FindFace

Release 11.240325

NtechLab

Dec 10, 2025

CONTENTS:

FindFace Server is a set of services for video analytics. FindFace Server is not a turnkey solution. It does not contain business logic and a user interface. The software is designed solely for civilian use and is not intended for military purposes.

Being integrated into specific solutions, FindFace 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 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.

FindFace Server can detect, identify, and analyze different objects in the video. Here are just a few of them:

- Human faces, along with recognition of such facial attributes as gender, age, emotions, glasses, face mask, beard, and many others.
- Human bodies (silhouettes), along with recognition of clothing type and color.
- Vehicles, with recognition of such vehicle attributes as make, model, body style, color, license plate number, and others.
- Snow pile, streetlight, graffiti, pet faces, shopping cart, visible gas, fire, bear, container, garbage, trash bin, and others.

Furthermore, it is possible to integrate 2D anti-spoofing system, which 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.

Core Features

- AI-based platform.
- Fast and robust AI-based object detection in still images and video.
- Fast and accurate AI-based object identification and verification.
- Scalable and fault-tolerant feature vector and metadata storage supporting sub-linear search time.
- AI recognition of gender, age, emotions, glasses, beard, face mask, and other face and body attributes.
- User-friendly APIs hiding complexities of distributed architecture.
- AI face liveness detector.
- Customized object processing directives.

Deployment environment

- Single- and multi-host deployment. Almost infinite scalability.
- Network or on-premise licensing.
- CPU- and GPU-based acceleration for your choice.
- Integration via HTTP API.

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

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

CHAPTER

ONE

SYSTEM ADMINISTRATOR'S GUIDE

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

1.1 Get Started

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

1. *Please, read the architecture and capabilities of FindFace Server*.
2. *Prepare hardware*.
3. *Deploy FindFace Server Services*.
4. *Detect objects in an image with the recognition of its attributes*.
5. *Configure video object detection*.
6. Consider *using advanced features*.

1.2 Architecture

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

In this chapter:

1.2.1 Recognition Objects and Recognition Process

FindFace Server can recognize a lot of objects and their attributes:

- human faces
- human bodies (silhouettes)
- vehicles
- snow pile
- streetlight
- graffiti
- pet faces
- shopping cart
- visible gas
- fire
- bear
- container
- garbage
- trash bin
- and other

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

1.2.2 Docker Platform

FindFace Server is delivered as a set of packages (container images), that are expected to be deployed using an OS-level virtualization technology commonly referred as "containers". Each FindFace Server service runs in a separate container. This manual uses Docker in all of its examples, but any OCI-compatible runtime may be used.

1.2.3 Architectural Elements

FindFace Server is a set of services for video analytics. You can implement the business logic and UI through 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.

FindFace Server implements a *Video Recorder*, an additional functionality that records, stores, and plays back video data from cameras.

Architecture scheme

FindFace Server Components

FindFace Server includes the following components:

Component	Ports in use	Description	Vendor
extraction-api	18666	Service that uses neural networks to detect an object in an image and extract its feature vector. It also recognizes object attributes (for example, gender, age, emotions, beard, glasses, face mask - for face objects). CPU- or GPU-acceleration.	NtechLab own deployment
sf-api	18411	Service that implements a unified user-friendly HTTP API for the `extraction-api` and `tntapi`.	
tntapi	8001 (API), 32001 (replication)	Feature vector and metadata storage built on top of Tarantool in-memory database.	
upload	3333	`nginx` -based web server used as a storage for normalized object images. If *Video Recorder* is enabled, `upload` can be used to store video data from cameras.	
facerouter	18820	Service used to define processing directives for detected objects.	
video-manager	18810 (API), 18811 (worker)	Service, part of the video object detection module, that is used for managing the video object detection functionality, configuring the video object detector settings and specifying the list of to-be-processed video streams.	
video-worker	18999	Service, part of the video object detection module, that recognizes an object in the video and posts its normalized image, full frame and metadata (such as detection time) to the router service (`facerouter`) for further processing according to given directives. If *Video Recorder* is enabled, `video-worker` sends video over to `video-storage`. Provides *face liveness detection* if enabled. CPU- or GPU-acceleration.	
ntls	3185 (API & UI), 3133 (licensing)	License server that interfaces with the NtechLab Global License Server, a USB dongle, or hardware fingerprint to verify the *license* of your FindFace Server instance.	
counter	18300	Feature vector deduplication, counting and aggregation service.	
liveness-api	18301	Besides the embedded functionality provided by `video-worker`, face liveness detection can also be harnessed as a standalone service `liveness-api`. The service takes a specific number of frames from a video chunk and returns the best quality face, and decimal liveness result. The service can be used in the course of user authentication by face.	
deduplicator	18310	Feature vector deduplication service.	
etcd	2379	Third-party software that implements a distributed key-value store for `video-manager`. Used as a coordination service in the distributed system, providing the video object detector with fault tolerance.	etcd
redis	6379	Third-party in-memory database that can be used by `sf-api` as a temporary storage for extracted object feature vectors before they are written to the feature vector storage.	redis
memcached	11211	Third-party software that implements a distributed memory caching system. Used by `sf-api` as a temporary storage for extracted object feature vectors before they are written to the feature vector database powered by Tarantool.	memcached

Video Recorder

A Video Recorder includes the following services:

Component	Ports in use	Description	Vendor
video-storage	18611	Service that implements video chunk management. It takes video chunks from the `video-worker` service, puts them into the storage (`upload`), and writes their meta-information to the MongoDB database. By request, it issues info about existing video chunks and Websocket-links to the corresponding streams. These links are further used by `video-streamer-cpu` to deliver the video to a user for viewing and downloading.	NtechLab own deployment
video-streamer	9000	After receiving an API request, this service extracts the requested video chunks from `video-storage` and `video-worker` (only the last chunk if it's not yet recorded to the storage). Then it merges the video chunks into a one-piece video and delivers it to a user for viewing and downloading using WebSocket.	
MongoDB	27017	Third-party software that implements the Video Recorder database. The database stores meta-information of the video chunks, including their location. The video chunks themselves are stored in the `upload` service.	MongoDB

See also:

Components in Depth

1.2.4 Single- and Multi-Host Deployment

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

- Deploy FindFace Server on a principal host and distribute additional `video-worker` instances across remote hosts.
- Distribute the FindFace Server services across multiple hosts. If necessary, set up load balancing. Contact our experts by support@ntechlab.com for guidance.

1.2.5 CPU- and GPU-acceleration

The `extraction-api` and `video-worker` services can be either CPU- or GPU-based.

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

If the cameras you are using have the resolution of `1920x1080@25`, and you need to process less than 5 such cameras, then it is enough to use a `video-worker-cpu`. But if you need to process more cameras, it is strongly recommended to use a `video-worker-gpu`.

When selecting CPU/GPU-acceleration of `extraction-api`, you should consider the number of objects in the camera frame and the number of features extracted from each object. When there is a high density of objects (about 50 objects in the frame of each camera), one instance of the `extraction-api-cpu` can extract features of objects from about 17 cameras, if you need more features or cameras, use the `extraction-api-gpu`.

Note: The number of objects and cameras depends on your implementation.

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

1.3 Requirements

In this chapter:

1.3.1 System Requirements for Basic Configuration

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

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

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

The system requirements depend on many factors, such as:

1. The number of video streams, their resolution and FPS.
2. The number, size and types of objects in the camera frame.
3. How often objects enter and exit the camera frame.
4. The number of selected detectors and attributes of the object.
5. Requests per second (RPS).
6. Settings and other additional factors that depend on the specific use of the server.

Requirements for `video-worker-cpu`

Important: The minimum technical requirements for processing a single video stream with a resolution of 1920x1080@25FPS, in the case when there are no more than 3 objects in the camera frame.

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 4+ GHz with Advanced Vector Extensions 2 (AVX2)
	Enable Turbo Boost to reach 4+ GHz
RAM	512+ Mb
HDD/SSD	256+ Mb
Operating system	Ubuntu 18+ 64-bit PC

Tip: If you need to process more video streams, use more physical cores, while requiring more RAM. For more accurate hardware selection, contact our support team by support@ntechlab.com.

Requirements for `video-worker-gpu`

Important: NVIDIA GPU accelerators with video decoding capability are supported, check out the NVIDIA document.

Important: The minimum technical requirements for processing a single video stream with a resolution of 1920x1080@25FPS, in the case when there are no more than 3 objects in the camera frame.

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 4+ GHz
	Enable Turbo Boost to reach 4+ GHz
RAM	2+ Gb
GPU	NVIDIA GeForce GTX 1080 Ti
	• Supported driver: NVIDIA R535 LTSB • Supported devices: NVIDIA, Pascal, Volta, Turing, Ampere architectures
GPU memory	2+ Gb
HDD/SSD	4.5+ Gb
Operating system	Ubuntu 18+ 64-bit PC

Tip: The NVIDIA GeForce GTX 1080 Ti graphics card is capable of processing approximately 23 video streams with a resolution of 1920x1080@25FPS when there are only 3 objects in the camera frame. For more accurate hardware selection, contact our support team by support@ntechlab.com.

Requirements for `extraction-api-cpu`

Important: The minimum technical requirements for processing are 4.4 RPS. For example, processing a video stream of 1920x1080@25FPS, in the case when there are no more than 3 objects in the camera frame, the `video-worker` generates approximately 0.2 RPS.

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 4+ GHz with Advanced Vector Extensions 2 (AVX2)
	Enable Turbo Boost to reach 4+ GHz
RAM	2.5+ Gb
HDD/SSD	256+ Mb
Operating system	Ubuntu 18+ 64-bit PC

Tip: If you need to process more RPS, use more physical cores, while requiring more RAM. For more accurate hardware selection, contact our support team by support@ntechlab.com.

Requirements for `extraction-api-gpu`

Important: The minimum technical requirements for processing 200 RPS. For example, processing a video stream of 1920x1080@25FPS, in the case when there are no more than 3 objects in the camera frame, the `video-worker` generates approximately 0.2 RPS.

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 4+ GHz
	Enable Turbo Boost to reach 4+ GHz
RAM	1.5+ Gb
GPU	NVIDIA GeForce GTX 1080 Ti
	• Supported driver: NVIDIA R535 LTSB • Supported devices: NVIDIA, Pascal, Volta, Turing, Ampere architectures
GPU memory	2+ Gb
HDD/SSD	4.5+ Gb
Operating system	Ubuntu 18+ 64-bit PC

Note: The NVIDIA GeForce GTX 1080 Ti graphics card is capable of processing approximately 500 RPS, but it will require more physical CPU cores and more RAM. For more accurate hardware selection, contact our support team by support@ntechlab.com.

Requirements for `sf-api`

sf-api

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	256+ Mb
Operating system	Ubuntu 18+ 64-bit PC

memcached

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	1+ Gb
HDD/SSD	128+ Mb
Operating system	Ubuntu 18+ 64-bit PC

redis

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	1+ Gb
HDD/SSD	256+ Mb
Operating system	Ubuntu 18+ 64-bit PC

upload

Important: Minimum technical requirements for storing 100,000 normalized object images.

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	1+ Gb
HDD/SSD	4+ Gb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `tntapi`

Important: Minimum technical requirements for storing 100,000 feature vectors of objects.

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	256+ Mb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `ntls`

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	128+ Mb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `video-manager`

video-manager

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	128+ Mb
Operating system	Ubuntu 18+ 64-bit PC

etcd

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	2+ Gb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `video-storage`

video-storage

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	128+ Mb
Operating system	Ubuntu 18+ 64-bit PC

mongo

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	1+ Gb
Operating system	Ubuntu 18+ 64-bit PC

upload

Important: Minimum technical requirements for storing video chunks of 1920x1080@25FPS per day.

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	1+ Gb
HDD/SSD	20+ Gb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `video-streamer`

video-streamer

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	128+ Mb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `deduplicator`

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	128+ Mb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `counter`

counter

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	128+ Mb
HDD/SSD	128+ Mb
Operating system	Ubuntu 18+ 64-bit PC

postgres

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	256+ Mb
HDD/SSD	256+ Mb
Operating system	Ubuntu 18+ 64-bit PC

Requirements for `liveness-api`

	Minimum
CPU	Intel Core i3 CPU with 1+ physical cores 2+ GHz
RAM	512+ Mb
HDD/SSD	256+ Mb
Operating system	Ubuntu 18+ 64-bit PC

1.3.2 Required Administrator Skills

Knowledge of the OS, on which the product instance is deployed, as well as of the Docker platform at the level of an advanced user is required for a FindFace Server administrator.

1.3.3 Video File Formats

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

Both CPU- and GPU-accelerated instances support most of the popular video file formats (MP4, MKV/WEBM, AVI, FLV, WMV, OGG, etc.). Video codec support differs between CPU and GPU versions:

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

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

1. Open the `video-worker.yaml` configuration file.

   ```
   sudo vi /opt/ffserver/configs/video-worker.yaml
   ```

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

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

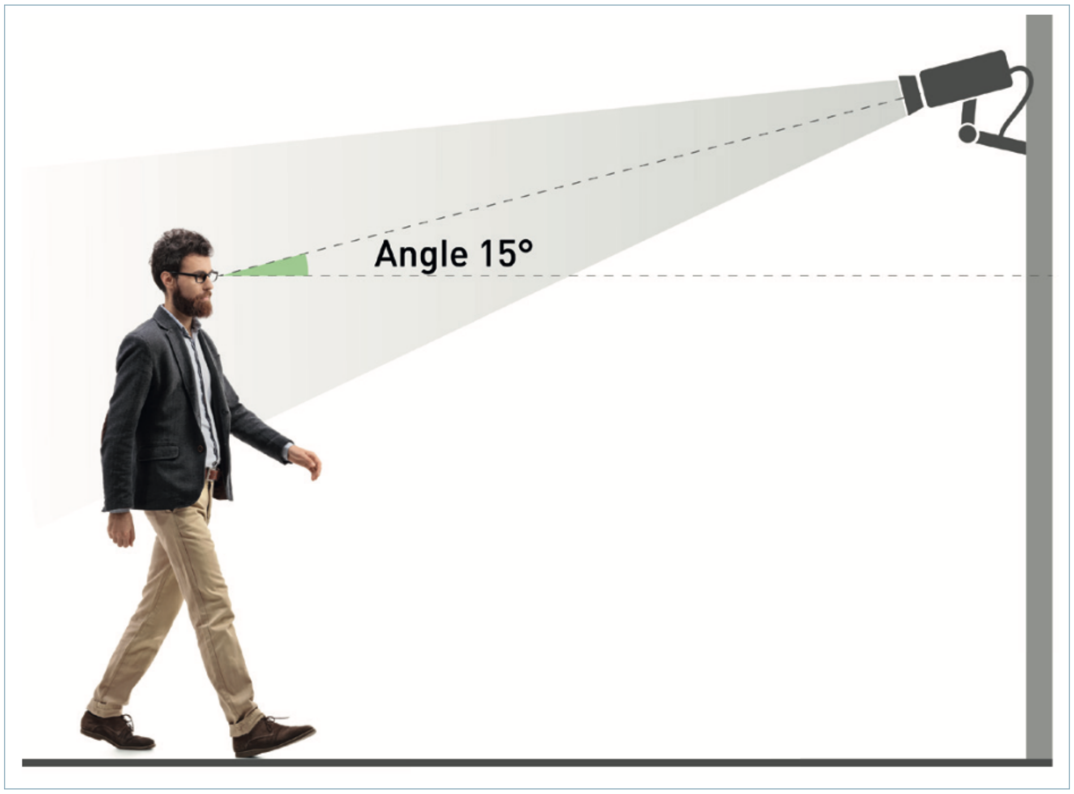
3. Rebuild `video-worker` container.

1.3.4 Requirements for CCTV Cameras

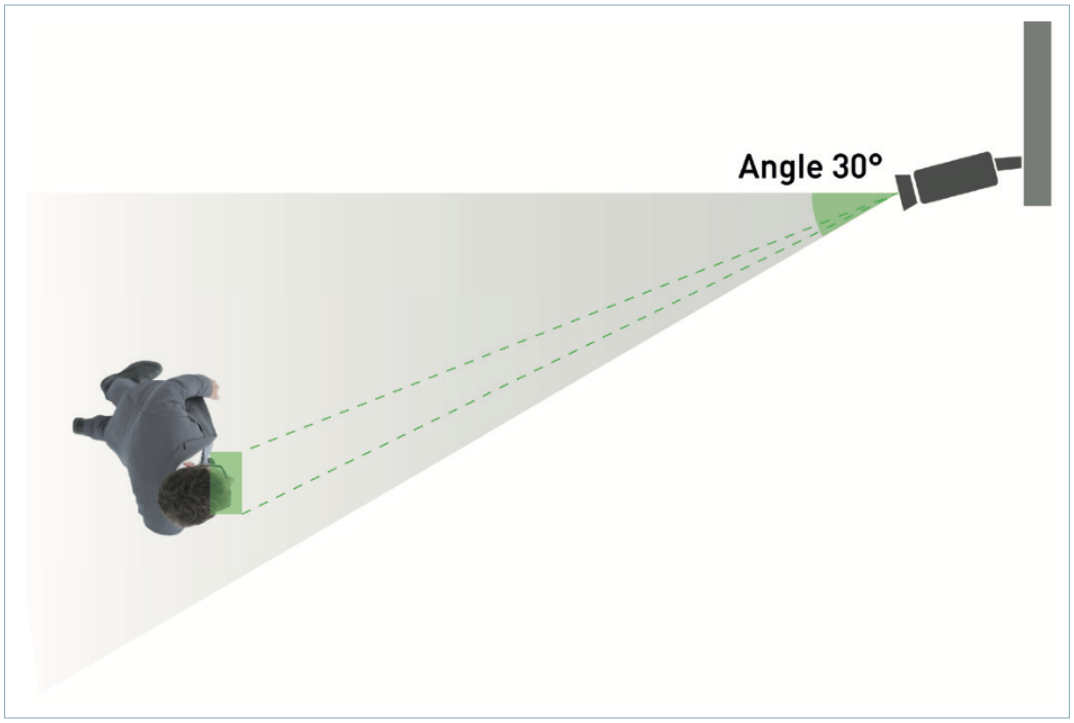
Face Recognition

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

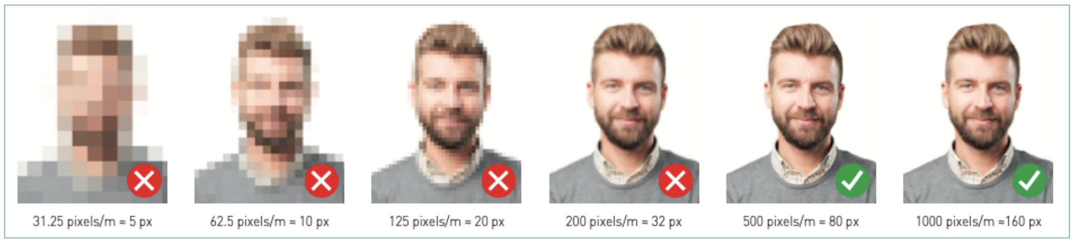
1. For correct face detection in a video stream, mount the camera so that the face of each individual entering the monitored area surely appears in the camera field of view.
2. The vertical tilt angle of the camera should not exceed 15°. The vertical tilt is a deviation of the camera's optical axis from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



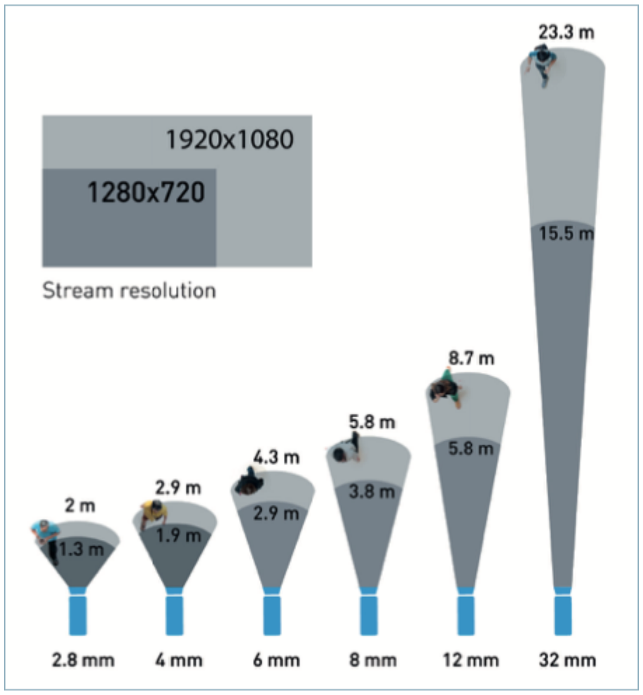
3. The horizontal deflection angle should not exceed 30°. The horizontal deflection is a deviation of the camera's optical axis from the motion vector of the main flow of objects subject to recognition.



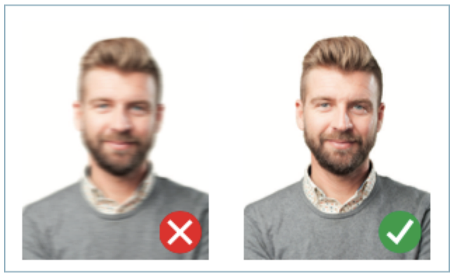
4. The minimum pixel density required for identification is 500 pixels/m (roughly corresponds to a face width of 80 pixels).



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



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



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

8. Video compression: refer to *Video File Formats*.
9. Video stream delivery protocols: RTSP, HTTP.

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

Body and Vehicle Recognition

Please contact our technical support team (support@ntechlab.com) to get requirements for installation and characteristics of CCTV cameras for body and vehicle recognition.

1.4 Deploy FindFace Server

Docker platform

FindFace Server is distributed using Docker images and is intended to be run using Docker-compatible execution environment (i.e. Docker, Podman, Docker Swarm or Kubernetes). You must install and start a set of Docker products before proceeding with the FindFace Server deployment. For your convenience, this chapter contains the *Ubuntu Server Preparation* section covering the intricacies of installing Docker on Ubuntu. For other platforms, please refer to the Docker documentation. On modern RedHat systems, you can substitute Docker with Podman, but you will need to refer to official NVIDIA docs if you intend to use GPU-acceleration with Podman.

NVIDIA driver and NVIDIA Container Runtime (GPU only)

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

Deployment

After you are finished with the server preparation, you are all set for the FindFace Server deployment. Each FindFace Server component is deployed step-by-step using the docker platform. It requires fundamental understanding of the product architecture. Specify environment variables. If needed, make changes to the configuration file of the required component and restart the container.

1. Deploying FindFace Server standalone. See *Deploy Components* for guidance.
2. Deploying FindFace Server in a multi-host environment. It requires a bit of technical expertise and knowledge of the product architecture.

Note: Keep in mind to install necessary neural network models along with the `video-worker-cpu` / `video-worker-gpu` and `extraction-api-cpu` / `extraction-api-gpu` components and enable them in corresponding configuration file.

Post-deployment procedures and how-to's

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

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

Important: FindFace Server does not include any means of access control and completely relies on surrounding network and cluster infrastructure to provide access control and authentication. You should avoid uncontrollably exposing FindFace Server components ports to untrusted networks without reverse proxies implementing some form of authorization or strict firewall policies.

Instance removal

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

1.4.1 Ubuntu Server Preparation

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

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

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

In this section:

GPU: Install NVIDIA Drivers

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

With the GPU-accelerated FindFace Server, you'll need the NVIDIA driver version `535` or above. Add the NVIDIA repository and install an applicable driver from it.

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

To install the `535` driver from a repository, do the following:

1. Install the repository signature key:

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

2. Install `aptitude`:

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

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

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

There are two ways to resolve them:

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

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

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

```
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock
```

(continues on next page)

(continued from previous page)

```
sudo rm /var/lib/dpkg/lock-frontend
sudo dpkg --configure -a
```

3. Install `nvidia-driver-535`:

```
sudo aptitude install nvidia-driver-535
```

Note: If you're using Ubuntu 18.04, please install the `530` nvidia driver.

4. Reboot the system:

```
sudo reboot
```

Install Docker Products

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

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

```
sudo apt-get update

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

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

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

3. Set up the Docker repository.

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

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

```
sudo apt-get update
```

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

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

5. Install the latest versions of Docker products.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
```

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

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

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

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

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

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

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

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

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

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

8. Configure the Docker network.

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

GPU: Install NVIDIA Container Runtime

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

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

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

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

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

3. Restart Docker.

```
systemctl restart docker
```

Now you are all set to install FindFace Server.

1.4.2 Deploy Components

This section will guide you through the FindFace Server installation process.

FindFace Server is configurable through a configuration file, various command-line flags, and environment variables.

A reusable configuration file is a yaml/env/ini file made with name and value of one or more command-line flags described below. In order to use this file, specify the file path as a value to the `-c` / `--config` flag or `CFG_CONFIG` environment variable. The sample configuration file can be used as a starting point to create a new configuration file as needed.

Options set on the command line take precedence over those from the environment.

The format of environment variable for flag `--my-flag` is `CFG_MY_FLAG`. It applies to all flags.

The instructions below are given only for a reference and a base configurations. See *Components in Depth* to build a configuration suitable for a specific project. For the latest available flags, use `--help` flag:

```
docker run --rm -ti --name <container_name> docker.int.ntl/ntech/universe/
↪<service_name>:ffserver-11.240325 --help
```

In this section:

Create a Docker Network

To create a docker network, use the following command:

```
docker network create --attachable server
```

Network name must be unique. The `--attachable` option used to enable manual container attachment. `server` is a name of your network.

Tip: Useful commands:

1. List networks.

   ```
   docker network ls
   ```

2. Display detailed information on your network.

   ```
   docker network inspect server
   ```

When a container is created and connected to a created network, Docker automatically creates a DNS record for that container, using the container name as the hostname and the IP address of the container as the record's value. This enables other containers on the same network to access each other by name, rather than needing to know the IP address of the target container.

Running `etcd` under Docker

The `etcd` is a third-party software that implements a distributed key-value store for `video-manager`. It is used as a coordination service in the distributed system, providing the video object detector with fault tolerance.

Run `etcd` under Docker:

```
docker run -tid --name etcd-1 --network server --restart always \
    quay.io/coreos/etcd:v3.5.11 /usr/local/bin/etcd \
    -advertise-client-urls http://0.0.0.0:2379 \
    -listen-client-urls http://0.0.0.0:2379
```

This `docker run` command will expose the `etcd` client API over port 2379. This will run `v3.5.11` version of `etcd`. You can specify a different version after consulting with our specialists.

Use docker options:

- `--name`: specify a custom identifier for a container, for example `etcd-1`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.

Use configuration flags to `etcd`:

- `-advertise-client-urls`: list of this member's client URLs to advertise to the public. Default `http://localhost:2379`.
- `-listen-client-urls`: list of URLs to listen on for client traffic. This flag tells the `etcd` to accept incoming requests from the clients on the specified scheme://IP:port combinations. Scheme can be either http or https. If 0.0.0.0 is specified as the IP, `etcd` listens to the given port on all interfaces. If an IP address is given as well as a port, `etcd` will listen on the given port and interface. Multiple URLs may be used to specify a number of

addresses and ports to listen on. The `etcd` will respond to requests from any of the listed addresses and ports. default: `http://localhost:2379`.

See etcd for more information.

Running `memcached` under Docker

The `memcached` is a third-party software that implements a distributed memory caching system. Used by `sf-api` as a temporary storage for extracted object feature vectors before they are written to the feature vector database powered by `Tarantool`.

Run `memcached` under Docker:

```
docker run -tid --name memcached --restart always --network server \
    docker.io/library/memcached:1.5.22 -v -m '1024' -I 16m -u memcache
```

This `docker run` command will expose the `memcached` client API over port 11211. This will run `v1.5.22` version of `memcached`. You can specify a different version after consulting with our specialists.

Use docker options:

- `--name`: specify a custom identifier for a container, for example `memcached`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.

Use configuration flags to `memcached`:

- `-v`: be verbose during the event loop; print out errors and warnings.
- `-m`: use 1 GB of memory to store features.
- `-I`: override the default size of each slab page.
- `-u`: assume the identity of `memcache`.

Running `redis` under Docker

The `redis` is a third-party software that implements a distributed memory caching system. Used by `sf-api` as a temporary storage for extracted object feature vectors before they are written to the feature vector database powered by `Tarantool`.

Run `redis` under Docker:

```
docker run -tid --name redis --restart always --network server \
    --volume /opt/ffserver/redis-data:/data \
    docker.io/redis:7 --appendonly no --save "" --maxmemory 1073741824 --
↪maxmemory-policy allkeys-lru
```

This `docker run` command will expose the `redis` client API over port 6379. This will run `7` version of `redis`. You can specify a different version after consulting with our specialists.

Use docker options:

- `--name`: specify a custom identifier for a container, for example `redis`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.

Use configuration flags to `redis`:

- `--appendonly`: asynchronously dumps the dataset on disk.
- `--save`: save a snapshot of the database to disk.
- `--maxmemory`: limiting memory usage to the specified number of bytes.
- `--maxmemory-policy`: how `redis` will select what to remove when `maxmemory` is reached.

Running `postgres` under Docker

The `postgres` is a third-party object-relational database system. Used by `counter` as storage for extracted object feature vectors and other information.

Run `postgres` under Docker:

```
docker run -tid --name db-postgres --restart always --network server \
    --env POSTGRES_USER=login \
    --env POSTGRES_PASSWORD=password \
    --env POSTGRES_DB=ntech \
    --volume /opt/ffserver/postgres-data:/var/lib/postgresql \
    docker.io/postgres:14.12
```

This `docker run` command will expose the `postgres` client API over port 5432. This will run `14.12` version of `postgres`. You can specify a different version after consulting with our specialists.

Use docker options:

- `--name`: specify a custom identifier for a container, for example `db-postgres`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.

Use environment variables to `postgres`:

- `POSTGRES_USER` and `POSTGRES_PASSWORD`: these are the credentials used to authenticate with the database.
- `POSTGRES_DB`: the name of the database.

Provide Licensing

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

The FindFace Server licensing is provided as follows:

1. Follow the steps required for your type of licensing. These steps are described here: *Licensing*
2. Deploy `ntls`, license server in the FindFace Server.

```
docker run -tid --name ntls --restart always --network server \
    --env CFG_LISTEN=0.0.0.0:3133 \
    --env CFG_UI=0.0.0.0:3185 \
    --volume /opt/ffserver/licenses:/ntech/license \
    --publish 127.0.0.1:3185:3185 \
    docker.int.ntl/ntech/universe/ntls:ffserver-11.240325
```

   Use docker options:

- `--name`: specify a custom identifier for a container, for example `ntls`.
- `--network`: connect a container to a network, named `server`. You should use the `host` namespace instead of `server` if your license type requires it. But be aware that in this case other services won't be able to connect to ntls using container name of `ntls`. As a workaround, you can specify the IP address of the `ntls` host instead of the `ntls` container name in the configuration files of licensable components.
- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.
- `--publish 127.0.0.1:3185:3185`: this allows you to connect to the `ntls` API from the host with `http://localhost:3185` (if you need to access not only from the local machine, remove `127.0.0.1:`). This option is needless if `--net=host` is specified.

Use environment variables:

- `CFG_LISTEN`: address to accept incoming client connections (IP:PORT) (`-l` flag).
- `CFG_UI`: bind address for embedded UI (IP:PORT) (`--ui` flag).

The directory you chose to store the licenses must be mounted to `/ntech/license`. In the example, `/opt/ffserver/licenses` is used as such a directory.

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

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

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

     Tip: Later on, use the `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 configuration file).
4. For the on-premise licensing, insert the USB dongle into a USB port.
5. If the licensable components are installed on remote hosts, specify the IP address of the `ntls` host in their configuration files. See *extraction-api*, *tntapi*, *Video Object Detection: video-manager and video-worker* for details.

See also:

Troubleshoot Licensing and ntls

Deploy `extraction-api`

The `extraction-api` is a service that uses neural networks to detect an object in an image and extract its feature vector. It also recognizes object attributes (for example, gender, age, emotions, beard, glasses, face mask - for face objects).

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

Important: This component requires the installation of neural network models. Load the desired models manually and mount a volume to the container.

Note: To deploy the `extraction-api` service with acceleration on the GPU, use the `extraction-api-gpu` image. Don't forget to use the `--runtime=nvidia` flag in the `docker run` command. You can specify the value of the GPU device identifier on which the output will be started using the `-gpu-device` or `CFG_GPU_DEVICE` variable flag.

1. Create a default `extraction-api` configuration file.

```
docker run --rm -ti docker.int.ntl/ntech/universe/extraction-api-cpu:ffserver-11.
↪240325 \
    --config-template > /opt/ffserver/configs/extraction-api.yaml
```

   - `/opt/ffserver/configs`: the directory on the host to store the configuration file.

2. Open the `extraction-api.yaml` configuration file.

```
sudo vi /opt/ffserver/configs/extraction-api.yaml
```

3. Enable recognition models, subject to your needs. Be sure to choose the right acceleration type for each model, matching the acceleration type of `extraction-api`: CPU or GPU. Be aware that `extraction-api` on CPU can work only with CPU-models, while `extraction-api` on GPU supports both CPU- and GPU-models.

```
detectors:
  models:
    jasmine:
      aliases:
      - face
      model: detector/facedet.jasmine_fast.004.cpu.fnk
      options:
        min_object_size: 32
        resolutions: [2048x2048]

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


normalizers:
  crop2x:
    model: facenorm/crop2x.v2_maxsize400.cpu.fnk

  norm200:
```

(continues on next page)

(continued from previous page)

```
    model: facenorm/bee.v3.cpu.fnk


extractors:
  models_root: /usr/share/findface-data/models
  models:
    face_emben:
      default:
        model: face/nectarine_m_160.cpu.fnk

    face_age:
      default:
        model: faceattr/faceattr.age.v3.cpu.fnk

    face_quality:
      default:
        model: faceattr/faceattr.quality.v5.cpu.fnk
```

4. Configure other parameters, if needed. For example, enable or disable image fetching from a remote server for some kind of request.

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

5. (Optional) Enable emben cutting.

```
extractors:
  models:
    face_emben:
      default:
        model: face/nectarine_xl_320.gpu.fnk
        params:
          emben_cut:
            - 64
```

   - Currently, only 1 value in the `ember_cut` list is supported.
   - The value must be divisible by 16 without remainder.
   - Make sure the model supports the emben cutting. Be sure to consult with our technical experts prior (support@ntechlab.com).

6. Specify upper limit on detection/extraction/normalization batch size, if needed.

```
detectors:
  max_batch_size: 1
extractors:
  max_batch_size: 1
normalizers:
  max_batch_size: 1
```

Note: The `max_batch_size` value determines the maximum number of images that are processed in parallel on the GPU or CPU. For GPU, it is recommended `max_batch_size: 8 or 16`. For CPU, you can specify max_batch_size: -1, this means max_batch_size is equal to the number of CPU cores.

Warning: The *-instances parameter is DEPRECATED. Its fields are outdated. It indicated how many extraction-api instances were used. The number of instances were specified from your license. The value (0) doesn't means that this number is equal to the number of CPU cores!

7. When you have edited the configuration file, run extraction-api component with the docker run command.

```
docker run -tid --name extraction-api --restart always --network server \
    --env CFG_LICENSE_NTLS_SERVER=ntls:3133 \
    --volume /opt/ffserver/models:/usr/share/findface-data/models \
    --volume /opt/ffserver/configs/extraction-api.yaml:/extraction-api.yaml \
    --publish 127.0.0.1:18666:18666 \
    docker.int.ntl/ntech/universe/extraction-api-cpu:ffserver-11.240325 \
    --config /extraction-api.yaml
```

Use docker options:

- --name: specify a custom identifier for a container, for example extraction-api.
- --network: connect a container to a network, named server.
- --restart: restart policy to apply when a container exits. Set always to always restart the container if it stops.

Use environment variables and configuration flags:

- CFG_LICENSE_NTLS_SERVER=ntls:3133: host and port of the ntls container. See *Licensing*.
- /opt/ffserver/configs: the directory on the host to store the configuration file.
- /opt/ffserver/models: the directory on the host to store the models.
- --publish 127.0.0.1:18666:18666: this allows you to connect to the extraction-api API from the host with http://localhost:18666 (if you need to access not only from the local machine, remove 127.0.0.1:).

Deploy tntapi

The tntapi component provides interaction of the sf-api component with the Tarantool database for efficient storage and search by feature vectors of objects. To increase search speed, multiple tntapi shards can be created on each Tarantool host. Their running concurrently leads to a remarkable increase in performance. Each shard can handle up to approximately 10,000,000 faces. In the case of 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 tntapi component, do the following:

1. Create a directory for snapshots and xlogs.

```
sudo mkdir -p /opt/ffserver/tnt/001-01/{snapshots,xlogs}
```

2. Run docker command and configure parameters:

```
docker run -tid --name tnt-1-1 --restart always --network server \
    --env CFG_LISTEN_HOST=0.0.0.0 \
    --env CFG_NTLS=ntls:3133 \
    --env TT_LISTEN=0.0.0.0:32001 \
    --env TT_MEMTX_MEMORY=$((1024 * 1024 * 1024)) \
    --volume /opt/ffserver/tnt/001-01:/opt/ntech/var/lib/tarantool/default \
    docker.int.ntl/ntech/universe/tntapi:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `tnt-1-1`. Throughout this page we use `tnt-<shard>-<replica>` naming convention for `tntapi` containers.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.

By default, the configuration is via environment variables (ENV), but it is also possible to use the configuration file `FindFace.lua`.

- `CFG_LISTEN_HOST=0.0.0.0`: host to public HTTP API.
- `CFG_NTLS=ntls:3133`: host and port of the `ntls` server.
- `TT_LISTEN=0.0.0.0:32001`: binary host/port, used for admin operations and replication.
- `TT_MEMTX_MEMORY=$((1024 * 1024 * 1024))`: the maximum memory usage in bytes.
- `/opt/ffserver/tnt/`: the directory on the host to store `tntapi` data.

If needed, configure via `FindFace.lua` file:

1. Download the `FindFace.lua` file and put it into some directory on the host (for example, `/opt/ffserver/configs`). Open the configuration file:

```
sudo vi /opt/ffserver/configs/FindFace.lua
```

2. 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,
```

3. When using the default `FindFace.lua` configuration file, you can set a set of `meta_scheme/meta_indexes` in the global variable `cfg_spaces`.

   Create a database structure to store the face recognition results. The structure is created as a set spaces of fields. Describe each field with the following parameters:

   - `id`: field id (starting from 1!);
   - `name`: field name, must be the same as the name of a relevant object parameter, string;
   - `field_type`: data type (`unsigned|string|set[string]|set[unsigned]`);
   - `default`: field default value. If a default value exceeds `1e14 - 1`, use a string data type to specify it, for example, `"123123.."` instead of `123123...`
   - `meta_indexes`: the names of the meta fields to build the index on.

You can find the custom `tnt-schema.lua` *here*.

Mount your edited `tnt-scheme.lua` file into the container when running the `tntapi` component:

```
docker run ... \
    --volume /opt/ffserver/configs/tnt-scheme.lua:/tnt-scheme.lua \
    ...
```

Use the environment variable `CFG_EXTRA_LUA` for the file `tnt-scheme.lua`:

```
CFG_EXTRA_LUA='dofile("/tnt-scheme.lua")'
```

4. Mount your edited `FindFace.lua` file into the container when running the `tntapi` component, otherwise, the default one will be used.

```
docker run ... \
    --volume /opt/ffserver/configs/FindFace.lua:/etc/tarantool/instances.enabled/
↪FindFace.lua \
    ...
```

Deploy `upload`

The `upload` service is a storage of normalized images of objects and video chunks. FindFace Server uses the saved normalized images of objects in order to migrate the database to the EmbeN of another neural network, and video chunks are used to implement the Video Recorder. The `upload` API is based on the WebDAV protocol.

Tip: If you don't need this functionality, skip this step.

