FindFace Enterprise Server

Release 4.0.3

NtechLab

Jun 15, 2023

Contents:

FindFace Enterprise Server is a cutting-edge fast and accurate AI-based face recognition technology.

Features:

- Fast and robust AI-based face detection in still images and video.
- Fast and accurate AI-based face identification and verification.
- Customized face processing directives.
- AI recognition of gender, age, emotions, and other face attributes.
- AI face liveness detector.
- Extended biometric API.
- Extended API for video face detection.
- Possibility of cluster deployment. Almost infinite scalability.
- Network or on-premise licensing.
- Integration via HTTP API.

Being integrated into specific solutions and Android/iOS applications, FindFace Enterprise Server can make for accomplishing diverse goals. Some of these goals are biometric identification and access control, customer analytics, customer offer tailoring, offline retargeting, managing whitelists/blacklists, sorting and optimizing media libraries, borrower scoring, crime prevention, employee productivity control, building SafeCities, to name a few.

FindFace Enterprise Server 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, and many others.

This guide is intended for developers and system integration engineers who are going to integrate the FindFace Enterprise Server functionality into their systems.

To get a general idea of the deployment process, first, take a look at the *6 steps to face recognition*. Let's get started!

CHAPTER 1

Get Started

Follow the **6 steps** below to implement the FindFace Enterprise Server's services to your system:

1. Choose deployment architecture.
2. Prepare hardware.
3. *Install FindFace Core*. Be sure to test the system *work*.
4. *Configure video face detection*. Specify *directives* for face processing.
5. Consider *using advanced features*.
6. *Finalize the system with coding*.

CHAPTER 2

Architecture

Be sure to take a minute to learn the FindFace Enterprise Server architecture. This knowledge is essential for the FindFace Enterprise Server deployment, integration, maintenance, and troubleshooting.

In this chapter:

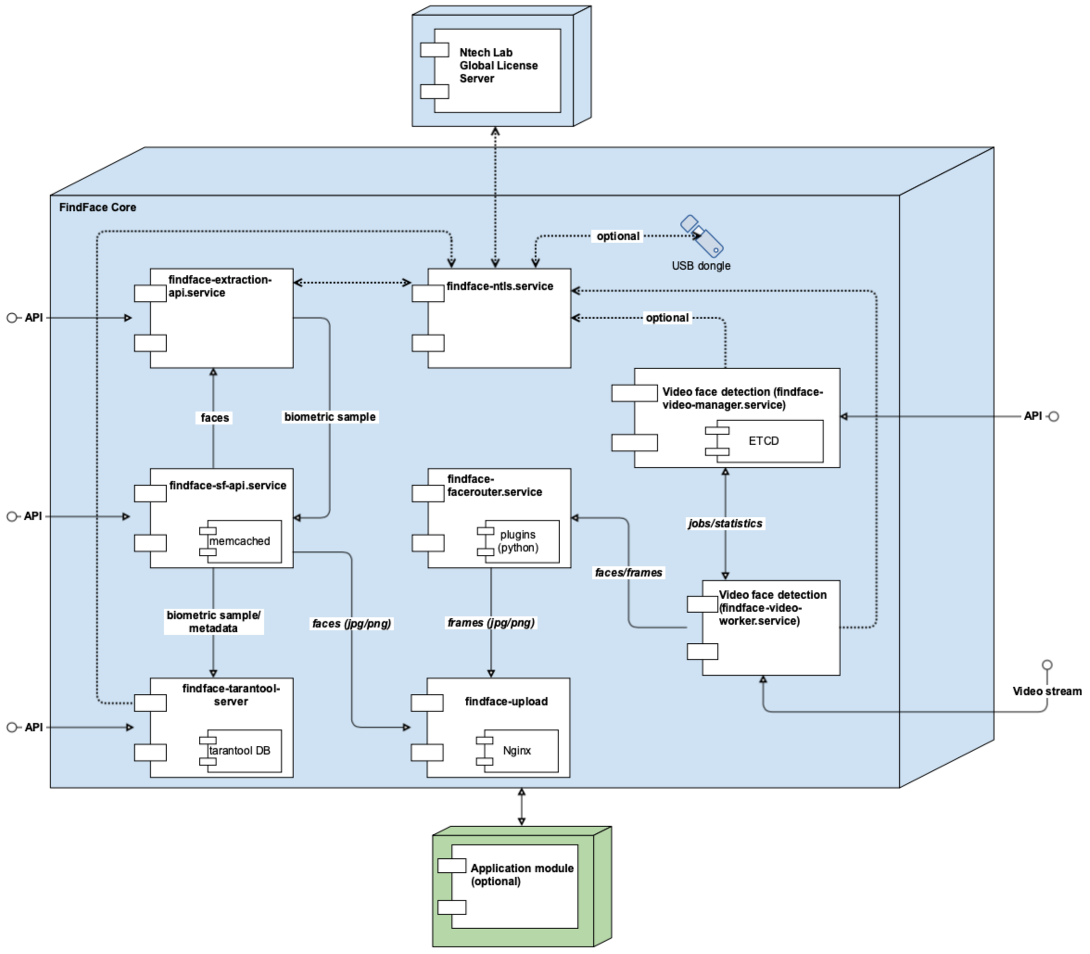
2.1 Architectural Elements

FindFace Enterprise Server consists of the following architectural elements:

- FindFace core,
- (optional) application modules.

Note: Application modules are not available in the basic configuration. To learn more about building a turnkey application with the help of our team, contact our experts by info@ntechlab.com.

The FindFace core includes the following components:


Ntech Lab Global License Server
FindFace Core
optional
USB dongle
findface-extraction-api.service
findface-ntls.service
optional
API
faces
biometric sample
Video face detection (findface-video-manager.service)
ETCD
API
findface-sf-api.service
memcached
findface-facerouter.service
plugins (python)
jobs/statistics
API
faces/frames
biometric sample/ metadata
faces (jpg/png)
frames (jpg/png)
Video face detection (findface-video-worker.service)
Video stream
findface-tarantool-server
tarantool DB
findface-upload
Nginx
API
Application module (optional)

Component	Description	Vendor
findface-extraction-api	Service that uses neural networks to detect a face in an image and extract a face biometric sample (feature vector), gender, age, emotions, and other face attributes. CPU- or GPU-acceleration.	NtechLab own deployment
findface-sf-api	Service that implements HTTP API for face detection and face recognition.	
findface-tarantool-server	Service that provides interaction between the `findface-sf-api` service and the biometric database (the Tarantool-powered database that stores face biometric samples).	
findface-upload	NGINX-based web server used as a storage for original images, thumbnails, and normalized face images.	
findface-facerouter	Service used to define processing directives for detected faces.	
findface-video-manager	Part of the video face detection module. Service that is used for managing the video face detection functionality, configuring the video face detector settings, and specifying the list of to-be-processed video streams.	
findface-video-worker	Part of the video face detection module. Service that recognizes a face in the video and posts its normalized image, full frame, and such metadata as the camera ID and detection time to the `findface-facerouter` service for further processing according to given directives. CPU- or GPU-acceleration.	
findface-ntls	License server which interfaces with the NtechLab Global License Server or a USB dongle to verify the *license* of your FindFace Enterprise Server instance.	
Tarantool	Third-party software which implements the biometric database that stores extracted biometric samples (feature vectors).	Tarantool
etcd	Third-party software that implements a distributed key-value store for `findface-video-manager`. Used as a coordination service in the distributed system, providing the video face detector with fault tolerance.	etcd
NGINX	Third-party software which implements the `findface-upload` component.	nginx
memcached	Third-party software which implements a distributed memory caching system. Used by `findface-extraction-api` as temporary storage for extracted face biometric samples before they are written to the biometric database powered by Tarantool.	memcached

See also:

Components in Depth

2.2 Single- and Multi-Host Deployment

Depending on your system characteristics and performance requirements, you can opt to install FindFace Enterprise Server standalone or in a cluster environment:

Deployment	Recommendation
Standalone	You can deploy FindFace Enterprise Server and neural network models on a single host (standalone) if the number of faces in the database does not exceed 1,000,000 (recommended limit). This variant makes it easier to start deployment and cater to the basic requirements of your system.
Cluster	If the number of faces in the database does exceed 1,000,000, we recommend you to deploy FindFace Enterprise Server in a cluster environment. In this case, FindFace Enterprise Server components will be distributed across several hosts. This is a medium and large deployment that can be scaled almost infinitely. It will also suit professional high load projects with severe requirements for performance.

If you opt for the cluster deployment, we offer you one of the following deployment schemes:

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

2.3 CPU- and GPU-acceleration

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

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

Important: Refer to Distribution requirements when choosing hardware configuration.

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

CHAPTER 3

System Requirements

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

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

In this chapter:

3.1 Basic Configuration

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

Note: When designing your system architecture in a cluster environment, bear in mind that the number of `findface-tarantool-server` shards on a single host must not exceed the number of CPU memory channels, multiplied by 2.

	Minimum	Recommended
CPU	Intel Core i5 CPU with 4 physical cores 2.8 GHz	Intel Xeon E5v3 with 6 physical cores, or higher or similar CPU
	The own needs of FindFace Enterprise Server require 2 cores HT > 2.5 GHz. The characteristics also depend on the number of cameras in use. A single camera 720p@25FPS requires 2 cores >2.5 GHz. AVX support	
GPU (optional)	Nvidia Geforce® GTX 980 4GB	Nvidia Geforce® GTX 1080+ with 8+Gb RAM
	Supported series: GeForce (Maxwell, Pascal, Turing, and above), Tesla (Maxwell, Pascal, Volta v100, Turing, and above)	
RAM	10 Gb	16+ Gb
	The own needs of FindFace Enterprise Server require 8 Gb. The RAM consumption also depends on the number of cameras in use. A single camera 720p@25FPS requires 2 GB RAM	
HDD	16 Gb	16+ Gb
	The own needs of the operating system and FindFace Enterprise Server require 15 GB.	
Operating system	Ubuntu 16.04 x64 only	

CHAPTER 4

Licensing Principles

FindFace Enterprise Server is licensed by the following criteria:

1. The number of biometric samples extracted from faces detected in static images and videos.
2. The number of cameras in use.
3. The number of `findface-extraction-api` model instances in use.
4. Face features recognition: gender/age/emotions/glasses/beard.
5. Face liveness detection.
6. Fast index.

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

- Online licensing requires a stable internet connection. Upon being disconnected from the internet, the system will continue working off-grid for about 1 hour.
- On-premise (offline) licensing requires a USB port on the physical server with the `findface-ntls` component, that will be used to plug in a provided USB dongle.

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

See also:

Retrieve Licensing Information

CHAPTER 5

Deploy FindFace Enterprise Server

For your convenience, we offer you several installation options:

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

After the installation, *test* your system work and configure a *fast index* search.

5.1 Install from Console Installer

To install FindFace Enterprise Server, use a developer-friendly console installer.

Tip: Be sure to consult the system requirements prior.

Do the following:

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

4. Execute the `.run` file.

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

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

   1. Product to install: `FindFace Server`.
   2. Installation type:

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

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

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

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

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

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

Should you choose the standalone installation, the FindFace Enterprise Server components will be automatically installed, configured, and/or started in the following configuration:

Service	Configuration
etcd	Installed and started.
memcached	Installed and started.
nginx	Installed and started.
findface-ntls	Installed and started.
findface-tarantool-server	Installed and started. The number of instances (shards) is calculated using the formula: `N = max(min(mem_mb // 2000, cpu_cores), 1)`, i.e. it is equal to the RAM size in MB divided by 2000, or the number of CPU physical cores (but at least 1 shard).
findface-extraction-api	Installed and started.
findface-sf-api	Installed and started.
findface-facerouter	Installed and started.
findface-upload	Installed.
findface-video-manager	Installed and started.
findface-video-worker-*	Installed and started.
findface-data-*	Neural network models for face and face features recognition (gender, age, emotions, glasses, beard). Installed.
findface-gpudetector-data/	NTechLab gpu detector data. Installed.
jq	Installed. Used to pretty-print API responses from FindFace Enterprise Server.

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

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

```
#############################################################################
#                       Installation is complete                            #
#############################################################################
- upload your license to http://127.0.0.1:3185/
- FindFace SF-API address: http://172.20.77.78:18411/
- FindFace VideoManager address: http://172.20.77.78:18411/
```

5. Upload the FindFace Enterprise Server license file via the `findface-ntls` web interface `http://<ntls_host_IP_address>:3185`.

   Note: The IP address is shown in the links to the 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.

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

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

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

5.2 Install Step-by-Step

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

In this section:

5.2.1 Install APT Repository

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

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

4. Execute the `.run` file.

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

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

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

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

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

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

5.2.2 Prerequisites

FindFace Enterprise Server requires such third-party software as etcd and memcached. Do the following:

1. Install the prerequisite packages as such:

```
sudo apt update
sudo apt install -y etcd memcached
```

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

```
sudo vi /etc/memcached.conf

-m 512
-I 16m
```

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

```
sudo systemctl enable etcd.service memcached.service
sudo systemctl start etcd.service memcached.service
```

5.2.3 Provide Licensing

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

The FindFace Enterprise Server licensing is provided as follows:

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

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

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

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

2. Upload the license file via the `findface-ntls` web interface in one of the following ways:
   - Navigate to the `findface-ntls` web interface `http://<NTLS_IP_address>:3185/#/`. Upload the license file.

     Tip: Later on, use the `findface-ntls` web interface to consult your license information, and upgrade or extend your license.

   - Directly put the license file into the license folder (by default, `/ntech/license`, can be changed in the `/etc/findface-ntls.cfg` configuration file).
3. For the on-premise licensing, insert the USB dongle into a USB port.
4. If the licensable components are installed on remote hosts, specify the IP address of the `findface-ntls` host in their configuration files. See *findface-extraction-api*, *findface-tarantool-server*, *Video face detection: findface-video-manager and findface-video-worker* for details.

See also:

Troubleshoot Licensing and findface-ntls

5.2.4 Deploy `findface-extraction-api`

To deploy the `findface-extraction-api` component, do the following:

Important: This component requires the installation of *neural network models*.

1. Install `findface-extraction-api` as such:

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

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

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

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

3. Specify the IP address of the `findface-ntls` host if `findface-ntls` is installed on a remote host. See licensing.

```
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 instead 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 `instances` parameter indicates how many `findface-extraction-api` instances are used. Specify the number of instances from your license. The default value (0) means that this number is equal to the number of CPU cores.

   Note: This parameter severely affects RAM consumption.

```
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
```

8. Enable recognition models for face features such as gender, age, emotions, glasses3, and/or beard, subject to your needs. Be sure to choose the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models.

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

The following models are available:

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

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

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

9. Enable the `findface-extraction-api` service autostart and launch the service.

```
sudo systemctl enable findface-extraction-api.service && sudo systemctl start␣
↪findface-extraction-api.service
```

5.2.5 Deploy `findface-tarantool-server`

The `findface-tarantool-server` component connects the Tarantool database and the `findface-sf-api` component, transferring search results from the database to `findface-sf-api` for further processing. To increase search speed, multiple `findface-tarantool-server` shards can be created on each Tarantool host. Their running concurrently leads to a remarkable increase in performance. Each shard can handle up to approximately 10,000,000 faces. In the case of the standalone deployment, you need only one shard (already created by default). In a cluster environment, the number of shards has to be calculated depending on your hardware configuration and database size (see details below).

To deploy the `findface-tarantool-server` component, do the following:

1. Install `findface-tarantool-server`

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

2. Disable autostart and stop the Tarantool exemplary service.

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

3. Open the configuration file:

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

4. Edit the maximum memory usage. The memory usage must be set in bytes, depending on the number of faces the shard handles, at the rate roughly 1280 byte per face. For example, the value `1.2*1024*1024*1024` corresponds to 1,000,000 faces:

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

5. Specify the IP address of the `findface-ntls` host if `findface-ntls` is installed on a remote host:

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

6. By default, you can access Tarantool only from the localhost (`127.0.0.1`). If you plan to be accessing Tarantool from a certain remote host, either specify this remote host IP address in the `FindFace.start` section, or change `127.0.0.1` to `0.0.0.0` in the same section to allow access to Tarantool from any IP address.

Tip: To allow access only from a certain remote host (`192.168.113.10` in the example), configure as follows:

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

To allow access from any IP address, apply `0.0.0.0` instead:

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

7. In the `meta_scheme` parameter, create a database structure to store the face recognition results. The structure is created as a set of fields. Describe each field with the following parameters:

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

```
box.cfg{
    listen = '127.0.0.1:33001',

    vinyl_dir = '/opt/ntech/var/lib/tarantool/name',
    work_dir = '/opt/ntech/var/lib/tarantool/name',
    memtx_dir = '/opt/ntech/var/lib/tarantool/name/snapshots',
    wal_dir = '/opt/ntech/var/lib/tarantool/name/xlogs',

    memtx_memory = 16 * 1024 * 1024 * 1024,

    checkpoint_interval = 3600*4,
    checkpoint_count = 3,

    -- force_recovery = true,
}

pcall(function() box.schema.user.grant('guest', 'execute,read,write', 'universe')␣
↪end)

FindFace = require("FindFace")
FindFace.start(
    "0.0.0.0",
    8001,
    {
        license_ntls_server="127.0.0.1:3133",
        facen_size=480,
        meta_scheme  = {

            {
                id = 1,
                name = 'm:timestamp',
                field_type = 'unsigned',
                default = 0
            },

            {
```

(continues on next page)

(continued from previous page)

```
                    id = 2,
                    name = 'feat',
                    field_type = 'string',
                    default = ""
                },

                {
                    id = 3,
                    name = 'normalized_id',
                    field_type = 'string',
                    default = ""
                },

                {
                    id = 4,
                    name = 'm:camera',
                    field_type = 'string',
                    default = ""
                },

                {
                    id = 5,
                    name = 'm:photo',
                    field_type = 'string',
                    default = ""
                },

                {
                    id = 6,
                    name = 'm:thumbnail',
                    field_type = 'string',
                    default = ""
                },

                {
                    id = 7,
                    name = 'm:score',
                    field_type = 'unsigned',
                    default = "10000000000000000000"
                },

                {
                    id = 8,
                    name = 'm:bbox',
                    field_type = 'string',
                    default = ""
                },

                {
                    id = 9,
                    name = 'm:labels',
                    field_type = 'set[string]',
                    default = {}
                },

                {
                    id = 10,
```

(continues on next page)

(continued from previous page)

```
                        name = 'm:is_friend',
                        field_type = 'unsigned',
                        default = 0
                  },
                  }
      }
)
```

8. (Optional) If there are more than 10,000,000 faces or so on a single shard, the search may take too long. In the case of a large installation, it is advised to create additional `findface-tarantool-server` shards, observing the following rules:

   - One shard can successfully handle up to approximately 10,000,000 faces.
   - The number of shards on a single host must not exceed the number of CPU memory channels, multiplied by 2. Bear it in mind, when designing your system architecture in a cluster environment.

   To create multiple shards, multiply the configuration file for the default shard `/etc/tarantool/instances.enabled/FindFace.lua`. Then override the default shard IP address and port with new values. To do so, write a bash script (e.g., `shard.sh`) that will automatically create configuration files for all shards on a particular host. The script below can be used as an excellent starting point for your own code. The exemplary script creates 4 shards listening to the ports: `findface-tarantool-server` `33001..33004` and HTTP `8001..8004`.

