries": ["default", "ppl"]
          "id": 316275,
          "meta": "Sam Berry",
          "photo": "http://static.findface.pro/sample.jpg",
          "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
          "timestamp": "2016-07-01T12:18:27.477653",
          "x1": 236,
          "x2": 311,
          "y1": 345,
          "y2": 419
        }
      },
      {
        "confidence": 0.723975,
        "face": {
          "galleries": ["default", "ppl"]
          "id": 316283,
          "meta": "Sam Berry",
          "photo": "http://test.flexify.io/img/sample2.jpg",
          "photo_hash": "9b1dd93259fe87df122cd678ce95b9f9",
          "timestamp": "2016-07-01T13:19:36.376548",
```

(continues on next page)

(continued from previous page)

```

        "x1": 78,
        "x2": 185,
        "y1": 114,
        "y2": 222
      }
    }
  ]
}
}

```

Method /face POST

Processes the uploaded image or provided URL, detects faces and adds the detected faces to the searchable database. If there are multiple faces on the photos, only the biggest face is added by default. You can add a custom string meta, such as name or ID, which uniquely identifies a person. Multiple face objects may have the same meta. We recommend that you don't assign the same meta to different persons. Thus when using person's name as a meta, make sure that all names are unique. You can optionally prefix it with a gallery id to upload into non-default gallery.

Parameters:

- `photo`: an uploaded image, or a publicly accessible URL, containing the image
- `meta` [optional]: some user-defined string identifier
- `bbox` [optional]: array of bounding boxes specifying face locations on the image
- `mf_selector` [optional]: specifies behavior in case if there are multiple faces found on the image or inside the specified rectangle; one of:
 - `"reject"`: return an error if more than one face was detected
 - `"biggest"` [default]: add the biggest face on the image
 - `"all"`: add all faces, found on the image. Please note that the meta will be the same for all faces added
- `galleries` [optional]: list of gallery names
- `cam_id` [optional]: UUID of the camera

Returns:

- A JSON representation of the added faces or a failure reason
- In the case multiple faces are detected and `mf_selector` is set to `reject`, this method returns 400 Bad Request and a list of bounding box coordinates for each detected face.

Example #1

Request

```

POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccd57a45a5c6d9aa7602e
Content-Type: application/json

```

(continues on next page)

(continued from previous page)

```
Content-Length: [length]

{
  "meta": "Sam Berry",
  "photo": "http://static.findface.pro/sample.jpg",
  "galleries": ["gall", "nicepl"]
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 06:04:02 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: [length]

{
  "results": [
    {
      "galleries": ["default", "gall", "nicepl"]
      "id": 2334,
      "meta": "Sam Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:11:29.425339",
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    }
  ]
}
```

Example #2

Request

```
POST /v0/face/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "mf_selector": "reject",
  "photo": "http://static.findface.pro/sample-multiface.jpg"
}
```

Response

```
HTTP/1.1 400 Bad Request
Date: Mon, 13 Jun 2016 06:04:02 GMT
```

(continues on next page)

(continued from previous page)

```
Content-Type: application/json; charset=UTF-8
Content-Length: [length]
```

```
{
  "code": 400,
  "faces": [
    {
      "x1": 1952,
      "x2": 2137,
      "y1": 838,
      "y2": 1023
    },
    {
      "x1": 1766,
      "x2": 1952,
      "y1": 1312,
      "y2": 1498
    },
    {
      "x1": 1385,
      "x2": 1540,
      "y1": 939,
      "y2": 1094
    },
    {
      "x1": 2452,
      "x2": 2607,
      "y1": 664,
      "y2": 818
    },
    {
      "x1": 1609,
      "x2": 1764,
      "y1": 767,
      "y2": 922
    }
  ],
  "reason": "Too many faces: 5"
}
```

Method /face/id/<id> GET

Returns detailed information about the face with id = FaceID.

Parameters:

- This method doesn't accept any additional parameters.

Returns:

- A JSON representation of the face with id = FaceID.

Example

Request

```
GET /v0/face/id/2333/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "galleries": ["default", "ppl"]
  "id": 2333,
  "meta": "Sam Berry",
  "photo": "http://static.findface.pro/sample.jpg",
  "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
  "timestamp": "2016-06-13T11:06:42.075754",
  "x1": 225,
  "x2": 307,
  "y1": 345,
  "y2": 428
}
```

Method `/face/id/<id> PUT`

This method can be used to modify certain fields of the face object with `id = FaceID`. Currently only changes to the meta attribute are supported.

Parameters:

- `meta`: new meta string
- `person_id`: unique identifier of the person
- `galleries`: JSON dictionary with one key and one value. Either `{"add":["list", "of", "galleries"]}`, `{"del":["list", "of", "galleries"]}`, `{"set":["list", "of", "galleries"]}`. Allows you to add face to galleries, remove from galleries or replace gallery list completely.

Returns:

- A JSON representation of the updated face with `id = FaceID`

Example

Request

```
PUT /v0/face/id/5/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]

{
  "meta": "Sam Berry #2"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "id": 2333,
  "meta": "Sam Berry #2",
  "photo": "http://static.findface.pro/sample2.jpg",
  "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
  "timestamp": "2016-06-13T11:06:42.075754",
  "x1": 225,
  "x2": 307,
  "y1": 345,
  "y2": 428
}
```

Method /face/id/<id> DELETE

Deletes a face with the id = FaceId.

Parameters:

- This method does not accept any additional parameters.

Returns:

- HTTP 204 No Content in the case of success, or the reason of failure

Example

Request

```
DELETE /v0/face/id/2332/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token ca7916cdac260628c411cb5d895dd566
Content-Length: 0
```

Response

```
HTTP/1.1 204 No Content
```

Method /face/meta/<meta> GET

Returns the list of faces with a given meta string. Note that the method is case-sensitive, so the given meta has to fully match the one from the database. A meta string has to be URL encoded, and according to the standard, spaces should be encoded as `%20` (not `+`) in this part of the URL.

Parameters:

- This method doesn't accept any additional parameters.

Returns:

- Returns the list of faces with a `<meta>`.

Example

Request

```
GET /v0/face/meta/Sam%20Berry/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    {
      "galleries": ["default", "ppl"],
      "id": 2333,
      "meta": "Sam Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:06:42.075754",
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    },
    {
      "galleries": ["default", "ppl"],
      "id": 2378,
```

(continues on next page)

(continued from previous page)

```

        "meta": "Sam Berry",
        "photo": "http://static.findface.pro/sample2.jpg",
        "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
        "timestamp": "2016-06-13T11:06:42.075754",
        "x1": 46,
        "x2": 502,
        "y1": 472,
        "y2": 789
    }
]
}

```

Method /faces GET

Parameters

- This method doesn't accept any additional parameters.

Returns:

- Returns the list of all faces stored in database.

Example

Request

```

GET /v0/faces/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e

```

Response

```

HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    {
      "galleries": ["default", "ppl"]
      "id": 2333,
      "meta": "Sam Berry",
      "photo": "http://static.findface.pro/sample.jpg",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "timestamp": "2016-06-13T11:06:42.075754",
      "x1": 225,
      "x2": 307,
      "y1": 345,
      "y2": 428
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```
    },  
    {  
      "galleries": ["default", "ppl"]  
      "id": 2335,  
      "meta": "",  
      "photo": "http://static.findface.pro/sample2.jpg",  
      "photo_hash": "9879efb38d2dae550460c9edb6f36982",  
      "timestamp": "2016-06-13T11:34:57.275394",  
      "x1": 8,  
      "x2": 152,  
      "y1": 406,  
      "y2": 550  
    }  
  ]  
}
```

Method /faces/gallery/<gallery> GET

Returns the list of all faces stored in a specified gallery.

Method /meta GET

This method retrieves all the meta string stored in the database along with one of the associated faces. To get more faces call GET /v0/face/meta/[Meta].

Parameters:

- This method doesn't accept any additional parameters

Returns:

- A list of objects containing meta string, number of faces marked with this meta string, and JSON representation of the first face object marked with this meta string

Example

Request