Install `upload` as such:

```
docker run -tid --name upload --restart always --network server \
    -v /opt/ffserver/upload:/var/lib/ffupload \
    docker.int.ntl/ntech/universe/upload:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `upload`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.
- `/var/lib/ffupload`: directory to store the data.

Deploy `sf-api`

Note: `sf-api` requires such third-party software as `memcached`. *Deploy* it first.

The `sf-api` is a service that implements an internal user-friendly HTTP API for the `extraction-api` and `tntapi`. That is, it provides an opportunity to detect an object in an image and extract its feature vector (or do it from a normalized object images), as well as save it in the Tarantool database.

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

1. Create a default `sf-api` configuration file.

   ```
   docker run --rm -ti docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
       --config-template > /opt/ffserver/configs/sf-api.yaml
   ```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Open the `/opt/ffserver/configs/sf-api.yaml` configuration file.

   ```
   sudo vi /opt/ffserver/configs/sf-api.yaml
   ```

3. Specify the addresses and ports of the `extraction-api` container (`extraction-api -> url`, in the format: `http://<domain_or_ip>:<port>`), the `tntapi` shards (`storage-api -> shards -> master`, in the format: `http://<domain_or_ip>:<port>/v2/`).

   ```
   extraction-api:
     url: http://extraction-api:18666
   ...

   storage-api:
     shards:
     - master: http://tnt-1-1:8001/v2/
       slaves: []
     - master: http://tnt-2-1:8001/v2/
       slaves: []
     read_slave_only: false
     read_slave_first: false
     galleries_read_slave_first: false
     max_slave_attempts: 2
     cooldown: 2s
   ....
   ```

4. The main part of the detected object data is stored in the `sf-api` cache.

   Depending on the load profile, the cache may be:

   - `inmemory`

     ```
     cache:
       type: inmemory
       inmemory:
         size: 16384
     ```

   - `memcache` (see Memcached).

     ```
     cache:
       type: memcache
       memcache:
         nodes:
         - memcached:11211
     ```

   - `redis`

     ```
     cache:
       type: redis
       redis: # https://redis.io/
     ```

(continues on next page)

(continued from previous page)

```
    nodes:
    - redis:6379
    timeout: 5s
    network: tcp
  password: ""
  db: 0
```

5. The `sf-api` service implements HTTP API to access the FindFace Server functions such as object detection and object recognition. In order to allow migration from one `EmbeN` model to another, the `sf-api` provides API to get normalized object images.

   One normalized object image can take from one to hundreds of kilobytes, so the storage for it should be several times larger than for vectors.

   Two storage types are currently supported:

   - `webdav`

   ```
   normalized-storage:
     type: webdav
     enabled: true
     webdav:
       # <http_client options>
       upload-url: http://upload:3333/uploads/
   ```

   - `s3` storage

   ```
   normalized-storage:
     type: s3
     enabled: true
     s3:
       endpoint: ""
       bucket-name: ""
       access-key: ""
       secret-access-key: ""
       secure: true
       region: ""
       public-url: ""
       operation-timeout: 30
   ```

   - If a vector migration is not needed, you can disable the storage of normalized object images:

   ```
   normalized-storage:
     enabled: false
   ```

6. When you have edited the configuration file, run `sf-api` component with the docker run command.

```
docker run -tid --name sf-api --restart always --network server \
    --volume /opt/ffserver/configs/sf-api.yaml:/sf-api.yaml \
    --publish 127.0.0.1:18411:18411 \
    docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
    --config /sf-api.yaml
```

   Use docker options:

   - `--name`: specify a custom identifier for a container, for example `sf-api`.

- `--restart`: restart policy to apply when a container exits. Set `always` to always restart the container if it stops.
- `/opt/ffserver/configs`: the directory on the host to store the configuration file.
- `--network`: connect a container to a network, named `server`.
- `--publish 127.0.0.1:18411:18411`: this allows you to connect to the `sf-api` API from the host with `http://localhost:18411` (if you need to access not only from the local machine, remove `127.0.0.1:`).

Use `sf-api` options:

- `--config`: path to `sf-api.yaml` configuration file.

Deploy `storage-api-proxy` (optional)

The `storage-api-proxy` is an optional component that proxies requests to the specified service that implements `storage-api` (for example, `tntapi` service or another `storage-api-proxy` services).

To run the `storage-api-proxy`, do the following:

1. Create a default `storage-api-proxy` configuration file.

```
docker run --rm -ti --entrypoint "/storage-api-proxy" docker.int.ntl/ntech/universe/
↪sf-api:ffserver-11.240325 \
    --config-template > /opt/ffserver/configs/storage-api-proxy.yaml
```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Modify configuration file depending on your needs, refer to the `storage-api` parameters:

```
storage-api:
  # <http_client options>
  shards:
  - master: http://tnt-1-1:8001/v2/
    slaves: [] # an array of URLs to the other shard nodes
  - master: http://tnt-2-1:8001/v2/
    slaves: []
  read_slave_only: false # If true: Ignore master on read requests. read_slave_
↪first will be ignored.
  read_slave_first: false # If true: Prefer slaves over master for requests.
  galleries_read_slave_first: false # If true: Prefer slaves over master for get/
↪list galleries requests.
  max_slave_attempts: 2 # Give up after trying to read from max_slave_attempts␣
↪slaves (default 2).
  cooldown: 2s # Cooldown timeout after communication error.
```

3. Run `storage-api-proxy` with the configuration file.

```
docker run --rm -ti --network server --entrypoint "/storage-api-proxy" \
    --volume /opt/ffserver/configs/storage-api-proxy.yaml:/storage-api-proxy.yaml \
    --publish 127.0.0.1:18411:18411 \
    docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
    --config /storage-api-proxy.yaml \
```

   Use docker options:

   - `/opt/ffserver/configs`: the directory on the host to store the configuration file.

- `--network`: connect a container to a network, named `server`.
- `--publish 127.0.0.1:18411:18411`: this allows you to connect to the `storage-api-proxy` API from the host with `http://localhost:18411` (if you need to access not only from the local machine, remove `127.0.0.1:`).

Use `storage-api-proxy` options:

- `--config`: path to `storage-api-proxy.yaml` configuration file.

Deploy Video Objects Detection

Video objects detection is provided by the `video-manager` and `video-worker` services.

Note: `video-manager` requires such third-party software as `etcd`. *Deploy* it first.

To deploy the `video-manager` component, run its image and specify the environment variables:

```
docker run -tid --name video-manager-1 --restart always --network server \
    --env CFG_ETCD_ENDPOINTS=http://etcd-1:2379,http://etcd-2:2379,http://etcd-
↪3:2379 \
    --env CFG_RPC_LISTEN=:18811 \
    --env CFG_MASTER_SELF_URL=video-manager-1:18811 \
    --env CFG_MASTER_SELF_URL_HTTP=video-manager-1:18810 \
    --publish 127.0.0.1:18810:18810 \
    docker.int.ntl/ntech/universe/video-manager:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `video-manager-1`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Specify `always` to always restart the container if it stops.

Environment variables and flags:

- `CFG_ETCD_ENDPOINTS=http://etcd-1:2379,http://etcd-2:2379,http://etcd-3:2379`: list of `etcd` URLs, where `etcd-1` is etcd container name.
- `CFG_RPC_LISTEN=:18811`: allow accessing the `video-manager` service from any IP address.
- `CFG_MASTER_SELF_URL=video-manager-1:18811`: self url.
- `CFG_MASTER_SELF_URL_HTTP=video-manager-1:18810`: self url http.
- `--publish 127.0.0.1:18810:18810`: this allows you to connect to the `video-manager` API from the host with `http://localhost:18810` (if you need to access not only from the local machine, remove `127.0.0.1:`).

If needed, configure the following parameters using environment variables/flags or through a configuration file:

1. If you need to deploy `video-worker` instances on remote hosts, specify `--publish 18811:18811` and in `CFG_MASTER_SELF_URL` and `CFG_MASTER_SELF_URL_HTTP` replace the container's hostname with the hostname or IP of the host.
2. If needed, in the `router_url` parameter, specify IP address and port of the `facerouter` component (if installed) which will receive detected faces from `video-worker` (env: `CFG_ROUTER_URL`).

```
--env CFG_ROUTER_URL=http://<facerouter-ip>:8085/
```

Note: <facerouter-ip> must be substituted into your IP address for the `facerouter` component.

3. If you need to run multiple `video-manager` instances, and you need the license limit on the number of cameras was checked by the `video-manager`, then specify the environment variables `CFG_NTLS_ENABLED=true` and `CFG_NTLS_URL=ntls:3185` where `video-manager` can send requests.
4. 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 `video-manager`, you will be able to individually configure processing settings for each video stream (see *Video Object Detection API*).

As an alternative way, you can use the configuration file:

1. Create a default `video-manager.yaml` configuration file.

```
docker run --rm -ti docker.int.ntl/ntech/universe/video-manager:ffserver-11.240325 \
    --config-template > /opt/ffserver/configs/video-manager.yaml
```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Open the `/opt/ffserver/configs/video-manager.yaml` configuration file and specify parameters as above.

```
sudo vi /opt/ffserver/configs/video-manager.yaml
```

3. Run the `video-manager` service using `--config /video-manager.yaml` flag with your modified configuration file mounted into the container.

```
docker run -tid --name video-manager-1 --restart always --network server \
    -v /opt/ffserver/configs/video-manager.yaml:/video-manager.yaml \
    docker.int.ntl/ntech/universe/video-manager:ffserver-11.240325 --config /video-
↪manager.yaml
```

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

Note: To deploy the `video-worker` service with acceleration on the GPU, use the `video-worker-gpu` image. Don't forget to use the `--runtime=nvidia` flag in the `docker run` command.

1. Create a default `video-worker.yaml` configuration file.

```
docker run --rm -ti docker.int.ntl/ntech/universe/video-worker-cpu:ffserver-11.
↪240325 \
    --config-template > /opt/ffserver/configs/video-worker.yaml
```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Open the `/opt/ffserver/configs/video-worker.yaml` configuration file.

```
sudo vi /opt/ffserver/configs/video-worker.yaml
```

3. Fill out the values for all required parameters. Make sure that `detectors`, `normalizers`, and `extractors` are specified in the `models` section and `objects` section has corresponding values. Below is an example of how the section should look like. It may vary depending on the recognition objects that you have selected.

```
models:
  cache_dir: /var/cache/findface/models_cache
  detectors:
    car:
      fnk_path: /usr/share/findface-data/models/detector/cardet.kali.008.cpu.fnk
      min_size: 60
    body:
      fnk_path: /usr/share/findface-data/models/detector/bodydet.kali.021.cpu.fnk
      min_size: 60
    face:
      fnk_path: /usr/share/findface-data/models/detector/facedet.jasmine_fast.004.
↪cpu.fnk
      min_size: 60
  normalizers:
    car_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    car_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    body_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    body_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    face_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.cpu.
↪fnk
    face_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/crop1x.v2_maxsize400.cpu.
↪fnk
  extractors:
    car_quality:
      fnk_path: /usr/share/findface-data/models/carattr/carattr.quality.v1.cpu.fnk
      normalizer: car_norm_quality
    body_quality:
      fnk_path: /usr/share/findface-data/models/pedattr/pedattr.quality.v0.cpu.fnk
      normalizer: body_norm_quality
    face_quality:
      fnk_path: /usr/share/findface-data/models/faceattr/faceattr.quality.v5.cpu.fnk
      normalizer: face_norm_quality
  objects:
    car:
      normalizer: car_norm
      quality: car_quality
      track_features: ''
    body:
      normalizer: body_norm
      quality: body_quality
      track_features: ''
    face:
      normalizer: face_norm
```

(continues on next page)

(continued from previous page)

```
      quality: face_quality
      track_features: ''
```

4. Run `video-worker` image and specify the environment variables. Mount volumes with models and modified configuration file.

```
docker run -tid --name video-worker --network server --restart always \
    --env CFG_MGR_STATIC=video-manager-1:18811 \
    --env CFG_NTLS_ADDR=ntls:3133 \
    --volume /opt/ffserver/models:/usr/share/findface-data/models \
    --volume /opt/ffserver/configs/video-worker.yaml:/video-worker.yaml \
    docker.int.ntl/ntech/universe/video-worker-cpu:ffserver-11.240325 \
    --config /video-worker.yaml
```

   Use docker options:

   - `--name`: specify a custom identifier for a container, for example `video-worker`.
   - `--network`: connect a container to a network, named `server`.
   - `--restart`: restart policy to apply when a container exits. Specify `always` to always restart the container if it stops.

   Environment variables and flags:

   - `CFG_MGR_STATIC=video-manager-1:18811`: videomanager grpc ip:port, which provides `video-worker` with settings and the video stream list.
   - `CFG_NTLS_ADDR=ntls:3133`: `ntls` server ip:port.
   - `/opt/ffserver/models`: the directory on the host to store models.
   - `/opt/ffserver/configs`: the directory on the host to store the configuration file.

Deploy Video Recorder: `video-storage` and `video-streamer-cpu`

The Video Recorder operation requires a third-party software, MongoDB. Install it as follows:

```
docker run -tid --name mongo --restart always --network server \
    --volume /opt/ffserver/mongo-data:/data/db \
    docker.io/library/mongo:4.4
```

To deploy the `video-storage` component, run its image and specify the environment variables:

```
docker run -tid --name video-storage --restart always --network server \
    --env CFG_STREAMER_ENDPOINTS=video-streamer:9000 \
    --env CFG_CHUNK_STORAGE_TYPE=webdav \
    --env CFG_CHUNK_STORAGE_WEBDAV_UPLOAD_URL=http://upload:3333/uploads/ \
    --env CFG_META_STORAGE_MONGO_URI=mongodb://mongo \
    --publish 127.0.0.1:18611:18611 \
    docker.int.ntl/ntech/universe/video-storage:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `video-storage`.
- `--network`: connect a container to a network, named `server`.

- `--restart`: restart policy to apply when a container exits. Specify `always` to always restart the container if it stops.

Environment variables and flags:

- `CFG_STREAMER_ENDPOINTS=video-streamer:9000`: list of streamer endpoint URLs.
- `CFG_CHUNK_STORAGE_TYPE=webdav`: Set `webdav` in chunk storage type.
- `CFG_CHUNK_STORAGE_WEBDAV_UPLOAD_URL=http://upload:3333/uploads/`: webdav storage URL.
- `CFG_META_STORAGE_MONGO_URI=mongodb://mongo`: meta storage mongo URI.
- `--publish 127.0.0.1:18611:18611`: this allows you to connect to the `video-storage` API from the host with `http://localhost:18611` (if you need to access not only from the local machine, remove `127.0.0.1:`).

To deploy the `video-streamer-cpu` component, run its image and specify the environment variables:

```
docker run -tid --name video-streamer --restart always --network server \
    --env CFG_VIDEO_STORAGE_URL=http://video-storage:18611 \
    --volume /opt/ffserver/video-streamer:/var/cache/findface/video-streamer \
    --publish 127.0.0.1:9000:9000 \
    docker.int.ntl/ntech/universe/video-streamer-cpu:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `video-streamer`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Specify `always` to always restart the container if it stops.

Environment variables and flags:

- `CFG_VIDEO_STORAGE_URL=http://video-storage:18611`
- `--volume /opt/ffserver/video-streamer:/var/cache/findface/video-streamer`: mount cache with volume.
- `--publish 127.0.0.1:9000:9000`: this allows you to connect to the `video-streamer` API from the host with `http://localhost:9000` (if you need to access not only from the local machine, remove `127.0.0.1:`).

Deploy `liveness-api`

To deploy the `liveness-api` component, run its image and specify the environment variables:

```
docker run -tid --name liveness-api --restart always --network server \
    --env CFG_EXTRACTION_API_EXTRACTION_API=http://extraction-api:18666 \
    --env CFG_SF_API_SF_API=http://sf-api:18411 \
    --publish 127.0.0.1:18301:18301 \
    docker.int.ntl/ntech/universe/liveness-api:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `liveness-api`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Specify `always` to always restart the container if it stops.

Environment variables and flags:

- `--publish 127.0.0.1:18301:18301`: this allows you to connect to the `liveness-api` API from the host with `http://localhost:18301` (if you need to access not only from the local machine, remove `127.0.0.1:`).

See also:

Liveness Detection as Standalone Service

Deploy `counter`

Note: `counter` requires such third-party software as `postgres`. *Deploy* it first.

The `counter` lets you get statistics of unique individuals.

To deploy the `counter` component, run its image and specify the environment variables:

```
docker run -tid --name counter --restart always --network server \
    --env CFG_DATABASE_CONNECTION_STRING=postgres://login:password@db-postgres/
↪ntech?sslmode=disable \
    --publish 127.0.0.1:18300:18300 \
    docker.int.ntl/ntech/universe/counter:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `counter`.
- `--network`: connect a container to a network, named `server`.
- `--restart`: restart policy to apply when a container exits. Specify `always` to always restart the container if it stops.

Environment variables and flags:

- `CFG_DATABASE_CONNECTION_STRING=postgres://login:password@db-postgres/ntech?sslmode=disable`: PostgreSQL connection string.
  - `login:password`: these are the credentials used to authenticate with the database.
  - `db-postgres`: name of the container, where PostgreSQL database is located, or remote host.
  - `ntech`: the name of the database.
- `--publish 127.0.0.1:18300:18300`: this allows you to connect to the `counter` API from the host with `http://localhost:18300` (if you need to access not only from the local machine, remove `127.0.0.1:`).

Deploy `deduplicator`

To deploy the `deduplicator` component, run its image and specify the environment variables and flags:

```
docker run -tid --name deduplicator --restart always --network server \
    --publish 127.0.0.1:18310:18310 \
    docker.int.ntl/ntech/universe/deduplicator:ffserver-11.240325
```

Use docker options:

- `--name`: specify a custom identifier for a container, for example `deduplicator`.
- `--network`: connect a container to a network, named `server`.

- `--restart`: restart policy to apply when a container exits. Specify `always` to always restart the container if it stops.

Environment variables and flags:

- `--publish 127.0.0.1:18310:18310`: this allows you to connect to the `deduplicator` API from the host with `http://localhost:18310` (if you need to access not only from the local machine, remove `127.0.0.1:`).

1.4.3 Post-deployment Procedures and How-to's

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

In this section:

License Instance

FindFace Server provides several licensing options. Whichever option you choose, you upload the FindFace Server license file via `ntls` web interface `http://<Host_IP_address>:3185/`.

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

Configure Logging

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

Useful Docker Commands

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

- View all Docker containers, including the stopped ones:

  ```
  docker ps -a
  ```

  To get a more compact and understandable output, execute:

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

  To extend the previous output, execute:

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

- Restart the Docker service:

```
sudo systemctl restart docker
```

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

```
journalctl CONTAINER_NAME=<findface-container_name> -f
```

- Stop a Docker container:

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

  Stop all Docker containers:

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

- Start a Docker container:

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

  Start all Docker containers:

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

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

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

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

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

1.4.4 Remove FindFace Server Instance

To remove your FindFace Server instance, remove all product components manually.

1. Stop running containers.

   To stop a single container, you can use the command `docker stop <container-name>` where `<container-name>` is the name or ID of the container you wish to stop. To stop multiple containers at once, you can specify their names or IDs separated by a space: `docker stop container1 container2 container3`.

2. Remove containers.

   To removes all stopped containers, use the command `docker container prune`. To Remove one or more containers, use the command `docker rm <container-name>`.

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

1.5 Administration and Basic Configuration

1.5.1 Licensing

In this chapter:

Licensing Principles

The FindFace Server licensing is granted using the following criteria:

1. The overall number of feature extractions, with grouping by feature.
2. Face attribute recognition: gender/age/emotions/glasses/beard/face mask/etc.
3. Body attribute recognition: clothing color/type/etc.
4. Vehicle attribute recognition: make/model/color/body style/etc.
5. License plate recognition.
6. Dumpster, container, trash bin, pet face, streetlight, garbage, merch, etc. attribute recognition.
7. The total `extraction-api` capacity is constrained by the `extapi` limit.
8. The total number of objects stored in all currently running `tntapi` instances.
9. `tntapi` fast index.
10. The number of cameras with a given detector.
11. Video recording.
12. Face liveness detection.

You can choose between the following licensing methods:

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

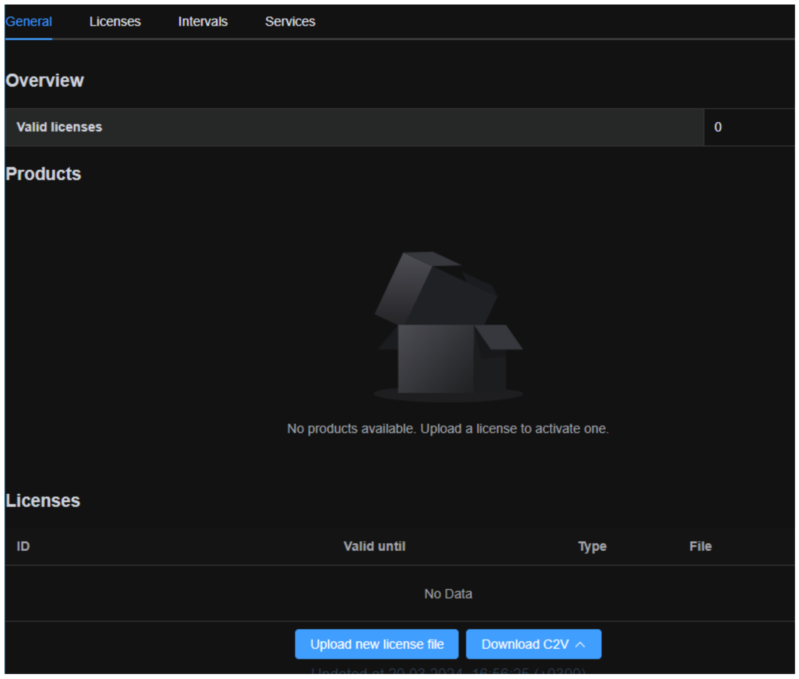
Note: It is possible to prolongate the off-grid period for up to 2 days. Inform your manager if you need that.

- The offline licensing via a USB dongle requires a USB port on the physical server with the `ntls` service.
- The Sentinel offline software licensing via hardware fingerprint requires Sentinel drivers installed on the physical server with the `ntls` service.
- The Guardant offline software licensing via hardware fingerprint requires Guardant Control Center installed on the physical server with the `ntls` service.

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

View and Update License

After installing FindFace Server, upload the license file you obtained from the manager into the system. Use `ntls` web interface `http://<NTLS_IP_address>:3185`.



Overview

Valid licenses	0

Products

No products available. Upload a license to activate one.

Licenses

ID	Valid until	Type	File
No Data			

Use *General*, *Licenses*, *Intervals*, *Services* tabs to consult current licensing information and upgrade your license.

Offline Licensing via USB dongle

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

1. Inform your manager that you intend to apply this licensing method and request your USB dongle and a license file.
2. Create a new **udev** rule.
   1. Download the `95-grdnt.rules` file (e.g., into the `/home/username/tmp/` directory).
   2. Copy the `95-grdnt.rules` file into the `/etc/udev/rules.d/` directory.

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

   ```
   udevadm control --reload
   udevadm trigger
   ```

4. Make sure the system has a device named `grdhid0`.

   ```
   /dev/grdhid0
   ```

5. Mount the `/dev/grdhid0` device into the `ntls` container by using volume option at runtime with the `docker run` command.

   ```
   -v /dev/grdhid0:/dev/grdhid0
   ```

Sentinel Offline Software Licensing via Hardware Fingerprint

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

Glossary:

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

To implement the Sentinel fingerprint licensing, do the following:

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

   Do the following:

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

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

   3. There is another archive `aksusbd-9.15.tar.gz` inside the archive. Unzip it and browse to the resulting directory.

      ```
      tar -xvzf aksusbd-9.15.tar.gz
      cd aksusbd-9.15/
      ```

   4. Run the installation command.

      ```
      sudo ./dinst
      ```

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

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

3. Mount the `/var/hasplm` and `/etc/hasplm` directories into the `ntls` container. To do so, use `-v` option at runtime with the `docker run` command.

```
-v /var/hasplm:/var/hasplm -v /etc/hasplm:/etc/hasplm
```

4. `ntls` must be running in the host's network namespace.

```
--net=host
```

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

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

6. Take a hardware fingerprint (C2V file) by sending the following API request to `ntls`.

```
wget <NTLS_IP_address>:3185/c2v -o sentinel.c2v
```

7. Send the License ID and the C2V file to your manager and receive your license file in return.
8. Upload the license file on the web interface `http://<NTLS_IP_address>:3185`.

Guardant Offline Software Licensing via Hardware Fingerprint

Note: This type of license is similar to Sentinel offline software licenses, but requires the Guardant service to function instead of Sentinel services.

To implement the Guardant fingerprint licensing, follow these steps:

1. Inform your manager that you intend to use this licensing method and request your unique license id.
2. Install the Guardant Control Center on the physical server with the `ntls` component.

   Do the following:

   1. Download Guardant Control Center from the official website. For Ubuntu/Debian, download the `grdcontrol*.deb` package. The most recent version that has been tested with is 3.27. Use later versions of Guardant Control Center, if any, at your own discretion.
   2. Install it. E.g., if you are running Ubuntu, execute:

      ```
      sudo dpkg -i /path/to/grdcontrol*.deb
      ```

   3. Make sure that the Guardant Control Center is up and running. On Ubuntu 22.04, you can do this as follows:

      ```
      sudo systemctl enable grdcontrol --now
      sudo systemctl status grdcontrol
      ```

3. `ntls` must be running in the host's network namespace.

```
--net=host
```

4. Take a hardware fingerprint (C2V file) by sending the following API request to `ntls`.

```
wget <NTLS_IP_address>:3185/c2v/guardant -o guardant.c2v
```

Warning: Be aware that Sentinel c2v is not compatible with Guardant c2v and vice versa.

Note: You can also download c2v using the `ntls` web interface.

5. Send the License ID and the C2V file to your manager and receive your license file in return.
6. Upload the license file on the web interface `http://<NTLS_IP_address>:3185`.

1.5.2 Allocate `video-worker` to Video Stream Source

In a distributed architecture, it is often necessary that video streams be processed *in situ*, without being redistributed across remote `video-worker` instances by the principal server.

Note: Among typical use cases are hotel chains, chain stores, several security checkpoints in the same building, etc.

In this case, allocate the local `video-worker` to the video stream source.

There are two ways to specify allocation labels:

- by `video-worker.yaml` configuration file
- by environment variable or flag.

Note: Use the same name of labels in the `video-worker` configuration file and in the video processing job, otherwise video stream source will not process.

Do the following:

1. Open the `/opt/ffserver/configs/video-worker.yaml` configuration file and specify the allocation labels in the dictionary format (label `district` in the example below).

```
sudo vi /opt/ffserver/configs/video-worker.yaml

labels: {"district": "SVAO"}
```

2. As an alternative way, you can use environment variable `CFG_LABELS` or `--labels` flag when running `video-worker` with docker run command:

```
CFG_LABELS=district=SVAO
```

3. Create a *video processing job* with labels.

```
"labels": {
      "district": "SVAO"
    },
```

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

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

1.5.3 Direct API requests to `extraction-api`

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

Note: `extraction-api` is the component that `sf-api` relies on for object detection and feature vector and attribute extraction. It is stateless and operates purely on request-response basis.

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

In this section:

Structure

API Requests structure

The `extraction-api` component accepts requests to `http://<extraction-api_ip>:18666/`.

There are an API v1 and v2 (multiobject).

Important:

Differences:

- The API v1 is selected automatically, if not route `/v2` in the request URL.

- The API v2 defines the object type and return response in appropriate fields. Also, it has new unified attribute names.

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
    "response_format": "msgpack" // return response in msgpack format
}
```

API Response Structure

A typical response from the `extraction-api` component contains a set of responses to the requests wrapped into the main API request:

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

API V2

API Request Format

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

- `"image"`: an uploaded image (use `multipart:part` to refer to a relevant request body `part`), or a publicly accessible image URL (`http:`, `https:`).
- `"roi"`: a region of interest in the image. If the region is not specified, the entire image is processed.
- `"detector"`: an object detector to apply to the image (`face`, `body`, `car` and etc or `prenormalized` or `original`). The `prenormalized` mode accepts normalized object images and omits detecting objects.
- `object_type`: it's a required parameter, if the value of the detector is `original`. For example, to extract `face_liveness` from the original image the value must be `face`, to extract frameattrs the value must be `none`.
- `bbox`: object bbox. It is used for extraction from `original`, extraction will be from the image with that value of bbox.
- `"need_normalized"`: returns a normalized object image encoded in base64. The normalized image can then be posted again to the `extraction-api` component as `prenormalized`.
- `"auto_rotate"`: if true, auto-rotates an original image to 4 different orientations and returns objects detected in each orientation.
- `"quality_estimator"`: if false, `detection_score` returns from the detector without `face_quality` attribute extract.

- "attributes": array of strings in the format ["face_gender", "face_age", "face_emotions"], enables recognition of the objects features passed in the array. Attribute name contains object type as prefix (face_, body_ and etc.)

```
{
    "image": "http://static.findface.pro/sample.jpg",
    "roi": {"left": 0, "right": 1000, "top": 0, "bottom": 1000},
    "detector": "face",
    "need_normalized": true,
    "auto_rotate": true,
    "attributes": ["face_emben", "face_gender", "face_age", "face_emotions", "face_beard
↪", "face_glasses3"]
}
```

API Response Format

Each response in the set contains the following JSON data:

- "objects": a structure with sets of detected objects 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").
- "timings": processing timings if "include_timings": true.

```
{
    "objects": {
        "face": [...] // detected face objects
        "car": [...], // detected car objects
        "head": [...] // detected head objects
        "body": [...] // detected body objects
    },
    "orientation": 1, // input image orientation
    "detector": "" // detector name
    "timings": ... // timings if requested
}
```

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

- "group_id": detection group identifier. All bboxes from one detects will have same value. For example, uses for N-in-1 detectors for group head, body, face from 3-in-1 detector.
- "bbox": coordinates of a bounding box with the object.
- "detection_score": either the object detection accuracy, or the object quality score. Upright objects in frontal position are considered the best quality. They result in values around 0, mostly negative (such as -0.00067401276, for example). Inverted objects and large object angles are estimated with negative values some -5 and less.
- "rotation_angle": angle of input image, which them object was detected.
- "attributes": object with results of attributes extraction. As keys uses full attribute name (face_age, face_gender), as value objects with the next data:
  - "extractor": extractor name.
  - "model_name": name of the extractor model.

  - "value": extraction result, may be different types for different extractors.

- "normalized": a normalized face image encoded in base64, if requested.
- "timings": face processing timings, if requested.

```
{
    "group_id": "28c97d15",
    "bbox": { "left": 1, "right": 2, "top": 3, "bottom": 4},
    "detection_score": 0.99,
    "normalized": "...",
    "attributes": {
      "face_age": {
          "extractor": "face_age",
          "model": "age.v2",
          "result": 25
      },
      "face_beard": {
          "extractor": "face_beard",
          "model": "beard.v0",
          "result": [
              { "confidence": 0.015328666, "name": "beard" }
          ]
      },
      "face_emotions": {
          "extractor": "face_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" }
          ]
      },
      ...
    }
    "timings": ...
}
```

Examples

Request #1

```
curl -X POST -F sample=@sample.jpg -F 'request={"requests":[{"image":"multipart:sample",
↪"detector":"face", "attributes": ["face_age", "face_gender", "face_emben"]}]}' http://
↪127.0.0.1:18666/v2 | jq .
```

Response

```
{
  "responses": [
    {
      "faces": null,
      "objects": {
        "face": [
          {
            "group_id": "b781670d",
            "bbox": {
              "left": 168,
              "top": 338,
              "right": 812,
              "bottom": 1234
            },
            "detection_score": 0.7689582,
            "rotation_angle": 0,
            "attributes": {
              "face_age": {
                "extractor": "face_age",
                "model": "age.v2",
                "result": 47
              },
              "face_emben": {
                "extractor": "face_emben",
                "model": "kiwi_320",
                "result": "..."
              },
              "face_gender": {
                "extractor": "face_gender",
                "model": "gender.v2",
                "result": [
                  {
                    "confidence": 1,
                    "name": "male"
                  },
                  {
                    "confidence": 5.503795e-08,
                    "name": "female"
                  }
                ]
              }
            }
          }
        ]
      },
      "orientation": 1,
      "detector": "face_jasmine"
    }
  ]
}
```

Important: If the requested attribute is not founded in the configuration file or isn't loaded, or object attribute does not match detect object type, this attribute will be ignored in the returned response.

Important: If `need_normalized: true` is specified in the request, normalization from `"objects:object:base_normalization"` config field will be used. If there is no base normalization in config, the default normalization will be used.

Request #2 A simple request with a 3-in-1 `headbodyface` detector

```
curl -s -X POST -F sample=@sample_3in1.jpg -F 'request={"requests":[{"image":
↪"multipart:sample", "detector":"headbodyface", "attributes": ["face_emben", "body_emben
↪", "head_motohelmet"]}]}' http://127.0.0.1:18666/v2 | jq
```

Response

```
{
  "responses": [
    {
      "faces": null,
      "objects": {
        "face": [
          {
            "group_id": "43c199aa",
            "bbox": {
              "left": 616,
              "top": 232,
              "right": 645,
              "bottom": 266
            },
            "detection_score": 0.67829776,
            "rotation_angle": 0,
            "attributes": {
              "face_emben": {
                "extractor": "face_emben",
                "model": "kiwi_320",
                "result": "..."
              }
            }
          }
        ],
        "head": [
          {
            "group_id": "43c199aa",
            "bbox": {
              "left": 615,
              "top": 225,
              "right": 652,
```

(continues on next page)

(continued from previous page)

```
              "bottom": 270
            },
            "detection_score": 0.94091797,
            "rotation_angle": 0,
            "attributes": {
              "head_motohelmet": {
                "extractor": "head_motohelmet",
                "model": "headattr.motohelmet.v1",
                "result": 0.109558105
              }
            }
          }
        ],
        "body": [
          {
            "group_id": "43c199aa",
            "bbox": {
              "left": 544,
              "top": 220,
              "right": 691,
              "bottom": 468
            },
            "detection_score": 0.7998271,
            "rotation_angle": 0,
            "attributes": {
              "body_emben": {
                "extractor": "body_emben",
                "model": "andariel",
                "result": "..."
              }
            }
          }
        ]
      },
      "orientation": 1,
      "detector": "headbodyface"
    }
  ]
}
```

Warning: `headbodyface` detector must be enabled in the `extraction-api` configuration file.

```
detectors:
    max_batch_size: 1
    instances: 1
    models:
    headbodyface:
      aliases:
      - headbodyface
      model: detector/headbodyface.gpu.fnk
      options:
        min_object_size: 32
        resolutions: [2048x2048]
```

Request #3 Request with "frameattr" extraction

```
curl -s -X POST -F sample=@/home/crowd.jpg -F 'request={"requests":[{"image":
↪"multipart:sample", "detector":"original", "attributes":["crowd_count"], "object_type
↪": "none"}]}' http://127.0.0.1:18666/v2/ | jq
```

Response

```
{
  "responses": [
    {
      "faces": null,
      "objects": {
        "none": [
          {
            "group_id": "",
            "bbox": {
              "left": 0,
              "top": 0,
              "right": 1276,
              "bottom": 608
            },
            "detection_score": 1,
            "rotation_angle": 0,
            "attributes": {
              "crowd_count": {
                "extractor": "crowd_count",
                "model": "frameattr.crowdcount.v0",
                "result": {
                  "count": 690.92346,
                  "heatmap_height": 76,
                  "heatmap_image": "...",
                  "heatmap_image_multiplier": 0.9641899,
                  "heatmap_width": 159
                }
              }
            }
          }
        ]
      },
      "orientation": 1,
      "detector": "original"
    }
  ]
}
```

Request #4 Request with liveness extraction from original

```
curl -s -X POST -F sample=@/home/sample2.jpg -F 'request={"requests":[{"image":
↪"multipart:sample", "detector":"original", "attributes":["face_liveness", "face_emben
↪"], "bbox": {"left": 10, "top": 10, "right": 444, "bottom": 444}, "object_type": "face
↪"}]}' http://127.0.0.1:18666/v2/ | jq
```

Response

```
{
  "responses": [
    {
      "faces": null,
      "objects": {
        "face": [
          {
            "group_id": "",
            "bbox": {
              "left": 10,
              "top": 10,
              "right": 445,
              "bottom": 445
            },
            "detection_score": 1,
            "rotation_angle": 0,
            "attributes": {
              "face_emben": {
                "extractor": "face_emben",
                "model": "kiwi_160",
                "result": "..."
              },
              "face_liveness": {
                "extractor": "face_liveness",
                "model": "liveness.pvn.v0",
                "result": 0.77615935
              }
            }
          }
        ]
      },
      "orientation": 1,
      "detector": "original"
    }
  ]
}
```

Method GET /v2/models-info

This method returns the information about enabled detectors, normalizers, extractors and objects.

Request

```
curl -s  http://127.0.0.1:18666/v2/models-info | jq
```

Response

```
{
  "detectors": {
    "body": {
      "object_types": [
        "body"
      ]
    },
    "car": {
      "object_types": [
        "car"
      ]
    },
    "gustav_body": {
      "object_types": [
        "body"
      ]
    },
    "gustav_car": {
      "object_types": [
        "car"
      ]
    },
    "headbodyface": {
      "object_types": [
        "head",
        "body",
        "face"
      ]
    },
    "license_plate": {
      "object_types": [
        "license_plate"
      ]
    },
    "license_plate_gustav_accurate": {
      "object_types": [
        "license_plate"
      ]
    },
    "shiloette": {
```

(continues on next page)

(continued from previous page)

```
      "object_types": [
        "body"
      ]
    }
  },
  "normalizers": {
    "carlicplate": {
      "normalization_type": "carlicplate"
    },
    "cropbbox": {
      "normalization_type": "cropbbox"
    },
    "norm200": {
      "normalization_type": "norm200"
    }
  },
  "extractors": {
    "car_color": {
      "normalization": "crop1x",
      "model_name": "carattr_color.v0"
    },
    "car_quality": {
      "normalization": "cropbbox",
      "model_name": "carattr.quality.v0"
    },
    "face_emben": {
      "normalization": "norm200",
      "model_name": "kiwi_160"
    },
    "face_quality": {
      "normalization": "crop1x",
      "model_name": "quality_fast.v1"
    },
    "license_plate_quality": {
      "normalization": "cropbbox",
      "model_name": "carlicplateattr.quality.v0"
    }
  },
  "objects": {
    "car": {
      "quality_attribute": "car_quality",
      "base_normalizer": "cropbbox"
    },
    "face": {
      "quality_attribute": "face_quality",
      "base_normalizer": "crop2x"
    },
    "license_plate": {
      "quality_attribute": "license_plate_quality",
      "base_normalizer": "carlicplate"
    }
  }
```

(continues on next page)

(continued from previous page)

```
}
```

API V1

API Request Format

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 `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
}
```

API Response Format

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 extraction-api.yaml). 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,
 "facen": "...",
 "gender": {
     "gender": "MALE",
     "score": "1.123"
 },
 "age": 23.59,
```

(continues on next page)

(continued from previous page)

```
"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": {
      "attribute": "glasses3",
```

(continues on next page)

(continued from previous page)

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

Examples

Request #1

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

Response

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

Request #2

```
curl -X POST  -F 'request={"requests": [{"need_age": true, "need_gender": true, "detector
↪": "nnd", "roi": {"left": -2975, "top": -635, "right": 4060, "bottom": 1720}, "image":
↪"https://static.findface.pro/sample.jpg", "need_emotions": true}]}' http://127.0.0.
↪1:18666/ |jq
```

Response

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

(continues on next page)

(continued from previous page)

```
            {
              "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,
            "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,
```

(continues on next page)

(continued from previous page)

```
              "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": {
          "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"
            },
            {
```

(continues on next page)

(continued from previous page)

```
                  "confidence": 8.4465064e-05,
                  "name": "sun"
                }
              ]
            }
          }
        }
      ],
      "orientation": 1
    }
  ]
}
```

1.5.4 Direct API requests to `tntapi`

You can use HTTP API to extract data directly from the `tntapi`.

In this section:

General Information

API requests to `tntapi` are to be sent to `http://<tnt_api_host_ip:port>`.

Tip: The port for API requests can be found in the environment `CFG_LISTEN_PORT`:

```
docker exec -ti tnt-1-1 env
docker exec -ti tnt-1-1 env | grep CFG_LISTEN
CFG_LISTEN_HOST=0.0.0.0
CFG_LISTEN_PORT=8001
```

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

- API version only v2 at the moment.
- `:name`: gallery name.

Node status

```
GET /v2/status
```

This method returns `read_only` node status and repeat header `"X-read-only"` value. If the value is `"true"` it means that the `tntapi` node has `follower` status and only read operations are available. Otherwise, if the value is `"false"`, it means that the `tntapi` node has `leader` status, read and write operations with the node are available.

Returns:

- HTTP 200 and list with null values on success.
- HTTP with a status other than 200 and an error description in the body on failure.

Example

Request

```
curl -D - -X GET -s 'http://localhost:8001/v2/status'
```

Response

```
HTTP/1.1 200 Ok
X-request-id: TN:cU4qiMGc
Content-type: application/json
X-read-only: false
Content-length: 19
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

{"read_only":false}
```

Add Face

```
POST /v2/faces/add/:name[?return_face=false]
```

Parameters in path segments:

- `:name`: gallery name.
- `return_face`: non-required parameter (default value false).

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.
- `"facen_model"`: facen model name, string.
- `"meta"`: face metadata, dictionary.

The field `"facen_model"` is not required. When adding the first face to the gallery, the value will be saved in the gallery. Otherwise, if there is already a saved value in the gallery, it will be checked. If the values in the request and in the gallery are different, the error 400 `Gallery[:name]: facen model mismatch: required :exp_val, got :val` will be returned.

Returns:

- HTTP 200 and a list with null values on success. If `"return_face=true"` is specified in the request, `"id"` and `"meta"` will be returned for successfully added faces.
- 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/z0BQTk9rcirOyZrE72yi047zl9gPSER2zxKyds9CjfCPQvR071z02S9mU0/
↪vVOVEj3SiZw8Q4Q9vViRFjo4Hqi8lc4wvDxcHT1kfpC82uVCPYEHr73RRMS7TVQpPCPUNjxXqZs99NqOvXbPEzxYsOQ9SL/
↪CvG3w4L0h4pG96nvBvTo85TwLByY8H2XMPQ3anb03FV68q/
↪UKPLCdlr3K6Kk9H4NSvAQnHD3nAmc7A1RAPZ8BgLzMZu47KwRovbC+PzzISL68vaYPvUfP1D1//
↪EM9xYZpvEaeGb3Gnti9/
↪n+VPZnONz2xQVW8fGNCPduV2DwGsaG74eCWvNdLy7yjFUe9skvtPJStxbvXk0O9C6nRPJj14zzR+iq9L8WqPWK8vr1OFAy91M8APZ
↪1Uy8ML+Euxl+1LzxbpG9FIDxvEPqur3WdHW9IDRRPfro6L0mSp88z6aivSmzrr3a4Qa+2Q3luyOatDwhaCI9AFecvPjA3jwb5B29u
↪9x49GeDyu+jjRr0RqPu8GqYgvf9piz183C490197Pe76gTvMUoY8oQBUPRjL1z2E95C83jcSPJIFML15SxG8ukGMPP1/
↪8bxoXIw9vYkevSMcKb2vmw69PM4+vftBxz068tA97uNLvSwazjxpMJu9N1akPHBDmz12c8Y8W08hPA3DpD1cOsC8lOc6PZCxh73EX
↪L3y2Ta77Bu+PbpY57ymrfU8IayRvZaLEz00w/
↪a9n42WOk23jbxZ4t49q1soPdB0ED39OXu8DhMYvbPRYz011C88WBiBvae8zrwL6Yk8GdTrPONJOz2GIxO8R+lmO2rwR73WOGm9JC0
↪r2XDVC7XdhtPb9vgrxdUZQ9GJ06PTCGJTsDAS09/
↪dgqPYT0ej20BjW+NPE7PLP3C7mTy4s85D5iPHrbUT3+3xA9Fcg0PCSdLD3AFbg82byAPC4doDymsHS8cvk3vJHogruI+ig9+qiWvS
↪O8S19rvRPXZz2aVcI9+OUcvCfdjT0ocIm9c3vju97AE72sI4w9rHuYPexB+TywgcU7fkQyumxQgr0Ik9g66oFGPSeljjxayPG8WBj
↪UPJ5Fsr3P46y9fOpGvfC9ojtO5Ao9nqRUvbUNYz00+KE8WgZyvKsWlD0YTma9hu2tPTX2ZD3kZ967tCJuvEADurxYUnO9Ezs1vdIG
```

(continues on next page)

(continued from previous page)

```
↪hD0/NpO9MdHavW1WuT0=",
    "meta": {
      "cam_id": "223900",
      "person_id": "123"
    }
  }
]'
```

Response

```
HTTP/1.1 200 Ok
X-request-id: TN:T39tIFME
Content-type: application/json
X-read-only: false
Content-length: 60
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

[null,{"error":{"code":400,"message":"face 'id' required"}}]
```

Remove Face

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

Parameters in path segments:

- `:name`: gallery name.