```
#!/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 inited"
done;
```

Tip: Download the `exemplary script`.

Run the script from the home directory.

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

Check the configuration files created.

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

(continues on next page)

(continued from previous page)

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

9. Enable the `findface-tarantool-server` shard autostart and launch the shard.

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

In the case of multiple shards, you can do so by analogy with the following example (launching 4 shards):

```
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;
```

5.2.6 Deploy `findface-upload`

To store all original images ever sent to the system for processing, as well as such FindFace core artifacts as face thumbnails and normalized images, you will need the `findface-upload` service.

Tip: Skip the `findface-upload` deployment if you do not want to store these data on the FindFace Enterprise Server host. In this case, the system will be saving only face features vectors (biometric samples) in the Tarantool-powered biometric database.

Install `findface-upload` as such:

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

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

5.2.7 Deploy `findface-sf-api`

To deploy the `findface-sf-api` component, do the following:

1. Install `findface-sf-api` as such:

```
sudo apt update
sudo apt install -y findface-sf-api
```

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

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

3. If FindFace Enterprise Server is being deployed in a cluster environment, specify the IP addresses and ports of the `findface-extraction-api` host (the `extraction-api` parameter), the `findface-tarantool-server` shards (`storage-api`, in the format: `http://IP_address:port/v2/`), and the `findface-upload` host (`upload_url`).

```
extraction-api:
  extraction-api: http://10.220.85.120:18666
storage-api:
 shards:
```

(continues on next page)

(continued from previous page)

```
 - master: http://10.200.85.115:8003/v2/
 - master: http://10.200.85.120:8004/v2/
 - master: http://10.200.85.120:8005/v2/
 - master: http://10.200.85.120:8006/v2/
   slave: ''
upload_url: http://127.0.0.1:3333/uploads/
```

4. Enable the `findface-sf-api` service autostart and launch the service.

```
sudo systemctl enable findface-sf-api.service && sudo systemctl start findface-sf-
↪api.service
```

5.2.8 Deploy `findface-facerouter`

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

1. Install `findface-facerouter` as such:

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

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

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

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

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

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

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

5.2.9 Deploy Video Face Detection

Video face detection is provided by the `findface-video-manager` and `findface-video-worker` components.

To deploy the `findface-video-manager` component, do the following:

1. Install `findface-video-manager`:

```
sudo apt install -y findface-video-manager
```

2. Open the `/etc/findface-video-manager.conf` configuration file.

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

3. In the `router_url` parameter, specify the IP address and port of the `findface-facerouter` component which will receive detected faces from `findface-video-worker`.

```
router_url: http://127.0.0.1:18820/v0/frame
```

4. In the `ntls -> url` parameter, specify the IP address of the `findface-ntls` host if `findface-ntls` is installed on a remote host.

```
ntls:
    url: http://127.0.0.1:3185/
```

5. If necessary, configure the video processing settings which apply to all video streams in the system.

   Tip: You can skip this step: when creating a job for `findface-video-manager`, you will be able to individually configure processing settings for each video stream (see *Video Face Detection API*).

6. Enable the `findface-video-manager` service autostart and launch the service.

```
sudo systemctl enable findface-video-manager.service && sudo systemctl start␣
↪findface-video-manager.service
```

To deploy the `findface-video-worker` component, do the following:

1. Install `findface-video-worker`:

```
sudo apt update
sudo apt install -y findface-video-worker-cpu
```

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

2. Open the `/etc/findface-video-worker-cpu.ini` (`/etc/findface-video-worker-gpu.ini`) configuration file.

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

3. In the `ntls-addr` parameter, specify the IP address of the `findface-ntls` host if `findface-ntls` is installed on a remote host.

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

4. In the `mgr-static` parameter, specify the IP address of the `findface-video-manager` host that will be providing `findface-video-worker` with settings and the list of to-be-processed video streams.

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

5. In the `capacity` parameter, specify the maximum number of video streams that `findface-video-worker` is allowed to process.

```
capacity=10
```

6. Enable the `findface-video-worker` autostart and launch the service.

```
sudo systemctl enable findface-video-worker-cpu.service && sudo systemctl start␣
↪findface-video-worker-cpu.service
```

5.3 Test Requests

Before you can proceed with development to implement the face recognition services to your system, 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**.

In this section:

5.3.1 How to Pretty-Print Responses

Use **jq** to parse JSON data in responses. The **jq** tool is automatically installed from the console installer.

Tip: If it is not so, install **jq** as such:

```
sudo apt install jq
```

Note: Since **jq** approximates integers larger than 2^53 (e.g., for `"id":1210786732394996822 8`, the output is `"id": 1210786732394996700 0`, etc.), you may want to use **json_pp** instead.

5.3.2 Create Gallery

The following request creates a new gallery `galleryname`. Relevant HTTP API method: `/galleries/<gallery> POST`.

Request

```
curl -s -X POST http://localhost:18411/v2/galleries/galleryname | jq
```

Response

```
{}
```

5.3.3 List Galleries

The following request returns the names of existing galleries (`galleryname`). Relevant HTTP API method: `/galleries GET`.

Request

```
curl -s http://localhost:18411/v2/galleries | jq
```

Response

```
{
  "galleries": [
    {
      "name": "galleryname",
      "faces": 0
    }
  ]
}
```

5.3.4 Detect Face in Image

The 1st request detects a face in a sample Internet image and returns coordinates of the rectangle around the face (a.k.a. bbox) and the face orientation. Relevant HTTP API method: `/detect POST`.

Request #1

```
curl -s -H 'Content-Type: text/x-url' -d https://static.findface.pro/sample.jpg -X␣
↪POST http://localhost:18411/v2/detect | jq
```

Response

```
{
  "faces": [
    {
      "bbox": {
        "left": 595,
        "top": 127,
        "right": 812,
        "bottom": 344
      },
      "features": {
        "score": 0.9999999
```

(continues on next page)

(continued from previous page)

```
      }
    }
  ],
  "orientation": 1
}
```

If `facen=on`, the detection result is saved in `memcached`. In the 2nd request, the image is the same but this time `facen=on`, along with enabled gender, age and emotions recognition.

Tip: To retrieve the detection result from `memcached`, use the `/detect GET` method.

Request #2

```
curl -s -H 'Content-Type: text/x-url' -d https://static.findface.pro/sample.jpg -X␣
↪POST 'http://localhost:18411/v2/detect?facen=on&gender=on&age=on&emotions=on' | jq
```

Response

```
{
  "faces": [
    {
      "id": "bhse5elubdg0ajgm2nkg",
      "bbox": {
        "left": 595,
        "top": 127,
        "right": 812,
        "bottom": 344
      },
       "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
```

(continues on next page)

(continued from previous page)

```
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      }
    }
  ],
  "orientation": 1
}
```

The 3rd request detects a face in another image and is used merely for the purpose of database population. The detection result is saved in `memcached` (`facen=on`).

Request #3

```
curl -s -H 'Content-Type: text/x-url' -d https://static.findface.pro/sample2.jpg -X␣
↪POST 'http://localhost:18411/v2/detect?facen=on
```

```
{
  "faces": [
    {
      "id": "bhse45dubdg0ajgm2nk0",
      "bbox": {
        "left": 515,
        "top": 121,
        "right": 821,
        "bottom": 427
      },
      "features": {
        "score": 0.9999982
      }
    }
  ],
  "orientation": 1
}
```

5.3.5 Retrieve Detection Result from `memcached`

The following request retrieves the detection result from `memcached` by id. Related HTTP API method: `/detect GET`.

Note: `bhse5elubdg0ajgm2nkg` is the id of the detection results in memcached. This id is provided only for reference. To create valid requests out of the example, replace the identifier with those actually received in the previous responses.

Important: Before you proceed, open the `findface-sf-api` configuration file and make sure that the `allow-return-facen` parameter is `on`.

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

allow-return-facen: on
```

Request #1

```
curl -s 'http://localhost:18411/v2/detect/bhse5elubdg0ajgm2nkg'
```

Response

```
{
  "id": "bhse5elubdg0ajgm2nkg",
  "bbox": {
    "left": 595,
    "top": 127,
    "right": 812,
    "bottom": 344
  },
  "features": {
    "score": 0.9999999
  }
}
```

To retrieve a face feature vector (facen) in a detection result, open the `/etc/findface-sf-api.ini` configuration file and set `allow-return-facen: true`. Restart `findface-sf-api` and append the `return_facen=on` query string parameter to the previous command:

Request #2

```
curl -s 'http://localhost:18411/v2/detect/bhse5elubdg0ajgm2nkg?return_facen=on' | jq
```

Response

```
{
  "id": "bhse5elubdg0ajgm2nkg",
  "bbox": {
    "left": 595,
    "top": 127,
    "right": 812,
    "bottom": 344
  },
  "features": {
    "score": 0.9999999
```

(continues on next page)

(continued from previous page)

```
  },
  "facen": "1ji...Vr3TEQg8"
}
```

5.3.6 Add Face from `memcached` to Gallery

The following requests not only retrieve the detection results from `memcached` by their id's but also add the results to the gallery `galleryname` under different custom ids (to be specified in a request). Relevant HTTP API method: `/v2/galleries/<gal>/faces/<id>`.

Request #1

```
curl -s -X POST -H 'Content-Type: application/json' --data '{"from":
↪"detection:bd2blott8f63g8nbhi50"}' http://localhost:18411/v2/galleries/galleryname/
↪faces/1 | jq
```

Response

```
{
  "id": {
    "gallery": "galleryname",
    "face": 1
  },
  "features": {
    "gender": {
      "gender": "FEMALE",
      "score": -2.6415923
    },
    "age": 26.04833,
    "score": 0.9999999,
    "emotions": [
      {
        "emotion": "neutral",
        "score": 0.99958
      },
      {
        "emotion": "sad",
        "score": 0.0004020398
      },
      {
        "emotion": "happy",
        "score": 8.603504e-06
      },
      {
        "emotion": "surprise",
        "score": 8.076798e-06
      },
      {
        "emotion": "disgust",
        "score": 6.653509e-07
      },
```

(continues on next page)

(continued from previous page)

```
      {
        "emotion": "angry",
        "score": 6.14346e-07
      },
      {
        "emotion": "fear",
        "score": 7.33713e-10
      }
    ]
  },
  "meta": {},
  "normalized_id": "3_bd323i5t8f66ph0eafq0.png"
}
```

Request #2

```
curl -s -X POST -H 'Content-Type: application/json' --data '{"from":"detection:␣
↪bd44p6dt8f66ph0eahkg "}' http://localhost:18411/v2/galleries/galleryname/faces/2 |␣
↪jq
```

5.3.7 List Gallery Faces

The following request returns the list of faces in the gallery `galleryname`. Relevant HTTP API method: `/galleries/<gallery>/faces` with the active `limit=` filter (maximum number of returned faces).

Request

```
curl -s 'http://localhost:18411/v2/galleries/galleryname/faces?limit=2' | jq
```

```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 1
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
```

(continues on next page)

(continued from previous page)

```
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "1_bd321tlt8f66ph0eaflg.png"
    },
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
```

(continues on next page)

(continued from previous page)

```
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
    }
  ],
  "next_page": "3"
}
```

5.3.8 Search Face in Gallery

The following request searches the gallery `galleryname` for faces similar to a detected face (detection result stored in `memcached`) with threshold similarity equal to 0.5. Relevant HTTP API request: `/galleries/<gallery>/faces` with enabled `detection:id` and `similarity` filters.

Request

```
curl -s 'http://localhost:18411/v2/galleries/galleryname/faces?
↪detection:bd3hv4tt8f66ph0eag1g=0.5&limit=1' | jq
```

Response

```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
```

(continues on next page)

(continued from previous page)

```
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png",
      "confidence": 0.9999
    }
  ],
  "next_page": "There are more than 1 results, but pagination is not supported when␣
↪filtering by FaceN"
}
```

The following request searches the gallery `galleryname` for faces similar to a given face in the same gallery with threshold similarity equal to 0.5. Relevant HTTP API request: `/galleries/<gallery>/faces` with enabled `face:<gallery>/<db_id>` and `similarity` filters.

```
curl -s 'http://localhost:18411/v2/galleries/galleryname/faces?face:galleryname/1=0.1&
↪limit=1' | jq
```

```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": null,
      "meta": {},
      "confidence": 0.999
    }
  ],
  "next_page": "There are more than 1 results, but pagination is not supported when␣
↪filtering by FaceN"
```

(continues on next page)

(continued from previous page)

```
}
```

5.3.9 Compare Faces

The following requests compare a pair of faces and return a probability of their belonging to the same person. Relevant HTTP API method: `/verify POST`.

The first request compares 2 results of the `/detect POST` method, stored in `memcached`.

Request #1

```
curl -s 'http://localhost:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=detection:bd3hv8dt8f66ph0eag2g' | jq
```

Response

```
{
  "confidence": 0.92764723
}
```

The 2nd request compares a result of the `/detect POST` method and a face in a gallery.

Request

```
curl -s 'http://localhost:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=face:galleryname/2' | jq
```

Response

```
{
  "confidence": 0.999996
}
```

5.4 Fast Index

To speed up the search, create a fast index for each gallery, using the `findface-tarantool-build-index` utility delivered with the console installer. The utility is independent of the `findface-tarantool-server` component and can be installed either on the localhost or on a remote host with access to Tarantool.

To create a fast index, do the following:

1. If you have installed the FindFace core step-by-step, install the `findface-tarantool-build-index` utility.

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

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

   Important: The gallery must not be empty. See *Direct API Requests to Tarantool* to fill the gallery with faces.

   Note: You have to repeat the fast index creation on each `findface-tarantool-server` 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-tarantool-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 --output /opt/ntech/var/lib/
↪tarantool/default/index/testgal.idx --facen_size 320
Config values:
.input = /tmp/preindex.bin
.output = /opt/ntech/var/lib/tarantool/default/index/testgal.idx
.facen_size = 320
.param_m = 12
.param_ef = 500
.limit = 4294967295

Building index: [XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX] 100% ; 3 / 3
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, the index will also be uploaded.

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

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

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

Tip: Be sure to consult the system requirements prior.

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

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

4. Execute the `.run` file.

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

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

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

After that, the installation process will automatically begin.

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

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

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

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

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

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

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

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

5.6 Neural Network Models Installation

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

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

1. Execute the prepared `<findface-security-and-server-xxx>.run` file.

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

2. Select the installation type: `Fully customized installation`.
3. Select a component to install: `findface-data`. To do so, first, deselect all the listed components by entering `-*` in the command line, then select the required component by entering its sequence number (keyword): `1`. Enter `done` to save your selection and proceed to another step.
4. In the same manner, select models to install. After that, the installation process will automatically begin.

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

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

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

5.7 Guide to Typical Cluster Installation

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

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

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

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

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

- The necessity to distribute the biometric sample extraction high load.
- A large number of faces to search through that requires the implementation of a distributed face database.

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

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

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

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

5.7.1 Deploy Principal Server

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

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

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

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

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

```
#############################################################################
#                       Installation is complete                            #
#############################################################################
- upload your license to http://127.0.0.1:3185/
- FindFace SF-API address: http://172.20.77.78:18411/
- FindFace VideoManager address: http://172.20.77.78:18411/
```

2. Upload the FindFace Enterprise Server license file via the `findface-ntls` web interface `http://<ntls_host_IP_address>:3185`.

Note: The IP address is shown in the links to the 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.

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

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

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

5.7.2 Deploy Video Processing Servers

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

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

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

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

Note: After the installation, specify the `findface-ntls` and/or `findface-video-manager` IP addresses in the `findface-video-worker` configuration file.

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

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

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

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

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

5.7.3 Deploy Biometric Servers

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

- Product to install: `FindFace Server`.
- Installation type: `Fully customized installation`.
- FindFace Enterprise Server components to install: `findface-extraction-api` and `findface-data`. To make a selection, first, deselect all the listed components by entering `-*` in the command line. Select `findface-extraction-api` and `findface-data` by entering their sequence number (keyword): `1 7`. Enter `done` to save your selection and proceed to another step.
- Type of `findface-extraction-api` acceleration: CPU or GPU.
- Modification of the `findface-extraction-api` configuration file: specify the IP address of the `findface-ntls` server.
- Neural network models to install: CPU or GPU model for face biometrics (mandatory), and (optional) CPU/GPU models for gender, age, emotions, glasses and/or beard recognition. To make a selection, first, deselect all the listed models by entering `-*` in the command line. Select required models by entering their sequence number (keyword), for example, `8 2` to select the GPU-models for biometric sample extraction and age recognition. Enter `done` to save your selection and proceed to another step. Be sure to choose the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models. See *Face Features Recognition* for details.