```
GET /v0/meta/ HTTP/1.1  
Host: 127.0.0.1  
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
```

Response

```
HTTP/1.1 200 OK  
Date: Mon, 13 Jun 2016 12:23:56 GMT  
Content-Type: application/json  
Content-Length: [length]
```

(continues on next page)

(continued from previous page)

```

{
  "results": [
    {
      "count": 1,
      "face": {
        "galleries": ["default", "ppl"]
        "id": 2333,
        "meta": "Sam Berry",
        "photo": "http://static.findface.pro/sample.jpg",
        "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
        "timestamp": "2016-06-13T11:06:42.075754",
        "x1": 225,
        "x2": 307,
        "y1": 345,
        "y2": 428
      },
      "meta": "Sam Berry"
    },
    {
      "galleries": ["default", "ppl"]
      "count": 15,
      "face": {
        "id": 2563,
        "meta": "Angelina Jolie",
        "photo": "http://static.findface.pro/sample2.jpg",
        "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
        "timestamp": "2016-06-13T11:06:42.075754",
        "x1": 225,
        "x2": 307,
        "y1": 345,
        "y2": 428
      },
      "meta": "Angelina Jolie"
    }
  ]
}

```

Method /galleries GET

List all your galleries.

Returns:

- A JSON dictionary with list of gallery ids

Example

Request

```

GET /v0/galleries/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e

```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]

{
  "results": [
    "default",
    "test"
    "57bd75f941741d36ab4614a0",
    "57bd76a241741d377bf881ac",
  ]
}
```

Method /galleries/<gallery> POST

Creates a new gallery under a given name. The gallery name can contain English letters, numbers, underscore and minus sign ([a-zA-Z0-9_-]+). It shouldn't be longer than 48 characters.

Parameters:

This method doesn't accept any additional parameters.

Example

Request

```
POST /v0/galleries/testgal HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
```

Response

```
HTTP/1.1 201 Created
Date: Mon, 13 Jun 2016 06:04:02 GMT
```

Method /galleries/<gallery> DELETE

Deletes the gallery and all faces in it.

Returns:

- HTTP 204 No content.

Example

Request

```
DELETE /v0/galleries/niceppl HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Length: 0
```

Response

```
HTTP/1.1 204 No Content
```

Method /docs GET

Lists documented API versions. Available without authorization.

Method /docs/<version> GET

Gets documentation for specified API version. Available without authorization.

Method /person/id/<id> GET

Parameters:

- This method doesn't accept any additional parameters

Returns:

- A JSON representation of the person with id = FaceID

Example

Request

```
GET /person/history/id/2001 HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]
{
  "cam_ids": [1, 25, 26, 27],
  "start": "2016-06-13T11:00:00.000000",
  "end": "2016-06-14T11:00:00.000000"
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 13 Jun 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]
{
  "results":
  [
    {
      "person_id": 2001,
      "face_id": 240344,
      "cam_id": 25,
      "meta": "Sam Berry",
      "screenshot": "https://static.findface.pro/57726179d6946f02f3763824/
↪dc7ac54590729669ca869a18d92cd05e_thumb.jpg",
      "timestamp": "2016-06-13T11:06:42.075754",
    },
    {
      "person_id": 2001,
      "face_id": 240422,
      "cam_id": 25,
      "meta": "Sam Berry",
      "screenshot": "https://static.findface.pro/57726179
d6946f02f3763824/dc7ac54590729669ca869a18d92cd05e_thumb.jpg",
      "timestamp": "2016-06-13T11:08:44.073452",
    }
  ]
}
```

Method /history/search POST

This method retrieves all events from camera history of the given parameters.

Parameters:

- "person_id" [optional]: unique person id
- "cam_ids" [optional]: array of camera ids.
- "start" [optional]: search history interval, start time as ISO8601 string
- "end" [option]: search history interval, end time as ISO8601 string
- "friend" [optional]: friend or foe identification
- "limit" [optional]: records per page, if 0 (default) - unlimited

Returns:

- A list of history events.
- next_page: URL to the next page (path and query portion only). If no such field in response - no more pages exist.

Example

Request

```
POST /v0/history/search HTTP/1.1
Host: 127.0.0.1
Authorization: Token e93437ccdae66d57a45a5c6d9aa7602e
Content-Type: application/json
Content-Length: [length]
{
  "limit": 2,
}
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 12 Oct 2016 12:23:56 GMT
Content-Type: application/json
Content-Length: [length]
{
  "next_page": "/v0/history/search?max_id=4",
  "results": [
    {
      "friend": false,
      "meta": "",
      "photo_hash": "9fda49f2444f93c33ad8aa914e20e53b",
      "cam_id": "12345678123456781234567812345678",
      "person_id": 8,
      "timestamp": "2016-10-11T14:36:27.450000",
      "photo": "",
      "id": 20146,
      "y1": 77,
      "x1": 285,
      "x2": 552,
      "y2": 345
    },
    {
      "friend": false,
      "meta": "",
      "photo_hash": "dc7ac54590729669ca869a18d92cd05e",
      "cam_id": "12345678123456781234567812345678",
      "person_id": 8,
      "timesamp": "2016-10-12T12:57:07.509000",
      "photo": "",
      "id": 20147,
      "x1": 236,
      "y1": 345,
      "x2": 311,
      "y2": 419
    }
  ]
}
```

9.1.3 Galleries

There is always a gallery titled default. Faces are always added to the default gallery and cannot be removed from it. The default gallery cannot be stopped.

In addition to the default gallery, you can create custom galleries and add faces into them. Custom galleries allows you to have several datasets in one environment. This might be useful if you need to search through different face lists, for example if you have several products or several customers with different face datasets.

To create a custom gallery, use the method *POST /galleries/gallery_name*.

By default, all API methods apply to the default gallery. However, you can narrow down usage of most methods to a specific gallery (see the table below). To do so, provide the gallery name in your API request URI. For example, to search a person in a gallery 'ppl', use *POST /faces/gallery/ppl/identify/* instead of *POST /identify/*.

Default gallery method	Custom gallery method
<i>POST /identify/</i>	<i>POST /faces/gallery/<Gallery>/identify/</i>
<i>GET /faces/</i>	<i>GET /faces/gallery/<Gallery>/</i>
<i>GET /face/meta/<Meta></i>	<i>GET /face/gallery/<Gallery>/meta/<Meta></i>
<i>GET /meta/</i>	<i>GET /meta/gallery/<Gallery>/</i>

9.1.4 Methods for Video Face Detection

These methods extend *general API methods* of FindFace Enterprise Server SDK.

In this section:

- *Method /camera POST*
- *Method /camera GET*
- *Method /camera/<camera_id> GET*
- *Method /camera/<camera_id> PUT*
- *Method /camera/<camera_id> DELETE*

Method /camera POST

Description

Creates a new camera.

Parameters:

- `meta` [optional]: some user-defined string identifier
- `enabled` [boolean]: enable video processing in FindFace web interface
- `url` [optional]: url address of the camera's stream
- `detector` [optional]: some user-defined string identifier

- `rot [W,H,X,Y]` [optional]: enable detecting and tracking faces only inside a clipping rectangle (ROT, region of tracking).
- `roi [W,H,X,Y]` [optional]: enable posting faces detected only inside a region of interest (ROI).

Returns:

A JSON representation of the added camera or a failure reason.

Example**Request**

```
POST /v1/camera/ HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
Content-Type: application/json
Content-Length: [length]

{
  "meta": "meta",
  "enabled": true,
  "url": "http://test.com:1234/stream.flv",
  "detector": "detec1"
}
```

Response

```
HTTP/1.1 201 Created
Content-Length: [length]
Content-Type: application/json; charset=UTF-8

{
  "meta": "meta",
  "enabled": true,
  "url": "http://test.com:1234/stream.flv",
  "detector": "detec1",
  "id": "7bb35e9d-9f4f-4e5b-8811-e1dded6de811"
}
```

Method /camera GET**Description**

Lists all cameras.

Parameters:

This method doesn't accept any additional parameters.

Returns:

The list of all cameras.