Parameters in body:

JSON-encoded array of face ids to be removed.

Returns:

- **HTTP 200 and a list of error values on success:**
  - `null` if face was successfully removed.
  - `{"error": {"code": 404, ...}}` if a face did not exist.
- 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/delete/testgal' --data '[1, 4, 922, 3]'
```

Response

```
HTTP/1.1 200 Ok
X-request-id: TN:sgi6xdVO
Content-type: application/json
X-read-only: false
Content-length: 6
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

[null, null, null, null]
```

Face Search

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

Parameters in path segments:

- `:name`: gallery name.

Parameters in body:

JSON-encoded search request with the following fields:

- `limit`: maximum number of faces in the response.
- `without_facen`: (default=false) do not return facen in response.
- `without_meta`: (default=false) do not return meta in 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`: similar to `like` in SQL requests: only 'aa%', 'aa%', and '%aa%' are supported. Only for strings and set[string]. In the case of set[string], the filter will return result if at least one value meets the filter condition.

  - `ilike`: similar to `like` but case-insensitive, only for strings and set[string].

The logic of the `set/subset` filters for fields of type `set[string]` is as follows:

- `set` – if any of the request values are in the set of values stored in database (the non-empty intersection of sets).
- `subset` – if all of the request values are in the set of values stored in database (the intersection of sets is equal to the set from request).

There are supported types of sorts (`sort`):

- `id` – asc (1) or desc (-1). It works only if filters by facen and live index (fast index) are used (both conditions require).
- `score` – only desc (-1) type supported. It works only when facen filter is used.

If `facen_model` is set, and the values in the gallery and request are different, an error will be returned in response.

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 will not be used if any of id or meta filters are set.

- 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/search/testgal' --data '
{
   "limit": 2,
   "sort": {
       "score": -1
   },
   "filter": {
       "facen": {
           "data":
↪"xaGhPRPZODu7p8Q9+YV+PFQbRTxXboY92YCCPW31sT22s5o6FvNMPevNATxNpqA8x8QXPXbZsj2cX5w9rBUsPR0BeD1/
↪4jo8gNIbPY4BXz39jAk9ZxhlPUyvgz0kyqU91ddxPU8YMz3lV109PbjdPMvqyjy3u3881PCrPMKtFDz93LQ9Gi/
↪XO3K1+Dx+0k89NVOqPRZ0ZD24FaY98kKDPZjCkz1k84A9RFFkPdbfpT0hsmw9pL0XPdkuhTxLE589FqpwPRjvgTsZbWA9ix+kPdsk
↪Gfj1L3zA9yV64PYr2ezwiEyw9wwqvPQfllD1C1Fo9JxDEPWUaOD21Nww7QivkPCDMMz3n730904xYPEW9QzyAcO88WjCJPW3lhz0d
↪Dxk7kM9TuxOPet6pD2lqKs9uW8HPRNj+TwuUfc8g9epPdPSTz1NYZA9eQoxPeO2Qz2MlcM95wNyPSb1+jvteb08Rxu5OxP1lz0W7h
↪tgM9Jwc1PS/
↪XVjx49qA9uK+ZPYCaqT38U4k9a7OXOVbDAD3Acas91xIvO5IThz0AncY9Bzy9PRYCjz3AoTY92pktPZQgPj1lXow99AN6PaoMrzzp
↪PbEfSj1OzbU9PzOTPQs/
↪JzxBNL48hoEXPUdHpT1M1g49tUOHPMxETj3sf9Y8r9Ol08kgkTw5Pbo915F6PSMplj3Xxwk8Rin+PKsdjTxBG5A951FOPS/
↪SZT0nDSw9WP8zPQxbDj2dNY89HhxXPPKeFj1OOzs8twc7PXJutTyTLo48ZiuRPUDa0Dy/
↪bJs99t3GPEybjjzri6w885S3PS7Ksj3iacQ9ZQM8PIZdHz2sqH4835U/
↪PLBlUTxP8748776wPTSMpT0qIUs8Maa1PQqWvD2mA3I8d9cWPYqPvz16oic9r1wjPd7wZT0UeTE9KZd3PAJ/
↪tD0v87U6h6+NPXLVHj2TuTE9tEBDPHleDj0gu4A9Jl05PZ9Ihj3WBp89aptbPDTxYTwRnrs9Q71+PTZ/
↪Gz3mVsQ9gx9jPeWYDjzFrYU96J0yPYBAkzr27P48EoufPBMRdT07JCQ9dhM7PR7ZtD0XJKE62+qnPOmVRD16anM9uP70Msoaz1Se
```

(continues on next page)

(continued from previous page)

```
↪Paq8nT09s8U9CSgyPYRQIz0bE0s9KGk2PAHCHz1d4ao9CACmPTLw4zzVOzg6G6y9PeilATxQ/
↪i093pAyPGnfkTzt05s7y3WsPZirOj2uTbE8q7qdPeo6Ez1yD7o9BryOPWb6aDxln609FCLgPJHPQT2QbXg91w/
↪VO4NaaT2bVQE9z8oFPQzF7DywcDM919ZvPaDXez3usK07Xne4PK4yvT2lk+Q8DqrHPASwtj0=",
            "score": [0.75, 1]
        },
        "id": {
            "range": [9223372036854776003, 9223372036854776009]
        },
        "meta": {
            "person_id": {
                "range": [1, 3]
            },
            "tags": {
                "set": ["2", "4"]
            }
        }
    }
}'
```

Response

```
HTTP/1.1 200 Ok
X-request-id: TN:fCQ3wdqQ
X-search-stat: batch_size:1, linearLinear:(idx:1,seen:7);linearIndexed:(idx:0,seen:0);
↪fastIndex:no;
Content-type: application/json
X-read-only: false
Content-length: 3786
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

{
   "facen_size": 320,
   "facen_model": "test",
   "results": [
     {
       "facen":
↪"xaGhPRPZODu7p8Q9+YV+PFQbRTxXboY92YCCPW31sT22s5o6FvNMPevNATxNpqA8x8QXPXbZsj2cX5w9rBUsPR0BeD1/
↪4jo8gNIbPY4BXz39jAk9ZxhlPUyvgz0kyqU91ddxPU8YMz3lV109PbjdPMvqyjy3u3881PCrPMKtFDz93LQ9Gi/
↪XO3K1+Dx+0k89NVOqPRZ0ZD24FaY98kKDPZjCkz1k84A9RFFkPdbfpT0hsmw9pL0XPdkuhTxLE589FqpwPRjvgTsZbWA9ix+kPdsk
↪Gfj1L3zA9yV64PYr2ezwiEyw9wwqvPQfllD1C1Fo9JxDEPWUaOD21Nww7QivkPCDMMz3n730904xYPEW9QzyAcO88WjCJPW3lhz0d
↪Dxk7kM9TuxOPet6pD2lqKs9uW8HPRNj+TwuUfc8g9epPdPSTz1NYZA9eQoxPeO2Qz2MlcM95wNyPSb1+jvteb08Rxu5OxP1lz0W7h
↪tgM9Jwc1PS/
↪XVjx49qA9uK+ZPYCaqT38U4k9a7OXOVbDAD3Acas91xIvO5IThz0AncY9Bzy9PRYCjz3AoTY92pktPZQgPj1lXow99AN6PaoMrzzp
↪PbEfSj1OzbU9PzOTPQs/
↪JzxBNL48hoEXPUdHpT1M1g49tUOHPMxETj3sf9Y8r9Ol08kgkTw5Pbo915F6PSMplj3Xxwk8Rin+PKsdjTxBG5A951FOPS/
↪SZT0nDSw9WP8zPQxbDj2dNY89HhxXPPKeFj1OOzs8twc7PXJutTyTLo48ZiuRPUDa0Dy/
↪bJs99t3GPEybjjzri6w885S3PS7Ksj3iacQ9ZQM8PIZdHz2sqH4835U/
↪PLBlUTxP8748776wPTSMpT0qIUs8Maa1PQqWvD2mA3I8d9cWPYqPvz16oic9r1wjPd7wZT0UeTE9KZd3PAJ/
↪tD0v87U6h6+NPXLVHj2TuTE9tEBDPHleDj0gu4A9Jl05PZ9Ihj3WBp89aptbPDTxYTwRnrs9Q71+PTZ/
```

(continues on next page)

(continued from previous page)

```
↪Gz3mVsQ9gx9jPeWYDjzFrYU96J0yPYBAkzr27P48EoufPBMRdT07JCQ9dhM7PR7ZtD0XJKE62+qnPOmVRD16anM9uP70OMsoaz1Se
↪Paq8nT09s8U9CSgyPYRQIz0bE0s9KGk2PAHCHz1d4ao9CACmPTLw4zzVOzg6G6y9PeilATxQ/
↪i093pAyPGnfkTztO5s7y3WsPZirOj2uTbE8q7qdPeo6Ez1yD7o9BryOPWb6aDxln609FCLgPJHPQT2QbXg91w/
↪VO4NaaT2bVQE9z8oFPQzF7DywcDM919ZvPaDXez3usK07Xne4PK4yvT2lk+Q8DqrHPASwtj0=",
      "meta": {
        "timestamp": 0,
        "photo_hash": "",
        "tags": ["1", "2", "3"],
        "person_id": 1,
        "cam_id": "223901"
      },
      "score": 0.9999,
      "id": 9223372036854776003
    },
    {
      "facen": "W7/+PE/
↪4rzygS7s9DQKYPdCsgDwKnK097YC4PZRjvT0dnwE9EQGRPKqo4DwosNA8dtsFPRcv8zzoAr89auiMPBuiEz19BRw95+p5PV51Sj0G
↪4w9JZsCPYYGRT1NAAc9eaAKOw0UDD1UQcE968mTPaKgxT0j95I9Gw3VPO79kT2PIWI9fKa7PLI9kD3Kup89qeaNPUEkfj0DMRI74F
↪0DzcW4892wkwPbBeuz1pjDI9f7EiPXjV2DzSQak834mHPQj0PD2Ssp89dzuNPICKvT2yOY89RCQcPagt/
↪zzZ+LE9ioUyOy0w8Txb8709t0ZmPSlrxDwk7qo8WkdAPcDH4Dz4XAU9vF9aPU82Jj2yzDY97cPBPXwgQz3Q46g8I7p9PUYMvj2oa1
↪PcFibT02QfE70GeoPYnKqj3y96Q91UAkPYKzij3gsOc8Mw2oPQwdiD2NBqk9hvNcPGFJEj2yZH89mLC5PWYQgD0hTFY8h1irPYbmA
↪MT49wLjkPErkiT1Ha088lf5bPftnajuJ2Yw9AK16Pa51sz3pvCg7oteWOpatoD13KXE9nBI3PX1TOjuuVA89JJE1Otu+JT1QtC09J
↪xD37kQw9g3pJPHGGFj3SAlQ95T6HPYzKtz16jbA9JPifPepCij08biI9OgSCPfMGvT3iKrM8T1YGPXjVrz0HxUc7vLeJPV1wEz0+n
↪Y48SmqaPUZdrzwPYRE9hcTEPGrTiD1w2Ag9Ki1pPSAWMz3E/
↪oE9w1vnPHo0zjuuwIU9MA7mOy+BuDwEo2E9Y2gRPcErOj1qmi89ehOLPSxpvzvn/
↪JU9bGWKPRGzrj3zPIs9sAOhPTYTWzs8Bb07f/
↪ATPABYmD3X61w8tgIUPSTBWz3NqYc86+anPVOuOTzoSJg9ICczPfEHZj24n78946Z0PWsMjTsfFLE9GodDOxFkxT1JpCg9/
↪vIKPQvdOTvJAS89jEKXPZVqfD1NXhw98iu5PQ0oTD271Fs8rLyrPFWbeD1BArg9bQWBPA8+hD148qk8l5WlPbQKgj2+DHw9KMsLPb
↪iY9zuNlPRl1sjy2zeA8MO5pPclxYD16s6o9PwS5PW4vTDykZBw9Y9i7PVn0xjzIGAE98IVfPSzrlj3lPp09dA6VPYaBwTzZE509wc
↪przwiT4w9c5uOPIqUsD0s0yQ8drucPS1RjT30Fqg8lyhSPQVexT2ujqs8FkS9O++yqT2/
↪pLw9xiSaPGlbTj08EnA9GzGYPVwEjD2GNEU98z0vPea/
↪kz0yKx49aMKUPSOD+DyyW6g93ZVtPV8NeT1ZVYY9CUztPIPzvT3s6Bs8m7y6PX5+mz0qAqM7lb8YPUwpUD0=",
      "meta": {
        "timestamp": 0,
        "photo_hash": "",
        "tags": ["2", "4"],
        "person_id": 3,
        "cam_id": "223901"
      },
      "score": 0.8818,
      "id": 9223372036854776005
    }
  ]
}
```

Multi Search

```
POST /v2/faces/search_multi
```

Parameters in body:

JSON-encoded search request with the following fields:

- `galleries`: list of galleries for search.
- `limit`: maximum number of faces in the response.
- `without_facen`: (default=false) do not return facen in response.
- `without_meta`: (default=false) do not return meta in 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`: similar to `like` in SQL requests: only 'aa%', 'aa%', and '%aa%' are supported. Only for strings and set[string]. In the case of set[string], the filter will return result if at least one value meets the filter condition.
  - `ilike`: similar to `like` but case-insensitive, only for strings and set[string].

The options and notes are similar to the search method in one gallery, except:

- JSON-encoded array with galleries search results objects.
- Any galleries objects on response contains array with results likes search method and gallery name.

Returns:

- 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/search_multi' --data '
{
    "galleries": ["testgal", "newgll"],
    "limit": 2,
    "sort": {
        "score": -1
    },
```

(continues on next page)

(continued from previous page)

```
    "filter": {
        "facen": {
            "data":
↪"xaGhPRPZODu7p8Q9+YV+PFQbRTxXboY92YCCPW31sT22s5o6FvNMPevNATxNpqA8x8QXPXbZsj2cX5w9rBUsPR0BeD1/
↪4jo8gNIbPY4BXz39jAk9ZxhlPUyvgz0kyqU91ddxPU8YMz3lV109PbjdPMvqyjy3u3881PCrPMKtFDz93LQ9Gi/
↪XO3K1+Dx+0k89NVOqPRZ0ZD24FaY98kKDPZjCkz1k84A9RFFkPdbfpT0hsmw9pL0XPdkuhTxLE589FqpwPRjvgTsZbWA9ix+kPdsk
↪Gfj1L3zA9yV64PYr2ezwiEyw9wwqvPQfllD1C1Fo9JxDEPWUaOD21Nww7QivkPCDMMz3n730904xYPEW9QzyAcO88WjCJPW3lhz0d
↪Dxk7kM9TuxOPet6pD2lqKs9uW8HPRNj+TwuUfc8g9epPdPSTz1NYZA9eQoxPeO2Qz2MlcM95wNyPSb1+jvteb08Rxu5OxP1lz0W7h
↪tgM9Jwc1PS/
↪XVjx49qA9uK+ZPYCaqT38U4k9a7OXOVbDAD3Acas91xIvO5IThz0AncY9Bzy9PRYCjz3AoTY92pktPZQgPj1lXow99AN6PaoMrzzp
↪PbEfSj1OzbU9PzOTPQs/
↪JzxBNL48hoEXPUdHpT1M1g49tUOHPMxETj3sf9Y8r9Ol08kgkTw5Pbo915F6PSMplj3Xxwk8Rin+PKsdjTxBG5A951FOPS/
↪SZT0nDSw9WP8zPQxbDj2dNY89HhxXPPKeFj1OOzs8twc7PXJutTyTLo48ZiuRPUDa0Dy/
↪bJs99t3GPEybjjzri6w885S3PS7Ksj3iacQ9ZQM8PIZdHz2sqH4835U/
↪PLBlUTxP8748776wPTSMpT0qIUs8Maa1PQqWvD2mA3I8d9cWPYqPvz16oic9r1wjPd7wZT0UeTE9KZd3PAJ/
↪tD0v87U6h6+NPXLVHj2TuTE9tEBDPHleDj0gu4A9Jl05PZ9Ihj3WBp89aptbPDTxYTwRnrs9Q71+PTZ/
↪Gz3mVsQ9gx9jPeWYDjzFrYU96J0yPYBAkzr27P48EoufPBMRdT07JCQ9dhM7PR7ZtD0XJKE62+qnPOmVRD16anM9uP70OMsoaz1Se
↪Paq8nT09s8U9CSgyPYRQIz0bE0s9KGk2PAHCHz1d4ao9CACmPTLw4zzVOzg6G6y9PeilATxQ/
↪i093pAyPGnfkTzt05s7y3WsPZirOj2uTbE8q7qdPeo6Ez1yD7o9BryOPWb6aDxln609FCLgPJHPQT2QbXg91w/
↪VO4NaaT2bVQE9z8oFPQzF7DywcDM919ZvPaDXez3usK07Xne4PK4yvT2lk+Q8DqrHPASwtj0=",
            "score": [0.75, 1]
        },
        "id": {
            "range": [9223372036854776003, 9223372036854776009]
        },
        "meta": {
            "person_id": {
                "range": [1, 3]
            },
            "tags": {
                "set": ["2", "4"]
            }
        }
    }
}'
```

Response

```
HTTP/1.1 200 Ok
X-request-id: TN:c4atIug6
Content-type: application/json
X-read-only: false
Content-length: 7610
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

[
   {
     "gallery": "testgal",
     "facen_size": 320,
```

(continues on next page)

(continued from previous page)

```
    "facen_model": "test",
    "results": [
      {
        "facen":
↪"xaGhPRPZODu7p8Q9+YV+PFQbRTxXboY92YCCPW31sT22s5o6FvNMPevNATxNpqA8x8QXPXbZsj2cX5w9rBUsPR0BeD1/
↪4jo8gNIbPY4BXz39jAk9ZxhlPUyvgz0kyqU91ddxPU8YMz3lV109PbjdPMvqyjy3u3881PCrPMKtFDz93LQ9Gi/
↪XO3K1+Dx+0k89NVOqPRZ0ZD24FaY98kKDPZjCkz1k84A9RFFkPdbfpT0hsmw9pL0XPdkuhTxLE589FqpwPRjvgTsZbWA9ix+kPdsk
↪Gfj1L3zA9yV64PYr2ezwiEyw9wwqvPQfllD1C1Fo9JxDEPWUaOD21Nww7QivkPCDMMz3n730904xYPEW9QzyAcO88WjCJPW3lhz0d
↪Dxk7kM9TuxOPet6pD2lqKs9uW8HPRNj+TwuUfc8g9epPdPSTz1NYZA9eQoxPeO2Qz2MlcM95wNyPSb1+jvteb08Rxu5OxP1lz0W7h
↪tgM9Jwc1PS/
↪XVjx49qA9uK+ZPYCaqT38U4k9a7OXOVbDAD3Acas91xIvO5IThz0AncY9Bzy9PRYCjz3AoTY92pktPZQgPj1lXow99AN6PaoMrzzp
↪PbEfSj1OzbU9PzOTPQs/
↪JzxBNL48hoEXPUdHpT1M1g49tUOHPMxETj3sf9Y8r9Ol08kgkTw5Pbo915F6PSMplj3Xxwk8Rin+PKsdjTxBG5A951FOPS/
↪SZT0nDSw9WP8zPQxbDj2dNY89HhxXPPKeFj1OOzs8twc7PXJutTyTLo48ZiuRPUDa0Dy/
↪bJs99t3GPEybjjzri6w885S3PS7Ksj3iacQ9ZQM8PIZdHz2sqH4835U/
↪PLBlUTxP8748776wPTSMpT0qIUs8Maa1PQqWvD2mA3I8d9cWPYqPvz16oic9r1wjPd7wZT0UeTE9KZd3PAJ/
↪tD0v87U6h6+NPXLVHj2TuTE9tEBDPHleDj0gu4A9Jl05PZ9Ihj3WBp89aptbPDTxYTwRnrs9Q71+PTZ/
↪Gz3mVsQ9gx9jPeWYDjzFrYU96J0yPYBAkzr27P48EoufPBMRdT07JCQ9dhM7PR7ZtD0XJKE62+qnPOmVRD16anM9uP70OMsoaz1Se
↪Paq8nT09s8U9CSgyPYRQIz0bE0s9KGk2PAHCHz1d4ao9CACmPTLw4zzVOzg6G6y9PeilATxQ/
↪i093pAyPGnfkTzt05s7y3WsPZirOj2uTbE8q7qdPeo6Ez1yD7o9BryOPWb6aDxln609FCLgPJHPQT2QbXg91w/
↪VO4NaaT2bVQE9z8oFPQzF7DywcDM919ZvPaDXez3usK07Xne4PK4yvT2lk+Q8DqrHPASwtj0=",
        "meta": {
          "timestamp": 0,
          "photo_hash": "",
          "tags": [
            "1",
            "2",
            "3"
          ],
          "person_id": 1,
          "cam_id": "223901"
        },
        "score": 0.9999,
        "id": 9223372036854776003
      },
      {
        "facen": "W7/+PE/
↪4rzygS7s9DQKYPdCsgDwKnK097YC4PZRjvT0dnwE9EQGRPKqo4DwosNA8dtsFPRcv8zzoAr89auiMPBuiEz19BRw95+p5PV51Sj0G
↪4w9JZsCPYYGRT1NAAc9eaAKOw0UDD1UQcE968mTPaKgxT0j95I9Gw3VPO79kT2PIWI9fKa7PLI9kD3Kup89qeaNPUEkfj0DMRI74F
↪0DzcW4892wkwPbBeuz1pjDI9f7EiPXjV2DzSQak834mHPQj0PD2Ssp89dzuNPICKvT2yOY89RCQcPagt/
↪zzZ+LE9ioUyOy0w8Txb8709t0ZmPSlrxDwk7qo8WkdAPcDH4Dz4XAU9vF9aPU82Jj2yzDY97cPBPXwgQz3Q46g8I7p9PUYMvj2oa1
↪PcFibT02QfE70GeoPYnKqj3y96Q91UAkPYKzij3gsOc8Mw2oPQwdiD2NBqk9hvNcPGFJEj2yZH89mLC5PWYQgD0hTFY8h1irPYbmA
↪MT49wLjkPErkiT1Ha088lf5bPftnajuJ2Yw9AK16Pa51sz3pvCg7oteWOpatoD13KXE9nBI3PX1TOjuuVA89JJE1Otu+JT1QtC09J
↪xD37kQw9g3pJPHGGFj3SAlQ95T6HPYzKtz16jbA9JPifPepCij08biI9OgSCPfMGvT3iKrM8T1YGPXjVrz0HxUc7vLeJPV1wEz0+n
↪Y48SmqaPUZdrzwPYRE9hcTEPGrTiD1w2Ag9Ki1pPSAWMz3E/
↪oE9w1vnPHo0zjuuwIU9MA7mOy+BuDwEo2E9Y2gRPcErOj1qmi89ehOLPSxpvzvn/
↪JU9bGWKPRGzrj3zPIs9sAOhPTYTWzs8Bb07f/
↪ATPABYmD3X61w8tgIUPSTBWz3NqYc86+anPVOuOTzoSJg9ICczPfEHZj24n78946Z0PWsMjTsfFLE9GodDOxFkxT1JpCg9/
↪vIKPQvdOTvJAS89jEKXPZVqfD1NXhw98iu5PQ0oTD271Fs8rLyrPFWbeD1BArg9bQWBPA8+hD148qk815WlPbQKgj2+DHw9KMsLPb
↪iY9zuNlPRl1sjy2zeA8MO5pPclxYD16s6o9PwS5PW4vTDykZBw9Y9i7PVn0xjzIGAE98IVfPSzrlj3lPp09dA6VPYaBwTzZE509wc
↪przwiT4w9c5uOPIqUsD0s0yQ8drucPS1RjT30Fqg8lyhSPQVexT2ujqs8FkS9O++yqT2/
↪pLw9xiSaPGlbTj08EnA9GzGYPVwEjD2GNEU98z0vPea/
```

(continues on next page)

(continued from previous page)

```
↪kz0yKx49aMKUPSOD+DyyW6g93ZVtPV8NeT1ZVYY9CUztPIPzvT3s6Bs8m7y6PX5+mz0qAqM7lb8YPUwpUD0=",
          "meta": {
            "timestamp": 0,
            "photo_hash": "",
            "tags": ["2", "4"],
            "person_id": 3,
            "cam_id": "223901"
          },
          "score": 0.8818,
          "id": 9223372036854776005
        }
      ]
    },
    {
      "gallery": "newgll",
      "facen_size": 320,
      "facen_model": "",
      "results": [
        {
          "facen":
↪"xaGhPRPZODu7p8Q9+YV+PFQbRTxXboY92YCCPW31sT22s5o6FvNMPevNATxNpqA8x8QXPXbZsj2cX5w9rBUsPR0BeD1/
↪4jo8gNIbPY4BXz39jAk9ZxhlPUyvgz0kyqU91ddxPU8YMz3lV109PbjdPMvqyjy3u3881PCrPMKtFDz93LQ9Gi/
↪XO3K1+Dx+0k89NVOqPRZ0ZD24FaY98kKDPZjCkz1k84A9RFFkPdbfpT0hsmw9pL0XPdkuhTxLE589FqpwPRjvgTsZbWA9ix+kPdsk
↪Gfj1L3zA9yV64PYr2ezwiEyw9wwqvPQfllD1C1Fo9JxDEPWUaOD21Nww7QivkPCDMMz3n730904xYPEW9QzyAcO88WjCJPW3lhz0d
↪Dxk7kM9TuxOPet6pD2lqKs9uW8HPRNj+TwuUfc8g9epPdPSTz1NYZA9eQoxPeO2Qz2MlcM95wNyPSb1+jvteb08Rxu5OxP1lz0W7h
↪tgM9Jwc1PS/
↪XVjx49qA9uK+ZPYCaqT38U4k9a7OXOVbDAD3Acas91xIvO5IThz0AncY9Bzy9PRYCjz3AoTY92pktPZQgPj1lXow99AN6PaoMrzzp
↪PbEfSj1OzbU9PzOTPQs/
↪JzxBNL48hoEXPUdHpT1M1g49tUOHPMxETj3sf9Y8r9Ol08kgkTw5Pbo915F6PSMplj3Xxwk8Rin+PKsdjTxBG5A951FOPS/
↪SZT0nDSw9WP8zPQxbDj2dNY89HhxXPPKeFj1OOzs8twc7PXJutTyTLo48ZiuRPUDa0Dy/
↪bJs99t3GPEybjjzri6w885S3PS7Ksj3iacQ9ZQM8PIZdHz2sqH4835U/
↪PLBlUTxP8748776wPTSMpT0qIUs8Maa1PQqWvD2mA3I8d9cWPYqPvz16oic9r1wjPd7wZT0UeTE9KZd3PAJ/
↪tD0v87U6h6+NPXLVHj2TuTE9tEBDPHleDj0gu4A9Jl05PZ9Ihj3WBp89aptbPDTxYTwRnrs9Q71+PTZ/
↪Gz3mVsQ9gx9jPeWYDjzFrYU96J0yPYBAkzr27P48EoufPBMRdT07JCQ9dhM7PR7ZtD0XJKE62+qnPOmVRD16anM9uP70OMsoaz1Se
↪Paq8nT09s8U9CSgyPYRQIz0bE0s9KGk2PAHCHz1d4ao9CACmPTLw4zzVOzg6G6y9PeilATxQ/
↪i093pAyPGnfkTzt05s7y3WsPZirOj2uTbE8q7qdPeo6Ez1yD7o9BryOPWb6aDxln609FCLgPJHPQT2QbXg91w/
↪VO4NaaT2bVQE9z8oFPQzF7DywcDM919ZvPaDXez3usK07Xne4PK4yvT2lk+Q8DqrHPASwtj0=",
          "meta": {
            "timestamp": 0,
            "photo_hash": "",
            "tags": ["1", "2", "3"],
            "person_id": 1,
            "cam_id": "223901"
          },
          "score": 0.9999,
          "id": 9223372036854776003
        },
        {
          "facen": "W7/+PE/
↪4rzygS7s9DQKYPdCsgDwKnK097YC4PZRjvT0dnwE9EQGRPKqo4DwosNA8dtsFPRcv8zzoAr89auiMPBuiEz19BRw95+p5PV51Sj0G
↪4w9JZsCPYYGRT1NAAc9eaAKOw0UDD1UQcE968mTPaKgxT0j95I9Gw3VPO79kT2PIWI9fKa7PLI9kD3Kup89qeaNPUEkfj0DMRI74F
↪0DzcW4892wkwPbBeuz1pjDI9f7EiPXjV2DzSQak834mHPQj0PD2Ssp89dzuNPICKvT2yOY89RCQcPagt/
```

(continues on next page)

(continued from previous page)

```
↪zzZ+LE9ioUyOy0w8Txb8709t0ZmPSlrxDwk7qo8WkdAPcDH4Dz4XAU9vF9aPU82Jj2yzDY97cPBPXwgQz3Q46g8I7p9PUYMvj2oa1A
↪PcFibT02QfE70GeoPYnKqj3y96Q91UAkPYKzij3gsOc8Mw2oPQwdiD2NBqk9hvNcPGFJEj2yZH89mLC5PWYQgD0hTFY8h1irPYbmAT
↪MT49wLjkPErkiT1Ha088lf5bPftnajuJ2Yw9AK16Pa51sz3pvCg7oteWOpatoD13KXE9nBI3PX1TOjuuVA89JJE1Otu+JT1QtC09Jl
↪xD37kQw9g3pJPHGGFj3SAlQ95T6HPYzKtz16jbA9JPifPepCij08biI9OgSCPfMGvT3iKrM8T1YGPXjVrz0HxUc7vLeJPV1wEz0+nc
↪Y48SmqaPUZdrzwPYRE9hcTEPGrTiD1w2Ag9Ki1pPSAWMz3E/
↪oE9w1vnPHo0zjuuwIU9MA7mOy+BuDwEo2E9Y2gRPcErOj1qmi89ehOLPSxpvzvn/
↪JU9bGWKPRGzrj3zPIs9sAOhPTYTWzs8Bb07f/
↪ATPABYmD3X61w8tgIUPSTBWz3NqYc86+anPVOuOTzoSJg9ICczPfEHZj24n78946Z0PWsMjTsfFLE9GodDOxFkxT1JpCg9/
↪vIKPQvdOTvJAS89jEKXPZVqfD1NXhw98iu5PQ0oTD271Fs8rLyrPFWbeD1BArg9bQWBPA8+hD148qk8l5WlPbQKgj2+DHw9KMsLPby
↪iY9zuNlPRl1sjy2zeA8MO5pPclxYD16s6o9PwS5PW4vTDykZBw9Y9i7PVn0xjzIGAE98IVfPSzrlj3lPp09dA6VPYaBwTzZE509wc
↪przwiT4w9c5uOPIqUsD0s0yQ8drucPS1RjT30Fqg8lyhSPQVexT2ujqs8FkS9O++yqT2/
↪pLw9xiSaPGlbTj08EnA9GzGYPVwEjD2GNEU98z0vPea/
↪kz0yKx49aMKUPSOD+DyyW6g93ZVtPV8NeT1ZVYY9CUztPIPzvT3s6Bs8m7y6PX5+mz0qAqM7lb8YPUwpUD0=",
        "meta": {
          "timestamp": 0,
          "photo_hash": "",
          "tags": ["1","4"],
          "person_id": 3,
          "cam_id": "223901"
        },
        "score": 0.8818,
        "id": 9223372036854776005
      }
    ]
  }
]
```

Edit Face Metadata and/or Feature Vector

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

Parameters in path segments:

- `:name`: gallery 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 a list of update results on success:**
  - a dictionary with all face parameters, including not updated, on success.
  - `{error: {code: ..., ...}}` if there is an error.
- HTTP 404 and an error description if a face with the given id doesn't exist.
- HTTP with a status other than 200 and an error description in the body on failure.

Example

Request

```
curl -D - -s 'http://localhost:8001/v2/faces/update/testgal' --data '
[
   {
         "id": 9223372036854776002,
         "facen":
↪"sy2BPb3xrz1ffbY9QJ0sPTlpAD2D+5A91tgePCIdGDwjSfQ8SupkPNMv2jwS5o48BvHNPETWWD3qSFo6zDp6PaiDpT3FHjA9lK+y
↪SKTvPP8s8E4v2PB6lnz2A7SQ9UK0gPRTxLz3Ukrg991C7PfrsZz0thTs9BwQQPVwvuj3/iAM8/
↪TaLPHg4pjzwvy49jwzTPHk/
↪pz2QpTc9lAelPLV3Az3WJgI9sT2gPXdgwD0oKoc9mzaxPTe52TweKZo9+GW6PThnYz3i4HM9SgQrPVq4qD0CUQI8JrmfPcXoMD3vX
↪BXCPHMSrj1JNDw9nze5PRbYtT2xN5w9Ms0uPWWd5jwLixg9LLCBPb2sxzyLa4E8xpumPeM8gTzcuos8KdG+PctOWT2LmMo8ot58PV
↪VGhPUzHBz2GTtg81YT8PKAtjzsmPbM9jkebPRSWBD01ADQ9Pl1MPYRbkT0A2ME97T8fPULMoT1j7oc9FE0sOq71rj2VGz49gewcPe
↪F89WeurPW3FUjxOtQE9uz6NPdhKCT2vlII9xwETPQbQED0Uubg93vyrPXmgrTy/
↪HmM92vPCPcEdzjtPgTo86vY3PQpupT0gM389L7EUPQtfaTtljrA9fqGDPem41TvlMKc8PjV7PZs8hj2KJLg9RiX2PPopvT12GoM9v
↪ngj0fhzI9WPVDPZSnPT0PeuI7M/
↪C3PUemcj0HKOY8cIvDPYkwQT0Bx148YfvcOzgBXj1V19Y84oG6PfzCvT2baFs9MLy6PVfA+zzsTY48S6uoPUOoHz0ERhE8f4JgPa/
↪Skjwh94g9RiQ6PWJLXz1CCKo9SlTKPAQwDz2UKPw8BAH+PPjPFj1Sm6w9uRVHPbwX+TxMaF491OuxPd0CnT0KYo09DhsxPU00wTzu
↪qMs8M16JPLp7xD3EHh89BHyPPdSsmD2qcnA9S2GlPEBMoT2Ih7s9gyYOPfG2sDzLlP88lNx1PTA3mz0Befg8gryUPbBpyDsesqE9l
↪PfkQGz00sl09kZewOnUuBzsgm/I8oYojOi/
↪d7jxmrzk8G2ITPfxqiT3OqyY9WwRqPZ4OWT0BtI0983dOPeGxgD0CECA928C1PLbbujx8f2U8WpwSPfjElD3Hq6s94BRgPcsTuTx6
↪",
         "meta": {
           "timestamp": 350
         }
   }
]'
```

Response

```
HTTP/1.1 200 Ok
X-request-id: TN:hsMZrt0g
Content-type: application/json
X-read-only: false
Content-length: 1847
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

[{"meta":{"timestamp":350,"photo_hash":"","tags":[],"person_id":123,"cam_id":"223900"},
```

(continues on next page)

(continued from previous page)

```
↪"id":9223372036854776002,"facen":
↪"sy2BPb3xrz1ffbY9QJ0sPTlpAD2D+5A91tgePCIdGDwjSfQ8SupkPNMv2jwS5o48BvHNPETWWD3qSFo6zDp6PaiDpT3FHjA91K+y
↪/SKTvPP8s8E4v2PB6lnz2A7SQ9UK0gPRTxLz3Ukrg991C7PfrsZz0thTs9BwQQPVwvuj3\/iAM8\/
↪TaLPHg4pjzwvy49jwzTPHk\/
↪pz2QpTc9lAelPLV3Az3WJgI9sT2gPXdgwD0oKoc9mzaxPTe52TweKZo9+GW6PThnYz3i4HM9SgQrPVq4qD0CUQI8JrmfPcXoMD3vX
↪/
↪BXCPHMSrj1JNDw9nze5PRbYtT2xN5w9Ms0uPWWd5jwLixg9LLCBPb2sxzyLa4E8xpumPeM8gTzcuos8KdG+PctOWT2LmMo8ot58PV
↪/
↪VGhPUzHBz2GTtg81YT8PKAtjzsmPbM9jkebPRSWBD01ADQ9Pl1MPYRbkT0A2ME97T8fPULMoT1j7oc9FE0sOq71rj2VGz49gewcPe
↪/F89WeurPW3FUjxOtQE9uz6NPdhKCT2vlII9xwETPQbQED0Uubg93vyrPXmgrTy\/
↪HmM92vPCPcEdzjtPgTo86vY3PQpupT0gM389L7EUPQtfaTtljrA9fqGDPem41TvlMKc8PjV7PZs8hj2KJLg9RiX2PPopvT12GoM9v
↪/ngj0fhzI9WPVDPZSnPT0PeuI7M\/
↪C3PUemcj0HKOY8cIvDPYkwQT0Bx148YfvcOzgBXj1V19Y84oG6PfzCvT2baFs9MLy6PVfA+zzsTY48S6uoPUOoHz0ERhE8f4JgPa\
↪/
↪Skjwh94g9RiQ6PWJLXz1CCKo9SlTKPAQwDz2UKPw8BAH+PPjPFj1Sm6w9uRVHPbwX+TxMaF491OuxPd0CnT0KYo09DhsxPU00wTzu
↪/
↪qMs8M16JPLp7xD3EHh89BHyPPdSsmD2qcnA9S2GlPEBMoT2Ih7s9gyYOPfG2sDzLlP88lNx1PTA3mz0Befg8gryUPbBpyDsesqE91
↪/PfkQGz00sl09kZewOnUuBzsgm\/I8oYojOi\/
↪d7jxmrzk8G2ITPfxqiT3OqyY9WwRqPZ4OWT0BtI0983dOPeGxgD0CECA928C1PLbbujx8f2U8WpwSPfjElD3Hq6s94BRgPcsTuTx6
↪"}]
```

List Galleries

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

Returns:

JSON-encoded array with galleries with the following fields:

- `space`: space name,
- `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
X-request-id: TN:6iRqT5FR
Content-type: application/json
X-read-only: false
Content-length: 60
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

{"results":[{"space":"default","name":"testgal","faces":0}]}
```

Get Gallery Info

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

Parameters in path segments:

- `:name`: gallery name.