  The following models are available:

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

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

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

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

5.7.4 Distribute Load across Biometric Servers

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

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

To set up load balancing, do the following:

1. Designate the FindFace Enterprise Server principal server (recommended) or any other server with NGINX as a gateway to all the biometric servers.

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

   Tip: You can install NGINX as such:

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

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

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

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

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

4. Enable the load balancer in NGINX.

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

5. Restart nginx.

```
sudo service nginx restart
```

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

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

listen: 172.168.1.10:18666
```

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

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

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

5.7.5 Distribute Database

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

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

- Product to install: `FindFace Server`.
- Installation type: `Fully customized installation`.
- FindFace Enterprise Server components to install: `findface-tarantool-server`. To make a selection, first, deselect all the listed components by entering `-*` in the command line. Select `findface-tarantool-server` by entering its sequence number (keyword): `13`. Enter `done` to save your selection and proceed to another step.

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

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

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

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

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

You will get the following result:

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

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

5.7.6 Configure Network

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

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

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

   Specify the following parameters:

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

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

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

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

2. Open the findface-facerouter configuration file. Specify the IP address of the findface-sf-api host.

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

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

3. Open the findface-video-manager configuration file. In the router_url parameter, specify the IP address and port of the findface-facerouter host to receive detected faces from findface-video-worker.

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

...
router_url: http://127.0.0.1:18820/v0/frame
```

The FindFace Enterprise Server components interaction is now set up.

5.8 Fully Customized Installation

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

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

- Product to install: `FindFace Server`.
- Installation type: `Fully customized installation`.
- FindFace Enterprise Server components to install: whenever you have to make a selection, first, deselect all the listed components by entering `-*` in the command line. Select required components by entering their sequence number (keyword), for example: `1 7` (`findface-data`, `findface-extraction-api`), `13` (`findface-tarantool-server`), or `9` (findface-upload). Enter `done` to save your selection and proceed to another step.
- Related questions such as about the acceleration type: CPU or GPU.

CHAPTER 6

Biometric API

6.1 How to Use Biometric API

In this section:

6.1.1 Endpoint

Biometric API requests are to be sent to `http://<findface-sf-api IP address>:18411/`. API requests are executed by the `findface-sf-api` component.

6.1.2 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 is to be specified in the request path (for example, v2 in `/v2/detect/`).

The most recent version is v2.

Tip: When starting a new project, always use the latest stable API version.

6.1.3 Face as API Object

Biometric API operates with a `face` object which represents a human face.

Note: There can be several faces in a photo and thus several `face` objects associated with it.

Note: Different images of the same person are considered to be different `face` objects.

Each `face` object has the following attributes:

- `"id" (uint64)`: (only if the face has been added to the biometric database) face identifier (uint64) to be specified by a user in an API request when adding a face from `memcached` to the database. The identifier is passed as <id> in the `/galleries/<gallery>/faces/<id> POST` method.
- `"facen" (bytes)`: the face feature vector.
- `"meta" (string)`: set of metadata strings that you can use to store any information associated with the face, for example, the name of a person, the camera id, the detection date and time, etc.
- `"features" (dictionary)`: a dictionary {key (string):value (any datatype)}. Used to store face biometric parameters such as gender, age, emotions.

6.1.4 Parameters Format

There are two ways to pass a photo image to the system:

- as a publicly accessible URL,
- as a file.

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

- image/jpeg, image/png, image/webp, image/bmp: to pass a photo image as a file,
- text/x-url: to pass a photo image as an URL,
- query string: parameters appended to a URI request.

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

6.1.5 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://<findface-sf-api_ip>:18411/v2/docs/v2/overview.html` for the API version /v2.

6.1.6 Limits

FindFace Enterprise Server imposes the following limits.

Limit	Value
Image formats	JPG, PNG, WEBP, BMP
Maximum photo file size	To be configured via the `findface-sf-api` configuration file.
Minimal size of a face	50x50 pixels
Maximum number of detected faces per photo	Unlimited

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

6.1.7 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 `desc`.

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

Common Error Codes

Error code	Description	HTTP code
`UNKNOWN_ERROR`	Error with unknown origin.	500
`BAD_PARAM`	The request can be read, however, some method parameters are invalid. This response type contains additional attributes `param` and‘‘value‘‘ to indicate which parameters are invalid.	400
`CONFLICT`	Conflict.	409
`EXTRACTION_ERROR`	Error upon a face feature vector extraction.	503
`LICENSE_ERROR`	The system configuration does not match license.	503
`MALFORMED_REQUEST`	The request is malformed and cannot be read.	400
`OVER_CAPACITY`	The `findface-extraction-api` queue length has been exceeded.	429
`SOURCE_NOT_FOUND`	The face in the `from` parameter does not exist.	400
`SOURCE_GALLERY_NOT_FOUND`	The gallery in the `from` parameter does not exist.	400
`STORAGE_ERROR`	The biometric database not available.	503
`CACHE_ERROR`	Memcached not available.	503
`NOT_FOUND`	Matching faces not found.	404
`NOT_IMPLEMENTED`	This functionality not implemented.	501
`GALLERY_NOT_FOUND`	Matching galleries not found.	404

6.2 Biometric API Methods

In this section:

6.2.1 Detect Face in Image

```
/detect POST
```

This method detects a face in a provided image and returns coordinates of the rectangle around the face (a.k.a. bbox) and the face orientation.

Note: Face detection is done by the `findface-extraction-api` component, so the `findface-sf-api` component formats your initial request and forwards it to `findface-extraction-api`.

Important: Be sure to pass the enabled `facen` parameter in the `/detect POST` query string in order to save the returned result in `memcached`. To retrieve the returned result from `memcached`, use the `/detect GET` method.

Tip: To enable a boolean parameter (`gender`, `age`, etc.), use any of the following strings: `1`, `yes`, `true`, or `on`, in any letter case.

Important: To enable recognition of face features, you can use either the new (preferred) or old API parameters (see the query string parameters for details). The old API allows you to recognize gender, age, emotions, and country, while the new API provides recognition of gender, age, emotions, country, beard, and glasses. Each face feature (gender, age, emotions, country, beard, or glasses) must be mentioned only once in a request, either in the new or old API format.

Query string parameters:

- `"detector"`: string, face detector to be applied to the image: `nnd` (regular detector) or `normalized` (accepts a normalized face image, skipping the face detection stage).
- `"gender"`: Boolean, enables gender recognition (old API).
- `"age"`: Boolean, enables age recognition (old API).
- `"emotions"`: Boolean, enables emotions recognition (old API).
- `"facen"`: Boolean, the formatted request to `findface-extraction-api` will include such parameters as `need_facen` (extract a face feature vector) and `need_normalized` (obtain a normalized face image), while the full `findface-extraction-api` response will be saved in `memcached` under a temporary UUID. You can find this UUID in the `id` field of the response.
- `"countries47::` Boolean, enables country recognition (old API).
- `"autorotate"`: Boolean, auto-rotates an original image to 4 different orientations and returns faces detected in each orientation.
- `"return_facen"`: Boolean, returns a face feature vector in the response. Requires the enabled `allow-return-facen` flag in the `findface-sf-api` configuration file.
- `"attribute"`: Array of strings in the format `["gender", "age", "emotions", "countries47", "beard", "glasses3"]`, enables recognition of the face features passed in the array (new API).

Parameters in request body:

Image as a file of the `image/jpeg`, `image/png`, or `image/webp` MIME-type, or as a `text/x-url` link to a relevant public URL.

Returns:

- list of coordinates of the rectangles around the detected faces;
- temporary UUID of the detection result (`id`, if `facen` enabled);

> **Important:** When writing code, be sure to check the relevance of the temporary UUID before you refer to it as it tends to become irrelevant with time. If so, re-detect the face.

- feature vector (if `return_facen` enabled);
- gender (if `gender` enabled): `male` or `female`, with algorithm confidence in the result (`"score"`);
- age (if `age` enabled): number of years;
- emotions (if `emotions` enabled): 6 basic emotions + neutral (`angry`, `disgust`, `fear`, `happy`, `sad`, `surprise`, `neutral`) with algorithm confidence in each emotion expression;
- countries (if `countries47` enabled): probable countries of origin with algorithm confidence in the result;
- attributes (if passed): gender (`male` or `female`), age (number of years), emotions (predominant emotion), probable countries of origin, beard (`beard` or `none`), glasses (`sun`, `eye`, or `none`), along with algorithm confidence in the result;
- orientation.

Examples

Request #1. Old and new API simultaneous usage

```
curl -i -X POST 'http://127.0.0.1:18411/v2/detect?facen=on&age=on&gender=on&
↪emotions=on&attribute=glasses3' -H 'Content-Type: image/jpeg' --data-binary @sample.
↪jpg
HTTP/1.1 100 Continue
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:BpLnfgDs
Date: Thu, 23 May 2019 12:00:22 GMT
Content-Length: 713

{
    "faces": [
        {
            "id": "bjj8mlhjisgjrk6hj1v0",
            "bbox": { "left": 595, "top": 127, "right": 812, "bottom": 344 },
            "features": {
                "gender": { "gender": "FEMALE", "score": 0.9998938 },
                "age": 25,
                "score": -0.000696103,
                "emotions": [
                    { "emotion": "neutral", "score": 0.99958 },
                    { "emotion": "sad", "score": 0.0004020365 },
                    { "emotion": "happy", "score": 8.603454e-06 },
                    { "emotion": "surprise", "score": 8.076766e-06 },
                    { "emotion": "disgust", "score": 6.6535216e-07 },
                    { "emotion": "angry", "score": 6.1434775e-07 },
                    { "emotion": "fear", "score": 7.3372125e-10 }
                ],
                "attributes": {
                    "glasses3": {
                        "attribute": "glasses3",
                        "model": "glasses3.v0",
                        "result": [
                            { "confidence": 0.99958307, "name": "none" },
                            { "confidence": 0.00033243417, "name": "eye" },
                            { "confidence": 8.4465064e-05, "name": "sun" }
                        ]
                    }
                }
            }
        }
    ],
    "orientation": 1
}
```

Request #2. New API usage

```
curl -s -X POST 'http://master:18411/v2/detect?attribute=countries47&attribute=gender
↪' -H 'Content-Type: image/jpeg' --data-binary @pasha.jpg | jq
{
  "faces": [
    {
      "bbox": {
        "left": 1019,
        "top": 1138,
        "right": 1666,
        "bottom": 2041
      },
      "features": {
        "score": -0.00035252835,
        "attributes": {
          "countries47": {
            "extractor": "countries47",
            "model": "countries47.v1",
            "result": [
              {
                "confidence": 0.24693502,
                "name": "ASTL"
              },
              {
                "confidence": 0.0913519,
                "name": "POL"
              },
              {
                "confidence": 0.05358066,
                "name": "ARG"
              },
              {
                "confidence": 0.05128677,
                "name": "UKR"
              },
              {
                "confidence": 0.043531597,
                "name": "GER"
              },
              {
                "confidence": 0.03273358,
                "name": "CZEC"
              },
              {
                "confidence": 0.03272725,
                "name": "LEBN"
              },
              {
                "confidence": 0.02950912,
                "name": "CHIL"
              },
              {
                "confidence": 0.027654657,
                "name": "RUS"
              },
              {
```

(continues on next page)

(continued from previous page)

```
                "confidence": 0.026015146,
                "name": "SAFR"
              },
              {
                "confidence": 0.025724495,
                "name": "GRBR"
              },
              {
                "confidence": 0.025504896,
                "name": "CSTR"
              },
              {
                "confidence": 0.023839278,
                "name": "KOR"
              },
              {
                "confidence": 0.022917742,
                "name": "ISRL"
              },
              {
                "confidence": 0.015377127,
                "name": "PKST"
              },
              {
                "confidence": 0.015136372,
                "name": "TRKY"
              },
              {
                "confidence": 0.014596664,
                "name": "ROM"
              },
              {
                "confidence": 0.014106703,
                "name": "ELSL"
              },
              {
                "confidence": 0.013242814,
                "name": "IND"
              },
              {
                "confidence": 0.012512706,
                "name": "IDSA"
              },
              {
                "confidence": 0.010475861,
                "name": "SARB"
              },
              {
                "confidence": 0.010281001,
                "name": "GUAT"
              },
              {
                "confidence": 0.009767002,
                "name": "GRC"
              },
              {
                "confidence": 0.009712666,
```

(continues on next page)

(continued from previous page)

```
                "name": "VENZ"
              },
              {
                "confidence": 0.0096269725,
                "name": "JAM"
              },
              {
                "confidence": 0.009401413,
                "name": "HAT"
              },
              {
                "confidence": 0.008189098,
                "name": "CUBA"
              },
              {
                "confidence": 0.007926449,
                "name": "GHAN"
              },
              {
                "confidence": 0.0077127796,
                "name": "HOND"
              },
              {
                "confidence": 0.007496704,
                "name": "NRA"
              },
              {
                "confidence": 0.0072590904,
                "name": "ECUA"
              },
              {
                "confidence": 0.007247953,
                "name": "CHIN"
              },
              {
                "confidence": 0.0072472636,
                "name": "PHIL"
              },
              {
                "confidence": 0.006719906,
                "name": "JPN"
              },
              {
                "confidence": 0.0062700314,
                "name": "DOMR"
              },
              {
                "confidence": 0.005724779,
                "name": "MEX"
              },
              {
                "confidence": 0.005676317,
                "name": "PERU"
              },
              {
                "confidence": 0.0055375276,
                "name": "BRZL"
```

(continues on next page)

(continued from previous page)

```
              },
              {
                "confidence": 0.005422363,
                "name": "EGYP"
              },
              {
                "confidence": 0.005258461,
                "name": "MLAS"
              },
              {
                "confidence": 0.004999833,
                "name": "KENY"
              },
              {
                "confidence": 0.004916244,
                "name": "TWAN"
              },
              {
                "confidence": 0.0045134826,
                "name": "VTNM"
              },
              {
                "confidence": 0.004322668,
                "name": "HNK"
              },
              {
                "confidence": 0.004172176,
                "name": "THAI"
              },
              {
                "confidence": 0.0031889428,
                "name": "TRIN"
              },
              {
                "confidence": 0.0026484467,
                "name": "NEP"
              }
            ]
          },
          "gender": {
            "extractor": "gender",
            "model": "gender.v2",
            "result": [
              {
                "confidence": 0.9999999,
                "name": "male"
              },
              {
                "confidence": 7.935678e-08,
                "name": "female"
              }
            ]
          },
          "quality": {
            "extractor": "quality",
            "model": "quality.v0",
            "result": 0.00035252835
```

(continues on next page)

(continued from previous page)

```
          }
        }
      }
    }
  ],
  "orientation": 1
}
```

6.2.2 Retrieve Detection Result from `memcached`

```
/detect/:id GET
```

This method retrieves the detection results from `memcached` by their temporary UUID's (including feature vectors of the detected faces).

Parameters in path segments:

- `:id`: the detection result temporary UUID in `memcached`.

Returns:

JSON representation of the detection result.

Example

Request

```
curl -i -X GET 'http://127.0.0.1:18411/v2/detect/bg2gu31jisghl6pee09g'
```

Response:

```
{
    "bbox": { "bottom": 343, "left": 593, "right": 824, "top": 112 },
    "features": {
        "age": 26.096783,
        "emotions": [
            { "emotion": "neutral", "score": 0.9094986 },
            { "emotion": "happy", "score": 0.11464329 },
            { "emotion": "sad", "score": 0.005675929 },
            { "emotion": "surprise", "score": -0.02556022 },
            { "emotion": "fear", "score": -0.14499822 },
            { "emotion": "angry", "score": -0.19491306 },
            { "emotion": "disgust", "score": -0.31617728 }
        ],
        "gender": { "gender": "FEMALE", "score": -2.7309942 },
        "score": -0.000696103
    },
    "id": "bg2gu31jisghl6pee09g"
}
```

6.2.3 Create Detection Result out of `findface-extraction-api` Response

```
/detect/:id POST
```

This method creates a detection result out of a `findface-extraction-api` response.

Parameters in path segments:

- `:id`: specify UUID under which the newly created detection result will be stored in cache.

Returns:

Empty JSON on success.

Example

Request

```
$ curl -i -X POST 'http://127.0.0.1:18411/v2/detect/bg2gu31jisghl6peea9g' -H 'Content-
↪Type: application/json' --data-binary '@extapi-face.json'
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: jFSBuSPm
Date: Wed, 05 Dec 2018 08:08:56 GMT
Content-Length: 2

{}
```

6.2.4 List Database Galleries

```
/galleries GET
```

This method returns the list of all galleries in the biometric database.

Parameters:

The method doesn't accept any parameters.

Returns:

JSON dictionary with the list of gallery names.

Example

Request

```
GET /v2/galleries HTTP/1.1
Host: 172.17.47.19:18411
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 02 Feb 2018 10:11:43 GMT
Content-Length: 35

{"galleries":[{"name":"sandbox"}]}
```

6.2.5 Create Database Gallery

```
/galleries/:gallery POST
```

This method creates a gallery under a given name.

Parameters in path segments:

`:gallery`: a new gallery's name. It can contain English letters, numbers, underscore and minus sign (`[a-zA-Z0-9_-]+`) and must be no longer than 48 characters.

Returns:

- Empty JSON on success.
- JSON with a relevant error description on failure.

Example

Request

```
POST /v2/galleries/newone HTTP/1.1
Host: 172.17.47.19:18411
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 02 Feb 2018 10:18:01 GMT
Content-Length: 2

{}
```

6.2.6 Retrieve Gallery Details

```
/galleries/:gallery GET
```

This method checks a gallery existence and retrieves the number of faces in it.

Parameters in path segments:

`:gallery`: a gallery's name.

Returns:

- JSON dictionary with the number of faces and gallery name on success.
- JSON with a relevant error description on failure.