Example**Request**

```
GET /v1/camera HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
```

Response

```
HTTP/1.1 200 OK
Content-Length: [length]
Date: Thu, 13 Oct 2016 12:14:22 GMT
Content-Type: application/json; charset=UTF-8
[
  {
    "enabled": true,
    "id": "32bc21fb-0aa2-4d17-88bb-1a2bf76d88ea",
    "meta": "Camera 1",
    "url": "rtsp://127.0.0.1/Streaming/Channels/1"
  },
  {
    "enabled": true,
    "id": "32bc21fb-0aa2-4d17-88bb-1a2bf76d88ea",
    "meta": "Camera 1",
    "roi": [
      200,
      300,
      400,
      500
    ],
    "rot": [
      100,
      100,
      800,
      800
    ],
    "url": "rtsp://127.0.0.1/Streaming/Channels/1"
  }
]
```

Method /camera/<camera_id> GET**Description**

Gets information about the camera with `id = camera_id`.

Parameters:

This method doesn't accept any additional parameters.

Returns:

Info about the camera or a failure reason.

Example**Request**

```
GET /v1/camera/32bc21fb-0aa2-4d17-88bb-1a2bf76d88ea HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
```

Response

```
HTTP/1.1 200 OK
Content-Length: [length]
Content-Type: application/json; charset=UTF-8
{
  "enabled": true,
  "id": "32bc21fb-0aa2-4d17-88bb-1a2bf76d88ea",
  "meta": "Camera 1",
  "roi": [
    200,
    300,
    400,
    500
  ],
  "rot": [
    100,
    100,
    800,
    800
  ],
  "url": "rtsp://127.0.0.1/Streaming/Channels/1"
}
```

Method /camera/<camera_id> PUT**Description**

This method can be used to modify certain fields of the camera object with `id = camera_id`.

Parameters:

- `meta` [optional]: new meta string
- `url` [optional]: url address of the camera's stream

- `rot [W,H,X,Y]` [optional]: enable detecting and tracking faces only inside a clipping rectangle (ROT, region of tracking).
- `roi [W,H,X,Y]` [optional]: enable posting faces detected only inside a region of interest (ROI).

Returns:

A JSON representation of the updated camera with `id = <camera_id>`.

Example #1

Request

```
PUT /v1/camera/b28a898b-6334-4d37-8888-c9dd858ddc47 HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
Content-Type: application/json
Content-Length: [length]
{
  "meta": "newinfo",
  "url": "http://zzzz.com:1234/stream.flv"
}
```

Response

```
HTTP/1.1 200 OK
Content-Length: [length]
Content-Type: application/json; charset=UTF-8
{
  "url": "http://zzzz.com:1234/stream.flv",
  "id": "b28a898b-6334-4d37-8888-c9dd858ddc47",
  "meta": "newinfo"
}
```

Example #2

Request

```
PUT /v1/camera/b28a898b-6334-4d37-8888-c9dd858ddc47 HTTP/1.1
Host: 127.0.0.1
Authorization: Token 1234567890qwertyuiop
Content-Type: application/json
Content-Length: [length]
{
  "rot": [
    120,
    120,
    35,
    50
  ],
  "roi": [
```

(continues on next page)

(continued from previous page)

```
    100,  
    100,  
    40,  
    50  
  ]  
}
```

Response

```
HTTP/1.1 200 OK  
Content-Length: [length]  
Content-Type: application/json; charset=UTF-8  
{  
  "id": "b28a898b-6334-4d37-8888-c9dd858ddc47",  
  "rot": [  
    120,  
    120,  
    35,  
    50  
  ],  
  "roi": [  
    100,  
    100,  
    40,  
    50  
  ]  
}
```

Method /camera/<camera_id> DELETE

Description

Deletes the camera with `id = camera_id`.

Parameters:

This method doesn't accept any additional parameters.

Returns:

HTTP 204 No Content in the case of success, or the reason of failure.

Example

Request

```
DELETE /v1/camera/b28a898b-6334-4d37-8888-c9dd858ddc47 HTTP/1.1  
Host: 127.0.0.1  
Authorization: Token 1234567890qwertyuiop  
Content-Length: 0
```

Response

`HTTP 204 No Content`

Tip: You can also find the REST API documentation on our [website](#) and at `http://<facenapi_ip>:8000/v1/docs`.

9.2 Plugins

Plugins can interact with FindFace Enterprise Server SDK on the following levels:

1. Direct access to the `findface-facenapi` context. This is the basic level of interaction which allows you to directly access the database, extractors, decoders and detectors.

Tip: See *Basics* and *Classes and Methods* for the full reference.

2. Through HTTP API handlers implemented to the FindFace API Environment in addition to the embedded *REST API*. You can write your own HTTP API handler or inherit from the following ready-to-use ones:

- `facenapi.core.http.base_handler.BaseHandler` implements the FindFace Web Interface (without Video Processing)
- `facenapi.core.http.base_handler.BaseVideoHandler` implements Video Processing

Tip: See *Build Plugin around HTTP Handler* for examples.

Note: Both `BaseHandler` and `BaseVideoHandler` provide token-based authentication.

9.2.1 Basics

This section will give you a general idea of the `findface-facenapi` context.

In this section:

- *Wrapping MongoDB Objects into the Model Class*
 - *Implementing Authentication*
 - *Common Object Types*
 - *Rectangle*
 - `facenapi.core.extractors.ExtractionFace`
 - *Face object*
 - *Implementing Collections of MongoDB Objects*

Wrapping MongoDB Objects into the Model Class

Information about faces, galleries, cameras, persons, as well as authentication data are stored in the MongoDB database. The *MongoStore* class provides a base for interaction between the *findface-facenapi* component and MongoDB objects, being a wrapper around the MongoDB object collection. *MongoStore* wraps each object, returned in a MongoDB query, into an instance of the *Model* class (*[Objects].Model*, e.g. *Faces.Model*, *Galleries.Model*, etc.), so that the object can have its own methods and properties, and can be further processed through the *findface-facenapi* context.

Implementing Authentication

While the *MongoStore* class creates a base for interaction between *findface-facenapi* and MongoDB objects (see *Wrapping MongoDB Objects into the Model Class*), it doesn't call for authentication. To provide it, the following *MongoStore* subclasses are used:

- The *UserStore* class provides user-based authentication. It ensures that passing an unauthenticated request will cause an error, instead of security vulnerabilities. Each class that represents a collection of objects (faces, galleries, persons, etc.) inherits from *UserStore*.
- The *Users* class stores the authentication credentials as user objects. Both a user id and user object are qualified for authentication and can be passed to methods that require it.

Common Object Types

Rectangle

Stores coordinates of a face bounding box in the original image as *Rectangle('left', 'top', 'right', 'bottom')*, where:

- *'left'*: x coordinate of the top-left corner of the bounding box, *float*.
- *'top'*: y coordinate of the top-left corner of the bounding box, *float*.
- *'right'*: x coordinate of the bottom-right corner of the bounding box, *float*.
- *'bottom'*: y coordinate of the bottom-right corner of the bounding box, *float*.

Note: A bbox's coordinates can lie outside image boundaries and be negative.

facenapi.core.extractors.ExtractionFace

The *facenapi.core.extractors.ExtractionFace* object represents a dictionary of face data returned from *ctx.extractor.extract()*. It can be converted into a *Faces.Model* instance by using *Faces.Model.from_extraction_face()*.

The *facenapi.core.extractors.ExtractionFace* dictionary have the following keys:

- *bbox*: coordinates of the face region in the original image (bbox)
- *normalized*: normalized face image, bytes of the .png file
- *facen*: (optional) face feature vector, *base64*
- *gender*: (optional) 'male' or 'female', *str*
- *age*: (optional) estimated age, *float*

- `emotions`: (optional) emotions with relevant probability, *namedtuple*
- `'score'`: face image quality, *float*
- `'photo_hash'`: hash of the original photo, *str*

Face object

Being wrapped into the `Faces.Model` class, a face object represents a dictionary *face* with the following keys:

- `"id" (number)`: unique identifier of the face
- `'bbox'`: coordinates of the face region in the original image (bbox)
- `'facen'`: (optional) face feature vector, *base64*
- `'gender'`: (optional) 'male' or 'female', *str*
- `'age'`: (optional) estimated age, *float*
- `'emotions'`: (optional) 2 most prevalent emotions, *tuple*
- `'normalized'`: normalized face image, bytes of the *.png* file
- `'score'`: face image quality, *float*

Note: You can interpret the face image quality as follows. 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.