Returns:

- HTTP 200 and a 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
```

Response

```
HTTP/1.1 200 Ok
X-request-id: TN:rPLOWTbi
Content-type: application/json
X-read-only: false
Content-length: 49
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

{"facen_size":320,"faces":2,"facen_model":"test"}
```

Create Gallery

```
POST /v2/galleries/add/:name[?space=default]
```

Parameters in path segments:

- `:name`: gallery name.
- `"space"`: space name from schema configuration, string. Non-required. Default value: default.

Parameters in body:

JSON-encoded object with faces with the following fields:

- `"live_idx"`: live index settings, object. See live index settings for setup value.
- `"space"`: space name from schema configuration, string.

Returns:

- HTTP 201 and empty body on success.
- HTTP with a status other than 201 and error description in the body on failure.

Example

Request

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

Response

```
HTTP/1.1 201 Created
X-request-id: TN:BCdY11F4
Content-type: application/json
X-read-only: false
Content-length: 4
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

null
```

Remove Gallery

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

Parameters in path segments:

- `:name`: gallery name to be deleted.

Returns:

- HTTP 204 and empty on success.
- HTTP with a status other than 204 and an error description in the body on failure.

Example

Request

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

Response

```
HTTP/1.1 204 No content
X-request-id: TN:a5HYFcKK
Content-type: application/json
X-read-only: false
Content-length: 4
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)
```

Rename Gallery

```
POST /v2/galleries/rename/:name/:newname
```

Parameters in path segments:

- `:name`: gallery name.
- `:newname`: new gallery name.

Returns:

- HTTP 204 and empty on success.
- HTTP with a status other than 204 and an error description in the body on failure.

Example

Request

```
curl -D - -X POST -s 'http://localhost:8001/v2/galleries/rename/testgal/newgal'
```

Response

```
HTTP/1.1 204 No content
X-request-id: TN:a5HYFcKK
Content-type: application/json
X-read-only: false
Content-length: 4
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)
```

1.5.5 Replication settings `tntapi`

In this section:

Simple master->slaves replication

Replication allows multiple tntapi instances to work on copies of the same 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 start a created replica for the first time, do the following:

1. Start the master instance.

```
docker run -tid --name tnt-1-1 --restart always --network server \
    --env CFG_LISTEN_HOST=0.0.0.0 \
    --env CFG_LISTEN_PORT=8001 \
    --env CFG_NTLS=ntls:3133 \
    --env TT_LISTEN=0.0.0.0:32001 \
    --env TT_MEMTX_MEMORY=$((1024 * 1024 * 1024)) \
    --volume /opt/ffserver/tnt/001-01:/opt/ntech/var/lib/tarantool/default \
    --publish 127.0.0.1:8001:8001 \
    docker.int.ntl/ntech/universe/tntapi:ffserver-11.240325
```

2. In the replica container environments, specify the IP address and the listening port of the master instance.

```
TT_REPLICATION=tnt-1-1:32001 # master address in TT_LISTEN
TT_READ_ONLY=true
```

3. Create a directory for slave snapshots and xlogs.

```
sudo mkdir -p /opt/ffserver/tnt/001-02/{snapshots,xlogs}
```

4. Copy the latest snapshot of the master instance into the `/opt/ffserver/tnt/001-02/snapshots` directory of the replica.

5. Copy the latest xlog of the master instance into the `/opt/ffserver/tnt/001-02/xlogs` directory of the replica.

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

```
docker run -tid --name tnt-1-2 --restart always --network server \
    --env CFG_LISTEN_HOST=0.0.0.0 \
    --env CFG_LISTEN_PORT=8101 \
    --env CFG_NTLS=ntls:3133 \
    --env TT_LISTEN=0.0.0.0:32101 \
    --env TT_REPLICATION=tnt-1-1:32001 \
    --env TT_READ_ONLY=true \
    --env TT_MEMTX_MEMORY=$((1024 * 1024 * 1024)) \
    --volume /opt/ffserver/tnt/001-02:/opt/ntech/var/lib/tarantool/default \
    --publish 127.0.0.1:8101:8101 \
    docker.int.ntl/ntech/universe/tntapi:ffserver-11.240325
```

7. Check the status of the nodes.

```
curl -i http://localhost:8001/v2/status
```

```
HTTP/1.1 200 Ok
X-request-id: TN:xfKtuMxP
Content-type: application/json
X-read-only: false
Content-length: 19
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

{"read_only":false}
```

```
curl -i http://localhost:8101/v2/status
```

```
HTTP/1.1 200 Ok
X-request-id: TN:1ZMBu6Uf
Content-type: application/json
X-read-only: true
Content-length: 18
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

{"read_only":true}
```

Important: Live index does not work on replicas.

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

1.5.6 Direct API requests to `deduplicator`

You can use HTTP API to extract data directly from the `deduplicator` component. The `deduplicator` checks EmbeN against a new or existing deduplication group.

In this section:

General Information

The `deduplicator` component accepts requests to `http://<deduplicator_ip>:18310/`.

If there is no group with this ID, or if the parameters differ – then a group is (re)created with the provided parameters (status code 201).

If there is already a group with the same ID and the same parameters, EmbeN is probed against it (status code 200).

If a match is found in the group, this method returns `unique=false` and the `meta` object that was supplied with the matching EmbeN when we first saw it. The matching EmbeN in the deduplication group will have its timestamp updated, current vector and its metadata will be discarded.

If no match is found, this method returns `unique=true` and no `meta`. Current EmbeN and its metadata will be stored in the deduplication group.

API v1

```
POST /v1/groups/{group_id}/probe
```

This is the main API of `deduplicator`. It accepts an EmbeN, deduplication group parameters and EmbeN metadata.

Parameters in path segments:

- `group_id`: ID of the group to probe against, string.

Request body:

- application/json: the request body contains only JSON.

```
{
  "emben": "string",
  "params": {
    "deduplication_period": 30,
    "deduplication_threshold": 0.75
  },
  "meta": 123
}
```

- emben: EmbeN, string. Required parameter.
- params:
  - deduplication_period: deduplication period in seconds, integer. Required parameter.
  - deduplication_threshold: deduplication similarity threshold, number($float). Required parameter.
- meta: any JSON value describing this EmbeN.

Returns:

EmbeN successfully probed against existing group on success.

```
{
  "unique": false,
  "matched_meta": 123
}
```

Common Codes

Code	Description
201	Created new group or reconfigured existing.
400	Invalid request or parameters. BAD_PARAM.

1.5.7 Direct API requests to counter

You can use HTTP API to extract data directly from the counter component. The counter component accepts requests to http://<counter_ip>:18300/.

In this section:

Methods with Counters by ID

Get a counter by ID

```
GET /v1/counter/:counter_id
```

This method retrieves an existing counter.

Parameters in path segments:

- `counter_id`: ID of the counter, string.

Returns:

- JSON representation of the counter.

Example

Request

```
curl -i -X GET http://127.0.0.1:18300/v1/counter/test_counter
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:KGNbSuJE
Date: Mon, 02 Dec 2019 13:45:59 GMT
Content-Length: 213

{
    "id": "test_counter",
    "name": "test counter name",
    "deduplication_period": 10,
    "deduplication_threshold": 0.75,
    "deduplication_only": false,
    "aggregate_by_labels": [
        "gender",
        "gate",
        "foo",
        "bar"
    ],
    "meta": "{\"meta_label\":1}"
}
```

Create a new counter

```
POST /v1/counter/:counter_id
```

This method creates a counter under a given name.

Parameters in path segments:

- `counter_id`: ID of the counter, string.

JSON parameters in request:

- `name` (`required=false`): the name of the new counter.
- `deduplication_period` (`required=false`, `default=10`): the lifetime of events in the `inmemory` queue, which is used for face deduplication.
- `deduplication_threshold` (`required=false`, `default=0.75`): confidence to deduplicate events.
- `deduplication_only` (`required=false`, `default=False`): the counter will be used only for deduplication, not to create tables with event statistics data.
- `aggregate_by_labels` `required=false`: array of strings, labels that the counter will use for counting and aggregation. A separate column with an index in the database will be created for each label.
- `meta` (`required=false`): a json string with additional parameters that are stored in the json field.

Example

Request

```
curl -i -X POST \
  http://127.0.0.1:18300/v1/counter/test_counter \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "test counter name",
    "deduplication_period": 10,
    "deduplication_threshold": 0.75,
    "deduplication_only": false,
    "aggregate_by_labels": ["gender", "gate", "foo", "bar"],
    "meta": "{\"meta_label\":1}"
}'
```

Response

```
HTTP/1.1 201 Created
Content-Type: application/json
X-Request-Id: FC:fQ6pYSQ3
Date: Mon, 02 Dec 2019 13:33:20 GMT
Content-Length: 213

{
    "id": "test_counter",
    "name": "test counter name",
    "deduplication_period": 10,
    "deduplication_threshold": 0.75,
    "deduplication_only": false,
    "aggregate_by_labels": [
        "gender",
        "gate",
        "foo",
        "bar"
    ],
    "meta": "{}"
}
```

Update an existing counter

```
PATCH /v1/counter/:counter_id
```

Parameters in path segments:

- `counter_id`: ID of the counter, string.

This method allows you to edit an existing counter. The editable fields are: `name`, `deduplication_period`, `deduplication_threshold`, `meta`.

Example

Request

```
curl -i -X PATCH \
  http://127.0.0.1:18300/v1/counter/test_counter \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "new counter name",
    "deduplication_threshold": 0.8
}'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:h9h2fhfU
Date: Mon, 02 Dec 2019 14:08:07 GMT
Content-Length: 211

{
    "id": "test_counter",
    "name": "new counter name",
    "deduplication_period": 10,
    "deduplication_threshold": 0.8,
    "deduplication_only": false,
    "aggregate_by_labels": [
        "gender",
        "gate",
        "foo",
        "bar"
    ],
    "meta": "{\"meta_label\":1}"
}
```

Delete a counter

```
DELETE /v1/counter/:counter_id
```

This method deletes the counter.

Parameters in path segments:

- `counter_id`: ID of the counter, string.

Example

Request

```
curl -i -X DELETE http://127.0.0.1:18300/v1/counter/test_counter
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:VuS9jZ8u
Date: Mon, 02 Dec 2019 14:09:36 GMT
Content-Length: 211

{
    "id": "test_counter",
    "name": "test counter name",
    "deduplication_period": 10,
    "deduplication_threshold": 0.75,
    "deduplication_only": false,
    "aggregate_by_labels": [
        "gender",
        "gate",
        "foo",
        "bar"
    ],
    "meta": "{\"meta_label\":1}"
}
```

List counters

```
GET /v1/counter/
```

This method returns a list of all counters.

Example

Request

```
curl -i -X GET http://127.0.0.1:18300/v1/counter/
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:KGNbSuJE
Date: Mon, 02 Dec 2019 14:12:18 GMT
Content-Length: 413

{
    "counters": [
        {
            "id": "test_counter_1",
            "name": "test_counter",
            "deduplication_period": 5,
            "deduplication_threshold": 2,
            "deduplication_only": false,
            "aggregate_by_labels": [
                "foo",
                "gate",
                "gender",
                "age",
                "bar"
            ],
            "meta": "{}"
        },
        {
            "id": "test_counter",
            "name": "test counter name",
            "deduplication_period": 10,
            "deduplication_threshold": 0.75,
            "deduplication_only": false,
            "aggregate_by_labels": [
                "gender",
                "gate",
                "foo",
                "bar"
            ],
            "meta": "{\"meta_label\":1}"
        }
    ]
}
```

Clear the queues of the counter

```
POST /v1/counter/:counter_id/clean-queue
```

This method clears the queues of the counter.

There are no parameters, returns `200 OK`.

Example

Request

```
curl -i -X POST http://127.0.0.1:18300/v1/counter/test_counter/clean-queue
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:VbhV3vC5
Date: Mon, 01 Aug 2022 19:53:29 GMT
Content-Length: 0
```

Post a face to a counter

```
POST /v1/counter/test_counter/add
```

This method posts a face to a counter.

JSON parameters in request:

- `face_id` (`required=true`): integer, ID of the face.
- `facen` (`required=true`): base64-encoded feature vector. If it is empty or not specified, the row in the statistics is saved without deduplication by face.
- `timestamp` (`required=true`): the date of the event in RFC 3339 format (always UTC, regardless of the host timezone and transmitted in the request).
- `labels` (`required=false`): a dictionary of string labels that correspond to the event.
- `weight` (`required=false`): a weight of the face. If it exceeds 1, this face will be counted not as one, but as a `weight` of faces in the `count` output.
- `topic` (`required=false, default="default"`): a label for distinguishing deduplications within a single counter. If specified, it indicates a separate named queue.

Example

Request

```
curl -X POST \
  http://127.0.0.1:18300/v1/counter/test_counter/add \
  -H 'Content-Type: application/json' \
  -d '{
    "face_id": 4226310056074757343,
    "facen": "5nWCPZwO9Lxl5zC9+6O3PG8U47twtQS9qwnVvJXo1zw/c ...",
    "timestamp": "2019-11-22T14:50:36+03:00",
    "labels": {"gender": "female", "gate": null, "foo": "f", "bar": null}
}'
```

Returns

JSON field:

- unique: whether this face is unique or not. The search for duplicates takes place in the inmemory queue with the deduplication_threshold confidence, where the lifetime of the event in queue is the deduplication_period.

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:AAwdCxmM
Date: Mon, 02 Dec 2019 14:15:13 GMT
Content-Length: 16

{
    "unique": true
}
```

Count faces with a given grouping

```
GET /v1/counter/test_counter/count
```

This method counts faces with a given grouping.

Query string parameters:

- from (required=false): the start time of counting in RFC 3339 format.
- till (required=false): the end time of counting in RFC 3339 format.
- labels (required=true): a list of labels (separated by commas &labels=gender,age) by which the results are grouped. The available labels are set in the theaggregate_by_labels field of the counter. It is not necessary to pass the full list of aggregate_by_labels every time. Labels that are missing in aggregate_by_labels will be ignored.

Example

Request

```
curl -i -X GET \
 'http://127.0.0.1:18300/v1/counter/test_counter/count?from=2019-11-20T00%3A57%3A21.
↪751812%2B03%3A00&labels=gender%2Cage'
```

The response contains a list of results for different combinations of the requested labels. The fields with "" mean that the event doesn't have such label.

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:wRC5UHK2
Date: Mon, 02 Dec 2019 14:46:45 GMT
Content-Length: 257

{
    "count": [
        {
            "count": 37,
            "labels": {
                "age": "",
                "gender": ""
            }
        },
        {
            "count": 2,
            "labels": {
                "age": "27",
                "gender": "male"
            }
        },
        {
            "count": 2,
            "labels": {
                "age": "29",
                "gender": "male"
            }
        },
        {
            "count": 1,
            "labels": {
                "age": "30",
                "gender": "male"
            }
        },
        {
            "count": 1,
            "labels": {
```

(continues on next page)

(continued from previous page)

```
                "age": "31",
                "gender": "male"
            }
        }
    ]
}
```

Count faces with a given aggregation time interval

```
GET /v1/counter/test_counter/aggregate
```

This method counts faces with a given aggregation time interval.

Query string parameters:

- `aggregate_interval`: a time interval in `h m` format (for example: `10h30m`, `1h`, `30m`) for which the results should be grouped.
- `from` (`required=false`): the start time of counting in RFC 3339 format.
- `till` (`required=false`): the end time of counting in RFC 3339 format.
- `labels` (`required=true`): a list of labels (separated by commas `&labels=gender,age`) by which the results are grouped.

Example

Request

```
curl -i -X GET \
  'http://127.0.0.1:18300/v1/counter/test_counter_1/aggregate?from=2019-11-20T00%3A57
↪%3A21.751812%2B03%3A00&aggregate_interval=24h&labels=gender'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: FC:0P7OPmmS
Date: Mon, 02 Dec 2019 14:54:14 GMT
Content-Length: 161

{
    "aggregate": [
        {
            "count": 6,
            "from": "2019-12-02T03:00:00+03:00",
            "labels": {
                "gender": "male"
            }
```

(continues on next page)

(continued from previous page)

```
        },
        {
            "count": 37,
            "from": "2019-12-02T03:00:00+03:00",
            "labels": {
                "gender": ""
            }
        }
    ]
}
```

1.6 Configure Neural Networks

1.6.1 Neural Networks Summary

Here you can find a summary of neural network models created by our Lab and used in FindFace Server. The total number of models working with FindFace Server is over 150 and isn't presented here. Please contact our support and sales team on support@ntechlab.com to get more information.

Face, vehicle, and body detection

```
detector/facedet.jasmine_fast.004.{gpu,cpu}.fnk
detector/facedet.kali.005.{gpu,cpu}.fnk

detector/bodydet.kali_accurate.021.{gpu,cpu}.fnk
detector/bodydet.kali.021.{gpu,cpu}.fnk

detector/car.jasmine_fast.007.{gpu,cpu}.fnk
detector/cardet.kali.008.{gpu,cpu}.fnk
detector/cardet.kali_accurate.008.{gpu,cpu}.fnk
```

Face and body image normalization

```
facenorm/bee.v3.{gpu,cpu}.fnk
facenorm/bee_fast.{gpu,cpu}.fnk

facenorm/crop1x.v2_maxsize400.{gpu,cpu}.fnk
facenorm/crop2x.v2_maxsize400.{gpu,cpu}.fnk
facenorm/cropbbox.v2.{gpu,cpu}.fnk

facenorm/facenorm.multicrop_full_center_size400.{gpu,cpu}.fnk
facenorm/facenorm.multicrop_full_crop2x_size400.{gpu,cpu}.fnk
```

Face recognition

```
face/nectarine_m_160.{gpu,cpu}.fnk
face/nectarine_xs_320.{gpu,cpu}.fnk
face/nectarine_s_320.{gpu,cpu}.fnk
face/nectarine_l_320.{gpu,cpu}.fnk
face/nectarine_xl_320.{gpu,cpu}.fnk
face/mango_320.{gpu,cpu}.fnk
```

Face attribute recognition

```
faceattr/faceattr.age.v3.{gpu,cpu}.fnk
faceattr/faceattr.gender.v3.{gpu,cpu}.fnk
faceattr/faceattr.quality.v5.{gpu,cpu}.fnk
faceattr/faceattr.beard4.v1.{gpu,cpu}.fnk
faceattr/emotions.v1.{gpu,cpu}.fnk
faceattr/glasses3.v0.{gpu,cpu}.fnk
faceattr/headpose.v3.{gpu,cpu}.fnk
faceattr/faceattr.liveness_web.v1.{gpu,cpu}.fnk
faceattr/liveness.pvn.v2.{gpu,cpu}.fnk
faceattr/faceattr.liveness_pacs.v3.{gpu,cpu}.fnk
faceattr/faceattr.liveness_mobile.hart.{gpu,cpu}.fnk
faceattr/medmask3.v2.{gpu,cpu}.fnk
```

Vehicle image normalization

```
carnorm/carnorm.briacon.v2.{gpu,cpu}.fnk
```

Vehicle recognition

```
carrec/bottas.{gpu,cpu}.fnk
```

Vehicle attribute recognition

```
carattr/carattr.categories.v1.{gpu,cpu}.fnk
carattr/carattr.license_plate.v9.{gpu,cpu}.fnk
carattr/carattr.orientation.v0.{gpu,cpu}.fnk
carattr/carattr.quality.v1.{gpu,cpu}.fnk
carattr/carattr.license_plate_quality.v3.{gpu,cpu}.fnk
carattr/carattr.weight_types7.v0.{gpu,cpu}.fnk
carattr/carattr.description.v1.{gpu,cpu}.fnk
carattr/carattr.special_types12.v0.{gpu,cpu}.fnk
```

Body recognition

```
pedrec/pedrec.durga.{gpu,cpu}.fnk
```

Body attribute recognition

```
pedattr/pedattr.age_gender.v0.{gpu,cpu}.fnk
pedattr/pedattr.bags.v0.{gpu,cpu}.fnk
pedattr/pedattr.clothes_type.v0.{gpu,cpu}.fnk
pedattr/pedattr.color.v1.{gpu,cpu}.fnk
pedattr/pedattr.protective.v1.{gpu,cpu}.fnk
pedattr/pedattr.quality.v0.{gpu,cpu}.fnk
```

1.6.2 Enable Face and Face Attribute Recognition

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

Face and face attribute recognition can be manually enabled and configured. Face and face attribute recognition works on both GPU- and CPU-acceleration.

Note: There are examples of how the sections should look like below. It may vary depending on the models that you have selected.

1. To enable face recognition, do the following:

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

   Important: Be sure to choose the right acceleration type for each model, matching the acceleration type of `extraction-api`: CPU or GPU. Be aware that `extraction-api` on CPU can work only with CPU-models, while `extraction-api` on GPU supports both CPU- and GPU-models.

   1. Open the `extraction-api.yaml` configuration file.

      ```
      sudo vi /opt/ffserver/configs/extraction-api.yaml
      ```

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

      #### GPU

      ```
      detectors:
        ...
        models:
          ...
          face_jasmine:
            aliases:
            - face
      ```

(continues on next page)

(continued from previous page)

```
      - nnd
      - cheetah
      model: detector/facedet.jasmine_fast.004.gpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 2048x2048
  ...
```

CPU

```
detectors:
  ...
  models:
    ...
    face_jasmine:
      aliases:
      - face
      - nnd
      - cheetah
      model: detector/facedet.jasmine_fast.004.cpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 2048x2048
  ...
```

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

GPU

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

CPU

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

4. Specify the face normalizer models in the `normalizers` section by pasting the following code:

GPU

```
normalizers:
  ...
  models:
    crop1x:
      model: facenorm/crop1x.v2_maxsize400.gpu.fnk
    crop2x:
      model: facenorm/crop2x.v2_maxsize400.gpu.fnk
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
    multicrop_full_center:
      model: ''
    multicrop_full_crop2x:
      model: facenorm/facenorm.multicrop_full_crop2x_size400.gpu.fnk
    norm200:
      model: facenorm/bee.v3.gpu.fnk
  ...
```

CPU

```
normalizers:
  ...
  models:
    crop1x:
      model: facenorm/crop1x.v2_maxsize400.cpu.fnk
    crop2x:
      model: facenorm/crop2x.v2_maxsize400.cpu.fnk
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
    multicrop_full_center:
      model: ''
    multicrop_full_crop2x:
      model: facenorm/facenorm.multicrop_full_crop2x_size400.cpu.fnk
    norm200:
      model: facenorm/bee.v3.cpu.fnk
  ...
```

5. **Note:** This step is required to enable face attribute recognition.

   To enable face attribute recognition, do the following:

   In the `/opt/ffserver/configs/extraction-api.yaml` configuration file, specify the extraction models in the `extractors` section, subject to the extractors you want to enable. Be sure to indicate the right acceleration type for each model, matching the acceleration type of `extraction-api`: CPU or GPU.

GPU

```
extractors:
  ...
  models:
    face_age:
      default:
        model: faceattr/faceattr.age.v3.gpu.fnk
    face_beard:
      default:
        model: faceattr/beard.v0.gpu.fnk
    face_emben:
      default:
        model: face/nectarine_xl_320.gpu.fnk
    face_emotions:
      default:
        model: faceattr/emotions.v1.gpu.fnk
    face_gender:
      default:
        model: faceattr/faceattr.gender.v3.gpu.fnk
    face_glasses3:
      default:
        model: faceattr/glasses3.v0.gpu.fnk
    face_headpose:
      default:
        model: faceattr/headpose.v3.gpu.fnk
    face_liveness:
      default:
        model: faceattr/faceattr.liveness_web.v1.gpu.fnk
    face_medmask3:
      default:
        model: faceattr/medmask3.v2.gpu.fnk
    face_quality:
      default:
         model: faceattr/faceattr.quality.v5.gpu.fnk
  ...
```

CPU

```
extractors:
  ...
  models:
    face_age:
      default:
        model: faceattr/faceattr.age.v3.cpu.fnk
    face_beard:
      default:
        model: faceattr/beard.v0.cpu.fnk
    face_emben:
      default:
        model: face/nectarine_xl_320.cpu.fnk
    face_emotions:
      default:
        model: faceattr/emotions.v1.cpu.fnk
    face_gender:
      default:
        model: faceattr/faceattr.gender.v3.cpu.fnk
    face_glasses3:
      default:
        model: faceattr/glasses3.v0.cpu.fnk
    face_headpose:
      default:
        model: faceattr/headpose.v3.cpu.fnk
    face_liveness:
      default:
        model: faceattr/faceattr.liveness_web.v1.cpu.fnk
    face_medmask3:
      default:
        model: faceattr/medmask3.v2.cpu.fnk
    face_quality:
      default:
        model: faceattr/faceattr.quality.v5.cpu.fnk
  ...
```

The most used extraction models are the following:

Extractor	Acceleration	Configure as follows
age	CPU	`face_age: faceattr/faceattr.age.v3.cpu.fnk`
	GPU	`face_age: faceattr/faceattr.age.v3.gpu.fnk`
beard	CPU	`face_beard: faceattr/beard.v0.cpu.fnk`
	GPU	`face_beard: faceattr/beard.v0.gpu.fnk`
	CPU	`face_beard4: faceattr/faceattr.beard4.v1.cpu.fnk`
	GPU	`face_beard4: faceattr/faceattr.beard4.v1.cpu.fnk`
individual face feature vector	CPU	`face_emben: face/nectarine_xl_320.cpu.fnk`
	GPU	`face_emben: face/nectarine_xl_320.gpu.fnk`
gender	CPU	`face_gender: faceattr/faceattr.gender.v3.cpu.fnk`
	GPU	`face_gender: faceattr/faceattr.gender.v3.gpu.fnk`
emotions	CPU	`face_emotions: faceattr/emotions.v1.cpu.fnk`
	GPU	`face_emotions: faceattr/emotions.v1.gpu.fnk`
glasses	CPU	`face_glasses3: faceattr/glasses3.v0.cpu.fnk`
	GPU	`face_glasses3: faceattr/glasses3.v0.gpu.fnk`
	CPU	`face_glasses4: faceattr/faceattr.glasses4.v0.cpu.fnk`
head position	CPU	`face_headpose: faceattr/headpose.v3.cpu.fnk`
	GPU	`face_headpose: faceattr/headpose.v3.gpu.fnk`
face liveness	CPU	`face_liveness: faceattr/faceattr.liveness_web.v1.cpu.fnk`
	GPU	`face_liveness: faceattr/faceattr.liveness_web.v1.gpu.fnk`
face mask	CPU	`face_medmask3: faceattr/medmask3.v2.cpu.fnk`
	GPU	`face_medmask3: faceattr/medmask3.v2.gpu.fnk`
	CPU	`face_medmask4: faceattr/faceattr.medmask4.v0.cpu.fnk`
	GPU	`face_medmask4: faceattr/faceattr.medmask4.v0.gpu.fnk`
face quality	CPU	`face_quality: faceattr/faceattr.quality.v5.cpu.fnk`
	GPU	`face_quality: faceattr/faceattr.quality.v5.gpu.fnk`

2. To enable face recognition, modify the `/opt/ffserver/configs/video-worker.yaml` configuration file.

    1. In the `models` section, specify the face neural network models by analogy with the example below:

GPU

```
sudo vi /opt/ffserver/configs/video-worker.yaml
```

```
models:
  ...
  detectors:
    ...
    face:
      fnk_path: /usr/share/findface-data/models/detector/facedet.jasmine_fast.
↪004.gpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    face_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.
↪gpu.fnk
    face_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/crop1x.v2_maxsize400.
↪gpu.fnk
    ...
  extractors:
    ...
    face_quality:
      fnk_path: /usr/share/findface-data/models/faceattr/faceattr.quality.v5.
↪gpu.fnk
      normalizer: face_norm_quality
    ...
```

CPU

```
models:
  ...
  detectors:
    ...
    face:
      fnk_path: /usr/share/findface-data/models/detector/facedet.jasmine_fast.
↪004.cpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    face_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.
↪cpu.fnk
    face_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/crop1x.v2_maxsize400.
↪cpu.fnk
    ...
  extractors:
```

(continues on next page)

(continued from previous page)

```
    ...
    face_quality:
      fnk_path: /usr/share/findface-data/models/faceattr/faceattr.quality.v5.
↪cpu.fnk
      normalizer: face_norm_quality
    ...
```

2. Make sure that the `objects -> face` section is included:

```
objects:
  ...
  face:
    normalizer: face_norm
    quality: face_quality
    track_features: ''
```

3. To enable face recognition, open the `/opt/ffserver/configs/video-manager.yaml` configuration file and make sure it contains the `face` section in `detectors` that looks similar to the example below.

```
sudo vi /opt/ffserver/configs/video-manager.yaml
```

```
detectors:
  face:
    filter_min_quality: 0.45
    filter_min_size: 1
    filter_max_size: 8192
    roi: ""
    fullframe_crop_rot: false
    fullframe_use_png: false
    jpeg_quality: 95
    overall_only: false
    realtime_post_first_immediately: false
    realtime_post_interval: 1
    realtime_post_every_interval: false
    track_interpolate_bboxes: true
    track_miss_interval: 1
    track_overlap_threshold: 0.25
    track_max_duration_frames: 0
    track_send_history: false
    post_best_track_frame: true
    post_best_track_normalize: true
    post_first_track_frame: false
    post_last_track_frame: false
    tracker_type: simple_iou
    track_deep_sort_matching_threshold: 0.65
    track_deep_sort_filter_unconfirmed_tracks: true
    track_object_is_principal: false
    track_history_active_track_miss_interval: 0
    filter_track_min_duration_frames: 1
    extractors_track_triggers: {}
```

1.6.3 Enable Vehicle and Vehicle Attribute Recognition

FindFace Server allows you to recognize a single vehicle and its attributes.

The vehicle attributes are as follows:

- license plate number (for selected countries),

  Note: Currently, the following countries are supported:

  Europe: Russia, Lithuania, Latvia, Estonia, Finland, Czech Republic, Serbia, Belarus, Ukraine, Moldova, Georgia, Azerbaijan, Armenia.

  Asia: the UAE, Kazakhstan, Kyrgyzstan, Tajikistan, Turkmenistan, Uzbekistan, Saudi Arabia, Vietnam, India, Pakistan, Thailand.

  North America: Mexico.

  South America: Argentina, Brazil.

- color,
- make,
- model,
- body style,
- vehicle category,
- vehicle orientation (front, rear, or side),
- special vehicle type (taxi, route transport, carsharing, ambulance, police, rescue service, gas service, military, road service, other special),
- vehicle weight and body size.

  Note: Special vehicle recognition as well as vehicle weight and body size recognition work for selected countries only and may not work for your country. For more information please contact your manager or our support team (support@ntechlab.com).

This section describes how to enable vehicle and vehicle attribute recognition.

To enable recognition of vehicles and their attributes, do the following:

1. Specify neural network models for vehicle recognition and vehicle attribute recognition in the `/opt/ffserver/configs/extraction-api.yaml` configuration file.

   Important: Be sure to choose the right acceleration type for each model, matching the acceleration type of `extraction-api`: CPU or GPU. Be aware that `extraction-api` on CPU can work only with CPU-models, while `extraction-api` on GPU supports both CPU- and GPU-models.

   1. Open the `extraction-api.yaml` configuration file.

      ```
      sudo vi /opt/ffserver/configs/extraction-api.yaml
      ```

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

GPU

```
detectors:
  ...
  models:
    ...
    gustav:
      aliases:
      - car
      - efreitor
      model: detector/cardet.kali.008.gpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 2048x2048
  ...
```

CPU

```
detectors:
  ...
  models:
    ...
    gustav:
      aliases:
      - car
      - efreitor
      model: detector/cardet.kali.008.cpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 2048x2048
  ...
```

3. Make sure that the `objects -> car` section contains the `quality_attribute: car_quality` and the `base_normalizer: facenorm/cropbbox.v2.gpu.fnk` or the `base_normalizer: facenorm/cropbbox.v2.cpu.fnk`, depending on your acceleration type:

GPU

```
objects:
  ...
  car:
    base_normalizer: facenorm/cropbbox.v2.gpu.fnk
    quality_attribute: car_quality
  ...
```

CPU

```
objects:
  ...
  car:
    base_normalizer: facenorm/cropbbox.v2.cpu.fnk
    quality_attribute: car_quality
  ...
```

4. Specify the normalizers required for the extractors. If you need license plate recognition, specify the `carlicplate` normalizer. For other extractors, specify the `cropbbox` normalizer.

Normalizer	Normalizer model	Used for extractors
carlicplate	`carnorm/carnorm.briacon.v2.gpu.fnk` `carnorm/carnorm.briacon.v2.cpu.fnk`	`car_license_plate`
cropbbox	`facenorm/cropbbox.v2.gpu.fnk` `facenorm/cropbbox.v2.cpu.fnk`	`car_license_plate_quality`, `car_description`, `car_quality`, `car_special_types11`, `car_categories`, `car_orientation`, `car_weight_types7`

GPU

```
normalizers:
  ...
  models:
    ...
    carlicplate:
      model: carnorm/carnorm.briacon.v2.gpu.fnk
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
    ...
```

CPU

```
normalizers:
  ...
  models:
    ...
    carlicplate:
      model: carnorm/carnorm.briacon.v2.cpu.fnk
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
    ...
```

5. Specify the extraction models in the `extractors -> models` section, subject to the extractors you want to enable:

GPU

```
extractors:
   ...
   models:
     car_categories:
       default:
         model: carattr/carattr.categories.v1.gpu.fnk
     car_description:
       default:
         model: carattr/carattr.description.v1.gpu.fnk
     car_emben:
       default:
         model: carrec/bottas.gpu.fnk
     car_license_plate:
       default:
         model: carattr/carattr.license_plate.v9.gpu.fnk
     car_license_plate_quality:
       default:
         model: carattr/carattr.license_plate_quality.v3.gpu.fnk
     car_orientation:
       default:
         model: carattr/carattr.orientation.v0.gpu.fnk
     car_quality:
       default:
         model: carattr/carattr.quality.v1.gpu.fnk
     car_special_types11:
       default:
         model: carattr/carattr.special_types11.v1.gpu.fnk
     car_weight_types7:
       default:
         model: carattr/carattr.weight_types7.v0.gpu.fnk
   ...
```

CPU

```
extractors:
   ...
   models:
     car_categories:
       default:
         model: carattr/carattr.categories.v1.cpu.fnk
     car_description:
       default:
         model: carattr/carattr.description.v1.cpu.fnk
     car_emben:
       default:
         model: carrec/bottas.cpu.fnk
```

(continues on next page)

(continued from previous page)

```
    car_license_plate:
      default:
        model: carattr/carattr.license_plate.v9.cpu.fnk
    car_license_plate_quality:
      default:
        model: carattr/carattr.license_plate_quality.v3.cpu.fnk
    car_orientation:
      default:
        model: carattr/carattr.orientation.v0.cpu.fnk
    car_quality:
      default:
        model: carattr/carattr.quality.v1.cpu.fnk
    car_special_types11:
      default:
        model: carattr/carattr.special_types11.v1.cpu.fnk
    car_weight_types7:
      default:
        model: carattr/carattr.weight_types7.v0.cpu.fnk
  ...
```

The most used extraction models are the following:

Extractor	Configure as follows
vehicle feature vector	`car_emben: carrec/bottas.cpu.fnk`
	`car_emben: carrec/bottas.gpu.fnk`
license plate number	`car_license_plate: carattr/carattr.license_plate.v9.cpu.fnk car_license_plate_quality: carattr/carattr.license_plate_quality.v3.cpu.fnk`
	`car_license_plate: carattr/carattr.license_plate.v9.gpu.fnk car_license_plate_quality: carattr/carattr.license_plate_quality.v3.gpu.fnk`
set of attributes: make / color / model / body style	`car_description: carattr/carattr.description.v1.cpu.fnk`
	`car_description: carattr/carattr.description.v1.gpu.fnk`
vehicle image quality	`car_quality: carattr/carattr.quality.v1.cpu.fnk`
	`car_quality: carattr/carattr.quality.v1.gpu.fnk`
special vehicle	`car_special_types11: carattr/carattr.special_types11.v1.cpu.fnk`
	`car_special_types11: carattr/carattr.special_types11.v1.gpu.fnk`
vehicle category	`car_categories: carattr/carattr.categories.v1.cpu.fnk`
	`car_categories: carattr/carattr.categories.v1.gpu.fnk`
vehicle weight and body size	`car_weight_types7: carattr/carattr.weight_types7.v0.cpu.fnk`
	`car_weight_types7: carattr/carattr.weight_types7.v0.gpu.fnk`
vehicle orientation	`car_orientation: carattr/carattr.orientation.v0.cpu.fnk`
	`car_orientation: carattr/carattr.orientation.v0.gpu.fnk`

2. Modify the `/opt/ffserver/configs/video-worker.yaml` configuration file.

   1. In the `models` section, specify the vehicle detector, normalizer, and extractor models as shown in the ex-

ample below:

GPU

```
sudo vi /opt/ffserver/configs/video-worker.yaml
```

```
models:
  ...
  detectors:
    ...
    car:
      fnk_path: /usr/share/findface-data/models/detector/cardet.kali.008.gpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    car_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
    car_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
    ...
  extractors:
    ...
    car_quality:
      fnk_path: /usr/share/findface-data/models/carattr/carattr.quality.v1.gpu.
↪fnk
      normalizer: car_norm_quality
```

CPU

```
sudo vi /opt/ffserver/configs/video-worker.yaml
```

```
models:
  ...
  detectors:
    ...
    car:
      fnk_path: /usr/share/findface-data/models/detector/cardet.kali.008.cpu.fnk
      min_size: 60
    ...
  normalizers:
    ...
    car_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    car_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    ...
  extractors:
    ...
    car_quality:
```

(continues on next page)

(continued from previous page)

```
      fnk_path: /usr/share/findface-data/models/carattr/carattr.quality.v1.cpu.
↪fnk
      normalizer: car_norm_quality
```

2. Make sure that the `objects -> car` section is included:

```
objects:
  ...
  car:
    normalizer: car_norm
    quality: car_quality
    track_features: ''
```

3. Open the `/opt/ffserver/configs/video-manager.yaml` configuration file and make sure it contains the `car` section in `detectors` that looks similar to the example below. Note that the `filter_min_quality` parameter is set to `0.65` by default. You can increase it to get more accurate results from cameras and other video sources. The best practice is to set this value to `0.73`.