Example

Request

```
curl -i -X GET 'http://127.0.0.1:18411/v2/galleries/hello'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: Ard3exjn
Date: Wed, 05 Dec 2018 08:17:54 GMT
Content-Length: 29

{ "faces": 123, "name": "hello" }
```

6.2.7 Delete Gallery

```
/galleries/:gallery DELETE
```

This method deletes a given gallery with all the faces.

Parameters in path segments:

`:gallery`: the name of the gallery to be deleted.

Returns:

- Empty JSON on success.
- JSON with a relevant error description on failure.

Example

Request

```
DELETE /v2/galleries/newone HTTP/1.1
Host: 172.17.47.19:18411
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Fri, 02 Feb 2018 10:18:01 GMT
Content-Length: 2

{}
```

6.2.8 Add Face to Database

```
/galleries/:gallery/faces/:id POST
```

This method takes a detected face from `memcached` or a face from a gallery. It then adds the face with its feature vector to a given gallery under a custom id. The custom id and destination gallery are to be specified in the path segments. Along with the face, you can also add metadata which uniquely describes the person, for example, the person's name.

Parameters in path segments:

- `:gallery`: the name of the gallery to add the face in.
- `:id`: permanent face id in the gallery, uint64.

Parameters in request body:

- `"from"`: temporary UUID of the detected face in `memcached` (`"from": "detection:<id>"`) or face id in a gallery (`"from: "face:<gallery>/<id>"`).
- `"meta"` [optional]: the person's metadata such as the person's name, original image details, detection date and time, etc., dictionary.

Returns:

- JSON representation of the added face on success.
- Error on failure.

Example

Request

```
curl -i -X POST http://127.0.0.1:18411/v2/galleries/hello/faces/123/ -H 'Content-
↪Type: application/json' --data-binary '@-' <<EOF
{
    "from": "detection:bg2gu31jisghl6pee09g",
    "meta": {
        "camera": "openspace",
        "labels": ["foo", "bar"],
        "timestamp": "1543837276"
    }
}
EOF
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:OSSKbJg3
Date: Wed, 05 Dec 2018 08:27:59 GMT
Content-Length: 555

{
    "features": {
        "age": 26.096783,
        "emotions": [
            { "emotion": "neutral", "score": 0.9094986 },
            { "emotion": "happy", "score": 0.11464329 },
            { "emotion": "sad", "score": 0.005675929 },
            { "emotion": "surprise", "score": -0.02556022 },
            { "emotion": "fear", "score": -0.14499822 },
            { "emotion": "angry", "score": -0.19491306 },
            { "emotion": "disgust", "score": -0.31617728 }
        ],
        "gender": { "gender": "FEMALE", "score": -2.7309942 },
        "score": -0.000696103
    },
    "id": { "face": 123, "gallery": "hello" },
    "meta": {
        "camera": "openspace",
        "labels": ["foo", "bar"],
        "timestamp": "1543837276"
    },
    "normalized_id": "123_bg2hcupjisghl6pee0ag.png"
}
```

6.2.9 Retrieve Face from Gallery

```
/galleries/:gallery/faces/:id GET
```

This method retrieves a face from a database gallery by id.

Parameters in path segments:

- `:gallery`: the name of the gallery to retrieve the face from.
- `:id`: face id in the gallery, uint64.

Returns:

- JSON representation of the retrieved face on success.
- Error on failure.

Example

Request

```
curl -s 'http://172.17.47.19:18411/v2/galleries/galleryname/faces/2' | jq
```

Response

```
{
  "id": {
    "gallery": "galleryname",
    "face": 2
  },
  "features": {
    "gender": {
      "gender": "FEMALE",
      "score": -2.6415923
    },
    "age": 26.04833,
    "score": 0.9999999,
    "emotions": [
      {
        "emotion": "neutral",
        "score": 0.99958
      },
      {
        "emotion": "sad",
        "score": 0.0004020398
      },
      {
        "emotion": "happy",
        "score": 8.603504e-06
      },
      {
        "emotion": "surprise",
        "score": 8.076798e-06
      },
      {
        "emotion": "disgust",
        "score": 6.653509e-07
      },
      {
```

(continues on next page)

(continued from previous page)

```
        "emotion": "angry",
        "score": 6.14346e-07
      },
      {
        "emotion": "fear",
        "score": 7.33713e-10
      }
    ]
  },
  "meta": {
    "timestamp": 2
  },
  "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
}
```

6.2.10 Delete Face from Gallery

```
/galleries/:gallery/faces/:id DELETE
```

This method deletes a face from a database gallery by id.

Parameters in path segments:

- `:gallery`: the name of the gallery to delete the face from.
- `:id`: face id in the gallery, uint64.

Returns:

- Empty JSON on success.
- Error on failure.

Example

Request

```
curl -s -X DELETE 'http://172.17.47.19:18411/v2/galleries/galleryname/faces/1' | jq
```

Response

```
{}
```

6.2.11 Update Face Metadata in Gallery

```
/galleries/:gallery/faces/:id PATCH
```

The method updates a face metadata in a database gallery by id.

Parameters in path segments:

- `:gallery`: the gallery's name.
- `:id`: face id in the gallery, uint64.

Parameters in request body

- `"meta"`: dictionary with the face's new metastrings.

Returns:

- JSON representation of the updated face on success.
- Error on failure.

Example

Request

```
curl -s -X PATCH -H 'Content-Type: application/json' --data '{"meta":{"timestamp":2}}
↪' 'http://172.17.47.19:18411/v2/galleries/galleryname/faces/2' | jq
```

Response

```
{
  "id": {
    "gallery": "galleryname",
    "face": 2
  },
  "features": {
    "gender": {
      "gender": "FEMALE",
      "score": -2.6415923
    },
    "age": 26.04833,
    "score": 0.9999999,
    "emotions": [
      {
        "emotion": "neutral",
        "score": 0.99958
      },
      {
        "emotion": "sad",
        "score": 0.0004020398
      },
      {
        "emotion": "happy",
        "score": 8.603504e-06
      },
      {
        "emotion": "surprise",
        "score": 8.076798e-06
```

(continues on next page)

(continued from previous page)

```
      },
      {
        "emotion": "disgust",
        "score": 6.653509e-07
      },
      {
        "emotion": "angry",
        "score": 6.14346e-07
      },
      {
        "emotion": "fear",
        "score": 7.33713e-10
      }
    ]
  },
  "meta": {
    "timestamp": 2
  },
  "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
}
```

6.2.12 Compare Faces

```
/verify POST
```

This method compares a pair of faces and returns a probability of their belonging to the same person (a.k.a. similarity, or confidence).

Query string parameters:

- `"face1"`: the first face, either a detection result (a result of the `/detect POST` method being stored in `memcached`), or one from the biometric database.
- `"face2"`: the second face, from the same possible sources as the first face.

Returns:

Algorithm confidence that the faces match.

Example

Request #1. Compare 2 detection results

```
curl -s 'http://172.17.47.19:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=detection:bd3hv8dt8f66ph0eag2g' | jq
```

Response

```
{
  "confidence": 0.92764723
}
```

Request #2. Compare a detection result and a face from a gallery

```
curl -s 'http://172.17.47.19:18411/v2/verify?face1=detection:bd3hv4tt8f66ph0eag1g&
↪face2=face:galleryname/2' | jq
```

Response

```
{
  "confidence": 0.999996
}
```

6.2.13 Retrieve Data from Gallery. Face Search

```
/v2/galleries/:gallery/faces GET
```

This method allows you to search faces in a gallery by using filters specified in the query string.

Parameters in path segments:

`:gallery`: the name of the gallery to search in.

Query string parameters:

- `?limit=`: (mandatory) maximum number of faces in the response.
- `?sort=`: sorting order. Pass one of the following values: `id`: increasing order by id, `-id`: decreasing order by id, `-confidence`: decreasing order by face similarity (only if you have specified a feature vector to search for).
- `?page=<page>`: cursor of the next page with search results. The `<page>` value is returned in the response in the `next_page` parameter along with the previous page results (see details below).
- `?ignore_errors`: By default, if one or several `findface-tarantool-server` shards are out of service during face identification, `findface-sf-api` returns an error. Enable this Boolean parameter to use available `findface-tarantool-server` shards to obtain face identification results.
- `?meta:in:meta1=val1&meta:in:meta1=val2&...`: select a face if a meta string `meta1` is equal to one of the values `val1`/`val2`/`...`, etc. (*uint64*, *string*).
- `?meta:gte:meta1=val1`: select all faces with a meta string `meta1` greater or equal to `val1` (*uint64*).
- `?meta:lte:meta1=val1`: select all faces with a meta string `meta1` less or equal to `val1` (*uint64*).
- `?id:in=value_id`: select all faces with `id` equal to `value_id`.
- `?id:gte=value_id`: select all faces with `id` greater or equal to `value_id`.
- `?id:lte=value_id`: select all faces with id less or equal to `value_id`.

- `?meta:subset:meta1=val1&meta:subset:meta1=val2&...:` select a face if a meta string `meta1` includes all the values `val1`, `val2`, `...`, etc. (*[]string*).
- `?<id>=<confidence>`: specifies a feature vector to search for in the biometric database, via the `<id>` parameter, as well as the threshold similarity in the search results as `<confidence>`. The `<id>` parameter can be either a face ID in a database gallery (specify `<id>` as `face:<gallery>/<db_id>`), or the temporary UUID of a detection result stored in `memcached` (`detection:<memcached_id>`) (see the `/detect POST` method and examples below). The <confidence> ranges from 0 to 1.

Returns:

JSON representation of an array with found faces. By default, faces in the response are sorted by id. Should you specify a feature vector to search for, faces will be sorted in decreasing order by similarity.

The response format is the following:

```
{
    "faces": [
      {
        ... face 1 data ...
        "confidence": 0.123 // if you search for a feature vector
      },
      {
        ... face 2 data ...
        "confidence": 0.123 // if you search for a feature vector
      },
      ...
    ],
    "next_page": "vgszk2bkexbl" // next page cursor
}
```

The `next_page` parameter is a URL-safe string that you will have to pass in the `?page=` in the next request in order to get the next page of results. Pagination is available only if the filtration by feature vector is disabled.

Request #1. Face identification (search a gallery for a face)

```
curl -s 'http://172.17.47.19:18411/v2/galleries/galleryname/faces?
↪detection:bd3hv4tt8f66ph0eag1g=0.5&limit=1' | jq
```

Response

```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
```

(continues on next page)

(continued from previous page)

```
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png",
      "confidence": 0.9999
    }
  ],
  "next_page": "There are more than 1 results, but pagination is not supported when␣
↪filtering by FaceN"
}
```

Request #2. List faces in gallery

```
curl -s 'http://172.17.47.19:18411/v2/galleries/galleryname/faces?limit=2' | jq
```

Response

```
{
  "faces": [
    {
      "id": {
        "gallery": "galleryname",
        "face": 1
```

(continues on next page)

(continued from previous page)

```
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "1_bd321tlt8f66ph0eaflg.png"
    },
    {
      "id": {
        "gallery": "galleryname",
        "face": 2
      },
      "features": {
        "gender": {
          "gender": "FEMALE",
          "score": -2.6415923
        },
        "age": 26.04833,
        "score": 0.9999999,
        "emotions": [
          {
            "emotion": "neutral",
```

(continues on next page)

(continued from previous page)

```
            "score": 0.99958
          },
          {
            "emotion": "sad",
            "score": 0.0004020398
          },
          {
            "emotion": "happy",
            "score": 8.603504e-06
          },
          {
            "emotion": "surprise",
            "score": 8.076798e-06
          },
          {
            "emotion": "disgust",
            "score": 6.653509e-07
          },
          {
            "emotion": "angry",
            "score": 6.14346e-07
          },
          {
            "emotion": "fear",
            "score": 7.33713e-10
          }
        ]
      },
      "meta": {},
      "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
    }
  ],
  "next_page": "3"
}
```

Request #3. Advanced face identification

```
curl -i -X GET http://127.0.0.1:18411/v2/galleries/history/faces/?limit=5&
↪meta:in:camera=openspace&meta:in:camera=entrance&meta:lte:timestamp=1543845934&
↪meta:gte:timestamp=1514801655&detection:bg2gu31jisghl6pee09g=0.4 | jq
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:ibKVYpcb
Date: Wed, 05 Dec 2018 08:37:33 GMT
Transfer-Encoding: chunked

{
    "faces": [
        {
            "confidence": 0.6026,
```

(continues on next page)

(continued from previous page)

```
            "features": { "score": 1 },
            "id": { "face": 4141715030051545133, "gallery": "history" },
            "meta": {
                "bbox": "[607, 802, 738, 933]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999998079040,
                "timestamp": 1542909082
            },
            "normalized_id": "4141715030051545133_bfrep71jisghl6pedvk0.png"
        },
        {
            "confidence": 0.5325,
            "features": { "score": 1 },
            "id": { "face": 4141715086422990894, "gallery": "history" },
            "meta": {
                "bbox": "[741, 905, 953, 1117]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999993877300,
                "timestamp": 1542909103
            },
            "normalized_id": "4141715086422990894_bfrepc9jisghl6pedvl0.png"
        },
        {
            "confidence": 0.531,
            "features": {
                "age": 41.2622,
                "gender": { "gender": "FEMALE", "score": -0.880698 },
                "score": 1
            },
            "id": { "face": 4141716493024780347, "gallery": "history" },
            "meta": {
                "bbox": "[90, 869, 166, 945]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 10000000000000000013,
                "timestamp": 1542909627
            },
            "normalized_id": "4141716493024780347_bfretf9jisghl6pee020.png"
        },
        {
            "confidence": 0.5236,
            "features": {
                "age": 48.949913,
                "gender": { "gender": "FEMALE", "score": -0.7653318 },
                "score": 1
            },
            "id": { "face": 4141716498393489468, "gallery": "history" },
            "meta": {
                "bbox": "[56, 853, 125, 923]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
```

(continues on next page)

(continued from previous page)

```
                "score": 9999999999999999053,
                "timestamp": 1542909629
            },
            "normalized_id": "4141716498393489468_bfretg1jisghl6pee030.png"
        },
        {
            "confidence": 0.5212,
            "features": {
                "age": 33.3112,
                "gender": { "gender": "MALE", "score": 1.9504981 },
                "score": 1
            },
            "id": { "face": 4141715338752319538, "gallery": "history" },
            "meta": {
                "bbox": "[-36, 862, 60, 958]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999999999425,
                "timestamp": 1542909197
            },
            "normalized_id": "4141715338752319538_bfreq4pjisghl6pedvp0.png"
        }
    ],
    "next_page": "There are more than 5 results, but pagination is not supported when␣
↪filtering by FaceN"
}
```

Tip: You can also find the biometric API documentation on our website and at `http://<findface-sf-api_ip>:18411/v2/docs`.

CHAPTER 7

Video Face Detection API

7.1 How to Use Video Face Detection API

In this section:

7.1.1 Endpoint

Video face detection API requests are to be sent to `http://<findface-video-manager IP address>:18810/`. API requests are executed by the `findface-video-manager` component.

7.1.2 Job Object

Video face detection API operates on a `job` object which represents a video processing task that the `findface-video-manager` component issues to `findface-video-worker`.

Each `job` object has the following attributes:

- `id`: job id specified by a user.
- `stream_url`: URL/address of video stream/file to process.
- `labels`: tag(s) that will be used by the `findface-facerouter` component to find processing directives for faces detected in this stream.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).

- `router_url`: IP address and port of the `findface-facerouter` component to receive detected faces from the `findface-video-worker` component for processing.
- `status`: job status.
- `status_msg`: additional job status info.
- `statistic`: job progress statistics (progress duration, number of posted faces).
- `worker_id`: id of the `findface-video-worker` instance executing the job.

7.1.3 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 `desc`.

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

Common Error Codes

Error code	Description	HTTP code
`UNKNOWN_ERROR`	Error with unknown origin.	500
`BAD_REQUEST`	The request cannot be read, or some method parameters are invalid.	400
`CONFLICT`	Conflict.	409
`NOT_FOUND`	Job not found.	404
`DELETING`	The previously requested job removal is in progress.	423

7.2 Video Face Detection API Methods

In this section:

7.2.1 Create Job

```
POST /job/:id
```

This method creates a video processing task job for the `findface-video-worker` component.

Parameters in path segments

`:id`: job id

Parameters in request body:

- `stream_url`: URL/address of a video stream/file to process.
- `labels`: tag(s) that will be used by the `findface-facerouter` component to find processing directives for faces detected in this video stream.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).
- Other video stream parameters that differ from common video stream parameters specified in the `findface-video-manager` configuration file.

Returns:

A `job` object: all parameters from the request, as well as some read-only attributes.

Example

Request

```
curl -s 'http://localhost:18810/job/myid-123' --data '{"stream_url":"http://1.2.3.4/
↪stream.mp4", "labels": {"district": "SVAO"}}' | jq
```

Response

```
{
  "id": "myid-123",
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://localhost:1514/",
  "single_pass": false,
  "status": "AWAITING",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0
  },
  "worker_id": ""
}
```

7.2.2 List Existing Jobs

```
GET /jobs
```

This method returns the list of all existing jobs.

Parameters:

This method doesn't accept any parameters.

Returns:

JSON representation with the list of all jobs.

Example

Request

```
curl -s 'http://localhost:18810/jobs' | jq
```

Response

```
[
    {
        "id": "b9c73bhg74hnekpaa0o0",
        "stream_url": "http://1.2.3.4/stream.mp4",
        "labels": {
            "district": "SVAO"
        },
        "router_url": "http://localhost:1514/",
        "single_pass": false,
        "status": "AWAITING",
        "status_msg": "",
        "worker_id": ""
    },
    {
        "id": "b9c73rhg74hnekpaa0og",
        "stream_url": "http://xxx.ru/stream.mp4",
        "labels": {
            "district": "ZAO"
        },
        "router_url": "http://localhost:1514/",
        "single_pass": false,
        "status": "AWAITING",
        "status_msg": "",
        "worker_id": ""
    }
]
```

7.2.3 Retrieve Job Parameters

```
GET /job/:id
```

This method retrieves a job parameters by id.