- `'timestamp'`: (optional) time of the face object creation as ISO8601 string
- `'photo'`: (optional) URL of the original image used to create the face object
- `'photo_hash'`: hash of the original photo, *str*
- `'thumbnail'`: (optional) URL of the face thumbnail
- `'meta'`: (optional) metadata string that you can use to store any information associated with the face
- `'galleries'`: (optional) list of galleries that feature the face.

A collection of face objects is stored in the `Faces` class.

Implementing Collections of MongoDB Objects

Each collection of MongoDB objects is implemented as the `[Objects]` class and consists of instances of the relevant `[Objects].Model` class. For example, to work with a face and person collections, you have to refer respectively to the `Faces` and `Persons` classes; with a face and person objects - to `Faces.Model` and `Persons.Model`, etc.

Each collection class inherits from *UserStore*.

9.2.2 Classes and Methods

This section provides a full reference to the `findface-facenapi` context you can use in your plugin.

Tip: Use `ctx` from the activate parameters to access classes and methods directly from a plugin, e.g. use `ctx.faces.Model.from_extraction_face(eface)` to invoke `Faces.Model.from_extraction_face()`. To refer to classes and methods in a class that inherits from a HTTP API handler, use `self.ctx`: `self.ctx.faces.Model.from_extraction_face(eface)`.

In this section:

- *Basic Classes*
- *User Enrollment*
- *Working with Faces and Galleries*
- *Adding Camera*
- *Working with Persons*
- *Auxiliary Classes*

Basic Classes

class MongoStore

Information about faces, galleries, cameras, persons, as well as authentication data are stored in the MongoDB database. The `MongoStore` class provides a base for interaction between `findface-facenapi` and MongoDB objects, being a wrapper around the MongoDB object collection. `MongoStore` wraps each object returned in a MongoDB query into an instance of the `Model` class (`[Objects].Model`), so that the object can have its own methods and properties and can be further processed through the `findface-facenapi` context.

Note: Objects can be of any type: face, gallery, camera, user, etc. `MongoStore` wraps them respectively into `Faces.Model`, `Galleries.Model`, `Cameras.Model`, `Users.Model`, etc.

find (*self*, *filters*, *sort=None*, *skip=None*, *limit=None*)

Searches for MongoDB objects that meet given requirements.

Parameters

- **filters** (*elements of the MongoDB query dictionary or None*) – filters
- **sort** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for sorting MongoDB objects
- **skip** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for skipping some of MongoDB objects
- **limit** (*int or None*) – (optional) maximum number of returned objects

Returns Objects of a certain type

Return type List of instances of the `[Objects].Model` class

find_one (*self*, *filters*, *sort=None*, *skip=None*)

Returns the first MongoDB object that meets given requirements.

Parameters

- **filters** (*elements of the MongoDB query dictionary or None*) – filters
- **sort** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for sorting MongoDB objects
- **skip** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for skipping some of MongoDB objects

Returns Object of a certain type

Return type [Objects].Model instance

count (*self, filters, sort=None, skip=None, limit=None*)

Returns the total number of MongoDB objects of a certain type.

Parameters

- **filters** (*elements of the MongoDB query dictionary or None*) – filters
- **sort** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for sorting MongoDB objects
- **skip** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for skipping some of MongoDB objects
- **limit** (*int or None*) – (optional) maximum number of counted objects

Returns Number of objects of a certain type

Return type int

remove_one (*self, filters, sort=None*)

Removes the first MongoDB object that meets given requirements.

Parameters

- **filters** (*elements of the MongoDB query dictionary or None*) – filters
- **sort** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for sorting MongoDB objects

Returns Removed object

Return type [Objects].Model instance

update_one (*self, filters, update, sort=None, return_document=ReturnDocument.AFTER, **kwargs*)

Updates a specified MongoDB object.

Parameters

- **filters** (*elements of the MongoDB query dictionary or None*) – filters
- **update** – object property to update
- **sort** (*elements of the MongoDB query dictionary or None*) – (optional) criteria for sorting MongoDB objects

Returns Updated object

Return type [Objects].Model instance

wrap_in_model (*self, obj*)

Wraps a MongoDB object into the Model class.

Parameters *obj* (*dict* or *OrderedDict*) – MongoDB object

Returns Object of a certain type

Return type [Objects].Model instance

class UserStore (*MongoStore*)

Inherits from the *MongoStore* class. While *MongoStore* creates a base for interaction between findface-facenapi and MongoDB objects, the *UserStore* class provides user-based authentication for such interaction. It ensures that passing an unauthenticated request will cause an error instead of security vulnerabilities.

find (*self, user, filters, sort=None, skip=None, limit=None*)

Searches for MongoDB objects that meet given requirements.

Parameters

- **user** (ObjectId or Users.Model) – user id or user object (for authentication)
- **filters** (*elements of the MongoDB query dictionary* or *None*) – filters
- **sort** (*elements of the MongoDB query dictionary* or *None*) – (optional) criteria for sorting MongoDB objects
- **skip** (*elements of the MongoDB query dictionary* or *None*) – (optional) criteria for skipping some of MongoDB objects
- **limit** (*int* or *None*) – (optional) maximum number of returned objects

Returns Objects of a certain type

Return type List of instances of the [Objects].Model class

find_one (*self, user, filters, sort=None, skip=None*)

Returns the first MongoDB object that meets given requirements.

Parameters

- **user** (ObjectId or Users.Model) – user id or user object (for authentication)
- **filters** (*elements of the MongoDB query dictionary* or *None*) – filters
- **sort** (*elements of the MongoDB query dictionary* or *None*) – (optional) criteria for sorting MongoDB objects
- **skip** (*elements of the MongoDB query dictionary* or *None*) – (optional) criteria for skipping some of MongoDB objects

Returns Object of a certain type

Return type [Objects].Model instance

count (*self, user, filters, sort=None, skip=None, limit=None*)

Returns the total number of MongoDB objects of a certain type.

Parameters

- **user** (ObjectId or Users.Model) – user id or user object (for authentication)
- **filters** (*elements of the MongoDB query dictionary* or *None*) – filters

- **sort** (*elements of the MongoDB query dictionary or `None`*) – (optional) criteria for sorting MongoDB objects
- **skip** (*elements of the MongoDB query dictionary or `None`*) – (optional) criteria for skipping some of MongoDB objects
- **limit** (*`int` or `None`*) – (optional) maximum number of counted objects

Returns Number of objects of a certain type

Return type `int`

remove_one (*self, user, filters, sort=None*)

Removes the first MongoDB object that meets given requirements.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)
- **filters** (*elements of the MongoDB query dictionary or `None`*) – filters
- **sort** (*elements of the MongoDB query dictionary or `None`*) – (optional) criteria for sorting MongoDB objects

Returns Removed object

Return type `[Objects].Model` instance

update_one (*self, user, filters, update, sort=None, return_document=ReturnDocument.AFTER, **kwargs*)

Updates a specified MongoDB object.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)
- **filters** (*elements of the MongoDB query dictionary or `None`*) – filters
- **update** – object property to update
- **sort** (*elements of the MongoDB query dictionary or `None`*) – (optional) criteria for sorting MongoDB objects

Returns Updated object

Return type `[Objects].Model` instance

update_many (*self, user, filters, update, sort=None, return_document=ReturnDocument.AFTER, **kwargs*)

Updates specified MongoDB objects.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)
- **filters** (*elements of the MongoDB query dictionary or `None`*) – filters
- **update** – object property to update
- **sort** (*elements of the MongoDB query dictionary or `None`*) – (optional) criteria for sorting MongoDB objects

Returns Updated objects

Return type List of instances of the `[Objects].Model` class

User Enrollment

class Users (*MongoStore*)

Represents a collection of user objects. Each user object in the collection has the following properties:

- `_id` - primary key, *ObjectId*
- `token` - authentication token, must be unique, *string*
- `active` - allows user to perform requests, *bool*

Each face and gallery in the system must belong to a certain user. User objects are also used for authentication.

add(self, user)

Enrolls a new user to the MongoDB database and returns it as an instance of the `Users.Model` class.