```
sudo vi /opt/ffserver/configs/video-manager.yaml
```

```
detectors:
  ...
  car:
    filter_min_quality: 0.65
    filter_min_size: 100
    filter_max_size: 8192
    roi: ''
    fullframe_crop_rot: false
    fullframe_use_png: false
    jpeg_quality: 95
    overall_only: true
    realtime_post_first_immediately: false
    realtime_post_interval: 1
    realtime_post_every_interval: false
    track_interpolate_bboxes: true
    track_miss_interval: 1
    track_overlap_threshold: 0.25
    track_max_duration_frames: 0
    track_send_history: false
    post_best_track_frame: true
    post_best_track_normalize: true
    post_first_track_frame: false
    post_last_track_frame: false
    tracker_type: simple_iou
    track_deep_sort_matching_threshold: 0.65
    track_deep_sort_filter_unconfirmed_tracks: true
    track_object_is_principal: false
    track_history_active_track_miss_interval: 0
```

1.6.4 Enable Body and Body Attribute Recognition

FindFace Server allows you to recognize individual human bodies and body attributes.

The body attributes are as follows:

- gender:
  - male;
  - female;
- age (by group):
  - 0-16 years;
  - 17-35 years;
  - 36-50 years;
  - 50+ years;
- clothing type:
  - generalized category of upper body wear: long sleeves, short sleeves, no sleeve;
  - specific type of upper body wear: jacket, coat, sleeveless vest, sweatshirt, T-shirt, shirt, dress;
  - type of lower body wear: pants, skirt, shorts, nondescript;
  - type of headgear: hat/cap, hood/headscarf, none;
- clothing color (top/bottom);
- presence of personal protective equipment (PPE):
  - PPE item: vest, helmet;
  - PPE color;
  - PPE recognition score;
- whether a person has a bag:
  - on the back;
  - in hand(s).

This section describes how to enable body and body attribute recognition.

To enable recognition of human bodies and their attributes, do the following:

1. Specify neural network models for body object and body attribute recognition in the `/opt/ffserver/configs/extraction-api.yaml` configuration file.

   Important: Be sure to choose the right acceleration type for each model, matching the acceleration type of `extraction-api`: CPU or GPU. Be aware that `extraction-api` on CPU can work only with CPU-models, while `extraction-api` on GPU supports both CPU- and GPU-models.

   1. Open the `extraction-api.yaml` configuration file.

      ```
      sudo vi /opt/ffserver/configs/extraction-api.yaml
      ```

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

GPU

```
detectors:
  ...
  models:
    ...
    body_gustav:
      aliases:
      - body
      - edie
      - shiloette
      - glen
      model: detector/bodydet.kali.021.gpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 2048x2048
  ...
```

CPU

```
detectors:
  ...
  models:
    ...
    body_gustav:
      aliases:
      - body
      - edie
      - shiloette
      - glen
      model: detector/bodydet.kali.021.cpu.fnk
      options:
        min_object_size: 32
        resolutions:
        - 2048x2048
  ...
```

3. Make sure that the `objects -> body` section contains the `quality_attribute: body_quality` and the `base_normalizer: facenorm/cropbbox.v2.gpu.fnk` or the `base_normalizer: facenorm/cropbbox.v2.cpu.fnk`, depending on your acceleration type:

GPU

```
objects:
  ...
  body:
    base_normalizer: facenorm/cropbbox.v2.gpu.fnk
    quality_attribute: body_quality
  ...
```

CPU

```
objects:
  ...
  body:
    base_normalizer: facenorm/cropbbox.v2.cpu.fnk
    quality_attribute: body_quality
  ...
```

4. Make sure that the `normalizers` section contains a model for the `cropbbox` normalizer, as shown in the example below. This normalizer is required for the extractors.

GPU

```
normalizers:
  ...
  models:
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.gpu.fnk
    ...
```

CPU

```
normalizers:
  ...
  models:
    ...
    cropbbox:
      model: facenorm/cropbbox.v2.cpu.fnk
    ...
```

5. Specify the extraction models in the `extractors -> models` section, subject to the extractors you want to enable:

GPU

```
extractors:
   ...
   models:
     body_age_gender:
       default:
         model: pedattr/pedattr.age_gender.v0.gpu.fnk
     body_bags:
       default:
         model: pedattr/pedattr.bags.v0.gpu.fnk
     body_clothes:
       default:
         model: pedattr/pedattr.clothes_type.v0.gpu.fnk
     body_color:
       default:
         model: pedattr/pedattr.color.v1.gpu.fnk
     body_emben:
       default:
         model: pedrec/pedrec.durga.gpu.fnk
     body_protective_equipment:
       default:
         model: pedattr/pedattr.protective.v1.gpu.fnk
     body_quality:
       default:
         model: pedattr/pedattr.quality.v0.gpu.fnk
   ...
```

CPU

```
extractors:
   ...
   models:
     body_age_gender:
       default:
         model: pedattr/pedattr.age_gender.v0.cpu.fnk
     body_bags:
       default:
         model: pedattr/pedattr.bags.v0.cpu.fnk
     body_clothes:
       default:
         model: pedattr/pedattr.clothes_type.v0.cpu.fnk
     body_color:
       default:
         model: pedattr/pedattr.color.v1.cpu.fnk
     body_emben:
       default:
         model: pedrec/pedrec.durga.cpu.fnk
     body_protective_equipment:
       default:
         model: pedattr/pedattr.protective.v1.cpu.fnk
```

(continues on next page)

(continued from previous page)

```
    body_quality:
      default:
        model: pedattr/pedattr.quality.v0.cpu.fnk
  ...
```

The most used extraction models are the following:

Extractor	Configure as follows
age and gender	`body_age_gender: pedattr/pedattr.age_gender.v0.gpu.fnk`
	`body_age_gender: pedattr/pedattr.age_gender.v0.cpu.fnk`
presence of bag	`body_bags: pedattr/pedattr.bags.v0.gpu.fnk`
	`body_bags: pedattr/pedattr.bags.v0.cpu.fnk`
clothing type	`body_clothes: pedattr/pedattr.clothes_type.v0.gpu.fnk`
	`body_clothes: pedattr/pedattr.clothes_type.v0.cpu.fnk`
clothing color	`body_color: pedattr/pedattr.color.v1.gpu.fnk`
	`body_color: pedattr/pedattr.color.v1.cpu.fnk`
individual body feature vector	`body_emben: pedrec/pedrec.durga.gpu.fnk`
	`body_emben: pedrec/pedrec.durga.cpu.fnk`
presence of protective equipment	`body_protective_equipment: pedattr/pedattr.protective.v1.gpu.fnk`
	`body_protective_equipment: pedattr/pedattr.protective.v1.cpu.fnk`
body quality	`body_quality: pedattr/pedattr.quality.v0.gpu.fnk`
	`body_quality: pedattr/pedattr.quality.v0.cpu.fnk`

2. Modify the `/opt/ffserver/configs/video-worker.yaml` configuration file.

   1. In the `models` section, specify the body neural network models by analogy with the example below:

GPU

```
sudo vi /opt/ffserver/configs/video-worker.yaml
```

```
models:
  ...
  detectors:
    ...
    body:
      fnk_path: /usr/share/findface-data/models/detector/bodydet.kali.021.gpu.
↪fnk
      min_size: 60
    ...
  normalizers:
    ...
    body_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
    body_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.gpu.fnk
```

(continues on next page)

(continued from previous page)

```
    ...
  extractors:
    ...
    body_quality:
      fnk_path: /usr/share/findface-data/models/pedattr/pedattr.quality.v0.gpu.
↪fnk
      normalizer: body_norm_quality
```

CPU

```
sudo vi /opt/ffserver/configs/video-worker.yaml
```

```
models:
  ...
  detectors:
    ...
    body:
      fnk_path: /usr/share/findface-data/models/detector/bodydet.kali.021.cpu.
↪fnk
      min_size: 60
    ...
  normalizers:
    ...
    body_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    body_norm_quality:
      fnk_path: /usr/share/findface-data/models/facenorm/cropbbox.v2.cpu.fnk
    ...
  extractors:
    ...
    body_quality:
      fnk_path: /usr/share/findface-data/models/pedattr/pedattr.quality.v0.cpu.
↪fnk
      normalizer: body_norm_quality
```

2. Make sure that the `objects -> body` section is included:

```
objects:
  ...
  body:
    normalizer: body_norm
    quality: body_quality
    track_features: ''
```

3. Open the `/opt/ffserver/configs/video-manager.yaml` configuration file and make sure it contains the `body` section in `detectors` that looks similar to the example below.

```
sudo vi /opt/ffserver/configs/video-manager.yaml
```

```
detectors:
  ...
  body:
    filter_min_quality: 0.6
    filter_min_size: 70
    filter_max_size: 8192
    roi: ''
    fullframe_crop_rot: false
    fullframe_use_png: false
    jpeg_quality: 95
    overall_only: true
    realtime_post_first_immediately: false
    realtime_post_interval: 1
    realtime_post_every_interval: false
    track_interpolate_bboxes: true
    track_miss_interval: 1
    track_overlap_threshold: 0.25
    track_max_duration_frames: 0
    track_send_history: false
    post_best_track_frame: true
    post_best_track_normalize: true
    post_first_track_frame: false
    post_last_track_frame: false
    tracker_type: simple_iou
    track_deep_sort_matching_threshold: 0.65
    track_deep_sort_filter_unconfirmed_tracks: true
    track_object_is_principal: false
    track_history_active_track_miss_interval: 0
```

1.6.5 Enable Face Liveness Detection

The FindFace Server face liveness detector tells apart live faces from face representations, such as face images, videos, or masks. The liveness detector estimates face liveness with a certain level of confidence and returns the confidence score along with a binary result `real/fake`, depending on the pre-defined liveness threshold.

Following the instructions below to enable face liveness detector.

Note: The face liveness detector functions on both GPU- and CPU-acceleration. However, it is much slower on CPU.

In this section:

Enable Face Liveness Detector

To enable the face liveness detector, do the following:

1. Open the `/opt/ffserver/configs/video-worker.yaml` configuration file. In the `liveness` section, specify the neural network models as shown in the example:

   #### GPU

   ```
   sudo vi /opt/ffserver/configs/video-worker.yaml

   liveness:
     fnk: /usr/share/findface-data/models/faceattr/liveness.pacs.v2.gpu.fnk
     norm: /usr/share/findface-data/models/facenorm/facenorm.multicrop_full_crop2x_
   ↪size400.gpu.fnk
   ...
   ```

   #### CPU

   ```
   sudo vi /opt/ffserver/configs/video-worker.yaml

   liveness:
     fnk: /usr/share/findface-data/models/faceattr/liveness.pacs.v2.cpu.fnk
     norm: /usr/share/findface-data/models/facenorm/facenorm.multicrop_full_crop2x_
   ↪size400.cpu.fnk
   ...
   ```

1.6.6 Liveness Detection as Standalone Service

See also:

Enable Face Liveness Detection

Besides the *integrated* anti-spoofing system that distinguishes a live face from a face image, FindFace Server provides an API-based face liveness detection service `liveness-api`.

The `liveness-api` service takes a specific number of frames from a provided video fragment and returns the best quality face, along with a decimal liveness result for it, averaged across the taken frames. If configured, the service can also return full-frame and normalized face images and save the detection result in the `sf-api` cache, returning `detection_id`.

To install and use the `liveness-api` service standalone, see *liveness installation*.

In this section:

Configure `liveness-api`

To configure the `liveness-api` component, do the following:

1. Create a default `liveness-api.yaml` configuration file (e.g., into the `/opt/ffserver/configs` directory).

   ```
   docker run --rm -ti --name liveness-api-cfg \
       docker.int.ntl/ntech/universe/liveness-api:ffserver-11.240325 \
       --config-template > /opt/ffserver/configs/liveness-api.yaml
   ```

2. You can configure the `liveness-api` parameters according to your needs in the `/opt/ffserver/configs/liveness-api.yaml` configuration file. You can find its default content `here`. Enable liveness recognition in the `/opt/ffserver/configs/extraction-api.yaml` configuration file by specifying the extraction model in the `face_liveness` section. See *Enable Face and Face Attribute Recognition*.

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

Command line flags	Type	Description
-attributes	value	Array of attributes to extract.
-config	string	Path to config file
-config-template		Output config template and exit.
-debug		Debug.
-extraction-api-extraction-api	string	Extraction API address (default http://127.0.0.1:18666).
-extraction-api-request-batch-size	int	Extraction API request batch size (default 16).
-extraction-api-timeouts-connect	duration	extraction-api-timeouts-connect (default 5s).
-extraction-api-timeouts-idle-connection	duration	extraction-api-timeouts-idle-connection (default 10s).
-extraction-api-timeouts-overall	duration	extraction-api-timeouts-overall (default 35s).
-extraction-api-timeouts-response-header	duration	extraction-api-timeouts-response-header (default 30s).
-fullframe-jpeg-quality	int	JPEG quality of full frames in the photo field (default 75).
-help		Print help information.
-limits-video-fps	float	Maximum video frames per second (default 60).
-limits-video-height-px	int	Maximum video height in px (default 1080).
-limits-video-length-sec	float	Maximum video length in seconds (default 60).
-limits-video-size	int	Maximum size of video supplied in request body, bytes. (default 10485760).
-limits-video-width-px	int	Maximum video width in pixels (default 1920)
-listen	string	IP:port to listen on (default :18301).
-liveness-threshold	float	Liveness threshold (default 0.994).
-max-decoded-frames	int	Finish decoding after reaching the specified number of frames (default 30).
-mf-selector	string	Service behavior upon having multiple faces in the video frame: reject - reject this frame, biggest - use the biggest face for liveness detection. (default reject).
-min-selected-frames	int	The minimum number of final frames successfully passed through decoding and liveness extraction. Must be equal or less than max-decoded-frames. (default 10).
-processing-script	string	Path to LUA file with processing_script function (default processing_score.lua).
-reject-ignore-secondary-faces-by-proportion	float	Proportion of the secondary faces for them to be ignored.
-sf-api-sf-api	string	SF API address (default http://127.0.0.1:18411).
-sf-api-timeouts-connect	duration	sf-api-timeouts-connect (default 5s).
-sf-api-timeouts-idle-connection	duration	sf-api-timeouts-idle-connection (default 10s).
-sf-api-timeouts-overall	duration	sf-api-timeouts-overall (default 35s).
-sf-api-timeouts-response-header	duration	sf-api-timeouts-response-header (default 30s).

The format of environment variable for flag -my-flag is CFG_MY_FLAG.

Priority:

1. Defaults from source code (lowest priority).
2. Configuration file.
3. Environment variables.
4. Command line.

Note: When working with processing_<attribute>.lua don't forget to get the appropriate attribute by setting it in the configuration file in the attributes section.

The default processing_score.lua file describes the logic of choosing the best face:

```
function process(candidates)

    local best_frame_idx = nil
    local best_score = nil
    for idx, candidate in pairs(candidates) do
        local score = candidate.score
        if (best_frame_idx == nil) or (best_score < score) then
            best_score = score
            best_frame_idx = idx
        end
    end

    return best_frame_idx
end
```

HTTP API Requests to `liveness-api`

To interact with the `liveness-api` service, use HTTP API requests to `http://liveness-api:18301/`.

```
POST /v1/video-liveness
```

This method determines liveness by video.

Query string parameters:

- `return_detection` (`required=false`, `default=False`): boolean, save the best face in the `sf-api` cache and return its `detection_id`.
- `return_normalized` (`required=false`, `default=False`): boolean, return the face normalized image in the `normalized` field.
- `return_photo` (`required=false`, `default=False`): boolean, return the full frame in the `photo` field.

Request Body:

Contains a video file.

Responses with Common Codes:

Code	Description
200	OK. Returns `application/json`.
400	Bad Request.
405	Invalid input.
406	Not Acceptable.
500	Internal Server Error.

Example

Request

```
curl -i -X POST \
    '127.0.0.1:18301/v1/video-liveness?return_detection=true&return_normalized=true&
↪return_photo=true' \
    --header 'Content-Type: video/mp4' \
    --data-binary '@/home/my_video.mp4'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "alive":true,
   "average_liveness":0.9521886,
   "best_face":{
      "liveness":0.9557304,
      "quality":0.91568387,
      "bbox":{
         "left":138,
         "top":172,
         "right":363,
         "bottom":468
      },
      "detection_id":"btfiva1o8cjc1k44q1vg",
      "photo":"/9j/2wCEAAgGBgcGBQgHBwcJ...",
      "normalized":"iVBORw0KGgoAAAANSUU...",
      "frame_no":14,
      "frame_ts":0.56
   }
}
```

1.7 Maintenance and Troubleshooting

1.7.1 Migrate Feature Vectors to a Different Neural Network Model

This section is about how to migrate object feature vectors to another neural network model.

See also:

sf-api-migrate options.

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

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

```
docker exec -it sf-api bash -c "mkdir dumps; cd dumps && /storage-api-dump -config /
↪sf-api.yaml"
docker cp sf-api-:/dumps /opt/ffserver
```

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

   1. Navigate to the directory for stored data (for example, `/opt/ffserver/data/tnt`) and find out the number of shards by counting the number of directories.

      Note: There are two shards in the example below.

      ```
      cd /opt/ffserver/data/tnt

      ls -l

      shard-001
      shard-002
      ```

   2. Create directories that will host files of the new shards.

      ```
      sudo mkdir -p shard-01{1..2}/{index,snapshots,xlogs}
      ```

3. Open the `/opt/ffserver/configs/extraction-api.yaml` configuration file and replace the extraction models with the new ones in the emben parameters (`body_emben`, `car_emben` and `face_emben`), depending on the object types you want to migrate.

   Important: Be sure to choose the right acceleration type for each model, matching the acceleration type of `extraction-api`: CPU or GPU. Be aware that `extraction-api` on CPU can work only with CPU-models, while `extraction-api` on GPU supports both CPU- and GPU-models.

   ```
   sudo vi /opt/ffserver/configs/extraction-api.yaml
   ```

   ### CPU

   ```
   extractors:
     ...
     models:
       ...
        face_emben:
           default:
             model: face/<new_model_face>.cpu.fnk
        ...
   ```

GPU

```
extractors:
  ...
  models:
    ...
     face_emben:
        default:
          model: face/<new_model_face>.gpu.fnk
```

4. Create new `tntapi` containers for each new shard. To do that, copy the `docker run` command of the existing service and replace the name of the shard and the path to the shard in the `--volume` flag.

   ```
   docker run -tid --name tnt-new-1 --restart always --network server \
       --env CFG_LISTEN_HOST=0.0.0.0 \
       --env CFG_NTLS=ntls:3133 \
       --env TT_LISTEN=0.0.0.0:32001 \
       --env TT_MEMTX_MEMORY=$((1024 * 1024 * 1024)) \
       --volume /opt/ffserver/tnt/001-new:/opt/ntech/var/lib/tarantool/default \
       docker.int.ntl/ntech/universe/tntapi:ffserver-11.240325
   ```

   Use environment environment and configuration flags:

   - `CFG_LISTEN_HOST=0.0.0.0`: host to public HTTP API.
   - `CFG_NTLS=ntls:3133`: host and port of the `ntls` server.
   - `TT_LISTEN=0.0.0.0:32001`: binary host/port, used for admin operations and replication.
   - `TT_MEMTX_MEMORY=$((1024 * 1024 * 1024))`: the maximum memory usage in bytes.
   - `/opt/ffserver/tnt/`: the directory on the host to store `tntapi` data.

5. Create a default `sf-api-migrate` configuration file. You can find its default content here.

   ```
   docker run --rm -ti --entrypoint "/sf-api-migrate" docker.int.ntl/ntech/universe/sf-
   ↪api:ffserver-11.240325 \
       --config-template > /opt/ffserver/configs/sf-api-migrate.yaml
   ```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

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

   ```
   storage-api-from: # current location of the gallery
     ...
     shards:
       - master: http://tnt-old-1:8001/v2/
         slaves: []
       - master: http://tnt-old-2:8001/v2/
         slaves: []
     ...
   ```

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

   ```
   storage-api-to:
     ...
     shards:
   ```

(continues on next page)

(continued from previous page)

```
    - master: http://tnt-new-1:8001/v2/
      slaves: []
    - master: http://tnt-new-2:8001/v2/
      slaves: []
  ...
```

8. Run the `sf-api-migrate` utility with the configuration file.

```
docker run --rm -ti --network server --entrypoint "/sf-api-migrate" \
    --volume /opt/ffserver/configs/sf-api-migrate.yaml:/sf-api-migrate.yaml \
    docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
    --config /sf-api-migrate.yaml \
```

Use docker options:

- `--network`: connect a container to a network, named `server`.

Use configuration flags:

- `--volume /opt/ffserver/configs/sf-api-migrate.yaml:/sf-api-migrate.yaml`: mount the configuration file from the directory on the host into the container.
- `--config /sf-api-migrate.yaml`: provide path to the `sf-api-migrate.yaml` configuration file.

Note: The migration process can take up a significant amount of time if there are many features of object vectors in the `tntapi` database.

9. Open the `/opt/ffserver/configs/sf-api.yaml` configuration file and adjust the shards ports, subject to the new shards settings.

```
sudo vi /opt/ffserver/configs/sf-api.yaml
```

```
storage-api:
  ...
  shards:
  - master: http://tnt-new-1:8001/v2/
    slaves: []
  - master: http://tnt-new-2:8002/v2/
    slaves: []
```

10. To make sure that the migration was successful, send a request to the `sf-api`:

```
GET /v3/galleries/:objtype/:gallery
```

The `emben_model` field of the response must specify the model `<new_model_face>` to which the migration was performed, for example:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:NZfBDUu0
Date: Thu, 06 Jun 2024 09:41:01 GMT
Content-Length: 83

{"name":"hello","emben_model":"nectarine_m_160","emben_size":160,"objects":100114}
```

1.7.2 Custom Feature Vector Database Structure

The database structure is set via the `tnt-schema.lua` file.

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

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

The following service fields for objects (faces/bodies) in the database are required for correct operation of `sf-api`.

```
meta_scheme  = {
  {
      id = 1,
      name = 'feat',
      field_type = 'string',
      default = ""
  },
  {
      id = 2,
      name = 'normalized_id',
      field_type = 'string',
      default = ""
  },
  ...
}
```

If you need to use metadata, the meta fields must be defined in the `tntapi` configuration file. So the data schema should be as follows:

```
cfg_spaces = {
  default = {
    meta_scheme = {
      {
        id = 1,
        name = 'feat',
        field_type = 'string',
        default = ""
      },
     {
        id = 2,
        name = 'normalized_id',
        field_type = 'string',
        default = ""
      },
      {
        id = 3,
        name = 'timestamp',
        field_type = 'unsigned',
        default = 0
```

(continues on next page)

(continued from previous page)

```
      },
      {
        id = 4,
        name = 'photo_hash',
        field_type = 'string',
        default = ""
      },
      {
        id = 5,
        name = 'cam_id',
        field_type = 'string',
        default = ""
      },
      {
        id = 6,
        name = 'person_id',
        field_type = 'unsigned',
        default = 0
      },
      {
        id = 7,
        name = 'tags',
        field_type = 'set[string]',
        default = {}
      }
    },
    meta_indexes = {'cam_id', 'person_id'}
  },
  altscheme = {
    meta_scheme = {
      {
        id = 1,
        name = 'feat',
        field_type = 'string',
        default = ""
      },
      {
        id = 2,
        name = 'normalized_id',
        field_type = 'string',
        default = ""
      },
      {
        id = 3,
        name = 'name',
        field_type = 'string',
        default = ""
      }
    },
    meta_indexes = {}
  }
}
```

1.7.3 Check Component Status

Check the status of containers once you have encountered a system problem, using the following commands:

```
docker ps
docker container inspect <container_id>/<container_name>
docker container stats <container_id>/<container_name>
```

1.7.4 Logging

Consulting logs is one of the first things you should do to identify a cause of a problem. By default, the FindFace Multi processes are logged to Docker container logs, which can be accessed via the `docker logs` and `docker service logs` commands. In addition, Docker uses the json-file logging driver, which stores container logs in JSON. You can configure Docker to use another logging driver, choosing from the multiple logging mechanisms available.

This section describes how to set up Docker to use the `journald` logging driver, which sends container logs to the `systemd` journal. In this case, log entries are retrieved using the `journalctl` command, through the `journal API`, or the `docker logs` command. You can configure the `systemd` journal as well by using the instructions below.

In this section:

Configure Journald

To configure the `systemd-journal` service, do the following:

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

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

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

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

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

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

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

3. Restart the journald service.

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

Enabling Journald Logging Driver

To enable Docker to use the journald logging driver, do the following:

1. Add the following line to the /etc/docker/daemon.json configuration file.

Tip: This file may not be present on your system. Check whether it exists and if it does not, create the /etc/docker directory first and then the file.

```
sudo mkdir /etc/docker
sudo touch /etc/docker/daemon.json
```

```
sudo vi /etc/docker/daemon.json

{
  "log-driver": "journald"
}
```

2. Stop all Docker containers.

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

3. Restart the Docker service.

```
sudo systemctl restart docker
```

4. Start all Docker containers.

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

Consult Logs

Use any convenient method to work with the journald logs. The following commands are a good place to start:

- Display all logs:

```
journalctl -fa
```

- Display logs by container ID:

```
journalctl CONTAINER_ID=<container_id> -f
```

- Display logs by container name:

```
journalctl CONTAINER_NAME=<container-name> -f
```

1.7.5 Troubleshoot Licensing and `ntls`

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

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

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

In this section:

Get Licenses Information (v1)

Active Licenses

```
GET /v1/licenses.json
```

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

```
curl http://localhost:3185/v1/licenses.json -s | jq
```

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

Interpret the `.licenses[].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. You should check the `ntls` logs for additional information.
- (120; ∞) — the licensing source response has been timed out. Take action.
- `.licenses[].valid.valid == false`: connection with the licensing source was never established.

Another important field is `.products`. It describes the available features and resources for each product:

- `.products[].features`: available features.
- `.products[].resources`: available resource limits and usage of these resources.
- `.products[].extra`: miscellaneous license information.

The response also contains the following parameters:

- `.time`: the current server time, in Unix timestamp format.
- `.licenses[].expire_date`: license expiration date. Treat this field with respect to `.time`
- `.licenses[].type`: license type.
- `.licenses[].license_id`: license ID.
- `.licenses[].generated`: license generation time.
- `.licenses[].source`: the path to the license file.
- `.licenses[].products`: what features and resources each license provides. Basically it resembles structure of `.products`, but resources usage isn't reported here.
- `.licenses[].valid.description`: if `.licenses[].valid.valid == false`, then this field contains the reason why the license is not valid.
- `.services`: an array of FindFace Server services connected to `ntls`.

```
{
    "name": "NTLS",
    "time": 1709825691,
    "licenses": [
        {
            "expire_date": 1737208526,
            "generated": 1705586179,
            "last_updated": 2,
            "license_id": "018f4c50e6044f0aa618436ad9450ac0",
            "products": {
                "multi": {
                    "extra": {},
                    "features": {
                        "sec-external-vms": {
                            "value": true
                        },
                        "sec-genetec": {
                            "value": true
                        }
                    },
                    "resources": {}
```

(continues on next page)

(continued from previous page)

```
                },
                "server": {
                    "extra": {},
                    "features": {
                        "body_bags": {
                            "value": true
                        },
                        "body_emben": {
                            "value": true
                        },
                        ...
                    },
                    "resources": {
                        "cameras": {
                            "value": 60
                        },
                        "extapi": {
                            "value": 256
                        },
                        "objects_tntapi": {
                            "value": 200000000
                        }
                    }
                }
            },
            "source": "/ntech/license/Test_license_018f4c50e6044f0aa618436ad9450ac0.lic",
            "type": "offline",
            "valid": {
                "description": "",
                "valid": true
            }
        },
        {
            "expire_date": 0,
            "generated": 1581946976,
            "last_updated": 1,
            "license_id": "017c40d9401742ce9eff8b48d93ded9a",
            "products": {},
            "source": "/ntech/license/ntech_license_017c40d9401742ce9eff8b48d93ded9a.lic
↪",
            "type": "online",
            "valid": {
                "description": "License is not valid",
                "valid": false
            }
        }
    ],
    "products": {
        "multi": {
            "features": {
                "sec-external-vms": {
                    "value": true
```

(continues on next page)

(continued from previous page)

```
                },
                "sec-genetec": {
                    "value": true
                }
            },
            "resources": {},
            "extra": {}
        },
        "server": {
            "features": {
                "body_bags": {
                    "value": true
                },
                "body_emben": {
                    "value": true
                },
                ...
            },
            "resources": {
                "cameras": {
                    "current": 11,
                    "value": 60
                },
                "extapi": {
                    "current": 1,
                    "value": 256
                },
                "objects_tntapi": {
                    "current": 12321,
                    "value": 200000000
                }
            },
            "extra": {}
        }
    },
    "services": [
        { "name": "video-worker-gpu", "ip": "10.255.233.12:48844" },
        { "name": "findface-extraction-api", "ip": "10.255.233.11:44182" },
        { "name": "FindFace-tarantool", "ip": "10.255.233.13:46444" }
    ]
}
```

Report

```
GET /v1/usage-report.json
```

Get a report of the interval counters. It updates once an hour, so be patient.

```
{
    "b2c4658412f44184a6e34a2a369ce58c": { // license ID
        "by_interval": [ // array of statistical accounting intervals
            {
                "since": "2023-08-25T00:00:00+00:00", // start date of the interval,␣
↪inclusive
                "till": "2023-09-25T00:00:00+00:00", // date of the end of the interval,␣
↪not inclusive
                "counters": {
                    "ffserver": { // product
                        "face_emben": { // the code of the counter
                            "used": 123 // how many extracts are counted in the interval
                        }
                    }
                }
            },
            {
                "since": "2024-01-25T00:00:00+00:00", // start date of the interval,␣
↪inclusive
                "till": null, // the end of the interval is missing - until the end of␣
↪the license period
                "counters": {
                    "ffserver": {
                        "face_emben": {
                            "used": 0
                        }
                    }
                }
            }
        ],
        "active_limits": { // limits are active at the moment. It is not necessarily␣
↪match one of the intervals from by_interval
            "since": "2023-08-25T00:00:00+00:00", // the start date of the current limit
            "till": "2023-09-25T00:00:00+00:00", // the expiration date of the current␣
↪limits or NULL if the expiration date is until the end of the license
            "counters": {
                "ffserver": {
                    "face_emben": {
                        "used": 123,
                        "available": 1000
                    }
                }
            }
        }
    },
    "6b5392f68fa84861bb39ce86fa5e42f9": {
        "by_interval": [
```

(continues on next page)

(continued from previous page)

```
            {
                "since": "2021-09-18T00:00:00+00:00",
                "till": "2021-10-18T00:00:00+00:00",
                "counters": {
                    "ffserver": {
                        "face_emben": {
                            "used": 1103
                        }
                    }
                }
            },
            {
                "since": "2021-10-18T00:00:00+00:00",
                "till": null,
                "counters": {
                    "ffserver": {
                        "face_emben": {
                            "used": 338
                        }
                    }
                }
            }
        ],
        "active_limits": {
            "since": "2021-10-18T00:00:00+00:00",
            "till": null,
            "counters": {
                "ffserver": {
                    "face_emben": {
                        "used": 338,
                        "available": 1000
                    }
                }
            }
        }
    }
}
```

Get License Information (v0)

Active License

```
GET /license.json
```

This route is deprecated, and you should always use `GET /v1/licenses.json` instead. It can only show information about one license, and if there are multiple licenses — the one that expires last is selected. However, it still combines information about the features and resources provided by each license.

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

(continues on next page)

(continued from previous page)

```
{
    "name": "NTLS",
    "time": 1706794854,
    "type": "online",
    "license_id": "110d47188a62497888184e58cb5b08de",
    "generated": 1588854459,
    "last_updated": 5,
    "valid": {
        "value": true,
        "description": ""
     },
    "source": "/ntech/license/import_
↪5d679839ce7993aaa0a0e0cb797c2eb2bd7c0a3e66f0c2a319032c4d99c97322.lic",
    "limits": [
        {
            "type": "time", "name": "end",
            "value": 1746422763
        },
        {
            "type": "number", "name": "faces",
            "value": 250000,
            "current": 172585
        },
        {
            "type": "number", "name": "cameras",
            "value": 25,
            "current": 4
        },
        {
            "type": "number", "name": "extraction_api",
            "value": 64,
            "current": 1
        },
        {
            "type": "boolean", "name": "gender",
            "value": true
        },
        ...
    ],

    "products": {
        "multi": {
            "extra": {"line-crossing": true},
            "features": {
                "sec-external-vms": {
                    "value": true
                },
                "sec-genetec": {
                    "value": true
                }
            },
            "resources": {}
```

(continues on next page)

(continued from previous page)

```
        },
        "server": {
            "extra": {},
               "features": {
                "body_bags": {
                    "value": true
                },
                "body_emben": {
                    "value": true
                },
                ...
            },
            "resources": {
                "cameras": {
                    "current": 0,
                    "value": 60
                },
                "extapi": {
                    "current": 0,
                    "value": 256
                },
                "objects_tntapi": {
                    "current": 0,
                    "value": 200000000
                }
            }
        }
        "foo": {
            "extra": {
                "bar": 123,
                "baz": "qux",
            }
        }
    },

    "services": [
        { "name": "video-worker-gpu", "ip": "10.255.233.12:48844" },
        { "name": "findface-extraction-api", "ip": "10.255.233.11:44182" },
        { "name": "FindFace-tarantool", "ip": "10.255.233.13:46444" }
    ]
}
```

Report

```
GET /usage-report.json
```

Getting a report of interval counters. If there are multiple active licenses, it returns information on the one that expires last.

```
{
    "by_interval": [
        {
            "since": "2023-08-25T00:00:00+00:00",
            "till": "2023-09-25T00:00:00+00:00",
            "counters": {
                "ffserver": {
                    "nrq_facen": {
                        "used": 123
                    }
                }
            }
        },
        {
            "since": "2024-01-25T00:00:00+00:00",
            "till": null,
            "counters": {
                "ffserver": {
                    "nrq_facen": {
                        "used": 0
                    }
                }
            }
        }
    ],
    "active_limits": {
        "since": "2023-08-25T00:00:00+00:00",
        "till": "2023-09-25T00:00:00+00:00",
        "counters": {
            "ffserver": {
                "nrq_facen": {
                    "used": 123,
                    "available": 1000
                  }
            }
        }
    }
}
```

License Management

```
GET /c2v
```

Take a hardware fingerprint (C2V file) for subsequent activation of the offline Sentinel license.

```
GET /c2v/guardant
```

Take a hardware fingerprint (C2V file) for subsequent activation of the offline Guardant license.

```
POST /import
```

Importing a license (an alternative to copying a file to a directory with licenses).

Auxiliary

```
GET /metrics
```

Prometheus Metrics.

1.8 Storage API utilities

The utilities are not independent services, but serve as an entrypoint of the `sf-api` component.

1.8.1 Feature Vector Database Dump And Restore

Utilities for dump and restore tarantool databases are included to the `sf-api` component docker image.

In this section:

Database Dump

To get the information about command line parameters for `storage-api-dump` utility, run the following command:

```
docker run --rm -ti --entrypoint "/storage-api-dump" docker.int.ntl/ntech/universe/sf-
↪api:ffserver-11.240325 --help
```

The most important command line flags are the following:

Command line flags	Type	Description
`-config`	string	Path to config file.
`-config-template`	–	Output config template and exit.
`-continue-on-errors`	–	Continue on errors instead of exiting.
`-debug`	–	Enable debug logging
`-gallery`	string	Dump only galleries whose name match this regular expression (dump all if empty).
`-help`	–	Print help information.
`-output-dir`	string	Output directory (default ".").
`-recent-id`	uint	Last ID from previous run for incremental dump.

To run the `storage-api-dump` utility, do the following:

1. Create a default `storage-api-dump` configuration file.

```
docker run --rm -ti --entrypoint "/storage-api-dump" docker.int.ntl/ntech/universe/
↪sf-api:ffserver-11.240325 \
    --config-template > /opt/ffserver/configs/storage-api-dump.yaml
```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Modify section `storage-api -> shards` and setup actual values of shards:

```
...
storage-api:
    timeouts:
      connect: 5s
      response_header: 30s
      overall: 35s
      idle_connection: 10s
    max-idle-conns-per-host: 20
    keepalive: 24h0m0s
    trace: false
    shards:
    - master: http://tnt-1-1:8001/v2/
      slaves: []
    - master: http://tnt-1-2:8002/v2/
      slaves: []
...
```

3. Run `storage-api-dump` utility with the configuration file.

```
docker run --rm -ti --network server --entrypoint "/storage-api-dump" \
    --volume /opt/ffserver/configs/storage-api-dump.yaml:/storage-api-dump.yaml \
    --volume /opt/ffserver/dumps:/dumps \
    docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
    --config /storage-api-dump.yaml -output-dir /dumps
```

   Console output:

```
2024/05/28 15:23:02 6 Galleries in DB, going to dump 6:
2024/05/28 15:23:02  - default_0(default) (100 faces)
2024/05/28 15:23:02  - test_0(test) (100 faces)
```

(continues on next page)

(continued from previous page)

```
2024/05/28 15:23:02  - default_1(default) (100 faces)
2024/05/28 15:23:02  - test_1(test) (100 faces)
2024/05/28 15:23:02  - default_2(default) (100 faces)
2024/05/28 15:23:02  - test_2(test) (100 faces)
2024/05/28 15:23:02 Going to dump all of these
2024/05/28 15:23:02 Dumping default_0(default) from shard 0 with masterURL␣
↪http://tnt-1-1:8001/v2/
2024/05/28 15:23:02         52
2024/05/28 15:23:02 Dumping default_0(default) from shard 1 with masterURL␣
↪http://tnt-1-2:8002/v2/
2024/05/28 15:23:02        100
2024/05/28 15:23:02 Gallery default_0(default) last ID 99
2024/05/28 15:23:02 Dumping default_1(default) from shard 0 with masterURL␣
↪http://tnt-1-1:8001/v2/
2024/05/28 15:23:02         52
2024/05/28 15:23:02 Dumping default_1(default) from shard 1 with masterURL␣
↪http://tnt-1-2:8002/v2/
2024/05/28 15:23:02        100
2024/05/28 15:23:02 Gallery default_1(default) last ID 99
2024/05/28 15:23:02 Dumping default_2(default) from shard 0 with masterURL␣
↪http://tnt-1-1:8001/v2/
2024/05/28 15:23:02         52
2024/05/28 15:23:02 Dumping default_2(default) from shard 1 with masterURL␣
↪http://tnt-1-2:8002/v2/
2024/05/28 15:23:02        100
2024/05/28 15:23:02 Gallery default_2(default) last ID 99
2024/05/28 15:23:02 Dumping test_0(test) from shard 0 with masterURL http://
↪tnt-1-1:8001/v2/
2024/05/28 15:23:02         52
2024/05/28 15:23:02 Dumping test_0(test) from shard 1 with masterURL http://
↪tnt-1-2:8002/v2/
2024/05/28 15:23:02        100
2024/05/28 15:23:02 Gallery test_0(test) last ID 99
2024/05/28 15:23:02 Dumping test_1(test) from shard 0 with masterURL http://
↪tnt-1-1:8001/v2/
2024/05/28 15:23:02         52
2024/05/28 15:23:02 Dumping test_1(test) from shard 1 with masterURL http://
↪tnt-1-2:8002/v2/
2024/05/28 15:23:02        100
2024/05/28 15:23:02 Gallery test_1(test) last ID 99
2024/05/28 15:23:02 Dumping test_2(test) from shard 0 with masterURL http://
↪tnt-1-1:8001/v2/
2024/05/28 15:23:02         52
2024/05/28 15:23:02 Dumping test_2(test) from shard 1 with masterURL http://
↪tnt-1-2:8002/v2/
2024/05/28 15:23:02        100
2024/05/28 15:23:02 Gallery test_2(test) last ID 99
2024/05/28 15:23:02 Done
```

Database Restore

To get the information about command line parameters for `storage-api-restore` utility, run the following command:

```
docker run --rm -ti --entrypoint "/storage-api-restore" docker.int.ntl/ntech/universe/sf-
↪api:ffserver-11.240325 --help
```

The most important command line flags are the following:

Command line flags	Type	Description
`-config`	string	Path to config file.
`-config-template`	–	Output config template and exit.
`-debug`	–	Enable debug logging.
`-dont-create-gallery`	–	Don't create gallery, fail if doesn't exist.
`-help`	–	Print help information
`-rename`	string	Ignore dump header and use this string as gallery name.

To run the `storage-api-restore` utility, do the following:

1. Create a default `storage-api-restore` configuration file.

   ```
   docker run --rm -ti --entrypoint "/storage-api-restore" docker.int.ntl/ntech/
   ↪universe/sf-api:ffserver-11.240325 \
       --config-template > /opt/ffserver/configs/storage-api-restore.yaml
   ```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Modify section `storage-api -> shards` and setup actual values of shards:

   ```
   ...
   storage-api:
       timeouts:
         connect: 5s
         response_header: 30s
         overall: 35s
         idle_connection: 10s
       max-idle-conns-per-host: 20
       keepalive: 24h0m0s
       trace: false
       shards:
       - master: http://tnt-1-1:8001/v2/
         slaves: []
       - master: http://tnt-1-2:8002/v2/
         slaves: []
   ...
   ```

3. Run `storage-api-restore` utility with the configuration file.

   ```
   cat /opt/ffserver/dumps/test_0 | docker run --rm -i --network server --entrypoint "/
   ↪storage-api-restore" \
       --volume /opt/ffserver/configs/storage-api-restore.yaml:/storage-api-restore.
   ↪yaml \
       docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
       --config /storage-api-restore.yaml
   ```

Console output:

```
2024/05/28 16:10:48 No positional arguments specified, restoring from stdin
2024/05/28 16:10:48 Dump contains faces from "test_0"("test")
2024/05/28 16:10:48 Restoring into "test_0"("test")
2024/05/28 16:10:48 Gallery created
2024/05/28 16:10:48        100
2024/05/28 16:10:48 Waiting for workers (25 < 100)
```

1.8.2 Migration utility options

To migrate object feature vectors to another neural network model, you need the `sf-api-migrate` utility. You can find the detailed information on the usage in *this section*.