Parameters in path segments:

`id`: job id.

Returns:

JSON representation of the job object.

Example

Request

```
curl -s 'http://localhost:18810/job/b9c73rhg74hnekpaa0og' | jq
```

Response

```
{
    "id": "b9c73rhg74hnekpaa0og",
    "stream_url": "http://xxx.ru/stream.mp4",
    "labels": {
        "district": "ZAO"
    },
    "router_url": "http://localhost:1514/",
    "single_pass": false,
    "status": "AWAITING",
    "status_msg": "",
    "worker_id": ""
}
```

7.2.4 Delete Job

```
DELETE /job/:id
```

This method deletes a job by id.

Parameters in path segments:

`id`: job id.

Returns:

JSON representation of the deleted job object.

Example

Request

```
curl -s 'http://localhost:18810/job/myid-123' -X DELETE | jq
```

Response

```
{
  "id": "myid-123",
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://myrouter",
  "single_pass": false,
  "status": "DELETED",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0
  },
  "worker_id": "b9kqns1g74hm6mbmhbqg"
}
```

7.2.5 Update Job

```
PATCH /job/:id
```

The method updates a job parameters by id.

Parameters in path segments:

`id`: job id.

Parameters in request body:

- `id`: job id.
- `stream_url`: URL/address of a video stream/file to process.
- `labels`: tag(s) that will be used by the `findface-facerouter` component to find processing directives for faces detected in this video stream.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).
- `router_url`: IP address and port of the `findface-facerouter` component to receive detected faces from the `findface-video-worker` component for processing.
- `status`: job status.
- `status_msg`: additional job status info.
- `statistic`: job progress statistics (progress duration, number of posted faces).

- `worker_id`: id of the `findface-video-worker` instance executing the job.
- New values of to-be-modified `find-video-manager` configuration parameters. These value have priority over those specified in the `findface-video-manager` configuration file.

Returns:

JSON representation of the updated job object.

Example

Request

```
curl -s 'http://localhost:18810/job/myid-123' -X PATCH --data '{"router_url":"http://
↪myrouter"}' | jq
```

```
{
  "id": "myid-123",
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://myrouter",
  "single_pass": false,
  "status": "INPROGRESS",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0
  },
  "worker_id": "b9kqns1g74hm6mbmhbqg"
}
```

7.2.6 Restart Job

```
RESTART /job/:id
```

This method restarts a job by ID.

Parameters in path segments:

`id`: job id.

Returns:

`HTTP/1.1 200 OK` on success.

Example

Request

```
curl -s -D - -X RESTART http://localhost:18810/job/1
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: VbhV3vC5
Date: Tue, 24 Apr 2018 15:23:19 GMT
Content-Length: 0
```

Tip: You can also find the video face detection API documentation at `http://<findface-video-manager_ip>:18810/docs`.

CHAPTER 8

Set Face Processing Directives

In the course of configuring the system, you have to set directives that determine how the system processes a face after it has been detected in video. To do so, you need to write a Python plugin(s).

Plugins are enabled through the `findface-facerouter` configuration file. They allow you to configure video face detection outcome individually for each use case.

8.1 Configure `findface-facerouter` to Use Plugins

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

1. Put a plugin into a directory of your choice. You can distribute a set of plugins across several directories.
2. Open the `findface-facerouter` configuration file.

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

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

3. Uncomment the `plugins_dirs` parameter and specify the comma-separated list of plugin directories.

   ```
   plugins_dirs                    = '/etc/findface/plugins/video, /etc/findface/
   ↪plugins/html'
   ```

4. Save the configuration file.

8.2 Basics

In this section:

8.2.1 Plugin Architecture

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

The `activate` function has the following parameters:

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

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

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

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

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

8.2.2 The `preprocess` method

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

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

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

Parameters

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

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

Parameters

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

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

8.2.3 The `process` method

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

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

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

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

8.2.4 The `shutdown` method

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

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

shutdown (*self*)

8.3 Classes and Methods

In this section:

8.3.1 Basic Classes

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

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

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

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

Parameters

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

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

Return type tuple

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

Parameters

- **photo** (*bytes*) – JPEG video frame featuring a detected face
- **face0** (*bytes*) – normalized face image
- **bbox** (list of integers [[x1,y1,x2,y2]], where `x1`: x coordinate of the top-left corner, `y1`: y coordinate of the top-left corner, `x2`: x coordinate of the bottom-right corner, `y2`: y coordinate of the bottom-right corner) – coordinates of the face region in the video frame
- **cam_id** (*string*) – camera id
- **timestamp** (*datetime.datetime*) – video frame timestamp
- **detectorParams** (*dictionary*) – debug information from the video face detector

- **bs_type** (*string*) – best face search mode. Available options: overall (the `findface-video-worker` posts only one snapshot per track, but of the highest quality.), realtime (the `findface-video-worker` posts the best snapshot within each of consecutive time intervals).
- **labels** (*dictionary*) – (duplicates params.labels) a custom set of a frame labels, which are specified in a job parameters for `findface-video-worker` and then assigned to the frame

process (*self, request: FrHTTPRequest, photo: bytes, bbox: typing.List[int], event_id: int, detection: DetectFace*)
Accepts the detected face features.

Parameters

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

Returns n/a

Return type n/a

shutdown (*self*)
This method is invoked before the `findface-facerouter` shutdown.

Param n/a

Returns n/a

8.3.2 Object Classes

class objects.**BBox**
Represents coordinates of the rectangle around a face.

class objects.**DetectFace**
Represents a detection result with the following fields:

Parameters

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

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

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

class `objects.`**`FaceId`**(*namedtuple('FaceId', ('gallery', 'face'))*)
Represents a custom face identifier object in the gallery.

Parameters

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

class `objects.`**`Face`**
Represents a result of database search by biometric sample

Parameters

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

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

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

8.3.3 Face Detection and Gallery Management

class `ntech.sfapi_client.client.`**`Client`**
Represents basic methods to detect faces in images and work with galleries.

`detect`(*self*, ***, *url=None*, *image=None*, *facen=False*, *gender=False*, *age=False*, *emotions=False*, *return_facen=False*, *autorotate=False*, *detector: str = None*, *timeout=None*) → DetectResponse
Detects a face and returns the result of detection.

Parameters

- **`url`** (*`URL`*) – image URL if you pass an image that is publicly accessible on the internet
- **`image`** (*`bytes`*) – PNG/JPG/WEBP image file is you pass an image as a file
- **`facen`** (*`boolean`*) – extract a biometric sample from the detected face. To save the detection result in memcached pass `facen=True`
- **`gender`** (*`boolean`*) – extract and return information about gender
- **`age`** (*`boolean`*) – extract and return information about age
- **`emotions`** (*`boolean`*) – extract and return information about emotions
- **`return_facen`** (*`boolean`*) – return facen in the method result
- **`autorotate`** (*`boolean`*) – automatically rotate the image in 4 different orientations to detect faces in each of them. Overlapping detections with IOU > 0.5 will be merged
- **`detector`** (*`boolean`*) – nnd or normalized. The `normalized` detector is used to process normalized images, for example, those which are received from `findface-video-worker`.

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

Returns Detection result

Return type `DetectorResponse` object.

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

Parameters **name** (*string*) – gallery name

Returns a gallery object

Return type sfapi_client.Gallery

list_galleries(self, timeout=None):
Returns the list of galleries.

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

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

Return type list of GalleryListItem

class ntech.sfapi_client.gallery.**Gallery**
Provides methods to work with galleries and faces.

list (*self*, ***, *filters: typing.Iterable[filters.Filter] = None*, *limit: int = 1000*, *sort: str = ”*, *page=None*, *ignore_errors=False*, *timeout=None*) → ListResponse
Returns a list-like object with faces from the gallery, that match the given filters. The returned list-like object has an additional property `next_page` which can be used as a value for the `page` parameter in next requests.

Parameters

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

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

Return type ListResponse object

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

Creates a face in the gallery.

Parameters

- **new_id** (*integer or callable*) – custom face identifier (Face ID) in the database gallery. May be a (async) callable which returns the id. To generate id, you can use the `ctx.idgen()` function delivered with the context.
- **source** (*sfapi_client.DetectFace, sfapi_client.Face, sfapi_client.FaceId, or string*) – face source: create a face using another face in the database or a detection result as a source.
- **meta** (*dictionary*) – face metadata. Keys must be strings and values must be either ints, strings or lists of strings. Metadata keys and types must be previously specified in the storage (`findface-tarantool-server`) configuration files.
- **regenerate_attempts** – number of attempts to regenerate a unique Face ID with the `ctx.idgen()` function if new_id is callable
- **timeout** (*number*) – FindFace core response timeout, in seconds (if `none`, the default value is used)

Returns representation of the newly created face

Return type Face object

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

Removes a face from the gallery.

Parameters

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

Returns None

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

Retrieves a face from the gallery.

Parameters

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

Returns representation of the face

Return type Face object

create(*self, timeout=None*) → None

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

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

Returns None

drop(*self*, *timeout=None*) → None:

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

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

Returns None

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

Update face meta data in the gallery.

Parameters

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

Returns representation of the updated face

Return type Face object

8.3.4 Filters for Database Search

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

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

serialize(*self*)

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

Returns filter names and filter values

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

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

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

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

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

Parameters **value** (*integer*) – id value

Returns filter name (LTE) and its value.

Return type object of `Filter` class.

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

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

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

Parameters **value** (*integer*) – id value

Returns filter name (GTE) and its value.

Return type object of `Filter` class.

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

classmethod oneof(*cls, *value: typing.Union[int]*) → Filter
IN filter. Select a face(s) with id from a given set.

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

Returns filter name (IN) and its value.

Return type object of `Filter` class.

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

class `ntech.sfapi_client.filters.`**Meta**
Represents methods for filtering gallery content by metadata. Don't instantiate, use relevant classmethods to call a filter.

classmethod lte(*self, value: typing.Union[str, int]*) → Filter
LTE filter. Select all faces with a metastring less or equal to `value`

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

Returns filter name (LTE) and its value.

Return type object of `Filter` class.

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

classmethod gte(*self, value: typing.Union[str, int]*) → Filter
GTE filter. Select all faces with a metastring greater or equal to `value`

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

Returns filter name (GTE) and its value.

Return type object of `Filter` class.

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

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

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

param value list of metastring values

type value list of strings or integers

return filter name (IN) and its value.

rtype object of `Filter` class.

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

classmethod subset(*self, *value: str*) → Filter
SUBSET filter. Select all faces with a metastring featuring all values from a given set.

Parameters value (*list of strings or integers*) – list of metastring values
Returns filter name (SUBSET) and its value.
Return type object of `Filter` class.

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

class `ntech.sfapi_client.filters.`**Detection**(*Filter*)
Represents a method that identifies a detected face (searches the database for similar faces).

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

Parameters

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

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

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

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

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

Parameters

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

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

Several Filters Usage Example

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

8.3.5 Display Error Messages

class sfapi_client.**SFApiRemoteError**

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

The error body always includes at least two fields: `code` and `desc`.

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

Common Error Codes

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

class `sfapi_client.`**`SFApiMalformedResponseError`**

This error message appears if the error occurred due to a network failure, or if Client was unable to read an API response from `findface-sf-api`.

8.4 Examples

The following examples illustrate the basics of writing a plugin, as well as the use of classes and methods.

1. If a detected face contains a label `'emo'`, this plugin will request facen and emotions data extraction and then log the received data.

```
import logging

from ntech import sfapi_client

from facerouter.plugin import Plugin

logger = logging.getLogger(__name__)


class LogEmoPlugin(Plugin):
    async def preprocess(self, request, labels):
        if labels.get('emo'):
            return ('facen', 'emotions')

    async def process(self, request, photo, bbox, event_id, detection: sfapi_
↪client.DetectFace):
        logger.info('%r: %r', bbox, detection.features.get('emotions')[0]['emotion
↪'])
        logger.info('%r: params: ', bbox)
        for param in request.params._fields:
            param_repr = repr(getattr(request.params, param))
            if len(param_repr) > 100:
                param_repr = param_repr[:97] + "..."
```

(continues on next page)

(continued from previous page)

```
            logger.info("%r: %s", param, param_repr)


def activate(app, ctx, plugin_name, plugin_source):
    return LogEmoPlugin(ctx=ctx)
```

2. This plugin requests facen extraction, after that it saves a face in the `'ppl'` gallery of the biometric database. If such a gallery doesn't exist, it will be created.

```
import logging
import PIL.Image
import time
from io import BytesIO

from ntech import sfapi_client

from facerouter.plugin import Plugin

logger = logging.getLogger(__name__)


class EnrollPlugin(Plugin):
    async def preprocess(self, request, labels):
        if labels.get('lol') == 'kek':
            return ('facen',)

    async def process(self, request, photo, bbox, event_id, detection: sfapi_
↪client.DetectFace):
        img = PIL.Image.open(BytesIO(photo))
        thumb = img.crop(bbox)
        fname = '/tmp/%x.jpeg' % (event_id,)
        thumb.save(fname)
        while True:
            try:
                await self.ctx.sfapi['ppl'].add(event_id, detection, meta={
                    'timestamp': int(time.time()),
                    'photo_hash': fname,
                })
            except sfapi_client.SFApiRemoteError as e:
                if e.code == "GALLERY_NOT_FOUND":
                    await self.ctx.sfapi['ppl'].create()
                else:
                    raise
            else:
                break
        logger.info('%r: %r %r', bbox, event_id, fname)


def activate(app, ctx, plugin_name, plugin_source):
    return EnrollPlugin(ctx=ctx)
```

CHAPTER 9

Advanced Features

9.1 Direct API requests to `findface-extraction-api`

You can use HTTP API to extract data directly from the `findface-extraction-api` component.

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

Tip: Normalized images received from `findface-extraction-api` are qualified for posting to `findface-sf-api`.

In this section:

9.1.1 API Requests

The `findface-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]
    "include_timings": true|false // include face processing timing in response,␣
↪false by default
}
```

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

Important: To enable recognition of face features, you can use either the new (preferred) or old API parameters. The old API allows you to recognize gender, age, and emotions, while the new API provides recognition of gender, age, emotions, country, beard, and glasses. Each face feature (gender, age, emotions, country, beard, or glasses) must be mentioned only once in a request, either in the new or old API format.

- "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 (old API).
- "need_emotions": returns emotions (old API).
- "need_age": returns age (old API).
- "need_normalized": returns a normalized face image encoded in base64. The normalized image can then be posted again to the findface-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.
- "attributes": array of strings in the format ["gender", "age", "emotions", "countries47", "beard", "glasses3"], enables recognition of the face features passed in the array (new API).

```
{
    "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
}
```

9.1.2 API Response Format

A typical response from the `findface-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"`).
- `"facen_model"`: face extraction model if `"need_facen": true`.
- `"timings"`: processing timings if `"include_timings": true`.

```
{
    "faces": [face1, face2, .., faceN],
    "error": {
        "code": "IMAGE_DECODING_FAILED",
        "desc": "Failed to decode: reason"
    }
    "facen_model": "elderberry_576",
    "timings": ...

}
```

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"`: face feature vector.
- `"gender"`: gender information (MALE or FEMALE) with recognition accuracy if requested (old API).
- `"age"`: age estimate if requested (old API).
- `"emotions"`: all available emotions in descending order of probability if requested (old API).
- `"countries47"`: probable countries of origin with algorithm confidence in the result if requested (old API).
- `"attributes"`: gender (`male` or `female`), age (number of years), emotions (predominant emotion), probable countries of origin, beard (`beard` or `none`), glasses (`sun`, `eye`, or `none`), along with algorithm confidence in the result if requested (new API).
- `"normalized"`: a normalized face image encoded in base64 if requested.
- `"timings"`: face processing timings if requested.