Parameters `user` (*dictionary*) – user data

Returns User object

Return type `Users.Model` instance

Working with Faces and Galleries

class Faces.Model (*OrderedDict*)

Represents a face object.

classmethod from_extraction_face (*cls, eface*)

Creates a face object as an instance of the `Faces.Model` class.

Parameters `eface` (*dictionary*) – set of face data received from `ctx.extractor.extract()`

Returns Face object

Return type `Faces.Model` instance

class Faces (*UserStore*)

Represents a collection of faces. Each face in the collection has the following properties:

- `_id` - primary key, *uint64*
- `owner` - owner id, *ObjectId*
- `facen` - feature vector, *base64*
- `bbox` - coordinates of the face region in the original image (`bbox`), `Rectangle(self['x1'], self['y1'], self['x2'], self['y2'])`
- `photo_hash` - md5 of the original image
- `gallery` - galleries that feature the face, *list*
- `meta` - metadata, string or None

add(self, user, face)

Enrolls a face object to MongoDB, adding such parameters as `_id` and `owner`, and returns the updated face object. If the face has no id, this method generates a new id and inserts it into the face object. If the added face already has an id, this method fails at the attempt to insert a new id. In the case of a conflict, this method retries with another id up to 3 times.

Important: As this method updates only MongoDB and not the facen storage (`tntapi`), you will have to call `Faces.add_to_galleries()` after this method in order to add a face to a gallery.

Parameters

- **user** (`ObjectId` or `Users.Model`) – face owner passed as a user id or user object
- **face** (`Faces.Model` instance) – face object

Returns Updated face object

Return type `Faces.Model` instance

Usage:

```
face = ctx.faces.Model.from_extraction_face(eface)
face['meta'] = meta
face = await ctx.faces.add(self.user, face)
```

regenerate_id (*self, face*)

Replaces a face's id with a newly generated one.

Parameters **face** (`Faces.Model` instance) – face object

Return type `Void`

add_to_galleries (*self, face, galleries*)

Adds a face to specified galleries in MongoDB and the facen storage (`tntapi`). This method first attempts to add a face to galleries in the facen storage. In the case of success, it updates the `face_gallery` field in MongoDB. If the face already exists in the facen storage gallery, the method generates an error.

Parameters

- **face** (`Faces.Model` instance) – face object
- **galleries** (`list[str]`) – gallery names

Return type `Void`

del_from_galleries (*self, face, galleries*)

Removes a face from specified galleries in MongoDB and the facen storage (`tntapi`). This method first attempts to remove a face from galleries in the facen storage. In the case of success, it updates the `face_gallery` field in MongoDB. If the face doesn't exist in the facen storage gallery, it is considered to be successfully removed.

Parameters

- **face** (`Faces.Model` instance) – face object
- **galleries** (`list[str]`) – gallery names

Return type `Void`

identify (*self, user, gallery, face, limit, threshold, filters=None, ignore_errors=False*)

Searches galleries for faces that resemble a given `face` with matching confidence larger or equal to the `threshold`.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)

- **gallery** – galleries to search in
- **face** (`Faces.Model` or dictionary) – either a face object, or `eface` received from the `ctx.extractor.extract()` method
- **limit** (*int*) – maximum number of returned faces
- **threshold** (*float*) – minimum matching confidence between the given and returned faces, from 0 (lowest) to 1 (highest)
- **filters** (*elements of the MongoDB query dictionary or None*) – (optional) filters
- **ignore_errors** (*bool*) – (optional) If `false` and one or several `tntapi` shards are out of service, an error is returned. If `true`, no error is generated and available `tntapi` shards are used to obtain face identification results, indicating the number of live servers vs the total number of servers in the results.

Returns Results that feature properties of each found face in order of decreasing confidence.

Return type List-like object results

The `results` object features the following properties:

- `results.live_server`: number of `tntapi` shards used to obtain face identification results (only if `ignore_errors=True`)
- `results.total_servers`: total number of `tntapi` shards in the system (only if `ignore_errors=True`)
- `results[x]`: `namedtuple IdentifyResult` featuring face and confidence.
- `results[x].face`: face object
- `results[x].confidence`: matching confidence, *float*

class ServerFaces (*CoreFaces*)

Extends functionality of the `Faces` class. Supports upload of original images, normalized images and thumbnails to the Uploads folder.

gen_ffupload_key (*cls, face, suffix='.jpeg'*)

Populates the `photo`, `thumbnail` and `normalized` fields of a face object or a dictionary with relevant links to the original image, thumbnail and normalized image in the Uploads folder. These links will be used when invoking `ServerFaces.upload()`.

Parameters

- **face** (`Faces.Model` instance or dictionary) – face object or face data
- **suffix** (*str*) – (optional) suffix to the name of the image file

Returns Links

Return type `'%s/%s/%d_%s%s' % (face['owner'], now.strftime('%Y%m%d'), face['_id'], token_hex(6), suffix)`

upload_thumbnail (*self, face, img: Image, url=None*)

Uploads a face thumbnail to the Uploads folder or specified URL.

Parameters

- **face** (`Faces.Model` or dictionary) – either a face object, or `eface` received from the `ctx.extractor.extract()` method
- **img** – original image `facenapi.core.image.Image`

- **url** – upload URL

Return type Void

regenerate_id (*self*, *face*)

Regenerates a face id and URLs of the relevant original image, normalized face image, and the thumbnail (as they contain the face id).

Parameters **face** (`Faces.Model` instance) – face object

Return type Void

upload (*self*, *face*, *img*)

Uploads an original image, normalized face image and a thumbnail to the Uploads folder.

Parameters

- **face** (`Faces.Model` or dictionary) – either a face object, assigned the normalized property, or eface received from the `ctx.extractor.extract()` method
- **img** – original image `facenapi.core.image.Image`

Return type Void

class Galleries (*UserStore*)

Represents a collection of galleries. Each gallery in the collection has the following properties:

- **owner** - owner id, *ObjectId*
- **name** - gallery name, *string*

add (*self*, *user*, *gallery*)

Creates a gallery in MongoDB and returns it as an instance of the `Galleries.Model` class.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)
- **gallery** (*str*) – gallery name

Returns Gallery object

Return type `Galleries.Model` instance

Raises **ValueError** – if the gallery name is not specified or already exists in MongoDB

Adding Camera

class Cameras (*UserStore*)

Represents a collection of cameras for video face detection.

add (*self*, *user*, *camera*)

Adds a camera to your system and returns it as an instance of the `Cameras.Model` class.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)
- **camera** (*dictionary*) – camera data

Returns Camera object

Return type `Cameras.Model` instance

Working with Persons

class `Persons` (`UserStore`)

Represents a collection of persons. Used to implement the advanced functions of *Dynamic Person Creation* and *'Friend and Foe' Identification*.

Each person object in the collection has the following properties:

- `_id`: person_id, `uint64`
- `owner`: owner id, `ObjectId`

add (`self`, `user`, `person`)

Creates a person and returns it as a `Persons.Model` instance.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication and to populate the owner property)
- **person** (`dictionary`) – person data

Returns Person object

Return type `Persons.Model` instance

identify (`self`, `user`, `face`, `gallery`, `threshold`, `create=False`, `filters=None`)

Searches galleries for persons whose faces resemble a given `face` with matching confidence larger or equal to the `threshold`.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)
- **face** (`Faces.Model` or `dictionary`) – either a face object, or `eface` received from the `ctx.extractor.extract()` method
- **gallery** – galleries to search in
- **threshold** (`float`) – minimum matching confidence between the given and returned faces, from 0 (lowest) to 1 (highest)
- **create** (`bool`) – (optional) if `True` and no similar faces were found, the method creates a new person and returns `person_id`
- **filters** (`elements of the MongoDB query dictionary or None`) – (optional) filters

Returns `person_id` of found persons, or `person_id` of a newly created person if no similar faces were found.