To get the information about command line parameters for `sf-api-migrate` utility, run the following command:

```
docker run --rm -ti --entrypoint "/sf-api-migrate" docker.int.ntl/ntech/universe/sf-
↪api:ffserver-11.240325 --help
```

Configuration options:

Command line flags	Type	Description
-config	string	Path to config file.
-config-template	–	Output config template and exit
-extraction-api-extraction-api	string	Extraction API address legacy argument – please switch to `url`.
-extraction-api-keepalive	duration	keep-alive connection timeout (default 24h0m0s).
-extraction-api-max-idle-conns-per-host	int	max idle keep-alive connections per host (default 20).
-extraction-api-timeouts-connect	duration	extraction-api-timeouts-connect (default 5s).
-extraction-api-timeouts-idle-connection	duration	extraction-api-timeouts-idle-connection (default 10s).
-extraction-api-timeouts-overall	duration	extraction-api-timeouts-overall (default 35s).
-extraction-api-timeouts-response-header	duration	extraction-api-timeouts-response-header (default 30s).
-extraction-api-trace	–	Enable HTTP tracing (extremely verbose, slows everything down considerably).
-extraction-api-url	string	Extraction API address (default `http://127.0.0.1:18666`).
-extraction-batch-size	int	Extraction api batch size(8 is recommended) (default 8).
-help	–	Print help information
-normalized-storage-enabled	–	Enables normalize saving (default true).
-normalized-storage-s3-access-key	string	Access key for the object storage.
-normalized-storage-s3-bucket-name	string	S3 storage bucket name.
-normalized-storage-s3-endpoint	string	S3 compatible object storage endpoint.
-normalized-storage-s3-operation-timeout	int	Storage operations (Get,Put,Delete) timeout in seconds (default 30).
-normalized-storage-s3-public-url	string	Storage public url.
-normalized-storage-s3-region	string	Storage region.
-normalized-storage-s3-secret-access-key	string	Secret key for the object storage.
-normalized-storage-s3-secure	–	If 'true' API requests will be secure (HTTPS), and insecure (HTTP) otherwise (default true).
-normalized-storage-webdav-keepalive	duration	keep-alive connection timeout (default 24h0m0s).
-normalized-storage-webdav-max-idle-conns-per-host	int	max idle keep-alive connections per host (default 20).
-normalized-storage-webdav-timeouts-connect	duration	normalized-storage-webdav-timeouts-connect (default 5s).
-normalized-storage-webdav-timeouts-idle-connection	duration	normalized-storage-webdav-timeouts-idle-connection (default 10s).

continues on next page

Table 1 – continued from previous page

Command line flags	Type	Description
`-normalized-storage-webdav-timeouts-overall`	duration	normalized-storage-webdav-timeouts-overall (default 35s).
`-normalized-storage-webdav-timeouts-response-header`	duration	normalized-storage-webdav-timeouts-response-header (default 30s).
`-normalized-storage-webdav-trace`		Enable HTTP tracing (extremely verbose, slows everything down considerably).
`-normalized-storage-webdav-upload-url`	string	webdav storage for normalized, disable normalized if empty string (default `http://127.0.0.1:3333/uploads/`).
`-normalized_storage`	string	Normalized storage type: webdav, s3 (default `webdav`).
`-objects`	value	Supported object types (default face, body, car).
`-objects-limit`	int	Get objects_limit objects from (default 1000).
`-storage-api-from-cooldown`	duration	Cooldown timeout after communication error (default 2s).
`-storage-api-from-galleries-read-slave-first`		Prefer slaves over master for get/list galleries requests.
`-storage-api-from-keepalive`	duration	keep-alive connection timeout (default 24h0m0s).
`-storage-api-from-max-idle-conns-per-host`	int	max idle keep-alive connections per host (default 20).
`-storage-api-from-max-slave-attempts`	int	Give up after trying to read from max_slave_attempts slaves (default 2).
`-storage-api-from-read-slave-first`		Prefer slaves over master for requests.
`-storage-api-from-read-slave-only`		Ignore master on read requests. If true: ReadSlaveFirst will be ignored.
`-storage-api-from-timeouts-connect`	duration	storage-api-from-timeouts-connect (default 5s).
`-storage-api-from-timeouts-idle-connection`	duration	storage-api-from-timeouts-idle-connection (default 10s).
`-storage-api-from-timeouts-overall`	duration	storage-api-from-timeouts-overall (default 35s).
`-storage-api-from-timeouts-response-header`	duration	storage-api-from-timeouts-response-header (default 30s).
`-storage-api-from-trace`		Enable HTTP tracing (extremely verbose, slows everything down considerably).
`-storage-api-to-cooldown`	duration	Cooldown timeout after communication error (default 2s).
`-storage-api-to-galleries-read-slave-first`		Prefer slaves over master for get/list galleries requests.
`-storage-api-to-keepalive`	duration	keep-alive connection timeout (default 24h0m0s).
`-storage-api-to-max-idle-conns-per-host`	int	max idle keep-alive connections per host (default 20).
`-storage-api-to-max-slave-attempts`	int	Give up after trying to read from max_slave_attempts slaves (default 2).
`-storage-api-to-read-slave-first`		Prefer slaves over master for requests.
`-storage-api-to-read-slave-only`		Ignore master on read requests. If true: ReadSlaveFirst will be ignored.
`-storage-api-to-timeouts-connect`	duration	storage-api-to-timeouts-connect (default 5s).
`-storage-api-to-timeouts-idle-connection`	duration	storage-api-to-timeouts-idle-connection (default 10s).
`-storage-api-to-timeouts-overall`	duration	storage-api-to-timeouts-overall (default 35s).
`-storage-api-to-timeouts-response-header`	duration	storage-api-to-timeouts-response-header (default 30s).
`-storage-api-to-trace`		Enable HTTP tracing (extremely verbose, slows everything down considerably).
`-workers-num`	int	Extract and save concurrent workers number(100+ is ok) (default 100).

The format of environment variable for flag `-my-flag` is `CFG_MY_FLAG`.

Priority:

1. Defaults from source code (lowest priority).
2. Configuration file.
3. Environment variables.
4. Command line.

1.8.3 Simple Face API migrator from v2 to v3

The `sf-api-migrate-v2-v3` utility is used for galleries migration from different API versions. The utility will rename all old gallery names to the new format. All v3 gallery names will be saved.

To get the information about command line parameters for `sf-api-migrate-v2-v3` utility, run the following command:

```
docker run --rm -ti --entrypoint "/sf-api-migrate-v2-v3" docker.int.ntl/ntech/universe/
↪sf-api:ffserver-11.240325 --help
```

The most important command line flags are the following:

Command line flags	Type	Description
`-config`	string	Path to the config file.
`-config-template`	–	Output config template and exit.

To run the `sf-api-migrate-v2-v3` utility, do the following:

1. Create a default `sf-api-migrate-v2-v3` configuration file.

   ```
   docker run --rm -ti --entrypoint "/sf-api-migrate-v2-v3" docker.int.ntl/ntech/
   ↪universe/sf-api:ffserver-11.240325 \
       --config-template > /opt/ffserver/configs/storage-api-migrate-v2-v3.yaml
   ```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Modify section `storage-api -> shards` and setup actual values of shards:

   ```
   ...
   storage-api:
       timeouts:
         connect: 5s
         response_header: 30s
         overall: 35s
         idle_connection: 10s
       max-idle-conns-per-host: 20
       keepalive: 24h0m0s
       trace: false
       shards:
       - master: http://tnt-1-1:8001/v2/
         slaves: []
       - master: http://tnt-1-2:8002/v2/
         slaves: []
   ...
   ```

3. Run `sf-api-migrate-v2-v3` utility with the configuration file.

   ```
   docker run --rm -ti --network server --entrypoint "/sf-api-migrate-v2-v3" \
       --volume /opt/ffserver/configs/storage-api-migrate-v2-v3.yaml:/storage-api-
   ↪migrate-v2-v3.yaml \
       docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
       --config /storage-api-migrate-v2-v3.yaml
   ```

   Console output:

```
INFO[0000] FindFace Simple Face API migrator from v2 to v3 (version 11.
↪240325+158.gbfdcdab578) starting up
INFO[0000] Starting "face:test" gallery migration
INFO[0000] Gallery "face:test" doesn't need to be renamed
INFO[0000] Starting "default_0" gallery migration
INFO[0000] Renamed "default_0" gallery to "face:default_0"
INFO[0000] Starting "default_1" gallery migration
INFO[0000] Renamed "default_1" gallery to "face:default_1"
```

1.8.4 Resharding

The `storage-api-reshard` utility explores moving data from N old shards to M new shards, where M, N is the full shards count on installation. To get the information about command line parameters for `storage-api-reshard` utility, run the following command:

```
docker run --rm -ti --entrypoint "/storage-api-reshard" docker.int.ntl/ntech/universe/sf-
↪api:ffserver-11.240325 --help
```

The most important command line flags are the following:

Command line flags	Type	Description
`-config`	string	Path to config file.
`-config-template`	–	Output config template and exit.
`-delete-old`	–	Delete migrated faces on old shard (default true).
`-dry-run`	–	Do not add/delete faces, perform only read requests (default true).
`-gallery`	string	Gallery to reshard (default “history”).
`-help`	–	Print help information.
`-max-face-id`	uint	Max object id for execution (default 18446744073709551615).
`-min-face-id`	uint	Min object id for execution.
`-new-shards`	[]string	List of new shards url.
`-old-shards`	[]string	List of old shards url.
`-shard-idx`	int	Execute migration for old_shards[idx]. Required parameter. On one execution only 1 old shard can be migrated to new shards.

Warning: It’s recommended to create a *dump* before resharding.

To run the `storage-api-reshard` utility, do the following:

1. Create a default `storage-api-reshard` configuration file.

```
docker run --rm -ti --entrypoint "/storage-api-reshard" docker.int.ntl/ntech/
↪universe/sf-api:ffserver-11.240325 \
    --config-template > /opt/ffserver/configs/storage-api-reshard.yaml
```

   - `/opt/ffserver/configs/`: the directory on the host to store the configuration file.

2. Modify configuration file:

```
...
gallery: history
min_face_id: 0
max_face_id: 18446744073709551615
shard_idx: 0
dry_run: true
delete_old: true
old_shards:
- http://tnt-1-1:8001/v2/
- http://tnt-2-1:8001/v2/
new_shards: []
- http://tnt-1-1:8001/v2/
- http://tnt-2-1:8001/v2/
- http://tnt-3-1:8001/v2/
...
```

Important: `storage-api-resharder` makes decisions based solely on shard number, so the `new_shards` list should be either shortened version of `old_shards` when scaling down, or the same as `old_shards` plus a few new lines when scaling up.

3. Run `storage-api-reshard` utility with the configuration file in `dry_run` to check your configuration.

```
docker run --rm -ti --network server --entrypoint "/storage-api-reshard" \
    --volume /opt/ffserver/configs/storage-api-reshard.yaml:/storage-api-reshard.
↪yaml \
    docker.int.ntl/ntech/universe/sf-api:ffserver-11.240325 \
    --config /storage-api-reshard.yaml
```

Console output:

```
Config:
---
gallery: face:test
min_face_id: 0
max_face_id: 18446744073709551615
shard_idx: 0
dry_run: true
delete_old: true
old_shards:
- http://tnt-1-1:8001/v2
- http://tnt-2-1:8001/v2
new_shards:
- http://tnt-1-1:8001/v2
- http://tnt-2-1:8001/v2
- http://tnt-3-1:8001/v2
...
2024/06/06 06:58:47 SAR-BpLnfgDs | got faces(4 ... 6), will migrate 3/3
2024/06/06 06:58:47 SAR-BpLnfgDs | total:3 migrated:3 migrateRatio:1.00 duration:0.
↪00s speed:2138.16/s

2024/06/06 06:58:47 Done
```

4. Run the `storage-api-reshard` utility with the configuration file without `dry_run` mode for all shard idx for migration.

1.9 Appendices

1.9.1 Components in Depth

`extraction-api`

The `extraction-api` service uses neural networks to detect an object in an image and extract its feature vector. It also recognizes object attributes (for example, gender, age, emotions, beard, glasses, face mask - for face objects).

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

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

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

Functionality:

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

The `extraction-api` service can be based on CPU (installed from the `docker.int.ntl/ntech/universe/extraction-api-cpu` image) or GPU (installed from the `docker.int.ntl/ntech/universe/extraction-api-gpu` image). For both CPU- and GPU-accelerated services, configuration is done through environment variables and command line flags. You can also use the `extraction-api` configuration file. Its content varies subject to the acceleration type. You can find its default content `here for CPU` and `here for GPU` or by running docker command with a flag `-config-template`.

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

Command line flags	Type	Description
`-allow-cors`	–	Add CORS headers to allow cross-origin requests.
`-ascend-device`	uint	Ascend device ID on which to launch inference.
`-cache-dir`	string	Directory for GPU model's cache (default `/var/cache/findface/models_cache`).
`-config`	string	Path to config file.
`-config-template`	–	Output config template and exit.
`-debug`	–	Enable verbose logging.
`-detectors-instances`	int	DEPRECATED [use `detectors-max-batch-size`] Number of parallel detector instances.
`-detectors-max-batch-size`	int	Upper limit on detection batch size. When using the CPU, you can specify `max_batch_size: -1`, then the detector maximum batch size will correspond to the number of CPU cores (default 1).
`-extractors-instances`	int	DEPRECATED [use `extractors-max-batch-size`] Number of parallel extractor instances.
`-extractors-max-batch-size`	int	Upper limit on extraction batch size. When using the CPU, you can specify `max_batch_size: -1`, then the extractor maximum batch size will correspond to the number of CPU cores (default 1).
`-extractors-models`	value	Attribute models.
`-fetch-enabled`	–	Enable fetching from remote urls (default true).
`-fetch-size-limit`	int	File size limit (default 10485760).
`-gpu-device`	uint	GPU device ID on which to launch inference.
`-help`	–	Print help information.
`-license-ntls-server`	string	NTLS (ntechlab license server) address (default `127.0.0.1:3133`).
`-listen`	string	IP:port to listen on (default `:18666`).
`-max-dimension`	int	Maximum dimension (default 6000).
`-models-root`	string	Root directory for model files (.fnk) (default `/usr/share/findface-data/models`).
`-normalizers-instances`	int	DEPRECATED [use `normalizers-max-batch-size`] Number of parallel normalizer instances.
`-normalizers-max-batch-size`	int	Upper limit on normalization batch size. When using the CPU, you can specify `max_batch_size: -1`, then the normalizer maximum batch size will correspond to the number of CPU cores (default 1).
`-normalizers-models`	value	Map of facenkit normalizers and their parameters.
`-ticker-interval`	int	Interval between ticker lines in log in milliseconds (0 - disabled) (default 5000).

The format of environment variable for flag `-my-flag` is `CFG_MY_FLAG`.

Priority:

1. Defaults from source code (lowest priority).
2. Configuration file.
3. Environment variables.
4. Command line.

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

- *Enable Face and Face Attribute Recognition*.
- *Enable Face Liveness Detection*

- *Liveness Detection as Standalone Service*
- *Enable Body and Body Attribute Recognition*
- *Enable Vehicle and Vehicle Attribute Recognition*

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

`sf-api`

The `sf-api` service implements the HTTP API for the main functionality of the FindFace Server such as object detection and object recognition.

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

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

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

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

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

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

Functionality:

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

The `sf-api` configuration is done through environment variables and command line flags or through the `sf-api.yaml` configuration file. You can find its default content `here`.

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

Command line flags	Type	Description
`-cache`	string	Cache type: inmemory, redis or memcache (default `memcache`).
`-cache-inmemory-size`	int	Maximum number of items in ARC cache (default 16384).
`-cache-memcache-dns-cache-timeout`	duration	DNS cache timeout (default 1m0s).

continues on next page

Table 2 – continued from previous page

Command line flags	Type	Description
`-cache-memcache-nodes`	value	Comma-separated list of memcache shards (default `127.0.0.1:11211`).
`-cache-memcache-timeout`	duration	Specifies read/write timeout (default 100ms).
`-cache-redis-db`	int	Database to be selected after connecting to the server.
`-cache-redis-network`	string	Network type, either tcp or unix (default `tcp`).
`-cache-redis-nodes`	value	Array of Host:Port addresses (default `localhost:6379`).
`-cache-redis-password`	string	Optional password. Must match the password specified in the requirepass server configuration option.
`-cache-redis-timeout`	duration	Specifies dial/read/write timeout (default 5s).
`-config`	string	Path to config file.
`-config-template`		Output config template and exit.
`-debug`	–	Enable debug logging.
`-extraction-api-extraction-api`	string	Extraction API address legacy argument – please switch to '*url*'.
`-extraction-api-keepalive`	duration	Keep-alive connection timeout (default 24h0m0s).
`-extraction-api-max-idle-conns-per-host`	int	Max idle keep-alive connections per host (default 20).
`-extraction-api-timeouts-connect`	duration	extraction-api-timeouts-connect (default 5s).
`-extraction-api-timeouts-idle-connection`	duration	extraction-api-timeouts-idle-connection (default 10s).
`-extraction-api-timeouts-overall`	duration	extraction-api-timeouts-overall (default 35s).
`-extraction-api-timeouts-response-header`	duration	extraction-api-timeouts-response-header (default 30s).
`-extraction-api-trace`	–	Enable HTTP tracing (extremely verbose, slows everything down considerably).
`-extraction-api-url`	string	Extraction API address (default `http://127.0.0.1:18666`).
`-generate-openapi`	–	Flag for generating openapi docs and exit.
`-help`	–	Print help information.
`-limits-allow-return-facen`	–	Allow returning raw feature vectors to detect responses if `?return_facen=true` (v2) or `?return_emben=true` (v3).
`-limits-body-image-length`	int	Maximum length of image supplied in request body (default 33554432).
`-limits-deny-networks`	string	Comma-separated list of subnets that are not allowed to fetch from (default `127.0.0.0/8,192.168.0.0/16,10.0.0.0/8,::1/128,fe00::/8`).
`-limits-url-length`	int	Maximum supported url length in bytes (default 4096).
`-listen`	string	IP:port to listen on (default `:18411`).
`-normalized-storage-enabled`	–	Enables normalize saving (default true).
`-normalized-storage-s3-access-key`	string	Access key for the object storage.
`-normalized-storage-s3-bucket-name`	string	S3 storage bucket name.
`-normalized-storage-s3-endpoint`	string	S3 compatible object storage endpoint.
`-normalized-storage-s3-operation-timeout`	int	Storage operations (Get, Put, Delete) timeout in seconds (default 30).
`-normalized-storage-s3-public-url`	string	Storage public URL.
`-normalized-storage-s3-region`	string	Storage region.
`-normalized-storage-s3-secret-access-key`	string	Secret key for the object storage.
`-normalized-storage-s3-secure`	–	If `true` API requests will be secure (HTTPS), and insecure (HTTP) otherwise (default true).
`-normalized-storage-webdav-keepalive`	duration	Keep-alive connection timeout (default 24h0m0s).
`-normalized-storage-webdav-max-idle-conns-per-host`	int	Max idle keep-alive connections per host (default 20).
`-normalized-storage-webdav-timeouts-connect`	duration	normalized-storage-webdav-timeouts-connect (default 5s).
`-normalized-storage-webdav-timeouts-idle-connection`	duration	normalized-storage-webdav-timeouts-idle-connection (default 10s).
`-normalized-storage-webdav-timeouts-overall`	duration	normalized-storage-webdav-timeouts-overall (default 35s).
`-normalized-storage-webdav-timeouts-response-header`	duration	normalized-storage-webdav-timeouts-response-header (default 30s).
`-normalized-storage-webdav-trace`	–	Enable HTTP tracing (extremely verbose, slows everything down considerably).

continues on next page

Table 2 – continued from previous page

Command line flags	Type	Description
`-normalized-storage-webdav-upload-url`	string	Webdav storage for normalized, disable normalized if empty string (default `http://127.0.0.1:3333/uploads/`).
`-normalized_storage`	string	Normalized storage type: webdav, s3 (default `webdav`).
`-objects`	value	Supported object types (default face,body,car).
`-storage-api-cooldown`	duration	Cooldown timeout after communication error (default 2s).
`-storage-api-galleries-read-slave-first`	–	Prefer slaves over master for get/list galleries requests.
`-storage-api-keepalive`	duration	Keep-alive connection timeout (default 24h0m0s).
`-storage-api-max-idle-conns-per-host`	int	Maximum keep-alive connections per host (default 20).
`-storage-api-max-slave-attempts`	int	Give up after trying to read from max_slave_attempts slaves (default 2).
`-storage-api-read-slave-first`	–	Prefer slaves over master for requests.
`-storage-api-read-slave-only`	–	Ignore master on read requests. If true: ReadSlaveFirst will be ignored.
`-storage-api-timeouts-connect`	duration	storage-api-timeouts-connect (default 5s).
`-storage-api-timeouts-idle-connection`	duration	storage-api-timeouts-idle-connection (default 10s).
`-storage-api-timeouts-overall`	duration	storage-api-timeouts-overall (default 35s).
`-storage-api-timeouts-response-header`	duration	storage-api-timeouts-response-header (default 30s).
`-storage-api-trace`	–	Enable HTTP tracing (extremely verbose, slows everything down considerably).

The format of environment variable for flag `-my-flag` is `CFG_MY_FLAG`.

Priority:

1. Defaults from source code (lowest priority).
2. Config file.
3. Environment variables.
4. Command line.

`tntapi`

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

Tip: See Tarantool official documentation for details.

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

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

Functionality:

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

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

When configuring `tntapi`, refer to the following `box.cfg` parameters:

Parameter	Description
• `listen` • env `TT_LISTEN`	Binary host/port, used for admin operations and replication, e.g. `127.0.0.1:32001`.
• `work_dir` • env `TT_WORK_DIR`	Base working directory.
• `memtx_dir` • env `TT_MEMTX_DIR`	Directory for snapshots, can be relative to `work_dir`.
• `wal_dir` • env `TT_WAL_DIR`	Directory for xlogs, can be relative to `work_dir`.
• `memtx_memory` • env `TT_MEMTX_MEMORY`	Maximum RAM that can be used by a Tarantool shard. Set in bytes, depending on the number of objects the hard handles. Consult our experts by support@ntechlab.com before setting this parameter.
• `checkpoint_interval` • env `TT_CHECKPOINT_INTERVAL`	The interval between snapshots(), in seconds.
• `checkpoint_count` • env `TT_CHECKPOINT_COUNT`	Maximum number of snapshots that may exist on the `memtx_dir`.
• `force_recovery` • env `TT_FORCE_RECOVERY`	If `force_recovery` equals true, Tarantool tries to continue if there is an error while reading a snapshot file (at server instance start) or a write-ahead log file (at server instance start or when applying an update at a replica): skips invalid records, reads as much data as possible and lets the process finish with a warning.

See box.cfg for details.

Refer to the following parameters:

Parameter	Description
CFG_LISTEN_HOST	Default = 127.0.0.1. Host to public HTTP API.
CFG_LISTEN_PORT	Default = 8001. Port to public HTTP API.
CFG_NTLS	Default = "". IP address and port of the ntls license server.
CFG_LOG_REQUESTS	Default = true. Log requests.
CFG_FASTIDX_SEARCH_THREADS	Default = 1. Number of threads to search in the quick index.
CFG_ALLOW_META_SCHEME_MIGRATION	Default = false. Enable meta_scheme migration. Possible only when adding new fields to the schema, if the existing ones completely match.
CFG_META_SCHEME_MIGRATION_SINGLE_TRANSACTION	Default = true. If CFG_ALLOW_META_SCHEME_MIGRATION is enabled: • CFG_META_SCHEME_MIGRATION_SINGLE_TRANSACTION=true: migration of one gallery occurs in one transaction until all data has migrated to the new schema, there is no entry in xlog. Galleries with a large number of faces may require a significant amount of memory. • CFG_META_SCHEME_MIGRATION_SINGLE_TRANSACTION=false: before migration, a box.snapshot() is taken, a flag file is created in the wal_dir directory. In case of a crash during migration, it is necessary to delete all the contents of the wal_dir directory, the data will be downloaded from snapshot.
CFG_BATCH_SEARCH_MAX_SIZE	Default = 1. Maximum size of search batches. If you specify a value > 1, facen search for queries with the same other filters will be performed by batches of exactly this maximum size.
CFG_BATCH_SEARCH_MAX_DELAY_SECONDS	Default = 0.3. Maximum query timeout when batching a search (batch_search_max_size > 1). If exceeded, the search will be performed regardless of the simultaneous presence of batch_search_max_size of similar search queries.
CFG_EXTRA_LUA	Default = "". Additional LUA code that will be executed during initialization.
CFG_METRICS_HTTP_BUCKETS	Buckets for the histogram of query times in metrics, for example: CFG_METRICS_HTTP_BUCKETS="[0.2, 0.5, 1, 5]"

The database structure is set via the /opt/ffserver/configs/tnt-schema.lua file. You will have to *manually set* it via the configuration file.

upload

The upload component is an NginX-based web server used as a storage for normalized object images. If Video Recorder is *deployed*, upload can be used to store video data from cameras.

Tip: See NginX official documentation for details.

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

Video Object Detection: `video-manager` and `video-worker`

In this section:

Functions of `video-manager`

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

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

- It supplies `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 `video-worker` instances.

Note: The configuration settings passed via jobs have priority over the `video-manager.yaml` configuration file.

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

Functionality:

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

Functions of `video-worker`

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

The `video-worker` service interfaces with the `video-manager` and router services (e.g. `facerouter`) as follows:

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

Functionality:

- detects objects in the video,
- normalizes images of objects,
- tracking objects in real time and posting the best object snapshot.

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

- **`Motion detection`**. Used to reduce resource consumption. Only when the motion detector recognizes the motion of certain intensity that the object tracker can be triggered.
- **`Object tracking`**. The object tracker traces, detects, and captures objects in the video. It can simultaneously be working with several objects. It also searches for the best object snapshot using the embedded neural network. After the best object snapshot is found, it is posted to `facerouter`.

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

- Real-time
- Offline

Real-Time Mode

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

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

Offline Mode

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

Configure Video Object Detection

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

1. The `video-manager` configuration file can be built by command line flag `-config-template`. You can find its default content `here`.

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

Command line flags	Type	Description
`-config`	string	Path to config file.
`-config-template`	–	Output config template and exit.
`-etcd-dial-timeout`	duration	Timeout for failing to establish a connection (default 3s).
`-etcd-endpoints`	string	List of URLs, separated by comas (default `127.0.0.1:2379`).
`-etcd-key-prefix`	string	Prefix for several sharded managers.
`-exp-backoff-enabled`	–	Enabled.
`-exp-backoff-factor`	float	Factor for increasing delay (default 2).
`-exp-backoff-flush-interval`	duration	Flush delay to min_delay if now()-last_NOT_STARTED > x (default 2m0s).
`-exp-backoff-max-delay`	duration	Maximum delay (default 1m0s).
`-exp-backoff-min-delay`	duration	Initial delay (default 1s).
`-generate-openapi`		Flag for generating openapi docs and exit.
`-help`	–	Print help information.
`-job-scheduler-script`	string	Lua script to schedule jobs.
`-job-status-change-script`	string	Lua script run on job status change.
`-kafka-enabled`	–	Enabled.
`-kafka-endpoints`	string	List of URLs, separated by comas (default `127.0.0.1:9092`).
`-listen`	string	IP:port to listen on (HTTP server) (default `:18810`).
`-master-lease-ttl`	int	Lease time-to-live, in seconds (default 10).
`-master-self-url`	string	Self url (default `127.0.0.1:18811`).
`-master-self-url-http`	string	Self url http (default `127.0.0.1:18810`).
`-ntls-enabled`	–	Check limits on `ntls`. If true, `video-manager` will send a job to `video-worker` only if the total number of processed cameras does not exceed the allowed number of cameras from the license.
`-ntls-update-interval`	duration	`ntls` update interval (default 1m0s).
`-ntls-url`	string	URL of `ntls`, ui port (default `http://127.0.0.1:3185/`).
`-router-events-url`	string	Facerouter events URL.
`-router-url`	string	Facerouter URL to receive detected objects from `video-worker` (default `http://127.0.0.1:18820/v0/frame`).
`-rpc-heart-beat-timeout`	duration	rpc-heart-beat-timeout (default 4s).
`-rpc-listen`	string	Listen on, IP:PORT (default `127.0.0.1:18811`).

The format of environment variable for flag `-my-flag` is `CFG_MY_FLAG`.

Priority:

1. Defaults from source code (lowest priority).
2. Configuration file.
3. Environment variables.
4. Command line.

The following parameters are available for configuration `stream_settings`:

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

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

Option	Description
`filter_min_quality`	Minimum threshold value for an object image quality. Default value: subject to the object type. Do not change the default value without consulting with our technical experts (support@ntechlab.com).
`filter_min_size`	Minimum size of an object in pixels. Calculated as the square root of the relevant bbox area. Undersized objects are not posted. Default value: 1.
`filter_max_size`	Maximum size of an object in pixels. Calculated as the square root of the relevant bbox area. Oversized objects are not posted. Default value: 8192.
`roi`	Enable posting objects detected only inside a region of interest WxH+X+Y. Default value: region not specified.
`fullframe_crop_rot`	Crop posted full frames by ROT. Default value: false.
`fullframe_use_png`	Send full frames in PNG and not in JPEG as set by default. Do not enable this parameter without supervision from our team as it can affect the entire system functioning. Default value: false (send in JPEG).
`jpeg_quality`	Quality of an original frame JPEG compression, in percents. Default value: 95%.
`overall_only`	Enables the offline mode for the best object search. Default value: true (CPU), false (GPU).
`realtime_post_first_immediately`	Enable posting an object image right after it appears in a camera field of view (real-time mode). Default value: false.
`realtime_post_interval`	Only for the real-time mode. Defines the time period in seconds within which the object tracker picks up the best snapshot and posts it to `facerouter`. Default value: 1.
`realtime_post_every_interval`	Only for the realtime mode. Post best snapshots obtained within each `realtime_post_interval` time period. If false, search for the best snapshot dynamically and send snapshots in order of increasing quality. Default value: false.
`track_interpolate_bboxes`	Interpolate missed bboxes of objects in track. For example, if frames #1 and #4 have bboxes and #2 and #3 do not, the system will reconstruct the absent bboxes #2 and #3 based on the #1 and #4 data. Enabling this option allows you to increase the detection quality on account of performance. Default value: true.
`track_miss_interval`	The system closes a track if there has been no new object in the track within the specified time (seconds). Default value: 1.
`track_overlap_threshold`	Tracker IoU overlap threshold. Default value: 0.25.
`track_max_duration_frames`	The maximum approximate number of frames in a track after which the track is forcefully completed. Enable it to forcefully complete "eternal tracks," for example, tracks with objects from advertisement media. The default value: 0 (option disabled).
`track_send_history`	Send track history. Default value: false.
`post_best_track_frame`	Send full frames of detected objects. Default value: true.
`post_best_track_normalize`	Send normalized images for detected objects. Default value: true.
`post_first_track_frame`	Post the first frame of a track. Default value: false.
`post_last_track_frame`	Post the last frame of a track. Default value: false.
`tracker_type`	Tracker type (simple_iou or deep_sort). Default value: `simple_iou`.
`track_deep_sort_matching_threshold`	Track features matching threshold (confidence) for deep sort tracker. Default value: `0.65`.
`track_deep_sort_filter_unconfirmed_tracks`	Filter unconfirmed (too short) tracks in deep sort tracker. Default value: true.
`track_object_is_principal`	Track by this object in N-in-1 detector tracker. Default value: false.
`track_history_active_track_miss_interval`	Don't count track as active if N seconds have passed, only if `track_send_history=true`. Default value: `0`.
`filter_track_min_duration_frames`	Post only if object track length at least N frames. Default value: 1.
`extractors_track_triggers`	Track triggers that trigger the extractor.

2. The `video-worker` configuration file `video-worker-cpu.yaml` or `video-worker-gpu.yaml`, subject to the acceleration type in use.

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

Command line flags	Type	Description
`-h`, `--help`	{bool}	Display help and exit (env: `CFG_HELP`).
`-c`, `--config`	{string}	Path to config file (env: `CFG_CONFIG`).
`-C`, `--config-template`	{bool}	Echo default config to stdout (env: `CFG_CONFIG_TEMPLATE`).
`--config-template-format`	{string}	`config-template` output format (yaml/env/ini) (env: `CFG_CONFIG_TEMPLATE_FORMAT`).
`-i`, `--input`	{string}	Read streams from file, do not use `video-manager` (env: `CFG_INPUT`).
`--exit-on-first-finished`	{bool}	Exit on first finished job, only when `--input` specified (env: `CFG_EXIT_ON_FIRST_FINISHED`).
`--batch-size`	{number}	Batch size (env: `CFG_BATCH_SIZE`).
`--metrics-port`	{number}	Http server port for metrics, 0=do not start server (env: `CFG_METRICS_PORT`).
`--resize-scale`	{double}	Rescale video frames with the given coefficient, if 1 – do not resize (env: `CFG_RESIZE_SCALE`).
`--capacity`	{number}	Maximum number of video streams to be processed by `video-worker`. (env: `CFG_CAPACITY`).
`--ntls-addr`	{string}	`ntls` server ip:port (env: `CFG_NTLS_ADDR`).
`--save-dir`	{string}	Debug: save objects to dir (env: `CFG_SAVE_DIR`).
`--resolutions`	{ResolutionsVector}	Preinit detector for specified resolutions: “640x480;1920x1080” (env: `CFG_RESOLUTIONS`).
`--strict-resolutions`	{bool}	Use `resolutions` as only possible values, others will rescale (env: `CFG_STRICT_RESOLUTIONS`).
`--labels`	{workerLabels}	`video-worker` labels: labels = k=v;group=enter (env: `CFG_LABELS`).
`--use-time-from-sei`	{bool}	Use timestamps from SEI packet (env: `CFG_USE_TIME_FROM_SEI`).
`--frame-buffer-size`	{number}	Reader frame buffer size (env: `CFG_FRAME_BUFFER_SIZE`).
`--events-queue-size`	{number}	Internal events queue size (env: `CFG_EVENTS_QUEUE_SIZE`).
`--skip-count`	{number}	Skip count (env: `CFG_SKIP_COUNT`).
`--mgr-cmd`	{string}	Command to obtain `video-manager` ‘s grpc ip:port (env: `CFG_MGR_CMD`).
`--mgr-static`	{string}	`video-manager` grpc ip:port (env: `CFG_MGR_STATIC`).
`--mgr-id-prefix`	{string}	`video-worker` ID prefix (env: `CFG_MGR_ID_PREFIX`).
`--streamer-port`	{number}	Streamer/shots webserver port, 0=disabled (env: `CFG_STREAMER_PORT`).
`--streamer-url`	{string}	Streamer url to access this `video-worker` on `streamer_port` (env: `CFG_STREAMER_URL`).
`--streamer-tracks`	{bool}	Use tracks instead detects for streamer (env: `CFG_STREAMER_TRACKS`).
`--streamer-tracks-last`	{bool}	Use tracks with lastFrameId=currentFrameId (.tracks must be true) (env: `CFG_STREAMER_TRACKS_LAST`).
`--streamer-max-backpressure`	{number}	Max backpressure for client connection (bytes) (env: `CFG_STREAMER_MAX_BACKPRESSURE`).
`--liveness-fnk`	{string}	Path to liveness fnk (env: `CFG_LIVENESS_FNK`).
`--liveness-norm`	{string}	Path to normalization for liveness (env: `CFG_LIVENESS_NORM`).
`--liveness-batch-size`	{number}	Liveness batch size (env: `CFG_LIVENESS_BATCH_SIZE`).
`--liveness-interval`	{double}	Liveness internal algo param (env: `CFG_LIVENESS_INTERVAL`).
`--liveness-stdev-cnt`	{number}	Liveness internal algo param (env: `CFG_LIVENESS_STDEV_CNT`).
`--imotion-shared-decoder`	{bool}	Use shared decoder for imotion (experimental) (env: `CFG_IMOTION_SHARED_DECODER`).
`--send-threads`	{number}	Posting threads (env: `CFG_SEND_THREADS`).

continues on next page

Table 3 – continued from previous page

Command line flags	Type	Description
`--send-queue-limit`	{number}	Posting maximum queue size (env: `CFG_SEND_QUEUE_LIMIT`).
`--send-disable-drops`	{bool}	Disable send queue drops (env: `CFG_SEND_DISABLE_DROPS`).
`--recorder-enabled`	{bool}	Video recording enabled (env: `CFG_RECORDER_ENABLED`).
`--recorder-chunk-size`	{number}	Maximum size of video recording chunks (env: `CFG_RECORDER_CHUNK_SIZE`).
`--recorder-storage-dir`	{string}	Absolute path to the temporary storage folder (env: `CFG_RECORDER_STORAGE_DIR`).
`--recorder-video-storage-url`	{string}	Video storage api url (env: `CFG_RECORDER_VIDEO_STORAGE_URL`).
`--recorder-video-storage-threads`	{number}	Posting threads for uploads & requests (env: `CFG_RECORDER_VIDEO_STORAGE_THREADS`).
`--recorder-video-storage-timeout`	{double}	Video storage api requests timeout, seconds (env: `CFG_RECORDER_VIDEO_STORAGE_TIMEOUT`).
`--recorder-video-storage-max-retries`	{number}	Number of retries on request failure (env: `CFG_RECORDER_VIDEO_STORAGE_MAX_RETRIES`).
`--models-cache-dir`	{string}	Path to cache directory (env: `CFG_MODELS_CACHE_DIR`).
`--models-detectors`	{CfgDetectors}	Detectors (env: `CFG_MODELS_DETECTORS`).
`--models-normalizers`	{CfgNormalizer}	Normalizers (env: `CFG_MODELS_NORMALIZERS`).
`--models-extractors`	{CfgExtractor}	Extractors (env: `CFG_MODELS_EXTRACTORS`).
`--models-objects`	{CfgObject}	Objects (env: `CFG_MODELS_OBJECTS`).
`--face-min-size`	{number}	DEPRECATED [use models-detectors] detector param (env: `CFG_FACE_MIN_SIZE`).
`--face-detector`	{string}	DEPRECATED [use models-detectors] path to face detector (env: `CFG_FACE_DETECTOR`).
`--face-norm`	{string}	DEPRECATED [use models-objects/models-normalizers] path to normalizer (usually crop2x) (env: `CFG_FACE_NORM`).
`--face-quality`	{string}	DEPRECATED [use models-objects/models-extractors] path to face quality extractor (env: `CFG_FACE_QUALITY`).
`--face-norm-quality`	{string}	DEPRECATED [use models-extractors] path to face quality normalizer (env: `CFG_FACE_NORM_QUALITY`).
`--face-track-features`	{string}	DEPRECATED [use models-objects/models-extractors] path to face track features extractor (env: `CFG_FACE_TRACK_FEATURES`).
`--face-track-features-norm`	{string}	DEPRECATED [use models-extractors] path to face track features normalizer (env: `CFG_FACE_TRACK_FEATURES_NORM`).
`--body-min-size`	{number}	DEPRECATED [use models-detectors] detector param (env: `CFG_BODY_MIN_SIZE`).
`--body-detector`	{string}	DEPRECATED [use models-detectors] path to body detector (env: `CFG_BODY_DETECTOR`).
`--body-norm`	{string}	DEPRECATED [use models-objects/models-normalizers] path to normalizer (usually crop2x) (env: `CFG_BODY_NORM`).
`--body-quality`	{string}	DEPRECATED [use models-objects/models-extractors] path to body quality extractor (env: `CFG_BODY_QUALITY`).
`--body-norm-quality`	{string}	DEPRECATED [use models-extractors] path to body quality normalizer (env: `CFG_BODY_NORM_QUALITY`).
`--body-track-features`	{string}	DEPRECATED [use models-objects/models-extractors] path to body track features extractor (env: `CFG_BODY_TRACK_FEATURES`).

continues on next page

Table 3 – continued from previous page

Command line flags	Type	Description
--body-track-features-norm	{string}	DEPRECATED [use models-extractors] path to body track features normalizer (env: CFG_BODY_TRACK_FEATURES_NORM).
--car-min-size	{number}	DEPRECATED [use models-detectors] detector param (env: CFG_CAR_MIN_SIZE).
--car-detector	{string}	DEPRECATED [use models-detectors] path to car detector (env: CFG_CAR_DETECTOR).
--car-norm	{string}	DEPRECATED [use models-objects/models-normalizers] path to normalizer (usually crop2x) (env: CFG_CAR_NORM).
--car-quality	{string}	DEPRECATED [use models-objects/models-extractors] path to car quality extractor (env: CFG_CAR_QUALITY).
--car-norm-quality	{string}	DEPRECATED [use models-extractors] path to car quality normalizer (env: CFG_CAR_NORM_QUALITY).
--car-track-features	{string}	DEPRECATED [use models-objects/models-extractors] path to car track features extractor (env: CFG_CAR_TRACK_FEATURES).
--car-track-features-norm	{string}	DEPRECATED [use models-extractors] path to car track features normalizer (env: CFG_CAR_TRACK_FEATURES_NORM).

When configuring note the format of models-detectors, models-normalizers, models-extractors and models-objects:

1. `--models-detectors`: describes detectors.
   - format: `name1:/path/to/file.fnk,min_size=60;name2:file2.fnk,min_size=100;....`
   - `name1`: an unique name.
   - `/path/to/file.fnk`: a path to .fnk.
   - `min_size=60`: an optional parameter.

   Example:

```
CFG_MODELS_DETECTORS="bag:/usr/share/findface-data/models/detector/bag.gustav_
↪accurate.001.gpu.fnk;bear:/usr/share/findface-data/models/detector/bear.gustav_
↪accurate.001.gpu.fnk,min_size=100"
```

```
models:
  detectors:
    bag:
      fnk_path: /usr/share/findface-data/models/detector/bag.gustav_accurate.001.
↪gpu.fnk
      min_size: 60
    bear:
      fnk_path: /usr/share/findface-data/models/detector/bear.gustav_accurate.001.
↪gpu.fnk
      min_size: 100
```

2. `--models-normalizers`: describes normalizers, that can be used in `--models-extractors` and `--models-objects`.
   - format: `name1:/path/to/file.fnk;name2:file2.fnk;....`
   - `name1`: an unique name.
   - `/path/to/file.fnk`: a path to .fnk.

Example:

```
CFG_MODELS_NORMALIZERS="face_norm:/usr/share/findface-data/models/facenorm/crop2x.
↪v2_maxsize400.gpu.fnk;face_quality_norm:/usr/share/findface-data/models/facenorm/
↪crop1x.v2_maxsize400.gpu.fnk"
```

```
models:
  normalizers:
    face_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/crop2x.v2_maxsize400.gpu.
↪fnk
    face_quality_norm:
      fnk_path: /usr/share/findface-data/models/facenorm/crop1x.v2_maxsize400.gpu.
↪fnk
```

3. `--models-extractors`: describes extractors, that can be used in `--models-objects`.

   - format: `name1:/path/to/file.fnk,normalizer=normName;....`
   - `name1`: an unique name.
   - `/path/to/file.fnk`: a path to .fnk.
   - `normalizer=normName`: must be described in `--models-normalizers`
   - `batch_size=123`: an optional parameter sets the size of the batch for this extractor.