```
{
 "bbox": { "left": 1, "right": 2, "top": 3, "bottom": 4},
 "detection_score": 0.99,
```

(continues on next page)

(continued from previous page)

```
"facen": "...",
"gender": {
    "gender": "MALE",
    "score": "1.123"
},
"age": 23.59,
"emotions": [
    { "emotion": "neutral", "score": 0.95 },
    { "emotion": "angry", "score": 0.55 },
    ...
],
"normalized": "...",
"attributes": {
  "age": {
      "attribute": "age",
      "model": "age.v1",
      "result": 25
  },
  "beard": {
      "attribute": "beard",
      "model": "beard.v0",
      "result": [
          { "confidence": 0.015328666, "name": "beard" }
      ]
  },
  "countries47": {
      "attribute": "countries47",
      "model": "countries47.v1",
      "result": [
          { "confidence": 0.90330666, "name": "UKR" },
          { "confidence": 0.013165677, "name": "RUS" },
          { "confidence": 0.009136979, "name": "POL" },
          ...
      ]
  },
  "emotions": {
      "attribute": "emotions",
      "model": "emotions.v1",
      "result": [
          { "confidence": 0.99959123, "name": "neutral" },
          { "confidence": 0.00039093022, "name": "sad" },
          { "confidence": 8.647058e-06, "name": "happy" },
          { "confidence": 7.994732e-06, "name": "surprise" },
          { "confidence": 6.495376e-07, "name": "disgust" },
          { "confidence": 6.063106e-07, "name": "angry" },
          { "confidence": 7.077886e-10, "name": "fear"                }
      ]
  },
  "gender": {
      "attribute": "gender",
      "model": "gender.v2",
      "result": [
          { "confidence": 0.999894, "name": "female" },
          { "confidence": 0.00010597264, "name": "male" }
      ]
  },
  "glasses3": {
```

(continues on next page)

(continued from previous page)

```
        "attribute": "glasses3",
        "model": "glasses3.v0",
        "result": [
            { "confidence": 0.9995815, "name": "none" },
            { "confidence": 0.0003348241, "name": "eye" },
            { "confidence": 8.363914e-05, "name": "sun" }
        ]
    }
  }
  "timings": ...
}
```

9.1.3 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
```

(continues on next page)

(continued from previous page)

```
```

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
            }
          ]
        }
      ]
    }
  ]
}
```

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"
       }
     ]
   }
 ]
}
```

Request #4. New API usage (attributes: "beard", "emotions", "age", "gender", "glasses3", "face")

```
curl -s -F photo=@sample.jpg -Frequest='{"requests": [{"image":"multipart:photo",
↪"detector": "nnd", "attributes": ["beard", "emotions", "age", "gender", "glasses3",
↪"face"]}]}' http://127.0.0.1:18666 | jq
```

Response

```
{
  "responses": [
    {
      "faces": [
        {
          "bbox": {
            "left": 595,
```

(continues on next page)

(continued from previous page)

```
            "top": 127,
            "right": 812,
            "bottom": 344
          },
          "detection_score": -0.00067401276,
          "rotation_angle": 0,
          "attributes": {
            "age": {
              "attribute": "age",
              "model": "age.v1",
              "result": 25
            },
            "beard": {
              "attribute": "beard",
              "model": "beard.v0",
              "result": [
                {
                  "confidence": 0.015324414,
                  "name": "beard"
                }
              ]
            },
            "emotions": {
              "attribute": "emotions",
              "model": "emotions.v1",
              "result": [
                {
                  "confidence": 0.99958,
                  "name": "neutral"
                },
                {
                  "confidence": 0.0004020365,
                  "name": "sad"
                },
                {
                  "confidence": 8.603454e-06,
                  "name": "happy"
                },
                {
                  "confidence": 8.076766e-06,
                  "name": "surprise"
                },
                {
                  "confidence": 6.6535216e-07,
                  "name": "disgust"
                },
                {
                  "confidence": 6.1434775e-07,
                  "name": "angry"
                },
                {
                  "confidence": 7.3372125e-10,
                  "name": "fear"
                }
              ]
            },
            "face": {
```

(continues on next page)

(continued from previous page)

```
              "attribute": "face",
              "model": "elderberry_576",
              "result": "KjiHu6cWh70ppqa9l"
            },
            "gender": {
              "attribute": "gender",
              "model": "gender.v2",
              "result": [
                {
                  "confidence": 0.9998938,
                  "name": "female"
                },
                {
                  "confidence": 0.000106243206,
                  "name": "male"
                }
              ]
            },
            "glasses3": {
              "attribute": "glasses3",
              "model": "glasses3.v0",
              "result": [
                {
                  "confidence": 0.99958307,
                  "name": "none"
                },
                {
                  "confidence": 0.00033243417,
                  "name": "eye"
                },
                {
                  "confidence": 8.4465064e-05,
                  "name": "sun"
                }
              ]
            }
          }
        }
      ],
      "orientation": 1
    }
  ]
}
```

9.2 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 the standalone deployment, you can access Tarantool by default only locally (`127.0.0.1`). If you want to access Tarantool remotely, *alter* the Tarantool configuration file.

In this section:

9.2.1 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.

The response will feature JSON with the following fields:

- `next`: pagination cursor to retrieve the next page with results, pass it as *:start_id* in the following request
- `total`: total number of galleries on the shard
- `galleries`: gallery list with the following data:
  - `id`: gallery id
  - `name`: gallery name
  - `cnt_linear`: number of faces in the `linear` space (faces without fast index)
  - `cnt_preindex`: number of faces in the `preindex` space (intermediate stage when creating fast index)
  - `cnt_indexed`: number of faces in the `indexed` space (faces with fast index)

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

```
HTTP/1.1 200 Ok
Content-length: 170
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"next":3,"galleries":[{"cnt_indexed":3,"id":1,"cnt_preindex":0,"name":"a","cnt_linear
↪":0},{"cnt_indexed":1,"id":2,"cnt_preindex":0,"name":"b","cnt_linear":1}],"total":5}
```

9.2.2 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.

The response will feature JSON with the following fields:

- `id`: gallery id
- `name`: gallery name
- `cptr`: uint64_t address of the gallery object in the memory
- `cnt_linear`: number of faces in the `linear` space
- `cnt_preindex`: number of faces in the `preindex` space
- `cnt_preindex_deleted`: number of faces removed from the `preindex` space, which are physically still present in Tarantool
- `cnt_indexed`: number of faces in the `indexed` space
- `cnt_indexed_deleted`: number of faces removed from the `indexed` space, which are physically still present in Tarantool
- `index_file`: path to fast index file
- `index_loaded`: indicator of whether or not fast index is loaded

Example

Request

```
http://127.0.0.1:8001/stat/info/my_gal
or
curl http://127.0.0.1:8001/stat/info/my_gal | jq
```

Response

```
HTTP/1.1 200 Ok
Content-length: 196
Server: Tarantool http (tarantool v1.7.3-673-g23cc4dc)
Connection: keep-alive

{"cnt_indexed":2464344,"cnt_preindex_deleted":139,"index_file":"none","index_loaded
↪":false,"cnt_preindex":8310,"cnt_linear":959,"cptr":29253696,"id":1,"name":"my_gal",
↪"cnt_indexed_deleted":78811}
```

9.3 Direct API Requests to Tarantool

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

In this section:

9.3.1 General Information

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

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

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

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

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

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

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

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

9.3.2 Add Face

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

Parameters in body:

JSON-encoded array of faces with the following fields:

- `"id"`: face id in the gallery, uint64_t,
- `"facen"`: raw feature vector, base64,
- `"meta"`: face metadata, dictionary.

Returns:

- HTTP 200 and empty body on success.
- HTTP 404 if a gallery with the given name doesn't exist.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

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

    }
  }
]
```

Response

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

9.3.3 Remove Face

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

Parameters in body:

JSON-encoded array of face ids to be removed

Returns:

- HTTP 200 and empty body on success.
- HTTP 404 if a face with the given id is not found in the gallery.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

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

Response

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

9.3.4 Face Search

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

Parameters in body:

JSON-encoded search request with the following fields:

- `limit`: maximum number of faces in the response.
- `sort`: sorting order. Pass one of the following values: `id`: increasing order by id, `-id`: decreasing order by id, `-score`: decreasing order by face similarity (only if you search for faces with similar feature vectors).
- `filter` (filters): * `facen`: (optional) search for faces with similar feature vectors. Pass a dictionary with the following fields: `data`: raw feature vector, base64; `score`: range of similarity between faces [threshold similarity; 1], where `1` is 100% match. * `id` and `meta/<meta_key>`: search by face id and metastring content. To set this filter, use the following operators:
  - `range`: range of values, only for numbers.
  - `set`: id or metastring must contain at least one value from a given set, for numbers and strings.

- `subset`: id or metastring must include all values from a given subset, for numbers and strings.
- `like`: by analogy with `like` in SQL requests: only ‘aa%’, ‘aa%’, and ‘%aa%’ are supported. Only for strings set[string]. In the case of set[string], the filter will return result if at least one value meets the filter condition.
- `ilike`: by analogy with `like` but case-insensitive, only for strings set[string].

Returns:

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

  Note: Fast index is not used in API v2.

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

Example

Request

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

Response

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

(continues on next page)

(continued from previous page)

```
Connection: keep-alive

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

9.3.5 Edit Face Metadata and Feature Vector

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

Parameters in body:

JSON-encoded array with faces with the following fields:

- `"id"`: face id, uint64_t.
- `"facen"`: (optional) new feature vector, base64. If omitted or passed as `null`, the relevant field in the database won't be updated.
- `"meta"`: dictionary with metadata to be updated. If some metastring is omitted or passed as `null`, the relevant field in the database won't be updated.

Returns:

- HTTP 200 and dictionary with all face parameters, including not updated, on success.
- HTTP 404 and error description if a face with the given id doesn't exist.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

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

Response

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

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

9.3.6 List Galleries

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

Returns:

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

Example

Request

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

Response

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

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

9.3.7 Get Gallery Info

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

Returns:

- HTTP 200 and dictionary with gallery parameters on success.
- HTTP 404 and error description if a gallery with the given name doesn't exist.
- HTTP with a status other than 200 and error description in the body on failure.

Example

Request

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

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

{"faces":2}
```

9.3.8 Create Gallery

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

Returns:

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

Example

Request

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

Response

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

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

9.3.9 Remove Gallery

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

Returns:

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

Example

Request

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

Response

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

9.4 Hacks for `findface-tarantool-server`

In this section:

9.4.1 Additional Configuration Parameters

To configure interaction between findface-sf-api and Tarantool, specify additional parameters in the 3rd argument of the `FindFace.start` section in the `findface-tarantool-server` 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 =␣
↪576, log_requests = false})
```

Additional parameters:

Parameter	Default value	Description
log_requests	true	Enable request logging (/var/log/tarantool/FindFace.log).
facen_size	576	Feature vector size, subject to the neural network model in use. Before editing this parameter, be sure to consult our experts by support@ntechlab.com.
search_threads	1	Number of threads for fast index search.
replication	nil	Only for a replica. Master instance IP address.
soft_delete_mode	false	Enable the soft deletion mode, when the faces are not removed from the fast index, but hidden in search results.

9.4.2 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})
```

9.4.3 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 the 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 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.

9.5 Real-time Face Liveness Detection

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

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

To enable the face liveness detector, do the following:

1. Open the `findface-video-worker` configuration file. In the `liveness` –> `fnk` parameter, specify the path to the face liveness detector model as shown below.

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

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


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

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

2. Restart `findface-video-worker`.

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

Once the face liveness detector enabled, the `findface-video-worker` service will be posting face liveness data to `findface-facerouter` in the `liveness` key of the `detectorParams` dictionary. To process a face due to its liveness, *write a plugin*.

9.6 Multiple Video Cards Usage

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

In this section:

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

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

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

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

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

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

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

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

4. Open the just created configuration file and actualize the video card number to use.

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

## cuda device number
device_number = 1
```

5. Open the new service and actualize the configuration file to use by specifying the just created one.

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

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

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

```
sudo systemctl daemon-reload
```

7. Enable the new service autostart.

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

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

8. Launch the new service.

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

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

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

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

...

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

9.7 Face Features Recognition

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

To enable automatic recognition of face features, open the `/etc/findface-extraction-api` configuration file and enable relevant recognition models: gender, age, emotions, glasses3, and/or beard. Be sure to choose the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models.

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

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

The following models are available:

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

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

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

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

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

Restart `findface-extraction-api`.

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

CHAPTER 10

Maintenance and Troubleshooting

10.1 Checking Component Status

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

Component	Command to view service status
`findface-extraction-api`	sudo systemctl status findface-extraction-api.service
`findface-sf-api`	sudo systemctl status findface-sf-api.service
`findface-facerouter`	sudo systemctl status findface-facerouter.service
`findface-tarantool-server`	sudo systemctl status tarantool@FindFace.service
`findface-video-manager`	sudo systemctl status findface-video-manager.service
`findface-video-worker-cpu`	sudo systemctl status findface-video-worker-cpu.service
`findface-video-worker-gpu`	sudo systemctl status findface-video-worker-gpu.service
`findface-ntls`	sudo systemctl status findface-ntls
`etcd`	sudo systemctl status etcd.service
`NginX`	sudo systemctl status nginx.service
`memcached`	sudo systemctl status memcached.service

10.2 Analyze Log Files

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

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

10.3 Troubleshoot Licensing and `findface-ntls`

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

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

10.3.1 Retrieve Licensing Information

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

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

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

Interpret the `last_updated` value as follows:

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

```
curl http://localhost:3185/license.json -s | jq
{
  "name": "NTLS",
  "time": 1520844897,
  "type": "offline (extended)",
  "license_id": "001278983",
  "generated": 487568400,
  "last_updated": 4,
  "valid": {
    "value": true,
    "description": ""
```

(continues on next page)

(continued from previous page)

```
  },
  "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
    }
  ],
  "services": [
    {
      "name": "video-worker",
      "ip": "127.0.0.1:58970"
    },
    {
      "name": "FindFace-tarantool",
      "ip": "127.0.0.1:58978"
    },
    {
```

(continues on next page)

(continued from previous page)

```
      "name": "findface-extraction-api",
      "ip": "127.0.0.1:52376"
    }
  ]
}
```

10.4 Automatic Tarantool Recovery

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

To enable automatic database recovery, do the following:

Note: You have to repeat the following instructions on each Tarantool shard.

1. Open a shard configuration file.

```
sudo vi /etc/tarantool/instances.enabled/<shard_001>.lua
```

2. Uncomment `force_recovery = true`.

```
box.cfg{

    force_recovery = true,
}
```

3. Restart the shard.

```
sudo systemctl restart tarantool@<shard_001>.service
```

CHAPTER 11

Appendices

11.1 Neural Network Models

Here you can see a summary for Ntech Lab neural networks which are currently in use in FindFace Enterprise Server:

Note: The CPU and GPU benchmark setup is the following:

- CPU: Intel® Core™ i7-5930K CPU @ 3.50GHz × 12
- GPU: GeForce GTX 1080

Type	Name	Facen size, bytes	CPU, FPS	GPU, FPS
Face biometrics	face/elderberry_160	160	5.78	204.98
	face/elderberry_576.r2	576	1.86	60.62
	face/grapefruit_160	160	6.01	191.43
	face/grapefruit_480	480	1.93	64.97
Gender recognition	faceattr/gender.v2	N/A	15.01	523.22
Age recognition	faceattr/age.v1	N/A	14.99	529.35
Emotions recognition	faceattr/emotions.v1	N/A	10.99	235.59
Country recognition	faceattr/countries47.v1	N/A	14.97	532.53

11.2 Components in Depth

11.2.1 `findface-extraction-api`

The `findface-extraction-api` service uses neural networks to detect a face in an image, extract face biometric data (feature vector), and recognize gender, age, emotions, and other features.

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

- Gets original images with faces and normalized face images.
- Returns the coordinates of the face bounding box, and (optionally) feature vector, face feature data, should these data be requested by `findface-sf-api`.

Tip: You can use *HTTP API* to directly access `findface-extraction-api`.

Functionality:

- face detection in an original image (with return of the bbox coordinates),
- face normalization,
- feature vector extraction from a normalized image,
- gender/age/emotions/country recognition.

The `findface-extraction-api` service can be based on CPU (installed from the `findface-extraction-api` package) or GPU (installed from the `findface-extraction-api-gpu` package). For both CPU- and GPU-accelerated services, configuration is done through the `/etc/findface-extraction-api.ini` configuration file. Its content varies subject to the acceleration type.

CPU-service configuration file:

```
allow_cors: false
detector_instances: 0
dlib:
  model: /usr/share/findface-data/normalizer.dat
  options:
    adjust_threshold: 0
    upsample_times: 1
extractors:
  instances: 1
  max_batch_size: 16
  models:
    age: ""
    beard: ""
    emotions: ""
    face: face/grapefruit_480.cpu.fnk
    gender: ""
    glasses3: ""
    liveness: ""
  models_root: /usr/share/findface-data/models
fetch:
  enabled: true
  size_limit: 10485760
license_ntls_server: 127.0.0.1:3133
listen: 127.0.0.1:18666
max_dimension: 6000
nnd:
  model: /usr/share/nnd/nnd.dat
  options:
    max_face_size: .inf
    min_face_size: 30
    o_net_thresh: 0.9
    p_net_max_results: 0
    p_net_thresh: 0.5
```

(continues on next page)

(continued from previous page)

```
    r_net_thresh: 0.5
    scale_factor: 0.79
  quality_estimator: true
  quality_estimator_model: /usr/share/nnd/quality_estimator_v2.dat
ticker_interval: 5000
```

GPU-service configuration file:

```
listen: 127.0.0.1:18666
dlib:
  model: /usr/share/findface-data/normalizer.dat
  options:
    adjust_threshold: 0
    upsample_times: 1
nnd:
  model: /usr/share/nnd/nnd.dat
  quality_estimator: true
  quality_estimator_model: /usr/share/nnd/quality_estimator_v2.dat
  options:
    min_face_size: 30
    max_face_size: .inf
    scale_factor: 0.7900000214576721
    p_net_thresh: 0.5
    r_net_thresh: 0.5
    o_net_thresh: 0.8999999761581421
    p_net_max_results: 0
detector_instances: 0
extractors:
  models_root: /usr/share/findface-data/models
  max_batch_size: 3
  instances: 2
  models:
    age: ""
    beard: ""
    emotions: ""
    face: face/grapefruit_480.gpu.fnk
    gender: ""
    glasses3: ""
    liveness: ""
  cache_dir:
  gpu_device: 0
license_ntls_server: 172.17.46.26:3133
fetch:
  enabled: true
  size_limit: 10485760
max_dimension: 6000
allow_cors: false
ticker_interval: 5000
```

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

Parameter	Description
`nnd -> quality_estimator`	Enables face quality estimation. In this case, `findface-extraction-api` returns a face quality score in the `detection_score` field. Interpret the quality score further in analytics. Upright faces in frontal position are considered the best quality. They result in values around 0, mostly negative (such as -0.00067401276, for example). Inverted faces and large face angles are estimated with negative values some -5 and less.
`nnd -> min_face_size`	The minimum size of a face (bbox) guaranteed to be detected. The larger the value, the less resources required for face detection.
`nnd -> max_face_size`	The minimum size of a face (bbox) guaranteed to be detected.
`license_ntls_server`	The license server IP address and port.
`gpu_device`	(Only for GPU) The number of the GPU device used by `findface-extraction-api-gpu`.
`models -> instances`	The number of neural network instances (and, consequently, the number of simultaneously processed requests) that are loaded into RAM by `findface-extraction-api`. Specify the number of instances from your license. The default value (0) means that this number is equal to the number of CPU cores. This parameter severely affects RAM consumption.
`fetch -> enabled`	Enables fetching images from the Internet.
`fetch -> size_limit`	The maximum size of an Internet image to be fetched.