Return type `uint64` or `list[uint64]`

is_friend (`self`, `user`, `person_id`, `cam_id`)

Checks if a person is a friend, for a given camera.

Parameters

- **user** (`ObjectId` or `Users.Model`) – user id or user object (for authentication)
- **person_id** (`uint64`) – person_id of the person
- **cam_id** (`str`) – camera id

Returns `True` if the person is a friend, and `False` otherwise.

Return type `bool`

Auxiliary Classes

class Counters (*MongoStore*)

Used to generate a new id for an object.

next (*self, counter*)

Increments the counter and returns its new value.

Parameters **counter** (*int*) – current value of the counter

Returns New value of the counter *seq*

Return type *int*

Raises

- **TypeError** – if the current value is not found
- **ValueError** – if the current value is invalid

9.2.3 Build Plugin around HTTP Handler

Plugins are great as proxy scripts that manage communication between `fkvideo_detector` and `findface-facenapi` and redirect `findface-facenapi` responses to an application that can process and render them. Another practical use case is sending the facial recognition results to a websocket or saving them to a file.

When writing a plugin for these use cases, you can inherit from the following ready-to-use HTTP handlers:

- `facenapi.core.http.base_handler.BaseHandler` implements the FindFace Web Interface (without Video Processing)
- `facenapi.server.base_video_handler.BaseVideoHandler` implements Video Processing.

Note: `BaseVideoHandler` parses the `fkvideo_detector` requests and passes the parsed data to the `process_frame` method.

Note: To refer to the `findface-facenapi` context in a class that inherits from a HTTP API handler, use `self.ctx`, e.g. `self.ctx.faces.Model.from_extraction_face(eface)`.

The following examples will help you use these handlers in your plugin:

Important: By default, `fkvideo_detector` sends API requests directly to `findface-facenapi`. To use a plugin as a proxy script between the components, assign the plugin path to the `request-url` parameter of `fkvideo_detector`. The plugin path is specified inside `app.add_handlers()` in the plugin. It is `/static-demo/frame` for `html-demo-report.py`, and `/demo/frame` for `websocket-demo-plugin`.

- The `html-demo-report.py` plugin identifies faces detected in video by the `fkvideo_detector` component and saves the identification results to a static HTML file.
- The `websocket-demo-plugin` plugin identifies faces detected in video by the `fkvideo_detector` component and sends the identification results to a websocket.

9.2.4 Configure findface-facenapi to Use Plugins

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

1. Put a plugin into a directory of your choice. You can use several directories to store plugins.
2. Open the findface-facenapi configuration file.

```
sudo vi /etc/findface-facenapi.ini
```

Warning: The findface-facenapi.ini 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. Uncomment the plugins_enabled parameter and specify the comma-separated list of plugins to load, or an asterisk (*) to load all plugins.

```
plugins_enabled = '*'
```

Maintenance and Troubleshooting

10.1 Troubleshoot Licensing and NTLS

When troubleshooting licensing and *NTLS*, the first step is to retrieve the licensing information and NTLS status. You can do so by sending an API request to 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 info@ntechlab.com.

10.1.1 Retrieve Licensing Information

To retrieve the FindFace Enterprise Server SDK *licensing* information and NTLS status, execute on the 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": 1520844897,
  "type": "offline (extended)",
  "license_id": "001278983",
  "generated": 487568400,
  "last_updated": 4,
  "valid": {
    "value": true,
    "description": ""
  },
  "source": "/ntech/license/001278983.lic",
  "limits": [
    {
      "type": "time",
      "name": "end",
      "value": 25343
    },
    {
      "type": "number",
      "name": "faces",
      "value": 90071,
      "current": 230258
    },
    {
      "type": "number",
      "name": "cameras",
      "value": 9007,
      "current": 3
    },
    {
      "type": "number",
      "name": "extraction_api",
      "value": 900,
      "current": 8
    },
    {
      "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
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    },
    "services": [
      {
        "name": "fkvideo-detector",
        "ip": "127.0.0.1:58970"
      },
      {
        "name": "FindFace-tarantool",
        "ip": "127.0.0.1:58978"
      },
      {
        "name": "findface-extraction-api",
        "ip": "127.0.0.1:52376"
      }
    ]
  }
}

```

10.2 Analyze Log Files

Log files provide a complete record of each FindFace Enterprise Server SDK component activity.

findface-facenapi

```

sudo tail -f /var/log/syslog | grep facenapi
Jun 28 17:20:08 ubuntu findface-facenapi[17104]: [I 170628 17:20:08 base:234] 200_
↪POST /v0/face (127.0.0.1) 1114 487 241 12

```

The findface-facenapi log contains the following time values:

- 1114 — total response time (FindFace Enterprise Server SDK components + MongoDB + Python),
- 487 — face detection time,
- 241 — findface-extraction-api response time,
- 12 — tntapi response time.

extraction-api

```

sudo tail -f /var/log/syslog | grep extraction-api

```

fkvideo_detector

```

sudo tail -f /var/log/syslog | grep fkvideo_detector

```

Tarantool

```
sudo cat /var/log/tarantool/FindFace.log
```

10.3 Migrate to Different Detector or Model

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

Sometimes you have to migrate your FindFace Enterprise Server SDK instance to another face detector or neural network model. This usually happens when you decide to update to the latest version of the product.

Tip: You can find the models summary [here](#).

If you need to re-detect faces, you must regenerate both normalized face images, thumbnails and facens. If you just want to apply a different model, it usually suffices to regenerate only facens. FindFace Enterprise Server SDK provides tools that can handle most migration use cases.

Warning: Different detectors have diverse sensitivity to certain facial features. Be aware that, after re-detecting your database, you may miss out on some previously found faces.

In this section:

- [Tools](#)
- [Requirements](#)
- [Regenerate Face Data](#)
- [Copy Facens from MongoDB to Tarantool](#)

10.3.1 Tools

To migrate your instance, you will need the following tools:

Tool	Description
<code>findface-regenerate</code>	Script that regenerates and overrides face data in MongoDB by applying different detector settings or another model to the images in the <code>Uploads</code> folder.
<code>mongo2searchapi</code>	Script that copies newly generated facens from MongoDB to Tarantool.

Both tools are automatically installed with `findface-facenapi`.

10.3.2 Requirements

The `/var/lib/ffupload/uploads/` folder (Uploads) has to be populated with at least the original images. Its content can be viewed at `http://<findface_upload_IP:3333/uploads/` in your browser.

Overall, the `findface-regenerate` tool works with the Uploads folder in the following way:

Use case	How it works
Different detector settings	The <code>findface-regenerate</code> tool runs original images through the <code>facenapi-extraction-api</code> pipeline with different detector [and model] settings, and returns regenerated normalized images, thumbnails and facens.
Different model	The <code>findface-regenerate</code> tool runs normalized face images through <code>extraction-api</code> with different model settings, and returns regenerated facens.

10.3.3 Regenerate Face Data

Important: Before conducting any operations on your MongoDB database, be sure to create its backup.

Apply `findface-regenerate` as follows:

1. Navigate into `/usr/bin/`. Display and thoroughly examine the `findface-regenerate` help message:

```
cd /usr/bin/
findface-regenerate --help
```

```
## findface-regenerate --help

Usage: /usr/bin/findface-regenerate [OPTIONS]
Options:
  --help                show this help information
/usr/lib/python3/dist-packages/facenapi/core/decoders/decoder_threaded.py options:
  --max-size            Maximum allowed photo width/height (default
                        6000)
/usr/lib/python3/dist-packages/facenapi/core/detectors/detector_dlib.py options:
  --dlib-adjust-threshold Adjust face detector threshold (default 0.0)
  --dlib-max-size        images with width or height larger than
                        dlib_max_size will be scaled down before
                        being fed into detector (default 1200)
  --dlib-normalizer      path to normalizer data (default
                        /usr/share/findface-data/normalizer.dat)
/usr/lib/python3/dist-packages/facenapi/core/detectors/detector_nnd.py options:
  --nnd-max-face-size   Maximum face size in pixels (no limit if 0)
                        (default 0)
  --nnd-min-face-size   Minimum face size in pixels (default 30.0)
  --nnd-o-net-thresh    (default 0.9)
  --nnd-p-net-thresh    (default 0.5)
  --nnd-r-net-thresh    (default 0.5)
  --nnd-scale-factor    (default 0.79)
  --nnd-workers         Number of detector workers threads. (0 - as
                        much as there are cpus) (default 0)
/usr/lib/python3/dist-packages/facenapi/core/extractors/extraction_api/__init__.py options:
  --extraction-api-connect-timeout extraction-api connect timeout (default
                        10.0)
```

(continues on next page)

(continued from previous page)

```

--extraction-api-detector      detector to use in extraction-api (legacy or
                                nnd) (default nnd)
--extraction-api-max-clients   maximum connections to extraction-api
                                (default 100)
--extraction-api-request-timeout extraction-api request timeout (default
                                10.0)
--extraction-api-url           extraction-api url (default
                                http://localhost:18666)
/usr/lib/python3/dist-packages/facenapi/core/facen_storage/searchapi.py options:
--searchapi-url               findface-searchapi url (default
                                http://localhost:18080)
/usr/lib/python3/dist-packages/facenapi/core/facen_storage/searchapi_replicated.
py options:
--searchapi-config            path to searchapi client config (default
                                /etc/findface-searchapi.json)
--searchapi-max-clients       searchapi max connections (default 100)
--searchapi-timeout           searchapi request timeout (default 10)
/usr/lib/python3/dist-packages/facenapi/core/facen_storage/tntapi.py options:
--tntapi-connect-timeout      tntapi server connect timeout (default 1.0)
--tntapi-ignore-search-errors do not break search on server error - use
                                results from 'good' servers (default False)
--tntapi-max-clients          searchapi max connections (default 100)
--tntapi-request-timeout      tntapi server request timeout (default 5.0)
--tntapi-servers-file         tntapi servers table file (default
                                /etc/tntapi.json)
/usr/lib/python3/dist-packages/facenapi/core/main_utils.py options:
--decoder                     Image decoder (threaded) (default threaded)
--detector                    Face detector (nnd,dlib) (default nnd)
--extractor                   Feature extractor (extraction-api,nnapi)
                                (default extraction-api)
--facen-storage               Feature vector storage
                                (tntapi,searchapi,searchapi_replicated)
                                (default tntapi)
--id-generator                Face id generator (tntime,mongo) (default
                                tntime)
/usr/lib/python3/dist-packages/facenapi/server/context.py options:
--fetch-proxy                 Fetch images from urls via proxy, ex:
                                http://1.2.3.4:3128
--ffupload-url                url (without path) to PUT images uploaded to
                                /face, ex: http://127.0.0.1:1234
--friend-count                (default 5)
--friend-interval              (default 604800)
--gae                         enable Gender, Age and Emotions support
                                (default False)
--max-identify-limit          Hard limit for all identification requests
                                to facen storage, 0 means unlimited (default
                                0)
--mongo-db                    mongo database name (default facenapi)
--mongo-host                  mongo database host (default localhost)
--mongo-port                  mongo database port (default 27017)
--person-identify             identify persons (default False)
--person-identify-global      identify persons across all cameras (default
                                False)
--person-identify-threshold   threshold for persons identify (default
                                0.75)
--upload-path                 path of $ffupload_url (default uploads)
/usr/lib/python3/dist-packages/facenapi/server/regenerate_facens.py options:

```

(continues on next page)

(continued from previous page)

```

--config          path to config file
--coroutines      Number of parallel coroutines (default 30)
--debug           Enable debug output (default False)
--every-other     (default 1)
--every-other-offset (default 0)
--facen-size      Facen size in number of floats. (facens of
                  this sizes are not regenerated when smart
                  regeneration is enabled) (default -1)
--max-id          Maximum id (inclusive)
--min-id          Minimum id (inclusive)
--regenerate      What to regenerate: facens, thumbs,
                  normalized (comma-separated). (default
                  facens)

```

2. To change detector settings, uncomment and edit the detector-related parameters in the `findface-facenapi` configuration file.

```

sudo vi /etc/findface-facenapi.ini

detector          = 'nnd'
...

```

3. To switch the face biometric *model*, edit the `models -> facen` parameter in the `findface-extraction-api` configuration file:

```

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

models:
  facen: apricot_320

```

4. Configure `findface-regenerate` by using command line arguments as described in the help message. For example, to switch the face detector, execute from `/usr/bin`:

```

sudo findface-regenerate --regenerate=normalized,thumbs,facens --config=/etc/
↪findface-facenapi.ini

```

To switch the model, execute:

```

sudo findface-regenerate --regenerate=facens --config=/etc/findface-facenapi.ini

```

10.3.4 Copy Facens from MongoDB to Tarantool

Apply `mongo2searchapi` as follows:

1. Create a backup for Tarantool.
2. Stop Tarantool.

```

sudo systemctl stop tarantool@FindFace*

```

3. Delete snapshot `.snap`, `xlog` `.xlog` and *fast index* `.idx` files for all `tntapi` shards.

Tip: By default, these files are stored in the following folders:

- Standalone instance:

- /opt/ntech/var/lib/tarantool/default/snapshots
- /opt/ntech/var/lib/tarantool/default/xlogs
- /opt/ntech/var/lib/tarantool/default/index

- Cluster instance:

- /opt/ntech/var/lib/tarantool/shard_N/snapshots
 - /opt/ntech/var/lib/tarantool/shard_N/xlogs
 - /opt/ntech/var/lib/tarantool/shard_N/index
-

4. If facens *differ in size* for the old and new models, update the facen size in the FindFace.start section of the Tarantool configuration file /etc/tarantool/instances.enabled/FindFace_shard_N.lua. Do so for each shard.

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

FindFace.start("127.0.0.1", 8001, {license_ntls_server="127.0.0.1:3133", facen_
↪size = 320})
```

5. Run mongo2searchapi on the findface-facenapi host:

```
sudo python3 -m facenapi.server.tools.mongo2searchapi --config=/etc/findface-
↪facenapi.ini
```

6. Start Tarantool

```
sudo systemctl start tarantool@FindFace*
```

10.4 Update to The Latest Version

In this section:

- *Update with Data Preservation*
- *Reinstall in Full*

10.4.1 Update with Data Preservation

To update FindFace Enterprise Server SDK to the latest version while maintaining face data, do the following:

1. Stop all the FindFace Enterprise Server SDK services:

```
sudo service 'findface*' stop
sudo service 'fkvideo*' stop
sudo service 'ntls' stop
sudo service 'nginx*' stop
sudo service 'tarantool*' stop
sudo service 'mongod*' stop
```

Note: Starting with version 2.6, the `nnapi` component is deprecated, and `extraction-api` is used as a default facen extractor (as well as a default face detector). If you update from a `nnapi`-based version, install `extraction-api` after this step and then remove `nnapi`. It is important that you do so, minding the sequence, otherwise you can accidentally remove some important data.

```
sudo apt-get update
sudo apt-get install findface-extraction-api
sudo apt-get remove findface-nnapi
```

2. *Prepare* the new package on each designated host.

3. Upgrade the services by executing:

```
sudo apt-get update
sudo apt-get upgrade
```

Tip: Be sure to check configuration files after this step.

4. Start the FindFace Enterprise Server SDK services.

```
sudo service 'findface*' start
sudo service 'fkvideo*' start
sudo service 'ntls' start
sudo service 'nginx*' start
sudo service 'tarantool*' start
sudo service 'mongod*' start
```

5. *Migrate* FindFace Enterprise Server SDK to a different detector or neural network model if necessary.

10.4.2 Reinstall in Full

To fully reinstall FindFace Enterprise Server SDK, do the following:

1. *Remove* the old instance with all the enrolled faces.
2. Deploy the latest version, following the standard *deployment procedure*.