   Example:

```
CFG_MODELS_EXTRACTORS="face_quality:/usr/share/findface-data/models/faceattr/
↪quality_fast.v1.gpu.fnk,normalizer=face_quality_norm,batch_size=512"
```

```
models:
  extractors:
    face_quality:
      fnk_path: /usr/share/findface-data/models/faceattr/quality_fast.v1.gpu.fnk
      normalizer: face_quality_norm
      batch_size: 512
```

4. `--models-objects`: describes objects.

   - format: `name1:normalizer=normName,quality=extractorName,track_features=extractorName;....`
   - `name1`: an object name.
   - `normalizer=normName`: how to normalize an object when sending it to `router_url`. `normName` must be described in `--models-normalizers`. It may be empty.
   - `quality=extractorName`: how to extract the quality of an object. `extractorName` must be described in `--models-extractors`. It may be empty.
   - `track_features=extractorName`: how to extract features for the DeepSortTracker. `extractorName` must be described in `--models-extractors`. It may be empty.

   Example:

```
CFG_MODELS_OBJECTS="face:normalizer=face_norm,quality=face_quality,track_features="
```

```
models:
  objects:
    face:
      normalizer: face_norm
      quality: face_quality
      track_features: ""
```

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

- *Enable Face Liveness Detection*
- *Enable Body and Body Attribute Recognition*
- *Enable Vehicle and Vehicle Attribute Recognition*

Jobs

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

You can find a job example `here` and view the parameters of the job *here*.

`ntls`

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

- NtechLab global license center if you opt for the online licensing, direct or via a proxy server.
- Guardant USB dongle if you opt for the on-premise licensing.
- Guardant or Sentinel services if you opt for the offline software licensing.

Use the web interface `http://<NTLS_IP_address>:3185/#/` to manage `ntls` in the following way:

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

The following components are licensable:

- `tntapi`,
- `extraction-api`,
- `video-worker`.

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

The `ntls` configuration is done through environment variables and command line flags. You can also use a configuration file and find its default content `here`.

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

Command line flags	Type	Description
`-h`, `--help`	{bool}	Display help and exit (env: `CFG_HELP`).
`-c`, `--config`	{string}	Path to config file (env: `CFG_CONFIG`).
`-C`, `--config-template`	{bool}	Echo default config to stdout (env: `CFG_CONFIG_TEMPLATE`).
`--config-template-format`	{string}	`config-template` output format (yaml/env/ini) (env: `CFG_CONFIG_TEMPLATE_FORMAT`).
`-l`, `--listen`	{networkAddress}	Address to accept incoming client connections (IP:PORT) (env: `CFG_LISTEN`).
`-L`, `--license-dir`	{string}	Directory where license files are stored (env: `CFG_LICENSE_DIR`).
`--proxy`	{optionalNetworkAddress}	(Optional) IP address and port of your specified proxy server. Use specified proxy (MUST support HTTP CONNECT method) to access global license server (IP:PORT). (env: `CFG_PROXY`).
`--ui`	{optionalNetworkAddress}	IP address from which accessing the `ntls` web interface must originate (IP:PORT). To allow access from any remote host, specify `0.0.0.0:3185`. (env: `CFG_UI`).
`--ui-dir`	{string}	Directory with UI static files (env: `CFG_UI_DIR`).

Priority:

1. Defaults from source code (lowest priority).
2. Configuration file.
3. Environment variables.
4. Command line.

Video Recorder: `video-storage` and `video-streamer-cpu`

Video Recorder is an optional part of the FindFace Server. It works in the following way:

- The `video-storage` service implements video chunk management. It takes video chunks from the `video-worker` component, puts them into the storage (`upload`), and writes their meta-information and whereabouts to the MongoDB database. By request, it issues info about existing video chunks and Websocket-links to the corresponding streams. These links are further used by `video-streamer-cpu` to deliver the video to a user for viewing and downloading.
- After receiving an API request, this service extracts the requested video chunks from `video-storage` and `video-worker` (only the last chunk if it's not yet recorded to the storage). Then it merges the video chunks into a one-piece video and delivers it to a user for viewing and downloading using WebSocket.

Configure Video Recorder

1. The `video-storage` configuration is done through environment variables or command line flags. You can also use a `video-storage.yaml` configuration file and find its default content `here`.

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

Command line flags	Type	Description
`-chunk-storage-localfs-directory`	string	The directory to store files.
`-chunk-storage-s3-access-key`	string	Access key for the object storage.
`-chunk-storage-s3-bucket-name`	string	S3 storage bucket name.
`-chunk-storage-s3-endpoint`	string	S3 compatible object storage endpoint.
`-chunk-storage-s3-operation-timeout`	int	Storage operations (Get, Put, Delete) timeout in seconds (default 30).
`-chunk-storage-s3-public-url`	string	Storage public url.
`-chunk-storage-s3-region`	string	Storage region.
`-chunk-storage-s3-secret-access-key`	string	Secret key for the object storage.
`-chunk-storage-s3-secure`	–	If `true` API requests will be secure (HTTPS), and insecure (HTTP) otherwise (default true).
`-chunk-storage-type`	string	Chunk storage type: inmemory, localfs, webdav, s3 (default `inmemory`).
`-chunk-storage-webdav-keepalive`	duration	Keep-alive connection timeout (default 24h0m0s).
`-chunk-storage-webdav-max-idle-conns-per-host`	int	Max idle keep-alive connections per host (default 20).
`-chunk-storage-webdav-timeouts-connect`	duration	chunk-storage-webdav-timeouts-connect (default 5s).
`-chunk-storage-webdav-timeouts-idle-connection`	duration	chunk-storage-webdav-timeouts-idle-connection (default 10s).
`-chunk-storage-webdav-timeouts-overall`	duration	chunk-storage-webdav-timeouts-overall (default 35s).
`-chunk-storage-webdav-timeouts-response-header`	duration	chunk-storage-webdav-timeouts-response-header (default 30s).
`-chunk-storage-webdav-trace`	–	Enable HTTP tracing (extremely verbose, slows everything down considerably).
`-chunk-storage-webdav-upload-url`	string	Webdav storage URL (default `http://127.0.0.1:3333/uploads/video_storage`).
`-config`	string	A path to a config file.
`-config-template`	–	Output config template and exit.
`-debug`	–	Enable debug logging.
`-external-address`	string	Used to access `video-storage` via external links (default `http://127.0.0.1:18611/`).
`-help`	–	Print help information.
`-listen`	string	IP:port to listen on (default `:18611`).
`-meta-storage-database`	string	meta-storage-database (default `video-storage`).
`-meta-storage-mongo-uri`	string	meta-storage-mongo-uri (default `mongodb://127.0.0.1`).
`-meta-storage-timings-connect`	duration	meta-storage-timings-connect (default 3s).
`-streamer-endpoints`	value	List of streamer endpoint URLs (default `127.0.0.1:9000`).

   The format of environment variable for flag `-my-flag` is `CFG_MY_FLAG`.

   Priority:

   1. Defaults from source code (lowest priority).
   2. Configuration file.
   3. Environment variables.
   4. Command line.

2. The `video-streamer-cpu` configuration is done through environment variables or command line flags, or through the `video-streamer-cpu.yaml` configuration file. You can find its default content `here`.

When configuring `video-streamer-cpu`, refer to the following parameters:

Command line flags	Type	Description
`-h`, `--help`	{bool}	Display this help and exit (env: `CFG_HELP`).
`-c`, `--config`	{string}	A path to a config file (env: `CFG_CONFIG`).
`-C`, `--config-template`	{bool}	Echo default config to stdout (env: `CFG_CONFIG_TEMPLATE`).
`--config-template-format`	{string}	config-template output format (yaml/env/ini) (env: `CFG_CONFIG_TEMPLATE_FORMAT`).
`--streamer-port`	{number}	Streamer port (env: `CFG_STREAMER_PORT`).
`--streamer-max-backpressure`	{number}	Max backpressure for client connection (bytes) (env: `CFG_STREAMER_MAX_BACKPRESSURE`).
`--streamer-io-buffer-size`	{number}	Muxer's io buffer size (bytes) (env: `CFG_STREAMER_IO_BUFFER_SIZE`).
`--video-storage-url`	{string}	`video-storage` REST-api url (env: `CFG_VIDEO_STORAGE_URL`).
`--video-storage-timeout`	{double}	`video-storage` api requests timeout, seconds (env: `CFG_VIDEO_STORAGE_TIMEOUT`).
`--cache-dir`	{string}	Absolute path to the temporary chunk storage folder (env: `CFG_CACHE_DIR`).

`counter`

The `counter` configuration is done through environment variables and command line flags. You can also use a configuration file `counter.yaml` and find its default content `here`.

When configuring `counter`, refer to the following parameters:

Command line flags	Type	Description
`-CounterDefaults-DeduplicationPeriod`	int	CounterDefaults-DeduplicationPeriod (default 10).
`-CounterDefaults-DeduplicationThreshold`	float	CounterDefaults-DeduplicationThreshold (default 0.75).
`-config`	string	Path to config file.
`-config-template`	–	Output config template and exit.
`-database-connection-string`	string	Postgresql connection string (default `dbname=ffcounter host=/var/run/postgresql sslmode=disable`)
`-help`	–	Print help information.
`-listen`	string	IP:port to listen on (default `:18300`).
`-sharded`	–	Multiple instance running or not (default true).

The format of environment variable for flag `-my-flag` is `CFG_MY_FLAG`.

Priority:

1. Defaults from source code (lowest priority).
2. Configuration file.
3. Environment variables.
4. Command line.

deduplicator

The deduplicator configuration is done through environment variables and command line flags. You can also use a deduplicator.yaml configuration file and find its default content here.

When configuring deduplicator, refer to the following parameters:

Command line flags	Type	Description
-collection-interval	duration	Group collection interval. Set to 0 if you don't want groups themselves to expire (default 10s).
-collection-step	duration	Delay between collecting groups. 0 means no delay (default 10ms).
-config	string	Path to config file.
-config-template	–	Output config template and exit.
-debug		Enable debug logging.
-help	–	Print help information.
-listen	string	IP:port to listen on (default :18310).

The format of environment variable for flag -my-flag is CFG_MY_FLAG.

Priority:

1. Defaults from source code (lowest priority).
2. Configuration file.
3. Environment variables.
4. Command line.

1.9.2 Fast Index

A fast index is an opportunity to speed up the emben (facen) search tens times as compared to the normal mode of operation. The index has 3 main parameters affecting accuracy and speed:

- "m (M)": the number of bi-directional links created for every new element during construction. Reasonable range of M is 2–100. Higher values of M work better on datasets with high intrinsic dimensionality and/or high recall, while low values of M work better on datasets with low intrinsic dimensionality and/or low recalls.
- "search_ef (ef)": the size of the dynamic list for the nearest neighbors (used during the search).
- "ef (ef_construction)": the parameter has the same meaning as search_ef, but controls the index_time/index_accuracy. Bigger value of "ef" leads to longer construction, but better index quality.

The values in brackets are values in the fast index library documentation. See ALGO_PARAMS.md for more information.

Live Index

The Live index is a gallery working mode, where new objects are immediately turned into the index, which is periodically saved to a disk, and the search always uses the index. It's possible when there aren't any other search filters than emben(facen).

Create gallery with live index parameters:

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

{
    "live_idx": {
        "enabled": true,
        "snapshot_path": "/tmp/idx.bin",
        "snapshot_interval_seconds": 10000,
        "snapshot_changes_count": 99,
        "initial_size": 100,
        "m": 4,
        "ef": 100,
        "search_ef": 100
    }
}
```

- `"enabled"`: enables live index, boolean.
- `"snapshot_path"`: path to the file with live index snapshot, string. The directory must already exist.
- `"snapshot_interval_seconds"`: the interval of the index snapshot creation, uint64_t.
- `"snapshot_changes_count"`: the count of added/removed `indexed` space objects, after which snapshot will be created, uint64_t.
- `"initial_size"`: the count of objects in the gallery, after that index will create, uint64_t.
- `"m"`, `"ef"`, `"search_ef"`: fast index parameters.

Important: Gallery creation returns `null` in case of success. To check the parameters of newly created gallery use `POST /v2/galleries/get/:name`.

Note: All numeric index creation parameters of gallery are checked for >=4 to protect from mistakes.

This example uses `tntapi` *API*. You can also perform the same operation via *sf-api*.

Warning: Live index does not work on replicas!

Index snapshots

The index is saved to the snapshot file (snapshot_path) either once every snapshot_interval_seconds seconds or after snapshot_changes_count changes made to the gallery (whichever happens first). All records will be moved from the space linear to the space indexed after saving the snapshot file. When restarting, the tntapi service tries to load index from snapshot_path and add to snapshot all new records (after last snapshot) from space linear. Then all records with tag deleted will be removed from the space indexed. Snapshot operations are blocking, no interaction with tntapi is possible while they are in progress.

Important: It is recommended to take snapshots every N object additions and limit the size of the gallery with live index.

Objects removing

Internally, removed records are only marked as "removed" but still occupy space in fast index. Having a large amount of removed records may reduce both performance and accuracy.

Important: If your use-case involves a lot of deletions, we recommend to organize your workflow in a way that allows you to discard galleries as a whole (i.e. for historical data use a gallery per day or per week and delete oldest gallery on a regular basis instead of deleting individual records, or, for static data, regularly rotate galleries by copying live records into a newly created empty gallery) to keep accumulation of removed records low.

To enable a live index, do the following:

1. Create a directory for live index snapshot.

```
sudo mkdir -p /opt/ffserver/tnt/001/{snapshots,xlogs,live_index}
```

2. Start the tntapi docker container.

```
docker run -tid --name tnt-1-1 --restart always --network server \
    --env CFG_LISTEN_HOST=0.0.0.0 \
    --env CFG_NTLS=ntls:3133 \
    --env TT_LISTEN=0.0.0.0:32001 \
    --env TT_MEMTX_MEMORY=$((1024 * 1024 * 1024)) \
    --volume /opt/ffserver/tnt/001-01:/opt/ntech/var/lib/tarantool/default \
    --publish 127.0.0.1:8001:8001 \
    docker.int.ntl/ntech/universe/tntapi:ffserver-11.240325
```

3. Create a gallery:

```
curl -D - -X POST -s 'http://localhost:8001/v2/galleries/add/testgal' --data '{
    "live_idx": {
        "enabled": true,
        "snapshot_path": "/opt/ntech/var/lib/tarantool/default/live_index/idx.bin",
        "snapshot_interval_seconds": 30,
        "snapshot_changes_count": 99,
        "initial_size": 100,
        "m": 4,
        "ef": 100,
        "search_ef": 100
```

(continues on next page)

(continued from previous page)

```
    }
}'

HTTP/1.1 201 Created
X-request-id: TN:77gGv1a1
Content-type: application/json
X-read-only: false
Content-length: 4
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)

null
```

4. Add more than 100 objects to the gallery `testgal`.
5. Check index snapshot on the path `/opt/ffserver/tnt/001/live_index/idx.bin`.

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

6. Send emben search request to the gallery and check response header `"X-search-stat"`:

```
HTTP/1.1 200 Ok
X-request-id: TN:EiQdsLeF
X-search-stat: batch_size:1, fastIndex:yes;
Content-type: application/json
X-read-only: false
Content-length: 1882
Connection: keep-alive
Server: Tarantool http (tarantool v2.10.4-2-gd536a7aa5)
```

CHAPTER

TWO

MULTI-OBJECT API

2.1 How to Use Multi-Object API

In this section:

2.1.1 Endpoint

Multi-Object API requests are to be sent to `http://<sf-api_IP_address>:18411/`. API requests are executed by the `sf-api` component.

2.1.2 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, v3 in `/v3/detect/`).

There are versions v2 and v3 at the moment. The most recent version is v3.

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

2.1.3 API Objects

Multi-Object API operates the multiple objects that represent detected physical objects. The most basic are:

- `face` – a human face
- `body` – a human silhouette
- `car` – a vehicle

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

Note: Different images of the same object are considered to be different API objects.

2.1.4 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 multi-object 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.

2.1.5 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://<sf-api_IP_address>:18411/docs/` for the API version /v3.

2.1.6 Limits

FindFace Server imposes the following limits.

Limit	Value
Image formats	JPG, PNG, WEBP, BMP
Maximum photo file size	To be configured via the `sf-api` configuration file.
Minimal size of an object	50x50 pixels
Maximum number of detected objects 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).

2.1.7 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 an object 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 `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 multi-object database not available.	503
`CACHE_ERROR`	Memcached not available.	503
`NOT_FOUND`	Matching objects not found.	404
`NOT_IMPLEMENTED`	This functionality not implemented.	501
`GALLERY_NOT_FOUND`	Matching galleries not found.	404

2.2 Multi-Object API Methods v3

In this section:

2.2.1 Detect Methods

Detect Objects on the Photo

```
POST /v3/detect
```

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

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

Query string parameters:

- `detector`: string, object detector to be applied to the image (see objects in the `extraction-api` configuration file).
- `cache`: boolean, if true, the formatted request to `extraction-api` will include such feature vectors as `face_emben`, `body_emben`, `car_emben` and `need_normalized` (obtain a normalized object image). The objects in `extraction-api` response will be saved in `memcached` under a ID. You can find this ID in the `id` field of the response. If not specified, the `sf-api` response will not contain feature vectors.
- `autorotate`: boolean, auto-rotates an original image to 4 different orientations and returns objects detected in each orientation.
- `roi`: string, a region of interest in the image, takes an array of integers in the format roi=top,left,right,bottom, (e.g. {“left”: 0, “right”: 1000, “top”: 0, “bottom”: 1000}).
- `return_emben`: boolean, returns a object feature vector in the response. Requires the enabled `allow-return-facen` flag in the `sf-api` configuration file. Requires the corresponding object in the `sf-api` configuration file in the `objects` section.
- `quality_estimator`: boolean, enables object quality estimation.

- attribute: array of strings in the format face_gender, face_age, face_emotions, face_beard, face_glasses3, enables recognition of the object features passed in the array.

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

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:

Returns a temporary object that can be stored in the cashe memcache, redis, or sf-api memory. It contains:

- temporary ID of the detection result (id, if cache enabled);

  Important: When you form a request, be sure to check the relevance of the temporary ID before you refer to it as it tends to become irrelevant with time. If so, re-detect the object.

- bbox: list of coordinates of the rectangles around the detected objects;
- score: algorithm confidence in the result.
- attributes (if passed): feature vector (if return_emben enabled) and attributes specified in the request parameters with extractor, model and result:
  - age (if face_age passed): number of years;
  - emotions (if face_emotions passed): 6 basic emotions + neutral (angry, disgust, fear, happy, sad, surprise, neutral) with algorithm confidence in each emotion expression;
  - gender (if face_gender passed): male or female, with model and algorithm confidence in the result ("confidence");
  - other attributes (if passed): beard (beard), glasses (sun, eye, or none), along with algorithm confidence in the result;
- orientation: EXIF orientation of the photo.

Examples

Request

```
curl -i -X POST 'http://127.0.0.1:18411/v3/detect?cache=on&attribute=face_gender&
↪attribute=face_age&attribute=face_emotions&attribute=face_glasses3' -H 'Content-Type:␣
↪image/jpeg' --data-binary @sample.jpg
```

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

{
    "objects": {
        "face": [
            {
                "id": "bjj8mlhjisgjrk6hj1v0",
                "bbox": {"left": 595, "top": 127, "right": 812, "bottom": 344},
                "score": 0.9171224,
                "attributes": {
                    "face_age": {"extractor": "face_age", "model": "age.v1", "result":␣
↪24},
                    "face_emotions": {
                        "extractor": "face_emotions",
                        "model": "emotions.v1",
                        "result": [
                            {"confidence": 0.9998919, "name": "neutral"},
                            {"confidence": 9.195882e-05, "name": "sad"},
                            {"confidence": 1.4304824e-05, "name": "happy"},
                            {"confidence": 1.8077538e-06, "name": "surprise"},
                            {"confidence": 4.2787672e-08, "name": "angry"},
                            {"confidence": 2.0801991e-08, "name": "disgust"},
                            {"confidence": 1.9873138e-11, "name": "fear"}
                        ]
                    },
                    "face_gender": {
                        "extractor": "face_gender",
                        "model": "gender.v2",
                        "result": [
                            {"confidence": 0.9999119, "name": "female"},
                            {"confidence": 8.809325e-05, "name": "male"}
                        ]
                    },
                    "face_glasses3": {
                        "attribute": "face_glasses3",
                        "model": "glasses3.v0",
                        "result": [
                            {"confidence": 0.99958307, "name": "none"},
                            {"confidence": 0.00033243417, "name": "eye"},
                            {"confidence": 8.4465064e-05, "name": "sun"}
                        ]
                    }
                }
            }
        ]
    },
    "orientation": 1
```

(continues on next page)

(continued from previous page)

```
}
```

Get Detection Result from the Cache by ID

```
GET /v3/detect/:id
```

This method retrieves the detection results from the cache by their temporary ID (including feature vectors of the detected objects).

Parameters in path segments:

- `:id`: string, the detection result temporary ID in the cache.

Returns:

JSON representation of the detection result.

Example

Request

```
curl -i -X GET 'http://127.0.0.1:18411/v3/detect/bg2gu31jisghl6pee09g'
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:HlQBSnTu
Date: Wed, 05 Dec 2018 07:51:04 GMT
Content-Length: 477

{
    "face": {
        "id": "bjj8mlhjisgjrk6hj1v0",
        "bbox": {"left": 595, "top": 127, "right": 812, "bottom": 344},
        "score": 0.9171224,
        "attributes": {
            "face_age": {"extractor": "face_age", "model": "age.v1", "result": 24},
            "face_emotions": {
                "extractor": "face_emotions",
                "model": "emotions.v1",
                "result": [
                    {"confidence": 0.9998919, "name": "neutral"},
                    {"confidence": 9.195882e-05, "name": "sad"},
                    {"confidence": 1.4304824e-05, "name": "happy"},
                    {"confidence": 1.8077538e-06, "name": "surprise"},
```

(continues on next page)

(continued from previous page)

```
                    {"confidence": 4.2787672e-08, "name": "angry"},
                    {"confidence": 2.0801991e-08, "name": "disgust"},
                    {"confidence": 1.9873138e-11, "name": "fear"}
                ]
            },
            "face_gender": {
                "extractor": "face_gender",
                "model": "gender.v2",
                "result": [
                    {"confidence": 0.9999119, "name": "female"},
                    {"confidence": 8.809325e-05, "name": "male"}
                ]
            },
            "face_glasses3": {
                "attribute": "face_glasses3",
                "model": "glasses3.v0",
                "result": [
                    {"confidence": 0.99958307, "name": "none"},
                    {"confidence": 0.00033243417, "name": "eye"},
                    {"confidence": 8.4465064e-05, "name": "sun"}
                ]
            }
        }
    }
}
```

Create Detection Result out of `extraction-api` Response

```
POST /v3/detect/premade
```

This method creates a detection result out of a `extraction-api` response. Requires `allow-return-facen: true` in `sf-api` configuration file.

Parameters in path segments:

The method doesn't accept any parameters.

Parameters in request body:

Request body contains the object in a format similar to the response `GET /v3/detect/:id: { "object type": object from the extraction-api response }`

Returns:

JSON with "id" on success.

Example

Request

```
curl -i -X POST 'http://127.0.0.1:18411/v3/detect/premade' -H 'Content-Type: application/
↪json' --data-binary '@extapi-face.json'
```

Response:

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

{"id": "bg2gu31jisghl6peea9g"}
```

Get Normalized Detect by ID

```
GET /v3/detect/:id/normalized
```

This method retrieves normalized detect by ID.

Parameters in path segments:

- `:id`: ID.

Returns:

Normalized object images on success.

Example

Request

```
curl -i 'http://127.0.0.1:18411/v3/detect/bg2gu31jisghl6pee09g/normalized'
```

Response:

```
HTTP/1.1 200 OK
Content-Type: image/png
X-Request-Id: SF:mjv3Lyvf
Date: Thu, 09 Apr 2020 15:59:48 GMT
Transfer-Encoding: chunked

Warning: Binary output can mess up your terminal. Use "--output -" to tell
Warning: curl to output it to your terminal anyway, or consider "--output
Warning: <FILE>" to save to a file.
```

2.2.2 Compare Two Objects

```
GET /v3/verify
```

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

Query string parameters:

- `"obj1"`: the first object, either a detection result (ID of the result from the `POST /v3/detect` method), or one from the biometric database (in the format `persistent:<objtype>/<gallery>/<id>` or `emben:[model:]<base64 emben>` if `allow-return-facen:true`).
- `"obj2"`: the second object, from the same possible sources as the first object.

Returns:

Algorithm confidence that the objects match.

Example

Request #1. Compare 2 detection results

```
curl -s 'http://127.0.0.1:18411/v3/verify?obj1=detection%3Acp1nh3k8kuic72tu0ljg%22&
↪obj2=detection%3Acp1nh3k8kuic72tu0ljg%22' | jq
```

Response

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

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

```
curl -i 'http://127.0.0.1:18411/v3/verify?obj1=detection:bg2gu31jisghl6pee09g&
↪obj2=persistent:face/history/4141715733352439863' | jq
```

Response

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

2.2.3 Gallery Methods

List Database Galleries

```
GET /v3/galleries/
```

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

```
curl -i -X GET 'http://127.0.0.1:18411/v3/galleries/'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:BjI7i3GZ
Date: Wed, 05 Dec 2018 08:19:55 GMT
Content-Length: 125

{
    "galleries": {
        "face": [
            { "objects": 22, "name": "history" },
```

(continues on next page)

(continued from previous page)

```
            { "objects": 1, "name": "test" },
            { "objects": 0, "name": "hello" }
        ],
        "car": [
            { "objects": 11, "name": "test" }
        ],
        "body": [
            { "objects": 51, "name": "entrance" }
        ]
    }
}
```

Create a Gallery

```
POST /v3/galleries/:objtype/:gallery
```

This method creates a gallery under a given name.

Parameters in path segments:

- `:objtype`: string, an object type.
- `:gallery`: string, 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.
- `space`: name of the space in the tarantool, passed in the body. Optional parameter.
- `live_idx`: parameters for Live index in the tarantool, passed to the body. Optional parameter.

Returns:

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

Example

Request

```
curl -i -X POST 'http://127.0.0.1:18411/v3/galleries/face/hello' -d '{"space":"some_
↪space_name"}'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:whPaBcoV
Date: Wed, 05 Dec 2018 08:16:20 GMT
Content-Length: 2

{}
```

If the gallery already exists, 409 will be returned.

Request

```
curl -i -X POST 'http://127.0.0.1:18411/v3/galleries/face/hello'
```

Response

```
HTTP/1.1 409 Conflict
Content-Type: application/json
X-Request-Id: SF:9S0gvzfM
Date: Wed, 05 Dec 2018 08:15:00 GMT
Content-Length: 62

{ "code": "CONFLICT", "desc": "Gallery \"hello\" already exists" }
```

Retrieve Gallery Details

```
GET /v3/galleries/:objtype/:gallery
```

This method checks the existence of the gallery and request the number of objects in it.

Parameters in path segments:

- `:objtype`: string, an object type.
- `:gallery`: string, a gallery's name.

Returns:

- JSON dictionary with the number of objects 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/v3/galleries/face/hello'
```

Response

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

{"name":"ffsec_face_events","emben_model":"kiwi_320","emben_size":320,"objects":474}
```

Delete Gallery

```
DELETE /v3/galleries/:objtype/:gallery
```

This method deletes the gallery and all the objects in it.

Parameters in path segments:

- `:objtype`: string, the name of the object to be deleted.
- `:gallery`: string, the name of the gallery to be deleted.

Returns:

- Empty JSON dictionary on success.
- JSON with a relevant error description on failure (404 if the gallery didn't exist).

Example

Request

```
curl -i -X DELETE 'http://127.0.0.1:18411/v3/galleries/face/hello'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:oJh9FEtX
Date: Wed, 05 Dec 2018 08:15:04 GMT
Content-Length: 2

{}
```

Rename the Gallery

```
PATCH /v3/galleries/:objtype/:gallery
```

This method renames the gallery.

Parameters in path segments:

- `:objtype`: string, the name of the object to be renamed.
- `:gallery`: string, the name of the gallery to be renamed.

Parameters in request body:

`name`: string, new name of the gallery.

Returns:

- Empty JSON dictionary on success.
- JSON with a relevant error description on failure (409 if the target name is already occupied).

Example

Request

```
curl -i -X PATCH -d '{"name": "testgal2"}' -H "Content-Type: application/json" http://
↪127.0.0.1:18411/v3/galleries/face/testgal1
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:AAwdCxmM
Date: Tue, 04 Feb 2020 08:10:15 GMT
Content-Length: 2

{}
```

List of galleries of a specific type

```
GET /v3/galleries/:objtype
```

Parameters in path segments:

- `:objtype`: string, the name of the object.

Returns:

- JSON dictionary of list of galleries of a specific type on success.
- JSON with a relevant error description on failure.

Example

Request

```
curl -i -X GET 'http://127.0.0.1:18411/v3/galleries/face/'
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:BjI7i3GZ
Date: Wed, 05 Dec 2018 08:19:55 GMT
Content-Length: 125

{
    "galleries": [
        { "objects": 22, "name": "history" },
        { "objects": 1, "name": "test" },
        { "objects": 0, "name": "hello" }
    ]
}
```

2.2.4 Methods with Objects in the Gallery

Create an Object to Database

```
POST /v3/galleries/:objtype/:gallery/objects/:id
```

This method takes a detected object from the cache or an object from a gallery. It then adds the object 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 object, you can also add metadata which uniquely describes the person, for example, the person's name.

Parameters in path segments:

- `:objtype`: string, the name of the object.
- `:gallery`: string, the name of the gallery to add the object in.
- `:id`: integer, permanent object id in the gallery, uint64.

Parameters in request body:

- `from`: identifier of the object serving as a source for the one being created:
  - `detection:<id>`– result of the detection.
  - `persistent:<objtype>/<gallery>/<id>` – existing object in the gallery.
  - `inline-detection` – detection object attached to the request in the field detection. Available only when `allow-return-facen` is enabled. If the detector is not specified normalized, such an object cannot be migrated to a new model.
  - `emben:[model:]<base64 emben>` – emben (with or without model) in base64. Available only when `allow-return-facen` is enabled. Creates incomplete objects that cannot be migrated between models.
- `meta` [optional]: dictionary of meta-fields.
- `detection`: detection object similar to the one passed in `POST /v3/detect/premade` (only the object itself, without wrapping in `{"object type": ...}`, because the type is already known).

Returns:

- JSON representation of the added object on success. If the query-argument `return_emben` is passed and `allow-return-facen` is enabled in the configuration file, it will also return emben in the response field. If the query-argument `return_full_meta` is passed, it will return the full meta, including fields filled in from defaults.
- Error on failure.

Example

Request

```
curl -i -X POST http://127.0.0.1:18411/v3/galleries/face/hello/objects/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

{
    "id": { "type":"face", "object": 123, "gallery": "hello" },
    "meta": {
        "camera": "openspace",
        "labels": ["foo", "bar"],
        "timestamp": "1543837276"
    },
}
```

Get an Object by ID

```
GET /v3/galleries/:objtype/:gallery/objects/:id
```

This method retrieves an object from a database gallery by id.

Parameters in path segments:

- `:objtype`: string, the name of the object.
- `:gallery`: string, the name of the gallery to retrieve an object from.
- `:id`: an object id in the gallery, uint64.

Returns:

- JSON representation of the retrieved object on success. If the query-argument `return_emben` is passed and `allow-return-facen` is enabled in the configuration file, it will also return emben in the response field.
- Error on failure.

Example

Request

```
curl -i -X GET http://127.0.0.1:18411/v3/galleries/face/hello/objects/123/
```

Response

See the response format in the POST description.

Delete an Object from Gallery

```
DELETE /v3/galleries/:objtype/:gallery/objects/:id
```

This method deletes an object from a database gallery by id.

Parameters in path segments:

- `:objtype`: string, the name of the object to delete.
- `:gallery`: string, the name of the gallery to delete the object from.
- `:id`: object id in the gallery, uint64.

Returns:

- Empty JSON on success.
- Error on failure.

Example

Request

```
curl -i -X DELETE http://127.0.0.1:18411/v3/galleries/face/hello/objects/123/
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:9s1oFbFP
Date: Wed, 05 Dec 2018 08:32:14 GMT
Content-Length: 2
{}
```

Update Object Metadata in Gallery

```
PATCH /v3/galleries/:objtype/:gallery/objects/:id
```

The method updates an object metadata in a database gallery by id.

Parameters in path segments:

- `:objtype`: string, the name of the object.
- `:gallery`: string, the gallery's name.
- `:id`: object id in the gallery, uint64.

Parameters in request body:

Important: If the field is not mentioned in the query, its value is not changed. Values of all field types are replaced in their entirety, even if the value is composite.

- `from`: identifier of the object serving as a source for the one being created:
- `meta`: dictionary with the face's new metastrings.
- `detection`: detection object.

Returns:

- JSON representation of the updated face on success. If the query-argument `return_emben` is passed and `allow-return-facen` is enabled in configuration file, it also returns emben in the response field. If the from field is passed, it will overwrite emben and (if there is one in the source) normalized object image.
- Error on failure.

Example

Request

```
curl -i -X PATCH http://127.0.0.1:18411/v3/galleries/face/hello/objects/123/ -H 'Content-
↪Type: application/json' --data-binary '@-' <<EOF
{
    "meta": {
        "labels": ["foo", "bar"]
    }
}
EOF
```

Response

See response format in description of object creation.

Get a List of Objects in the Gallery

```
GET /v3/galleries/:objtype/:gallery/objects/
```

This method allows you to search objects 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`: maximum number of objects in the response. * Required parameter.
- `sort`: sorting order. Pass one of the following values: `id`: increasing order by id, `-id`: decreasing order by id, `-confidence`: sorting by ascending and descending id, descending confidence (only if a similarity filter is set).
- `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 `tntapi` shards are out of service during object identification, `sf-api` returns an error. Enable this Boolean parameter to use available `tntapi` shards to obtain object identification results.
- `return_emben`: if `allow_return_facen` is enabled in config, return emben for each object in response.

Filters in query string parameters:

- `meta:in:meta1=val1&meta:in:meta1=val2&...`: select an objects if a meta string `meta1` is equal to one of the values `val1`/ `val2`/ . . . , etc. (*uint64*, *string*).
- `meta:gte:meta1=val1`: select all objects with a meta string `meta1` greater or equal to `val1` (*uint64*).
- `meta:lte:meta1=val1`: select all objects with a meta string `meta1` less or equal to `val1` (*uint64*).
- `meta:subset:meta1=val1&meta:subset:meta1=val2&...`: select an object if a meta string `meta_field` contains all the values `val1`, `val2`, . . . , etc. (*[]string*).
- `id:in=value_id`: select all objects with `id` equal to `value_id`.
- `id:gte=value_id`: select all objects with `id` greater or equal to `value_id`.
- `id:lte=value_id`: select all objects with id less or equal to `value_id`.
- `looks_like=0.75=detection:bg2gu31jisghl6pee09g`: select if object is similar to detection `bg2gu31jisghl6pee09g` with confidence not lower than 0.75.
- `looks_like=0.78=persistent:face/hello/123`: select if the object looks like face `123` from gallery `hello` with a confidences not lower than 0.78.
- `looks_like=0.81=emben:[model:]<base64 emben>`: select if the object is similar to the passed feature vector with a confidences not lower than 0.81. Available only if `allow-return-facen` is enabled.
- `meta:like:meta1=val1%`: ~sql like: any combination of characters and % sign is supported, e.g. 'aa%', '%aa', '%aa%' or even '%aa%bb%'. That is, '%' is treated as 'any number of any characters'. For data type `array/list` - if any of the values passed the check. Always properly escape `%`.
- `meta:ilike:meta1=Val1%`: case-insensitive analogue of `meta:like` filter.

Do not forget that emben is encoded `base64.StdEncoding` and may contain characters =. Always properly escape your emben.

You can only set one similarity filter in one query (several are BAD_PARAM). If you set `looks_like`, the default sort will automatically switch to -confidence.

Returns:

JSON representation of an array with found objects.

The response format is the following:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: [...]

{
    "objects": [
      {
        ... object 1 data ...
        "confidence": 0.123 // if a similarity filter was specified
      },
      {
        ... object 2 data ...
        "confidence": 0.123 // if a similarity filter was specified
      },
     ...
    ],
    "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.

Example

Request #1.

```
curl -i -X GET http://127.0.0.1:18411/v3/galleries/face/history/objects/?limit=5&
↪meta:in:camera=openspace&meta:in:camera=entrance&meta:lte:timestamp=1543845934&
↪meta:gte:timestamp=1514801655&looks_like=0.4=detection:bg2gu31jisghl6pee09g
```

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

{
```

(continues on next page)

(continued from previous page)

```
    "objects": [
        {
            "confidence": 0.6026,
            "id": { "type":"face", "object": 4141715030051545133, "gallery": "history" },
            "meta": {
                "bbox": "[607, 802, 738, 933]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999998079040,
                "timestamp": 1542909082
            }
        },
        {
            "confidence": 0.5325,
            "id": { "type":"face", "object": 4141715086422990894, "gallery": "history" },
            "meta": {
                "bbox": "[741, 905, 953, 1117]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999993877300,
                "timestamp": 1542909103
            }
        },
        {
            "confidence": 0.531,
            "id": { "type":"face", "object": 4141716493024780347, "gallery": "history" },
            "meta": {
                "bbox": "[90, 869, 166, 945]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 10000000000000000013,
                "timestamp": 1542909627
            }
        },
        {
            "confidence": 0.5236,
            "id": { "type":"face", "object": 4141716498393489468, "gallery": "history" },
            "meta": {
                "bbox": "[56, 853, 125, 923]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999999999053,
                "timestamp": 1542909629
            }
        },
        {
            "confidence": 0.5212,
            "id": { "type":"face", "object": 4141715338752319538, "gallery": "history" },
```

(continues on next page)

(continued from previous page)

```
            "meta": {
                "bbox": "[-36, 862, 60, 958]",
                "camera": "openspace",
                "is_friend": 0,
                "labels": [],
                "score": 9999999999999999425,
                "timestamp": 1542909197
            }
        }
    ],
    "next_page": "There are more than 5 results, but pagination is not supported when␣
↪filtering by emben"
}
```

Request #2.

```
curl -s -X GET http://127.0.0.1:18411/v3/galleries/face/hello/objects/\?limit\=1\&
↪meta:like:camera\=%25pace\&meta:ilike:labels\=FO%25
```

Response

```
{
   "objects": [
   {
      "id": {
         "type": "face",
         "gallery": "hello",
         "object": 123
      },
      "meta": {
         "camera": "openspace",
         "labels": [
            "foo",
            "bar"
         ],
      }
   }
]
}
```

Get a Normalized Objects Image from the Gallery

```
GET /v3/galleries/:objtype/:gallery/objects/:id/normalized
```

Parameters in path segments:

- `:objtype`: string, the name of the object.
- `:gallery`: string, the gallery's name.
- `:id`: object id in the gallery, uint64.

Returns:

Returns Header `location` and `text/plain` with a link to the normalized photo (storage link from `normalized-storage` config). Successful response is available only if `normalized-storage.enabled` is enabled in config.

Example

Request

```
curl -vvv http://localhost:18411/v3/galleries/face/hello/objects/1234/normalized
```

Response

```
HTTP/1.1 302 Found
Content-Type: text/plain
Location: http://127.0.0.1:3333/uploads/ZmFjZTpoZWxsbw==/6c/df/1234_cf7umjq92d31006ou0rg.
↪png
X-Request-Id: SF:Edz0ThbX
Date: Tue, 24 Jan 2023 14:37:39 GMT
Content-Length: 87

http://127.0.0.1:3333/uploads/ZmFjZTpoZWxsbw==/6c/df/1234_cf7umjq92d31006ou0rg.png
```

2.2.5 Get Models Information

```
GET /v3/models-info
```

Parameters in path segments::

The method doesn't accept any parameters.

Returns:

Returns information about available (i.e. enabled) detectors, normalizers, extractors and objects in `extraction-api`.