You will also have to enable recognition models for face features such as gender, age, emotions, glasses3, and/or beard, subject to your needs. Be sure to choose the right acceleration type for each model, matching the acceleration type of `findface-extraction-api`: CPU or GPU. Be aware that `findface-extraction-api` on CPU can work only with CPU-models, while `findface-extraction-api` on GPU supports both CPU- and GPU-models.

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

The following models are available:

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

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

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

11.2.2 `findface-sf-api`

The `findface-sf-api` service implements HTTP API for the FindFace core main functionality such as face detection and face recognition (the mentioned functions themselves are provided by *findface-extraction-api*‘). It interfaces with the biometric database powered by Tarantool via the `findface-tarantool-server` service, as well as with `findface-extraction-api` (provides face detection and face recognition) and `findface-upload` (provides a storage for original images and FindFace core artifacts).

To detect a face in an image, you need to send the image as a file or URL in an *API request* to `findface-sf-api`. The `findface-sf-api` will then redirect the request to `findface-extraction-api` for face detection and recognition.

Tip: You can also *directly* access `findface-extraction-api`.

If there is a configured video face detection module in the system, `findface-sf-api` also interfaces with the `findface-facerouter` service. It receives data of detected in video faces along with processing directives from `findface-facerouter`, and then executes the received directives, for example, saves faces into a specific database gallery.

Functionality:

- *HTTP API* implementation (face detection and face recognition methods, performed via `findface-extraction-api`).
- saving face data to the biometric database (performed via `findface-tarantool-server`),

- saving original images, face thumbnails and normalized face images to an NginX-powered web server (via `findface-upload`).
- provides interaction between all the FindFace core components.

```
cache:
  inmemory:
    size: 16384
   memcache:
    nodes:
    - 127.0.0.1:11211
    timeout: 100ms
  redis:
    addr: localhost:6379
    db: 0
    network: tcp
    password: ''
    timeout: 5s
  type: memcache
extraction-api:
  extraction-api: http://127.0.0.1:18666
  timeouts:
    connect: 5s
    idle_connection: 10s
    overall: 35s
    response_header: 30s
limits:
  allow-return-facen: false
  body-image-length: 33554432
  deny-networks: 127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8
  url-length: 4096
listen: 127.0.0.1:18411
normalized-storage:
  enabled: true
  s3:
    access-key: ''
    bucket-name: ''
    endpoint: ''
    operation-timeout: 30
    public-url: ''
    region: ''
    secret-access-key: ''
    secure: true
  type: webdav
  webdav:
    timeouts:
      connect: 5s
      idle_connection: 10s
      overall: 35s
      response_header: 30s
    upload-url: http://127.0.0.1:3333/uploads/
storage-api:
  max-idle-conns-per-host: 20
  shards:
  - master: http://127.0.0.1:8101/v2/
    slave: ''
   - master: http://127.0.0.1:8102/v2/
    slave: ''
```

(continues on next page)

(continued from previous page)

```
  timeouts:
    connect: 5s
    idle_connection: 10s
    overall: 35s
    response_header: 30s
```

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

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

11.2.3 `findface-tarantool-server`

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

Tip: See Tarantool official documentation for details.

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

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

Functionality:

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

The `findface-tarantool-server` configuration is done through the `/etc/tarantool/instances.enabled/<*>.lua` configuration file. In a cluster environment, configuration has to be done for each shard.

```
--
-- Please, read the tarantool cfg doc:
-- https://tarantool.org/doc/reference/configuration/index.html#box-cfg-params
--

box.cfg{
    --port to listen, direct tarantool access
```

(continues on next page)

(continued from previous page)

```
    --Only need for admin operations
    --THIS IS NOT PORT YOU NEED FOR facenapi/sf-api
    listen = '127.0.0.1:33001',

    --Directory to store data
    vinyl_dir = '/opt/ntech/var/lib/tarantool/shard-001',
    work_dir = '/opt/ntech/var/lib/tarantool/shard-001',
    memtx_dir = '/opt/ntech/var/lib/tarantool/shard-001/snapshots',
    wal_dir = '/opt/ntech/var/lib/tarantool/shard-001/xlogs',

    --Maximum mem usage in bytes
    memtx_memory = 200 * 1024 * 1024,

    checkpoint_interval = 3600*4,
    checkpoint_count = 3,

    --uncomment only if you know what you are doing!!! and don't forget box.snapshot()
    -- wal_mode = 'none',

    --if true, tarantool tries to continue if there is an error while reading a␣
↪snapshot/xlog files:  skips invalid records, reads as much data as possible and re-
↪builds the file
    -- force_recovery = true,
}

pcall(function() box.schema.user.grant('guest', 'execute,read,write', 'universe') end)

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

-- host,port to bind for http server
-- this is what you need for facenapi
FindFace = require("FindFace")
FindFace.start("127.0.0.1", 8101, {
    license_ntls_server="127.0.0.1:3133",
    facen_size=480,
    meta_scheme = meta_scheme
})
```

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

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

The default database structure is passed from `/etc/ffsecurity/tnt_schema.lua` to the `meta_scheme` parameter if FindFace Enterprise Server is installed from the installer. If it is installed from the apt repository, you will have to manually set it via the configuration file.

11.2.4 `findface-upload`

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

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

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

11.2.5 `findface-facerouter`

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

The `findface-facerouter` service accepts a face bbox and normalized image along with the original image and other data (for example, the detection date and time) from the `findface-video-worker` service. In general, `findface-facerouter` allows you to apply arbitrary face processing directives, including directly sending faces to a partner application. In the basic configuration, `findface-facerouter` is pre-configured to redirect faces to `findface-sf-api` for further processing, but you will still have to set processing directives by creating a plugin.

Functionality:

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

The `findface-facerouter` configuration is done through a configuration file `/etc/findface-facerouter.py`.

~~~
```# main.py options:

# debug                           = False
## debug - debug mode
# host                            = ''
## host - host to listen
# port                            = 18820
## port - port to listen
# sfapi_url                       = 'http://localhost:18411'
## sfapi_url - SF-API URL
# version                         = False
## version - print version

# plugin_dir.py options:

# plugin_dir                      = ''
## plugin_dir - Plugin directory for plugin_source='dir'

# abstract_define.py options:

# plugin_source                   = 'dir'
~~~

(continues on next page)

(continued from previous page)

```
## plugin_source - Plugin source (dir)

# log.py options:

# log_file_max_size              = 100000000
## log_file_max_size - max size of log files before rollover
# log_file_num_backups           = 10
## log_file_num_backups - number of log files to keep
# log_file_prefix                = None
## log_file_prefix - Path prefix for log files. Note that if you are running
## multiple tornado processes, log_file_prefix must be different for each of
## them (e.g. include the port number)
# log_rotate_interval            = 1
## log_rotate_interval - The interval value of timed rotating
# log_rotate_mode                = 'size'
## log_rotate_mode - The mode of rotating files(time or size)
# log_rotate_when                = 'midnight'
## log_rotate_when - specify the type of TimedRotatingFileHandler interval other
## options:('S', 'M', 'H', 'D', 'W0'-'W6')
# log_to_stderr                  = None
## log_to_stderr - Send log output to stderr (colorized if possible). By default
## use stderr if --log_file_prefix is not set and no other logging is
## configured.
# logging                        = 'info'
## logging - Set the Python log level. If 'none', tornado won't touch the
## logging configuration.
```

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

Parameter	Description
`sfapi_url`	IP address and port of the `findface-sf-api` host.
`plugin_dir`	List of directories with plugins to define face processing directives.

11.2.6 Video face detection: `findface-video-manager` and `findface-video-worker`

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

In this section:

Functions of `findface-video-manager`

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

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

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

Note: Configuration settings passed via jobs have priority over the `findface-video-manager` configuration file.

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

Functionality:

- allows for configuring video face detection parameters,
- allows for managing the list of to-be-processed video streams,
- implements video face detection management.

Functions of `findface-video-worker`

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

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

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

Functionality:

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

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

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

- **`Face tracking`**. The face tracker tracks, detects and captures faces in video. It can simultaneously be working with several faces. It also searches for the best face snapshot, using an embedded neural network. After the best face snapshot is found, it is posted to `findface-facerouter`.

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

- Real-time
- Offline

Real-Time Mode

In the real-time mode, `findface-video-worker` posts a face immediately after it appears in the camera field of view.

- If `rt-perm=True`, the face tracker searches for the best face snapshot within each time period equal to `rt-delay` and posts it to `findface-facerouter`.
- If `rt-perm=False`, the face tracker searches for the best face snapshot dynamically:
  1. First, the face tracker estimates whether the quality of a face snapshot exceeds a pre-defined threshold value. If so, the snapshot is posted to `findface-facerouter`.
  2. The threshold value increases after each post. Each time the face tracker gets a higher quality snapshot of the same face, it is posted.
  3. When the face disappears from the camera field of view, the threshold value resets to default.

By default, the real-time mode is disabled (`realtime=false` in the `/etc/findface-video-manager.conf` file).

Offline Mode

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

By default, the offline mode is enabled (`overall=true` in the `/etc/findface-video-manager.conf` file).

Configure Video Face Detection

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

1. The `findface-video-manager` configuration file `/etc/findface-video-manager.conf`:

```
listen: 127.0.0.1:18810
etcd:
  endpoints: 127.0.0.1:2379
  dial_timeout: 3s
kafka:
   enabled: false
   endpoints: 127.0.0.1:9092
master:
  lease_ttl: 10
  self_url: 127.0.0.1:18811
  self_url_http: 127.0.0.1:18811
rpc:
```

(continues on next page)

(continued from previous page)

```
  listen: 127.0.0.1:18811
  heart_beat_timeout: 4s
router_url: http://127.0.0.1:18820/v0/frame
exp_backoff:
  enabled: false
  min_delay: 1s
  max_delay: 1m0s
  factor: 2
  flush_interval: 2m0s
ntls:
  enabled: false
  url: http://127.0.0.1:3185/
  update_interval: 1m0s
prometheus:
  jobs_processed_duration_buckets:
  - 1
  - 30
  - 60
  - 500
  - 1800
  - 3600
  - 21600
  - .inf
job_scheduler_script: ''
stream_settings:
  ffmpeg_params: []
  md_threshold: 0.002
  md_scale: 0.3
  fd_frame_height: -1
  uc_max_time_diff: 30
  uc_max_dup: 3
  uc_max_avg_shift: 10
  det_period: 8
  realtime: false
  npersons: 4
  disable_drops: false
  tracker_threads: 4
  parse_sei: false
  image_arg: photo
  additional_headers: []
  additional_body: []
  api_timeout: 15000
  api_ssl_verify: true
  post_uniq: true
  min_score: -2
  min_d_score: -1000
  realtime_dly: 500
  realtime_post_perm: false
  rot: ''
  roi: ''
  draw_track: false
  send_track: 0
  min_face_size: 0
  max_face_size: 0
  overall: true
  only_norm: false
  max_candidates: 0
```

(continues on next page)

(continued from previous page)

```
  jpeg_quality: 95
  ffmpeg_format: ''
stream_settings_gpu:
  play_speed: -1
  filter_min_quality: -2
  filter_min_face_size: 1
  filter_max_face_size: 8192
  normalized_only: false
  jpeg_quality: 95
  overall_only: true
  use_stream_timestamp: false
  ffmpeg_params: []
  router_timeout_ms: 15000
  router_verify_ssl: true
  router_headers: []
  router_body: []
  start_stream_timestamp: 0
  imotion_threshold: 0
  rot: ''
  roi: ''
  realtime_post_interval: 1
  realtime_post_every_interval: false
  ffmpeg_format: ''
  disable_drops: false
```

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

Option	Description
`router_url`	IP address and port of the `findface-facerouter` host to receive detected faces from `findface-video-worker`. Default value: `http://127.0.0.1:18820/v0/frame`.
`etcd` -> `endpoints`	IP address and port of the `etcd` service. Default value: `127.0.0.1:2379`.
`ntls` -> `enabled`	If true, `findface-video-manager` will send a job to `findface-video-worker` only if the total number of processed cameras does not exceed the allowed number of cameras from the license. Default value: false.
`ntls` -> `url`	IP address and port of the `findface-ntls` host. Default value: `http://127.0.0.1:3185/`.

You can also configure the following parameters:

Note: In the `stream_settings(-gpu)` section of the file, you will find settings common to all video streams. Settings of a particular stream, passed in a job, have priority over those in the configuration file (see *Jobs*).

CPU-option	GPU-option	Description
`additional_body`	`router_body`	Additional body fields in a request body when posting a face: ["key = value"]. Default value: body fields not specified.

Continued on next page

Table 1 – continued from previous page

CPU-option	GPU-option	Description
additional_headers	router_headers	Additional header fields in a request when posting a face: ["key = value"]. Default value: headers not specified.
api_ssl_verify	router_verify_ssl	Enables a https certificate verification when findface-video-worker and findface-facerouter interact over https. Default value: true. If false, a self-signed certificate can be accepted.
api_timeout	router_timeout_ms	Timeout for a findface-facerouter response to a findface-video-worker API request, in milliseconds. If the timeout has expired, the system will log an error. Default value: 15000.
disable_drops	disable_drops	Enables posting all appropriate faces without drops. By default, if findface-video-worker does not have enough resources to process all frames with faces, it drops some of them. If this option is active, findface-video-worker puts odd frames on the waiting list to process them later. Default value: false.
draw_track	N/a	Enables drawing a face motion track in a bbox. Default value: false.
fd_frame_height	N/a	Video frame height for the face tracker, in pixels. Scale down in the case of high resolution cameras, or close up faces, or if the CPU load is too high, to reduce the system resources consumption. Make sure that the scaled face size exceeds the min-face-size value. Default value: -1 (negative values corresponds to the original size). Optimal value to reduce load: 640-720.
ffmpeg_format	ffmpeg_format	Pass FFMPEG format (mxg, flv, etc.) if it cannot be detected automatically.
ffmpeg_params	ffmpeg_params	List of a video stream ffmpeg options with their values as a key=value array: ["rtsp_transpotr=tcp", .., "ss=00:20:00"]. Check out the FFmpeg web site for the full list of options. Default value: options not specified.
image_arg	N/a	Name of the argument containing a bbox with a face, in an API request. Default value: photo.
jpeg_quality	jpeg_quality	Quality of an original frame JPEG compression, in percents. Default value: 95%.
max_face_size	filter_max_face_size	Maximum size of a face in pixels. Oversized faces are not posted. Default value: 0 (filter disabled).
md_scale	N/a	Video frame scaling coefficient for the motion detector, relative to the original size from 0 to 1. Scale down in the case of high resolution cameras, or close up faces, or if the CPU load is too high, to reduce the system resources consumption. Make sure that the scaled face size exceeds the min-face-size value. Default value: 1 (original size).
md_threshold	motion_threshold	Minimum motion intensity to be detected by the motion detector. The threshold value is to be fitted empirically. Empirical units: zero and positive rational numbers. Milestones: 0 = detector disabled, 0.002 = default value, 0.05 = minimum intensity is too high to detect motion.
min_score	filter_min_quality	Minimum threshold value for a face image quality. A face is posted if it has better quality. The threshold value is to be fitted empirically. Empirical units: negative rational numbers to zero. Milestones: 0 = high quality faces, -1 = good quality, -2 = satisfactory quality, -5 = face recognition maybe inefficient. Default value: -2.
min_face_size	filter_min_face_size	Minimum size of a face in pixels. Undersized faces are not posted. Default value: 0 (filter disabled).

Continued on next page

Table 1 – continued from previous page

CPU-option	GPU-option	Description
`min_d_score`	N/a	Maximum deviation of a face from its frontal position. A face is posted if its deviation is less than this value. The deviation is to be fitted empirically. Empirical units: negative rational numbers to zero. Milestones: -3.5 = large face angles, face recognition may be inefficient, -2.5 = satisfactory deviation, -0.05 = close to the frontal position, 0 = frontal face. Default value: -1000.
`npersons`	N/a	Maximum number of faces simultaneously tracked by the face tracker. This parameter severely affects performance. Default value: 4.
`only_norm`	`normalized_only`	Enable posting only normalized face images without full frames. Default value: false.
`overall`	`overall_only`	Enables the offline mode for the best face search. Default value: true.
N/a	`play_speed`	If less than zero, the speed is not limited. In other cases, the stream is read with the given `play_speed`. Not applicable for live streams.
`post_uniq`	N/a	Enables face deduplication, i.e. posting only a certain number of faces belonging to one person, during a certain period of time. In this case, if `findface-video-worker` posts a face to `findface-facerouter` and then tracks another one within the time period `uc_max_time_diff`, and the distance between the two faces doesn't exceed `uc_max_avg_shift`, `findface-video-worker` estimates their similarity. If the faces are similar and the total number of similar faces during the `uc_max_time_diff` period does not exceed the number `uc_max_dup`, `findface-video-worker` posts the other face. Otherwise, the other face is not posted. Default value: true.
`realtime`	N/a	Enables the real-time mode for the best face search. Default value: false.
`realtime_dly`	`realtime_post_interval`	Only for the real-time mode. If `realtime_post_perm=True`, defines the time period in milliseconds within which the face tracker picks up the best snapshot and posts it to `findface-facerouter`. If `realtime_post_perm=False`, defines the minimum time period between 2 posts of the same face with increased quality. Default value: 500.
`realtime_post_perm`	`realtime_post_every_interval`	Only for the realtime mode. Post best snapshots obtained within each `realtime_dly` time period. If false, search for the best snapshot dynamically and send snapshots in order of increasing quality. Default value: false.
`roi`	`roi`	Enable posting faces detected only inside a region of interest WxH+X+Y. Default value: region not specified.
`rot`	`rot`	Enables detecting and tracking faces only inside a clipping rectangle WxH+X+Y. You can use this option to reduce `findface-video-worker` load. Default value: rectangle not specified.
`send_track`	N/a	Enables posting a face motion track as array of the bbox center coordinates. As the `send_track` value, specify the number of dots in the motion track. Default value: 0 (array not posted).
N/a	`start_stream_timestamp`	Add the specified number of seconds to timestamps from a stream.
`tracker_threads`	N/a	Number of tracking threads for the face tracker. This value should be less or equal to the `npersons` value. We recommend you to set them equal. If the number of tracking threads is less than the maximum number of tracked faces, resource consumption is reduced but so is the tracking speed. Default value: 1.

Continued on next page

Table 1 – continued from previous page

CPU-option	GPU-option	Description
uc_max_time_diff	N/a	Only if `post_uniq: true` (face deduplication enabled). Maximum time period in seconds during which a number of similar faces are considered as belonging to one person. Default value: 30.
uc_max_dup	N/a	Only if `post_uniq: true` (face deduplication enabled). Maximum number of faces during the `uc_max_time_diff` period that is posted for a person. Default value: 3.
uc_max_avg_shift	N/a	Only if `post_uniq: true` (face deduplication enabled). Distance in pixels within which a number of similar faces are considered as belonging to one person. Default value: 10.
N/a	use_stream_timestamp	If true, retrieve and post timestamps from a video stream. If false, post the actual date and time.

1. If you opt for the CPU-accelerated package `findface-video-worker-cpu`, use the `/etc/findface-video-worker-cpu.ini` configuration file:

```
## read streams from file, do not use VideoManager
input =

## exit on first finished job, only when --input specified
exit_on_first_finished = false

## batch size
batch_size = 4

## http server port for metrics, 0=do not start server
metrics_port = 0

## resize scale, 1=do not resize
resize_scale = 1.000000

## maximum number of streams
capacity = 10

## command to obtain videomanager's grpc ip:port
mgr_cmd =

## videomanager grpc ip:port
mgr_static = 127.0.0.1:18811

## ntls server ip:port
ntls_addr = 127.0.0.1:3133

## debug: save faces to dir
save_dir =

## minimum face size
min_face_size = 60

## preinit detector for specified resolutions: "640x480;1920x1080"
resolutions =

## worker labels: "k=v;group=enter"
labels =
```

(continues on next page)

(continued from previous page)

```
## use timestamps from SEI packet
use_time_from_sei = false

#------------------------------
[streamer]
#------------------------------
## streamer server port, 0=disabled
port = 18999

## streamer url - how to access this worker on streamer_port
url = ws://127.0.0.1:18999/stream/

#------------------------------
[liveness]
#------------------------------
## path to liveness fnk
fnk =

## liveness threshold
threshold = 0.945000

## liveness internal algo param
interval = 1.000000

## liveness internal algo param
stdev_cnt = 1

#------------------------------
[send]
#------------------------------
## posting faces threads
threads = 8

## posting faces maximum queue size
queue_limit = 256

#------------------------------
[tracker]
#------------------------------
## max face miss duration, sec
miss_interval = 1.000000

## overlap threshold
overlap_threshold = 0.250000

#------------------------------
[models]
#------------------------------
## path to detector fnk
detector = /usr/share/findface-data/models/facedet/mtcnn.cpu.fnk

## path to quality fnk
quality = /usr/share/findface-data/models/faceattr/quality.v0.cpu.fnk

## path to norm for quality fnk
norm_quality = /usr/share/findface-data/models/facenorm/ant.v2.cpu.fnk
```

(continues on next page)

(continued from previous page)

```
## path to norm200 fnk, for face send
norm_200 = /usr/share/findface-data/models/facenorm/ant.v2.cpu.fnk

## path to norm_crop2x fnk, for face send
norm_crop2x = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.cpu.
↪fnk
```

If you opt for the GPU-accelerated package `findface-video-worker-gpu`, use the `/etc/findface-video-worker-gpu.ini` configuration file.

```
## cuda device number
device_number = 0

## old gpu detector models directory
models_dir = /usr/share/findface-gpudetector/models

## read streams from file, do not use VideoManager
input =

## exit on first finished job, only when --input specified
exit_on_first_finished = false

## batch size
batch_size = 8

## http server port for metrics, 0=do not start server
metrics_port =

## resize scale, 1=do not resize
resize_scale = 1.000000

## maximum number of streams
capacity = 30

## command to obtain videomanager's grpc ip:port
mgr_cmd =

## videomanager grpc ip:port
mgr_static = 127.0.0.1:18811

## ntls server ip:port
ntls_addr = 127.0.0.1:3133

## debug: save faces to dir
save_dir =

## minimum face size
min_face_size = 60

## preinit detector for specified resolutions: "640x480;1920x1080"
resolutions =

## worker labels: "k=v;group=enter"
labels =

## use timestamps from SEI packet
```

(continues on next page)

(continued from previous page)

```
use_time_from_sei = false

#------------------------------
[streamer]
#------------------------------
## streamer server port, 0=disabled
port = 18999

## streamer url - how to access this worker on streamer_port
url = ws://172.17.46.17:18999/stream/

#------------------------------
[liveness]
#------------------------------
## path to liveness fnk
fnk =

## liveness threshold
threshold = 0.945000

## liveness internal algo param
interval = 1.000000

## liveness internal algo param
stdev_cnt = 1

#------------------------------
[send]
#------------------------------
## posting faces threads
threads = 8

## posting faces maximum queue size
queue_limit = 256

#------------------------------
[tracker]
#------------------------------
## max face miss duration, sec
miss_interval = 1.000000

## overlap threshold
overlap_threshold = 0.250000

#------------------------------
[models]
#------------------------------
## path to detector fnk
detector =

## path to quality fnk
quality =

## path to norm for quality fnk
norm_quality =

## path to norm200 fnk, for face send
```

(continues on next page)

(continued from previous page)

```
norm_200 = /usr/share/findface-data/models/facenorm/ant.v2.gpu.fnk

## path to norm_crop2x fnk, for face send
norm_crop2x = /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.gpu.
↪fnk

## path to cache directory
cache_dir =

#------------------------------
[video_decoder]
#------------------------------
## decode video on cpu
cpu = false
```

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

CPU	GPU	Description
`ntls-addr`		IP address and port of the `findface-ntls` host.
`mgr-static`		IP address of the `findface-video-manager` host to provide `findface-video-worker` with settings and the list of to-be-processed streams.
`capacity`		Maximum number of video streams to be processed by `findface-video-worker`.
`mgr-exec`		(Optional, instead of the `mgr-static` parameter) A script to describe dynamic IP address of the `findface-video-manager` host.
`labels`		Labels that determine processing directives to be applied by `findface-facerouter` to the found faces.
N/a	`fnk`	Path to the face *liveness* detector.
`input`		Process streams from file, ignoring stream data from `findface-video-manager`.
`exit_on_first_finished`		(Only if `input` is specified) Exit on the first finished job.
`resize_scale`		Rescale video frames with the given coefficient.
`save_dir`		(For debug) Save detected faces to the given directory.
`min_face_size`		Minimum face size to be detected.
`resolutions`		Preinitialize the `findface-video-worker` for specific resolutions to speed up its performance.
N/a	`device_number`	GPU device number to use.
N/a	`models_dir`	Old directory with GPU detector models. Otherwise, use the `[models]` section.
N/a	`cpu`	If necessary, decode video on CPU.

Jobs

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

The content of a typical job is shown in the example below.

```
curl http://127.0.0.1:18810/job/1 | jq
 % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 100  1771  100  1771    0     0   447k      0 --:--:-- --:--:-- --:--:--  576k
```

(continues on next page)

(continued from previous page)

```
{
  "id": "1",
  "enabled": true,
  "stream_url": "rtmp://restreamer.int.ntl/cams/openspace",
  "labels": {},
  "router_url": "http://172.17.46.13/video-detector/frame",
  "single_pass": false,
  "stream_settings": {
    "ffmpeg_params": [],
    "md_threshold": 0.002,
    "md_scale": 0.3,
    "fd_frame_height": -1,
    "uc_max_time_diff": 30,
    "uc_max_dup": 3,
    "uc_max_avg_shift": 10,
    "det_period": 8,
    "realtime": false,
    "npersons": 4,
    "disable_drops": false,
    "tracker_threads": 4,
    "parse_sei": false,
    "image_arg": "photo",
    "additional_headers": [
      "Authorization=Token b612396adc3a6dd71b82b5fe333a0a30"
    ],
    "additional_body": [],
    "api_timeout": 15000,
    "api_ssl_verify": true,
    "post_uniq": true,
    "min_score": -2,
    "min_d_score": -1000,
    "realtime_dly": 500,
    "realtime_post_perm": false,
    "rot": "",
    "roi": "",
    "draw_track": false,
    "send_track": 0,
    "min_face_size": 0,
    "max_face_size": 0,
    "overall": true,
    "only_norm": false,
    "max_candidates": 0,
    "jpeg_quality": 95,
    "ffmpeg_format": ""
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": -2,
    "filter_min_face_size": 1,
    "filter_max_face_size": 8192,
    "normalized_only": false,
    "jpeg_quality": 95,
    "overall_only": false,
    "use_stream_timestamp": false,
    "ffmpeg_params": [],
    "router_timeout_ms": 15000,
    "router_verify_ssl": true,
```

(continues on next page)

(continued from previous page)

```
    "router_headers": [
      "Authorization=Token b612396adc3a6dd71b82b5fe333a0a30"
    ],
    "router_body": [],
    "start_stream_timestamp": 0,
    "imotion_threshold": 0,
    "rot": "",
    "roi": "",
    "realtime_post_interval": 1,
    "realtime_post_every_interval": false,
    "ffmpeg_format": "",
    "disable_drops": true
  },
  "status": "INPROGRESS",
  "status_msg": "",
  "statistic": {
    "processed_duration": 14879,
    "faces_posted": 777,
    "faces_failed": 3,
    "faces_not_posted": 1206,
    "processing_fps": 18.816668,
    "frames_dropped": 0,
    "frames_processed": 0,
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "job_starts": 56
  },
  "restream_url": "",
  "worker_id": "ffsec40_213ab8c0ed5d954e",
  "version": "bl068taaa7tcafrfsmq0"
}
```

Each job has the following parameters:

- `id`: job id.
- `enabled`: active status.
- `stream_url`: URL/address of video stream/file to process.
- `labels`: tag(s) that will be used by the `findface-facerouter` component to find processing directives for faces detected in this stream.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).
- `router_url`: IP address and port of the `findface-facerouter` component to receive detected faces from the `findface-video-worker` component for processing.
- `stream_settings`, `stream_settings_gpu`: video stream settings that duplicate *those* in the `findface-video-manager` configuration file (while having priority over them).
- `status`: job status.
- `status_msg`: additional job status info.
- `statistic`: job progress statistics (progress duration, number of posted and not posted faces, processing fps, the number of processed and dropped frames, job start time, etc.).
- `worker_id`: id of the `findface-video-worker` instance executing the job.

11.2.7 findface-ntls

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

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

Use the main web interface to manage `findface-ntls`:

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

The following components are licensable:

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

Important: After connection between `findface-ntls` and a licensable component, or between `findface-ntls` and the global license server is broken, you will have 6 hours to restore it before the licensable components will be automatically stopped.

The `findface-ntls` configuration is done through a configuration file `/etc/findface-ntls.cfg`.

```
# Listen address of NTLS server where services will connect to.
# The format is IP:PORT
# Use 0.0.0.0:PORT to listen on all interfaces
# This parameter is mandatory and may occur multiple times
# if you need to listen on several specific interfaces or ports.
listen = 127.0.0.1:3133


# Directory with license files.
# NTLS use most recently generated one.
# Note: "recentness" of a license file is detected not by
#       mtime/ctime but from its internal structure.
#
# This parameter is mandatory and must occur exactly once.
license-dir = /opt/ntech/license

# You can specify proxy which NTLS will use to access
# global license server. The syntax is the same that is used by curl.
# Proxy is optional
#proxy = http://192.168.1.1:12345

# This is bind address for NTLS web-interface.
# Note: there're no authorization or access restriction mechanisms
#       in NTLS UI. If you need one, consider using nginx as proxy
#       with .htaccess / ip-based ACLs.
```

(continues on next page)

(continued from previous page)

```
# This parameter may be specified multiple times.
ui = 127.0.0.1:3185
```

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

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

11.3 Installation File

FindFace Enterprise Server installation configuration is automatically saved to a file `/tmp/<findface-installer-*>.json`. You can edit this file and use it to install FindFace Enterprise Server on other hosts without having to answer the installation questions again.

Tip: See *Install from Console Installer* to learn more about the FindFace Enterprise Server installer.

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

```
{
  "ignore_lowmem": true,
  "findface-security.config": {
    "EXTERNAL_ADDRESS": "http://172.20.77.78"
  },
  "inter_ip.bind": "127.0.0.1",
  "memcached.config": {
    "listen_host": "127.0.0.1",
    "max_memory": 1024,
    "item_size": 16
  },
  "findface-video-worker.config": {
    "FKVD_WRK_CAP": "10",
    "FKVD_NTLS_ADDR": "127.0.0.1:3133",
    "streamer": [
      "port = 18999",
      "url = ws://127.0.0.1:18999/stream/"
    ],
    "FKVD_MGR_ADDR": "127.0.0.1:18811"
  },
  "ext_ip.bind": "0.0.0.0",
  "findface-data.models": [
    "./findface-data-age.v1-cpu_3.0.0_amd64.deb",
    "./findface-data-age.v1-gpu_3.0.0_amd64.deb",
    "./findface-data-beard.v0-cpu_3.0.0_amd64.deb",
```

(continues on next page)

(continued from previous page)

```
    "./findface-data-beard.v0-gpu_3.0.0_amd64.deb",
    "./findface-data-grapefruit-160-cpu_3.0.0_amd64.deb",
    "./findface-data-grapefruit-160-gpu_3.0.0_amd64.deb",
    "./findface-data-grapefruit-480-cpu_3.0.0_amd64.deb",
    "./findface-data-grapefruit-480-gpu_3.0.0_amd64.deb",
    "./findface-data-emotions.v1-cpu_3.0.0_amd64.deb",
    "./findface-data-emotions.v1-gpu_3.0.0_amd64.deb",
    "./findface-data-gender.v2-cpu_3.0.0_amd64.deb",
    "./findface-data-gender.v2-gpu_3.0.0_amd64.deb",
    "./findface-data-glasses3.v0-cpu_3.0.0_amd64.deb",
    "./findface-data-glasses3.v0-gpu_3.0.0_amd64.deb",
    "./findface-data-liveness.v1-gpu_3.0.0_amd64.deb"
  ],
  "findface-video-worker.variant": "cpu",
  "inter_ip.advertised": "127.0.0.1",
  "product": "security",
  "findface-ntls.config": {
    "NTLS_LISTEN": "127.0.0.1:3133",
    "NTLS_LICENSE_DIR": "/opt/ntech/license",
    "NTLS_LISTEN_UI": "127.0.0.1:3185"
  },
  "ext_ip.advertised": "172.20.77.78",
  "tnt_instances": 2,
  "findface-facerouter.config": {
    "port": "18820",
    "host": "127.0.0.1",
    "plugin_source": "dir",
    "plugin_dir": "/etc/findface-facerouter-plugins",
    "sfapi_url": "http://127.0.0.1:18411"
  },
  "findface-sf-api.config": {
    "storage-api": {
      "shards": [
        {
          "slave": "",
          "master": "http://127.0.0.1:8101/v2/"
        },
        {
          "slave": "",
          "master": "http://127.0.0.1:8102/v2/"
        }
      ]
    },
    "listen": "127.0.0.1:18411",
    "extraction-api": {
      "extraction-api": "http://127.0.0.1:18666"
    }
  },
  "type": "stand-alone",
  "findface-extraction-api.variant": "cpu",
  "findface-video-manager.config": {
    "rpc": {
      "listen": "127.0.0.1:18811"
    },
    "listen": "127.0.0.1:18810",
    "master": {
      "self_url": "127.0.0.1:18811",
```

(continues on next page)

(continued from previous page)

```
      "self_url_http": "127.0.0.1:18811"
    },
    "ntls": {
      "url": "http://127.0.0.1:3185/",
      "enabled": false
    }
  },
  "components": [
    "findface-data",
    "memcached",
    "etcd",
    "redis",
    "postgresql",
    "findface-ntls",
    "findface-extraction-api",
    "findface-sf-api",
    "findface-upload",
    "findface-video-manager",
    "findface-video-worker",
    "findface-security",
    "findface-tarantool-server"
  ],
  "findface-tarantool-server.config": {
    "shard-002": {
      "TNT_LISTEN": "127.0.0.1:33002",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-002",
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_FF_LISTEN_PORT": "8102",
      "TNT_FF_LISTEN_IP": "127.0.0.1",
      "TNT_EXTRA_LUA": "\ndofile(\"/etc/ffsecurity/tnt_schema.lua\")\n",
      "TNT_FF_NTLS": "127.0.0.1:3133"
    },
    "shard-001": {
      "TNT_LISTEN": "127.0.0.1:33001",
      "TNT_DATA_DIR": "/opt/ntech/var/lib/tarantool/shard-001",
      "TNT_META_SCHEME": "meta_scheme",
      "TNT_FF_LISTEN_PORT": "8101",
      "TNT_FF_LISTEN_IP": "127.0.0.1",
      "TNT_EXTRA_LUA": "\ndofile(\"/etc/ffsecurity/tnt_schema.lua\")\n",
      "TNT_FF_NTLS": "127.0.0.1:3133"
    }
  },
  "findface-extraction-api.config": {
    "extractors": {
      "instances": 1,
      "models": {
        "emotions": "",
        "age": "",
        "gender": "",
        "face": "face/grapefruit_480.cpu.fnk"
      }
    },
    "nnd": {
      "quality_estimator": true
    },
    "listen": "127.0.0.1:18666",
    "license_nils_server": "127.0.0.1:3133"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

To automatically install FindFace Enterprise Server on another host without answering the installation questions, use the following command:

```
sudo ./findface-security-2.1.0-server-3.1.0.run -f /tmp/<findface-installer-*>.json
```

Python Module Index

Index