10.5 Remove Instance

You can automatically remove FindFace Enterprise Server SDK, and, optionally, MongoDB and Tarantool by using the `ffserver_uninstall.sh` script. Do the following:

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

```
chmod +x ffserver_uninstall.sh
```

3. Launch the `ffserver_uninstall.sh` script.

```
sudo ./ffserver_uninstall.sh
```

4. Answer the questions provided by the script interactive wizard to choose whether to remove FindFace Enterprise Server SDK completely (along with all the enrolled faces), or while maintaining face data.

10.6 Troubleshoot Uploads

Issues with the `findface-upload` component result in unavailability of the Uploads folder content at `http://<findface_upload_IP:3333/uploads/` and in the *FindFace web interface*.

Note: The Uploads folder contains the original images which have been processed by FindFace Server, and the FindFace Server artifacts such as face thumbnails and normalized images.

In this section:

- *Uploads in FindFace Web UI*

10.6.1 Uploads in FindFace Web UI

Issue: Original images, face thumbnails, and face normalized images are not displayed in the FindFace web interface after the `findface-upload` host IP address has been changed.

Each face object in the *MongoDB* database is provided with the following links to the Uploads folder:

- Link to the relevant original image
- Links to the relevant FindFace Server artifacts: the face thumbnail and normalized image

If the `findface-upload` host IP address happens to change, the links to the Uploads folder get broken, and the original images and artifacts can no longer be displayed in the *web interface*.

To fix the problem, bulk-edit the links in the `photo`, `thumbnail` and `normalized` fields of the face objects in MongoDB as follows:

1. On the console, navigate into MongoDB and then into the `facenapi` database.

```
mongo
use facenapi
```

2. Invoke a random face object to make sure that the old IP address is still in use in its `photo`, `normalized`, and `thumbnail` fields (127.0.0.1 in the case study).

```
db.faces.findOne()

{
  "_id" : NumberLong("3871027550645276"),
  "y2" : 383,
  "x2" : 397,
  "x1" : 84,
```

(continues on next page)

(continued from previous page)

```

    "y1" : 71,
    "facen" : BinData(0, "CKftuU5t6j+...+tdKD0E1M29="),
    "gender" : "female",
    "age" : 38.75063705444336,
    "emotions" : [
        "neutral",
        "sad"
    ],
    "meta" : "",
    "photo_hash" : "6209c1a017972f8b18fada3f9e4d2768",
    "timestamp" : ISODate("2017-12-01T09:22:16.950Z"),
    "gallery" : [
        "default"
    ],
    "person_id" : 13,
    "friend" : false,
    "owner" : ObjectId("5a0e96928acdc01dab404bdd"),
    "photo" : "http://127.0.0.1:3333/uploads/5a0e96928acdc01dab404bdd/
↪20171201/3871027550645276_92fc8aa39973_photo.jpeg",
    "normalized" : "http://127.0.0.1:3333/uploads/5a0e96928acdc01dab404bdd/
↪20171201/3871027550645276_41ec18ba44cd_norm.png",
    "thumbnail" : "http://127.0.0.1:3333/uploads/5a0e96928acdc01dab404bdd/
↪20171201/3871027550645276_3bc9e34b60aa_thumb.jpeg"
}

```

3. Apply the IP address replacement script to the photo, normalized, and thumbnail fields of the face objects. In the case study, the IP address 127.0.0.1 is being replaced with 192.168.2.158.

```

db.faces.find().forEach(function(e,i) {      e.photo=e.photo.replace("//127.0.0.1",
↪ "//192.168.2.158"); e.normalized=e.normalized.replace("//127.0.0.1","//192.168.
↪ 2.158"); e.thumbnail=e.thumbnail.replace("//127.0.0.1","//192.168.2.158");
↪ db.faces.save(e); });

```

4. Invoke a random face object once more to make sure that the IP address has been successfully changed.

```

db.faces.findOne()

...
"photo" : "http://192.168.2.158:3333/uploads/5a0e96928acdc01dab404bdd/20171201/
↪ 3871027550645276_92fc8aa39973_photo.jpeg",
"normalized" : "http://192.168.2.158:3333/uploads/5a0e96928acdc01dab404bdd/
↪ 20171201/3871027550645276_41ec18ba44cd_norm.png",
"thumbnail" : "http://192.168.2.158:3333/uploads/5a0e96928acdc01dab404bdd/
↪ 20171201/3871027550645276_3bc9e34b60aa_thumb.jpeg"
...

```

10.7 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.

Warning: The automatic recovery process may result in MongoDB and Tarantool being out of sync.

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,  
}
```


11.1 Neural Network Models

Here you can see a summary for neural network models created by our Lab and used in FindFace Enterprise Server SDK:

Note: The CPU and GPU benchmark setup is the following:

- CPU: OpenBLAS 0.2.18 (single thread), Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz
 - GPU: CUDA 8.0, GeForce GTX 1080
-

Type	Name	In use	Face size	CPU		GPU		Accuracy (Black-dirty)
				FPS	Memory, MB	FPS	Memory, MB	
Face biometrics	model_36	2016	160	N/A	N/A	N/A	N/A	N/A
	model_39	2016	160	N/A	N/A	N/A	N/A	N/A
	fr_1	2016-04/05/2017	160	N/A	N/A	N/A	N/A	N/A
	en_1	2016-03/03/2017	320	N/A	N/A	N/A	N/A	N/A
	en2_face0	since 03/14/2017	320	N/A	N/A	N/A	N/A	N/A
	apricot_16	since 07/31/2017	160	5.15	336	62.27	622	54.48
	apricot_32	since 07/31/2017	320	4.87	386	59.92	616	59.32
	banana_80	since 09/15/2017	800	0.71	2407	11.96	2638	61.25
	cherry_48	since 03/30/2018	480	1.24	1880	26.45	2144	71.91
Gender recognition	fr_1_gender	since 04/05/2017	N/A	N/A	N/A	N/A	N/A	N/A
Age recognition	fr_1_age	since 04/05/2017	N/A	N/A	N/A	N/A	N/A	N/A
Emotions recognition	model_39	04/05/2017-08/11/2017	N/A	N/A	N/A	N/A	N/A	N/A
	emotion_1	since 08/11/2017	N/A	N/A	N/A	N/A	N/A	N/A

A

`add()` (*Faces method*), 129
`add()` (*Galleries method*), 132
`add()` (*Persons method*), 133
`add_to_galleries()` (*Faces method*), 130

C

Cameras (*built-in class*), 132
`count()` (*MongoStore method*), 126
`count()` (*UserStore method*), 127
Counters (*built-in class*), 134

D

`del_from_galleries()` (*Faces method*), 130

F

Faces (*built-in class*), 129
Faces.Model (*built-in class*), 129
`find()` (*MongoStore method*), 125
`find()` (*UserStore method*), 127
`find_one()` (*MongoStore method*), 125
`find_one()` (*UserStore method*), 127
`from_extraction_face()` (*Faces.Model class method*), 129

G

Galleries (*built-in class*), 132
`gen_ffupload_key()` (*ServerFaces method*), 131

I

`identify()` (*Faces method*), 130
`identify()` (*Persons method*), 133
`is_friend()` (*Persons method*), 133

M

MongoStore (*built-in class*), 125

N

`next()` (*Counters method*), 134

P

Persons (*built-in class*), 133

R

`regenerate_id()` (*Faces method*), 130
`regenerate_id()` (*ServerFaces method*), 132
`remove_one()` (*MongoStore method*), 126
`remove_one()` (*UserStore method*), 128

S

ServerFaces (*built-in class*), 131

U

`update_many()` (*UserStore method*), 128
`update_one()` (*MongoStore method*), 126
`update_one()` (*UserStore method*), 128
`upload()` (*ServerFaces method*), 132
`upload_thumbnail()` (*ServerFaces method*), 131
Users (*built-in class*), 129
UserStore (*built-in class*), 127

W

`wrap_in_model()` (*MongoStore method*), 126