Example

Request

```
curl -s http://localhost:18411/v3/models-info | jq
```

Response

```
{
  "detectors": {
    "body": {
      "object_types": [
        "body"
      ]
    },
    "car": {
      "object_types": [
        "car"
      ]
    },
    "gustav_body": {
      "object_types": [
        "body"
      ]
    },
    "gustav_car": {
      "object_types": [
        "car"
      ]
    },
    "headbodyface": {
      "object_types": [
        "head",
        "body",
        "face"
      ]
    },
    "license_plate": {
      "object_types": [
        "license_plate"
      ]
    },
```

(continues on next page)

(continued from previous page)

```
    "license_plate_gustav_accurate": {
      "object_types": [
        "license_plate"
      ]
    },
    "shiloette": {
      "object_types": [
        "body"
      ]
    }
  },
  "normalizers": {
    "carlicplate": {
      "normalization_type": "carlicplate"
    },
    "cropbbox": {
      "normalization_type": "cropbbox"
    },
    "norm200": {
      "normalization_type": "norm200"
    }
  },
  "extractors": {
    "car_color": {
      "normalization": "crop1x",
      "model_name": "carattr_color.v0"
    },
    "car_quality": {
      "normalization": "cropbbox",
      "model_name": "carattr.quality.v0"
    },
    "face_emben": {
      "normalization": "norm200",
      "model_name": "kiwi_160"
    },
    "face_quality": {
      "normalization": "crop1x",
      "model_name": "quality_fast.v1"
    },
    "license_plate_quality": {
      "normalization": "cropbbox",
      "model_name": "carlicplateattr.quality.v0"
    }
  },
  "objects": {
    "car": {
      "quality_attribute": "car_quality",
      "base_normalizer": "cropbbox"
    },
    "face": {
      "quality_attribute": "face_quality",
      "base_normalizer": "crop2x"
```

(continues on next page)

(continued from previous page)

```
    },
    "license_plate": {
      "quality_attribute": "license_plate_quality",
      "base_normalizer": "carlicplate"
    }
  }
}
```

2.3 Object API Methods v2

In this section:

2.3.1 Detect Methods

Detect Face on the Photo

```
POST /v2/detect
```

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 `extraction-api` component, so the `sf-api` component formats your initial request and forwards it to `extraction-api`.

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 `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 `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 `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).
- `roi`: string, a region of interest in the image, takes an array of integers in the format roi=top,left,right,bottom, (e.g. {“left”: 0, “right”: 1000, “top”: 0, “bottom”: 1000}).

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

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 you form a request, 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.

Example

Request. 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
```

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": {
```

(continues on next page)

(continued from previous page)

```
                "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
}
```

Get Detection Result from the Cache by ID

```
GET /v2/detect/:id
```

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

Parameters in path segments:

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

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"
}
```

Create Detection Result out of `extraction-api` Response

```
POST /v2/detect/:id
```

This method creates a detection result out of a `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

{}
```

Get Normalized Detect by ID

```
GET /v2/detect/:id/normalized
```

This method retrieves normalized detect by ID.

Parameters in path segments:

- `:id`: ID.

Returns:

Normalized object images on success.

Example

Request

```
curl -i http://127.0.0.1:18411/v2/detect/bg2gu31jisghl6pee09g/normalized
```

Response:

```
HTTP/1.1 200 OK
Content-Type: image/png
X-Request-Id: SF:mjv3Lyvf
Date: Thu, 09 Apr 2020 15:59:48 GMT
Transfer-Encoding: chunked

Warning: Binary output can mess up your terminal. Use "--output -" to tell
```

(continues on next page)

(continued from previous page)

```
Warning: curl to output it to your terminal anyway, or consider "--output
Warning: <FILE>" to save to a file.
```

2.3.2 Compare Faces

```
GET /v2/verify
```

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 (in the format `detection:<id>`), or one from the biometric database (in the format `face:<gallery>/<id>` or `facen:[model:]<base64 facen>` if `allow-return-facen:true`).
- `"face2"`: the second face, from the same possible sources as the first object.

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
}
```

2.3.3 Gallery Methods

List Database Galleries

```
GET /v2/galleries
```

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

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

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:BjI7i3GZ
Date: Wed, 05 Dec 2018 08:19:55 GMT
Content-Length: 125

{
    "galleries": [
        { "faces": 22, "name": "history" },
        { "faces": 1, "name": "test" },
        { "faces": 0, "name": "hello" }
    ]
}
```

Create Database Gallery

```
POST /v2/galleries/:gallery
```

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

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

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:whPaBcoV
Date: Wed, 05 Dec 2018 08:16:20 GMT
Content-Length: 2

{}
```

Retrieve Gallery Details

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

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" }
```

Delete Gallery

```
DELETE /v2/galleries/:gallery
```

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

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

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:oJh9FEtX
Date: Wed, 05 Dec 2018 08:15:04 GMT
Content-Length: 2

{}
```

Rename the Gallery

```
PATCH /v2/galleries/:gallery
```

This method renames the gallery.

Parameters in path segments:

- `:gallery`: string, the name of the gallery to be renamed.

Parameters in request body:

`new_name`: string, new name of the gallery.

Returns:

- Empty JSON dictionary on success.
- JSON with a relevant error description on failure (409 if the target name is already occupied).

Example

Request

```
curl -i -X PATCH -d '{"new_name": "testgal2"}' -H "Content-Type: application/json" http:/
↪/127.0.0.1:18411/v2/galleries/testgal1
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:AAwdCxmM
Date: Tue, 04 Feb 2020 08:10:15 GMT
Content-Length: 2

{}
```

List of Galleries

```
GET /v2/galleries
```

Returns:

- JSON dictionary of list of galleries on success.
- JSON with a relevant error description on failure.

Example

Request

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

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
X-Request-Id: SF:BjI7i3GZ
Date: Wed, 05 Dec 2018 08:19:55 GMT
Content-Length: 125

{
    "galleries": [
        { "faces": 22, "name": "history" },
        { "faces": 1, "name": "test" },
        { "faces": 0, "name": "hello" }
    ]
}
```

2.3.4 Methods with Faces in the Gallery

Create a Face to Database

```
POST /v2/galleries/:gallery/faces/:id
```

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`: identifier of the object serving as a source for the one being created:
  - `detection:<id>`– result of the detection.
  - `face:<gallery>/<id>` – existing face in the gallery.
  - `inline-detection` – detection object attached to the request in the field detection. Available only when `allow-return-facen` is enabled. If the detector is not specified normalized, such a face cannot be migrated to a new model.
  - `facen:[model:]<base64 facen>` – facen (with or without model) in base64. Available only when `allow-return-facen` is enabled. Creates incomplete faces that cannot be migrated between models.
- `meta` [optional]: dictionary of meta-fields.
- `detection`: detection object similar to the one passed in `POST /v2/detect/:id.`

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"
```

(continues on next page)

(continued from previous page)

```
    }
}
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"
}
```

Get a Face from Gallery

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

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
```

(continues on next page)

(continued from previous page)

```
      },
      {
        "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": {
    "timestamp": 2
  },
  "normalized_id": "2_bd323f5t8f66ph0eafp0.png"
}
```

Delete Face from Gallery

```
DELETE /v2/galleries/:gallery/faces/:id
```

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

```
{}
```

Update Face Metadata in Gallery

```
PATCH /v2/galleries/:gallery/faces/:id
```

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

Important: If some metastring is omitted or passed as `null`, the relevant field in the database won't be updated.

- `"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
      },
      {
        "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"
}
```

Get a List of Faces in the Gallery

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

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`: 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 `tntapi` shards are out of service during face identification, `sf-api` returns an error. Enable this Boolean parameter to use available `tntapi` shards to obtain face identification results.
- `return_facen`: if `allow_return_facen` is enabled in config, return facen for each face in response.

Filters in query string parameters:

- `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*).
- `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: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`.
- `detection:bg2gu31jisghl6pee09g=0.75`: select if object is similar to detection `bg2gu31jisghl6pee09g` with confidence not lower than 0.75.
- `face:hello/123=0.78`: select if the object looks like face `123` from gallery `hello` with a confidences not lower than 0.78.
- `meta:like:meta1=val1%`: ~sql like: any combination of characters and % sign is supported, e.g. 'aa%', '%aa', '%aa%' or even '%aa%bb%'. That is, '%' is treated as 'any number of any characters'. For data type `array/list` - if any of the values passed the check. Always properly escape `%`.
- `meta:ilike:meta1=Val1%`: case-insensitive analogue of `meta:like` filter.
- `facen:[model:]<base64 facen>=0.78`: select if the object is similar to the passed feature vector with a confidences not lower than 0.81. Available only if `allow-return-facen` is enabled.

- <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 POST /v2/detect 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
      },
      "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
          }
        ]
      },
```

(continues on next page)

(continued from previous page)

```
      "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",
            "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,
            "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
            },
```

(continues on next page)

(continued from previous page)

```
            "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": [],
                "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"
}
```

Get a Normalized Objects Image from the Gallery

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

Parameters in path segments:

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

Returns:

Returns Header `location` and `text/plain` with a link to the normalized photo (storage link from `normalized-storage` config). Successful response is available only if `normalized-storage.enabled` is enabled in config.

Example

Request

```
curl -vvv http://localhost:18411/v2/galleries/face:test/faces/123/normalized
```

Response

```
HTTP/1.1 302 Found
Content-Type: text/plain
Location: http://127.0.0.1:3333/uploads/ZmFjZTpoZWxsbw==/6c/df/1234_cf7umjq92d31006ou0rg.
↪png
X-Request-Id: SF:Edz0ThbX
Date: Tue, 24 Jan 2023 14:37:39 GMT
Content-Length: 87

http://127.0.0.1:3333/uploads/ZmFjZTpoZWxsbw==/6c/df/1234_cf7umjq92d31006ou0rg.png
```

Tip: You can also find the multi-object API documentation at `http://<sf-api_IP_address>:18411/docs/`.

CHAPTER

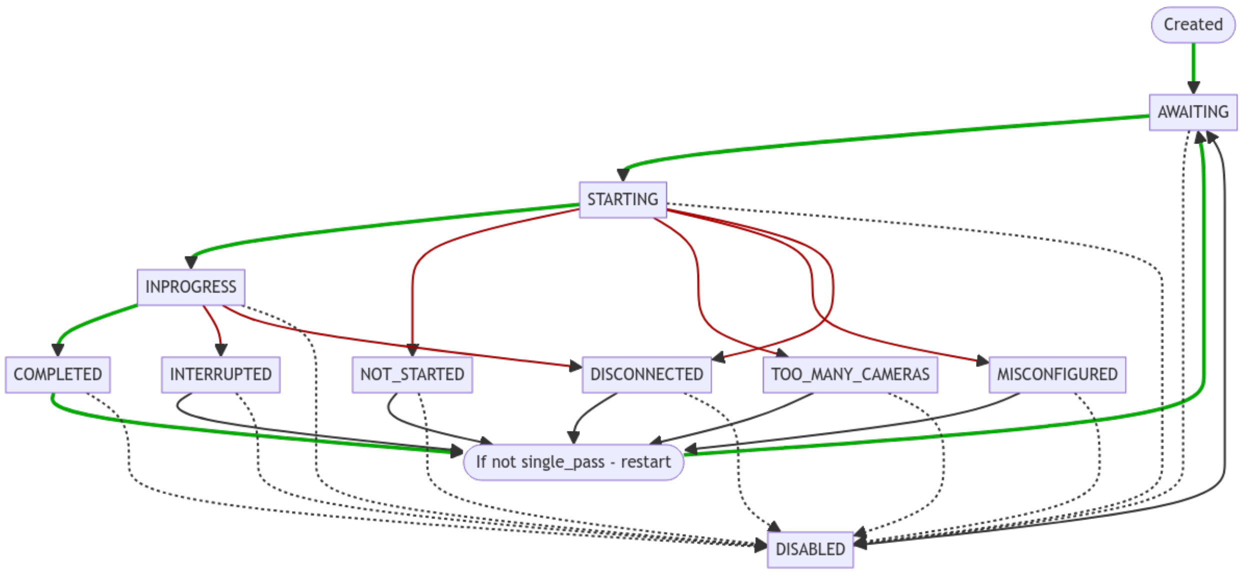
THREE

VIDEO OBJECT DETECTION API

3.1 How to Use Video Object Detection API

In this section:

3.1.1 Job Life Cycle



- AWAITING – the job is waiting for a suitable instance of the `video-worker`. The job gets this status immediately after the creation, or after stopping with an error if `single_pass==false` and it can be restarted. A job can be in this status while it has not yet been assigned, when there are no instances of the `video-worker` with suitable labels, or when the maximum number of video streams that `video-worker` allowed to process exceeds the `capacity` value.

- STARTING – the job has been sent to the `video-worked`, but hasn't processed any frame yet.
- INPROGRESS – the job is being processed by the `video-worker` and has processed at least one frame.
- COMPLETED – the job was completed successfully at the end of the file or stream. Only jobs with `single_pass==true` are permitted to remain in this status for a long time, other ones will be restarted.
- INTERRUPTED – the stream processing was interrupted with an error. If the job hasn't `single_pass` option, it will be restarted.
- MISCONFIGURED – the assigned `video-worker` can't process this job due to configuration inaccuracy. The job will try to run on another `video-worker`.
- TOO_MANY_CAMERAS – the stream is unable to be started due to license restrictions. The job can try to run again, if there will be free slots in the license.
- NOT_STARTED – the attempt to process is completed by an error without processing any frame. The job can try to run on another `video-worker`.
- DISCONNECTED – the connection with the assigned to process the job `video-worker` has been lost.
- DISABLED – the job is set to `enabled:  false`. This status is accessible from any status. After changing option `enabled` to `true`, the job will switch to AWAITING.

3.1.2 Endpoint

Video object detection API requests are to be sent to `http://<video-manager_IP_address>:18810/`. API requests are executed by the `video-manager` component.

3.1.3 Job

Video object detection API operates on a `job` object which represents a video processing task (ie, job) that the `video-manager` component issues to `video-worker`.

Each `job` object has the following attributes:

- `id`: job id.
- `enabled`: active status.
- `stream_url`: URL/address of video stream/file to process.
- `labels`: key-value labels, that will be used by the router component (for example `facerouter`) to find processing directives for objects detected in this stream.
- `router_url`: URL/address of the router component (for example, `facerouter`) to receive detected objects from the `video-worker` component for processing.
- `router_events_url`: URL/address of the router component (for example, `facerouter`), that uses events extraction.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).
- `stream_settings`: video stream settings that duplicate *those* in the `video-manager.yaml` configuration file (while having priority over them).
- `stream_settings_gpu`: deprecated video stream settings. Not recommended for use. Only for compatibility.
- `status`: job status.

- `status_msg`: additional job status info.
- `statistic`: job progress statistics (progress duration, number of posted and not posted objects, processing fps, the number of processed and dropped frames, job start time, etc.).
- `restream_url`: websocket URL where processing stream with detected objects streams live time.
- `restream_direct_url`: websocket URL where original stream with input quality streams live time.
- `shots_url`: HTTP URL where actual stream screenshot can be downloaded.
- `worker_id`: unique id of the `video worker` instance with a processing job.
- `version`: job version.

3.1.4 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
`MALFORMED_REQUEST`	The request is malformed and cannot be read.	400
`SOURCE_NOT_FOUND`	Detection not found in cache.	400
`CONFLICT`	Conflict.	409
`NOT_FOUND`	Job not found.	404
`EXTRACTION_ERROR | STORAGE_ERROR | LICENSE_ERROR | CACHE_ERROR`	Service Unavailable.	523

3.2 Video Object Detection API Methods

In this section:

3.2.1 Create Job

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

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

Parameters in path segments

`:id`: job id.

Parameters in request body:

- `enabled`: job enabled state.
- `stream_url`: URL/address of a video stream/file to process.
- `labels`: key-value labels, that will be used by the router component (for example `facerouter`) to find processing directives for objects detected in this video stream.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).
- `router_url`: URL/address of the router component (for example, `facerouter`) to receive detected objects from the `video-worker` component for processing.
- `router_events_url`: URL/address of the router component (for example, `facerouter`), that uses events extraction.
- `stream_settings`: stream processing common settings and detector parameters.
- `stream_settings_gpu`: deprecated video stream settings. Not recommended for use. Only for compatibility.

Note: Other video stream parameters that differ from common video stream parameters are specified in the `video-manager` configuration file (see *Video Object Detection: video-manager and video-worker* section) or in the service defaults.

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",
  "enabled": true,
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://127.0.0.1:7777/mocks/events/records",
  "router_events_url": "http://127.0.0.1:7777/mocks/actions/records",
  "single_pass": false,
  "stream_settings": {
    "play_speed": -1,
    "disable_drops": false,
    "imotion_threshold": 0,
    "router_timeout_ms": 15000,
    "router_verify_ssl": true,
    "router_headers": [],
    "router_body": [],
    "ffmpeg_params": [],
    "ffmpeg_format": "",
    "use_stream_timestamp": false,
    "start_stream_timestamp": 0,
    "rot": "",
    "stream_data_filter": "",
    "video_transform": "",
    "enable_recorder": false,
    "enable_liveness": false,
    "record_audio": false,
    "detectors": {
      "face": {
        "filter_min_quality": 0.45,
        "filter_min_size": 1,
        "filter_max_size": 8192,
        "roi": "",
        "fullframe_crop_rot": false,
        "fullframe_use_png": false,
        "jpeg_quality": 95,
        "overall_only": false,
```

(continues on next page)

(continued from previous page)

```
        "realtime_post_first_immediately": false,
        "realtime_post_interval": 1,
        "realtime_post_every_interval": false,
        "track_interpolate_bboxes": true,
        "track_miss_interval": 1,
        "track_overlap_threshold": 0.25,
        "track_max_duration_frames": 0,
        "track_send_history": false,
        "post_best_track_frame": true,
        "post_best_track_normalize": true,
        "post_first_track_frame": false,
        "post_last_track_frame": false,
        "tracker_type": "simple_iou",
        "track_deep_sort_matching_threshold": 0.65,
        "track_deep_sort_filter_unconfirmed_tracks": true,
        "track_object_is_principal": false,
        "track_history_active_track_miss_interval": 0,
        "filter_track_min_duration_frames": 1,
        "extractors_track_triggers": {}
      }
    }
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": 0.45,
    "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,
    "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,
    "router_full_frame_png": false,
    "router_disable_normalized": false,
    "crop_fullframe_rot": false,
    "realtime_post_first_immediately": false,
    "post_first_track_frame": false,
    "post_last_track_frame": false,
    "track_max_duration_frames": 0,
    "send_track_history": false,
```

(continues on next page)

(continued from previous page)

```
    "stream_data_filter": "",
    "video_transform": ""
  },
  "status": "AWAITING",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0,
    "faces_failed": 0,
    "faces_not_posted": 0,
    "processing_fps": 0,
    "frames_dropped": 0,
    "frames_processed": 0,
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "frame_width": 0,
    "frame_height": 0,
    "last_stream_timestamp": 0,
    "objects": null,
    "extractors": null,
    "job_starts": 0
  },
  "restream_url": "",
  "restream_direct_url": "",
  "shots_url": "",
  "worker_id": "",
  "version": ""
}
```

3.2.2 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": "myid-123",
    "enabled": true,
    "stream_url": "http://1.2.3.4/stream.mp4",
    "labels": {
      "district": "SVAO"
    },
    "router_url": "http://127.0.0.1:7777/mocks/events/records",
    "router_events_url": "http://127.0.0.1:7777/mocks/actions/records",
    "single_pass": false,
    "stream_settings": {
      "play_speed": -1,
      "disable_drops": false,
      "imotion_threshold": 0,
      "router_timeout_ms": 15000,
      "router_verify_ssl": true,
      "router_headers": [],
      "router_body": [],
      "ffmpeg_params": [],
      "ffmpeg_format": "",
      "use_stream_timestamp": false,
      "start_stream_timestamp": 0,
      "rot": "",
      "stream_data_filter": "",
      "video_transform": "",
      "enable_recorder": false,
      "enable_liveness": false,
      "record_audio": false,
      "detectors": {
        "face": {
          "filter_min_quality": 0.45,
          "filter_min_size": 1,
          "filter_max_size": 8192,
          "roi": "",
          "fullframe_crop_rot": false,
          "fullframe_use_png": false,
          "jpeg_quality": 95,
          "overall_only": false,
          "realtime_post_first_immediately": false,
          "realtime_post_interval": 1,
          "realtime_post_every_interval": false,
          "track_interpolate_bboxes": true,
```

(continues on next page)

(continued from previous page)

```
        "track_miss_interval": 1,
        "track_overlap_threshold": 0.25,
        "track_max_duration_frames": 0,
        "track_send_history": false,
        "post_best_track_frame": true,
        "post_best_track_normalize": true,
        "post_first_track_frame": false,
        "post_last_track_frame": false,
        "tracker_type": "simple_iou",
        "track_deep_sort_matching_threshold": 0.65,
        "track_deep_sort_filter_unconfirmed_tracks": true,
        "track_object_is_principal": false,
        "track_history_active_track_miss_interval": 0,
        "filter_track_min_duration_frames": 1,
        "extractors_track_triggers": {}
      }
    }
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": 0.45,
    "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,
    "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,
    "router_full_frame_png": false,
    "router_disable_normalized": false,
    "crop_fullframe_rot": false,
    "realtime_post_first_immediately": false,
    "post_first_track_frame": false,
    "post_last_track_frame": false,
    "track_max_duration_frames": 0,
    "send_track_history": false,
    "stream_data_filter": "",
    "video_transform": ""
  },
  "status": "STARTING",
```

(continues on next page)

(continued from previous page)

```
    "status_msg": "",
    "statistic": {
      "processed_duration": 0,
      "faces_posted": 0,
      "faces_failed": 0,
      "faces_not_posted": 0,
      "processing_fps": 0,
      "frames_dropped": 0,
      "frames_processed": 0,
      "frames_imotion_skipped": 0,
      "decoding_soft_errors": 0,
      "frame_width": 0,
      "frame_height": 0,
      "last_stream_timestamp": 0,
      "objects": {},
      "extractors": {},
      "job_starts": 0
    },
    "restream_url": "ws://127.0.0.1:9999/stream/myid-123",
    "restream_direct_url": "ws://127.0.0.1:9999/directstream/myid-123",
    "shots_url": "http://127.0.0.1:9999/shot/myid-123",
    "worker_id": "ntechlab_cpu_bed698ede98011e0311d7669f7ff0b18",
    "version": "cpa3slouvrb8u0vir3j0"
  }
]
```

3.2.3 Existing Jobs Count

```
GET /jobs-count
```

This method returns a set of counters for all jobs statuses. Counters with no jobs will not be shown.

Parameters:

This method doesn't accept any parameters.

Returns:

JSON representation with information about the count of jobs.

Example

Request

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

Response

```
{
  "ALL": 10,
  "AWAITING": 1,
  "COMPLETED": 1,
  "DISABLED": 1,
  "DISCONNECTED": 1,
  "INPROGRESS": 1,
  "INTERRUPTED": 1,
  "MISCONFIGURED": 1,
  "NOT_STARTED": 1,
  "STARTING": 1,
  "TOO_MANY_CAMERAS": 1
}
```

3.2.4 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/myid-123' | jq
```

Response

```
{
  "id": "myid-123",
  "enabled": true,
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://127.0.0.1:7777/mocks/events/records",
  "router_events_url": "http://127.0.0.1:7777/mocks/actions/records",
  "single_pass": false,
  "stream_settings": {
    "play_speed": -1,
    "disable_drops": false,
    "imotion_threshold": 0,
    "router_timeout_ms": 15000,
    "router_verify_ssl": true,
    "router_headers": [],
    "router_body": [],
    "ffmpeg_params": [],
    "ffmpeg_format": "",
    "use_stream_timestamp": false,
    "start_stream_timestamp": 0,
    "rot": "",
    "stream_data_filter": "",
    "video_transform": "",
    "enable_recorder": false,
    "enable_liveness": false,
    "record_audio": false,
    "detectors": {
      "face": {
        "filter_min_quality": 0.45,
        "filter_min_size": 1,
        "filter_max_size": 8192,
        "roi": "",
        "fullframe_crop_rot": false,
        "fullframe_use_png": false,
        "jpeg_quality": 95,
        "overall_only": false,
        "realtime_post_first_immediately": false,
        "realtime_post_interval": 1,
        "realtime_post_every_interval": false,
        "track_interpolate_bboxes": true,
        "track_miss_interval": 1,
```

(continues on next page)

(continued from previous page)

```
        "track_overlap_threshold": 0.25,
        "track_max_duration_frames": 0,
        "track_send_history": false,
        "post_best_track_frame": true,
        "post_best_track_normalize": true,
        "post_first_track_frame": false,
        "post_last_track_frame": false,
        "tracker_type": "simple_iou",
        "track_deep_sort_matching_threshold": 0.65,
        "track_deep_sort_filter_unconfirmed_tracks": true,
        "track_object_is_principal": false,
        "track_history_active_track_miss_interval": 0,
        "filter_track_min_duration_frames": 1,
        "extractors_track_triggers": {}
      }
    }
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": 0.45,
    "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,
    "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,
    "router_full_frame_png": false,
    "router_disable_normalized": false,
    "crop_fullframe_rot": false,
    "realtime_post_first_immediately": false,
    "post_first_track_frame": false,
    "post_last_track_frame": false,
    "track_max_duration_frames": 0,
    "send_track_history": false,
    "stream_data_filter": "",
    "video_transform": ""
  },
  "status": "STARTING",
  "status_msg": "",
```

(continues on next page)

(continued from previous page)

```
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0,
    "faces_failed": 0,
    "faces_not_posted": 0,
    "processing_fps": 0,
    "frames_dropped": 0,
    "frames_processed": 0,
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "frame_width": 0,
    "frame_height": 0,
    "last_stream_timestamp": 0,
    "objects": {},
    "extractors": {},
    "job_starts": 0
  },
  "restream_url": "ws://127.0.0.1:9999/stream/myid-123",
  "restream_direct_url": "ws://127.0.0.1:9999/directstream/myid-123",
  "shots_url": "http://127.0.0.1:9999/shot/myid-123",
  "worker_id": "ntechlab_cpu_bed698ede98011e0311d7669f7ff0b18",
  "version": "cpa3slouvrb8u0vir3j0"
}
```

3.2.5 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",
  "enabled": true,
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://127.0.0.1:7777/mocks/events/records",
  "router_events_url": "http://127.0.0.1:7777/mocks/actions/records",
  "single_pass": false,
  "stream_settings": {
    "play_speed": -1,
    "disable_drops": false,
    "imotion_threshold": 0,
    "router_timeout_ms": 15000,
    "router_verify_ssl": true,
    "router_headers": [],
    "router_body": [],
    "ffmpeg_params": [],
    "ffmpeg_format": "",
    "use_stream_timestamp": false,
    "start_stream_timestamp": 0,
    "rot": "",
    "stream_data_filter": "",
    "video_transform": "",
    "enable_recorder": false,
    "enable_liveness": false,
    "record_audio": false,
    "detectors": {
      "face": {
        "filter_min_quality": 0.45,
        "filter_min_size": 1,
        "filter_max_size": 8192,
        "roi": "",
        "fullframe_crop_rot": false,
        "fullframe_use_png": false,
        "jpeg_quality": 95,
        "overall_only": false,
        "realtime_post_first_immediately": false,
        "realtime_post_interval": 1,
        "realtime_post_every_interval": false,
        "track_interpolate_bboxes": true,
        "track_miss_interval": 1,
        "track_overlap_threshold": 0.25,
        "track_max_duration_frames": 0,
        "track_send_history": false,
        "post_best_track_frame": true,
        "post_best_track_normalize": true,
        "post_first_track_frame": false,
        "post_last_track_frame": false,
        "tracker_type": "simple_iou",
```

(continues on next page)

(continued from previous page)

```
        "track_deep_sort_matching_threshold": 0.65,
        "track_deep_sort_filter_unconfirmed_tracks": true,
        "track_object_is_principal": false,
        "track_history_active_track_miss_interval": 0,
        "filter_track_min_duration_frames": 1,
        "extractors_track_triggers": {}
      }
    }
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": 0.45,
    "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,
    "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,
    "router_full_frame_png": false,
    "router_disable_normalized": false,
    "crop_fullframe_rot": false,
    "realtime_post_first_immediately": false,
    "post_first_track_frame": false,
    "post_last_track_frame": false,
    "track_max_duration_frames": 0,
    "send_track_history": false,
    "stream_data_filter": "",
    "video_transform": ""
  },
  "status": "DELETED",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0,
    "faces_failed": 0,
    "faces_not_posted": 0,
    "processing_fps": 0,
    "frames_dropped": 0,
    "frames_processed": 0,
```

(continues on next page)

(continued from previous page)

```
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "frame_width": 0,
    "frame_height": 0,
    "last_stream_timestamp": 0,
    "objects": null,
    "extractors": null,
    "job_starts": 0
  },
  "restream_url": "ws://127.0.0.1:9999/stream/myid-123",
  "restream_direct_url": "ws://127.0.0.1:9999/directstream/myid-123",
  "shots_url": "http://127.0.0.1:9999/shot/myid-123",
  "worker_id": "ntechlab_cpu_bed698ede98011e0311d7669f7ff0b18",
  "version": "cpa3slouvrb8u0vir3j0"
}
```

3.2.6 Update Job

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

The method updates job parameters by id.

Parameters in path segments:

`id`: job id.

Parameters in request body:

- `enabled`: job enabled state.
- `stream_url`: URL/address of a video stream/file to process.
- `labels`: key-value labels, that will be used by the router component (for example, `facerouter`) to find processing directives for objects detected in this video stream.
- `single_pass`: if true, disable restarting video processing upon error (by default, false).
- `router_url`: URL/address of the router component (for example, `facerouter`) to receive detected objects from the `video-worker` component for processing.
- `router_events_url`: URL/address of the router component (for example, `facerouter`), that uses events extraction.
- `stream_settings`: stream processing common settings and detector parameters.
- `stream_settings_gpu`: deprecated video stream settings. Not recommended for use. Only for compatibility.

Returns:

JSON representation of the updated job object.

Non-editable fields in response:

- `id`: job id.
- `status`: job status.
- `status_msg`: additional job status info.
- `statistic`: job progress statistics (progress duration, number of posted and not posted objects, processing fps, the number of processed and dropped frames, job start time, etc.).
- `restream_url`: websocket URL where processing stream with detected objects streams live time.
- `restream_direct_url`: websocket URL where original stream with input quality streams live time.
- `shots_url`: HTTP URL where actual stream screenshot can be downloaded.
- `worker_id`: unique id of the `video worker` instance with a processing job.
- `version`: job version.

Simple example

Request

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

```
{
  "id": "myid-123",
  "enabled": true,
  "stream_url": "http://1.2.3.4/stream.mp4",
  "labels": {
    "district": "SVAO"
  },
  "router_url": "http://myrouter",
  "router_events_url": "http://127.0.0.1:7777/mocks/actions/records",
  "single_pass": false,
  "stream_settings": {
    "play_speed": -1,
    "disable_drops": false,
    "imotion_threshold": 0,
    "router_timeout_ms": 15000,
    "router_verify_ssl": true,
    "router_headers": [],
    "router_body": [],
    "ffmpeg_params": [],
    "ffmpeg_format": "",
    "use_stream_timestamp": false,
    "start_stream_timestamp": 0,
    "rot": "",
```

(continues on next page)

(continued from previous page)

```
    "stream_data_filter": "",
    "video_transform": "",
    "enable_recorder": false,
    "enable_liveness": false,
    "record_audio": false,
    "detectors": {
      "face": {
        "filter_min_quality": 0.45,
        "filter_min_size": 1,
        "filter_max_size": 8192,
        "roi": "",
        "fullframe_crop_rot": false,
        "fullframe_use_png": false,
        "jpeg_quality": 95,
        "overall_only": false,
        "realtime_post_first_immediately": false,
        "realtime_post_interval": 1,
        "realtime_post_every_interval": false,
        "track_interpolate_bboxes": true,
        "track_miss_interval": 1,
        "track_overlap_threshold": 0.25,
        "track_max_duration_frames": 0,
        "track_send_history": false,
        "post_best_track_frame": true,
        "post_best_track_normalize": true,
        "post_first_track_frame": false,
        "post_last_track_frame": false,
        "tracker_type": "simple_iou",
        "track_deep_sort_matching_threshold": 0.65,
        "track_deep_sort_filter_unconfirmed_tracks": true,
        "track_object_is_principal": false,
        "track_history_active_track_miss_interval": 0,
        "filter_track_min_duration_frames": 1,
        "extractors_track_triggers": {}
      }
    }
  },
  "stream_settings_gpu": {
    "play_speed": -1,
    "filter_min_quality": 0.45,
    "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,
    "router_headers": [],
    "router_body": [],
    "start_stream_timestamp": 0,
```

(continues on next page)

(continued from previous page)

```
    "imotion_threshold": 0,
    "rot": "",
    "roi": "",
    "realtime_post_interval": 1,
    "realtime_post_every_interval": false,
    "ffmpeg_format": "",
    "disable_drops": false,
    "router_full_frame_png": false,
    "router_disable_normalized": false,
    "crop_fullframe_rot": false,
    "realtime_post_first_immediately": false,
    "post_first_track_frame": false,
    "post_last_track_frame": false,
    "track_max_duration_frames": 0,
    "send_track_history": false,
    "stream_data_filter": "",
    "video_transform": ""
  },
  "status": "STARTING",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0,
    "faces_failed": 0,
    "faces_not_posted": 0,
    "processing_fps": 0,
    "frames_dropped": 0,
    "frames_processed": 0,
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "frame_width": 0,
    "frame_height": 0,
    "last_stream_timestamp": 0,
    "objects": null,
    "extractors": null,
    "job_starts": 0
  },
  "restream_url": "ws://127.0.0.1:9999/stream/myid-123",
  "restream_direct_url": "ws://127.0.0.1:9999/directstream/myid-123",
  "shots_url": "http://127.0.0.1:9999/shot/myid-123",
  "worker_id": "ntechlab_cpu_bed698ede98011e0311d7669f7ff0b18",
  "version": "cpa5ob0uvrb8u0vir3mg"
}
```

Disable detector example

Request

```
$ curl -s 'http://localhost:18810/job/myid-123' -X PATCH --data '{"stream_settings":{
↪"detectors":{"face":null}}}' | jq
```

```
{
   "id": "myid-123",
   "enabled": true,
   "stream_url": "http://1.2.3.4/stream.mp4",
   "labels": {
     "district": "SVAO"
   },
   "router_url": "http://myrouter",
   "router_events_url": "http://127.0.0.1:7777/mocks/actions/records",
   "single_pass": false,
   "stream_settings": {
     "play_speed": -1,
     "disable_drops": false,
     "imotion_threshold": 0,
     "router_timeout_ms": 15000,
     "router_verify_ssl": true,
     "router_headers": [],
     "router_body": [],
     "ffmpeg_params": [],
     "ffmpeg_format": "",
     "use_stream_timestamp": false,
     "start_stream_timestamp": 0,
     "rot": "",
     "stream_data_filter": "",
     "video_transform": "",
     "enable_recorder": false,
     "enable_liveness": false,
     "record_audio": false,
     "detectors": {}
   },
   "stream_settings_gpu": {
     "play_speed": -1,
     "filter_min_quality": 0.45,
     "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,
     "router_headers": [],
     "router_body": [],
     "start_stream_timestamp": 0,
```

(continues on next page)

(continued from previous page)

```
    "imotion_threshold": 0,
    "rot": "",
    "roi": "",
    "realtime_post_interval": 1,
    "realtime_post_every_interval": false,
    "ffmpeg_format": "",
    "disable_drops": false,
    "router_full_frame_png": false,
    "router_disable_normalized": false,
    "crop_fullframe_rot": false,
    "realtime_post_first_immediately": false,
    "post_first_track_frame": false,
    "post_last_track_frame": false,
    "track_max_duration_frames": 0,
    "send_track_history": false,
    "stream_data_filter": "",
    "video_transform": ""
  },
  "status": "STARTING",
  "status_msg": "",
  "statistic": {
    "processed_duration": 0,
    "faces_posted": 0,
    "faces_failed": 0,
    "faces_not_posted": 0,
    "processing_fps": 0,
    "frames_dropped": 0,
    "frames_processed": 0,
    "frames_imotion_skipped": 0,
    "decoding_soft_errors": 0,
    "frame_width": 0,
    "frame_height": 0,
    "last_stream_timestamp": 0,
    "objects": {},
    "extractors": {},
    "job_starts": 0
  },
  "restream_url": "ws://127.0.0.1:9999/stream/myid-123",
  "restream_direct_url": "ws://127.0.0.1:9999/directstream/myid-123",
  "shots_url": "http://127.0.0.1:9999/shot/myid-123",
  "worker_id": "ntechlab_cpu_bed698ede98011e0311d7669f7ff0b18",
  "version": "cpa5ob0uvrb8u0vir3mg"
}
```

3.2.7 Restart Job

```
RESTART /job/:id
POST /job/:id/restart
```

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: VM:n267l1VQ
Date: Mon, 27 May 2024 08:12:19 GMT
Content-Length: 0
```

3.2.8 Get Actual Stream Settings as JSON-schema

```
GET /schemas/stream_settings
```

This method gets actual stream settings parameters as JSON schema.

Parameters:

This method doesn't accept any parameters.

Returns:

JSON representation, free form of JSON schema.

Example

Request

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

Response

```
{
  "$schema": "http://json-schema.org/draft/2020-12/schema",
  "$id": "https://my/videomanager-api/my/stream-settings",
  "$ref": "#/$defs/StreamSettings",
  "$defs": {
    "Detector": {
      "properties": {
        "filter_min_quality": {
          "type": "number"
        },
        "filter_min_size": {
          "type": "integer"
        },
        "filter_max_size": {
          "type": "integer"
        },
        "roi": {
          "type": "string"
        },
        "fullframe_crop_rot": {
          "type": "boolean"
        },
        "fullframe_use_png": {
          "type": "boolean"
        },
        "jpeg_quality": {
          "type": "integer"
        },
        "overall_only": {
          "type": "boolean"
        },
        "realtime_post_first_immediately": {
          "type": "boolean"
        },
        "realtime_post_interval": {
          "type": "number"
        },
        "realtime_post_every_interval": {
          "type": "boolean"
```

(continues on next page)

(continued from previous page)

```
        },
        "track_interpolate_bboxes": {
          "type": "boolean"
        },
        "track_miss_interval": {
          "type": "number"
        },
        "track_overlap_threshold": {
          "type": "number"
        },
        "track_max_duration_frames": {
          "type": "integer"
        },
        "track_send_history": {
          "type": "boolean"
        },
        "post_best_track_frame": {
          "type": "boolean"
        },
        "post_best_track_normalize": {
          "type": "boolean"
        },
        "post_first_track_frame": {
          "type": "boolean"
        },
        "post_last_track_frame": {
          "type": "boolean"
        },
        "tracker_type": {
          "type": "string"
        },
        "track_deep_sort_matching_threshold": {
          "type": "number"
        },
        "track_deep_sort_filter_unconfirmed_tracks": {
          "type": "boolean"
        },
        "track_object_is_principal": {
          "type": "boolean"
        },
        "track_history_active_track_miss_interval": {
          "type": "number"
        },
        "filter_track_min_duration_frames": {
          "type": "integer"
        },
        "extractors_track_triggers": {
          "patternProperties": {
            ".*": {
              "$ref": "#/$defs/ExtractorTrackTriggers"
            }
          },
```

(continues on next page)

(continued from previous page)

```
          "type": "object"
        }
      },
      "additionalProperties": false,
      "type": "object"
    },
    "ExtractorTrackTriggers": {
      "properties": {
        "run_on_start": {
          "type": "boolean"
        },
        "run_on_end": {
          "type": "boolean"
        },
        "run_interval": {
          "type": "number"
        },
        "post_frame": {
          "type": "boolean"
        }
      },
      "additionalProperties": false,
      "type": "object"
    },
    "StreamSettings": {
      "properties": {
        "play_speed": {
          "type": "number"
        },
        "disable_drops": {
          "type": "boolean"
        },
        "imotion_threshold": {
          "type": "number"
        },
        "router_timeout_ms": {
          "type": "integer"
        },
        "router_verify_ssl": {
          "type": "boolean"
        },
        "router_headers": {
          "items": {
            "type": "string"
          },
          "type": "array"
        },
        "router_body": {
          "items": {
            "type": "string"
          },
          "type": "array"
```

(continues on next page)

(continued from previous page)

```
        },
        "ffmpeg_params": {
          "items": {
            "type": "string"
          },
          "type": "array"
        },
        "ffmpeg_format": {
          "type": "string"
        },
        "use_stream_timestamp": {
          "type": "boolean"
        },
        "start_stream_timestamp": {
          "type": "integer"
        },
        "rot": {
          "type": "string"
        },
        "stream_data_filter": {
          "type": "string"
        },
        "video_transform": {
          "type": "string"
        },
        "enable_recorder": {
          "type": "boolean"
        },
        "enable_liveness": {
          "type": "boolean"
        },
        "record_audio": {
          "type": "boolean"
        },
        "detectors": {
          "patternProperties": {
            ".*": {
              "$ref": "#/$defs/Detector"
            }
          },
          "type": "object"
        }
      },
      "additionalProperties": false,
      "type": "object"
    }
  }
}
```

3.2.9 Get Actual GPU Stream Settings as JSON-schema

```
GET /schemas/stream_settings_gpu
```

This method gets actual GPU stream settings (outdated format of the stream settings) parameters as JSON schema.

Parameters:

This method doesn't accept any parameters.

Returns:

JSON representation, free form of JSON schema.

Example

Request

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

Response

```
{
  "$schema": "http://json-schema.org/draft/2020-12/schema",
  "$id": "https://my/videomanager-api/my/stream-settings-gpu",
  "$ref": "#/$defs/StreamSettingsGpu",
  "$defs": {
    "StreamSettingsGpu": {
      "properties": {
        "play_speed": {
          "type": "number"
        },
        "filter_min_quality": {
          "type": "number"
        },
        "filter_min_face_size": {
          "type": "integer"
        },
        "filter_max_face_size": {
          "type": "integer"
        },
        "normalized_only": {
          "type": "boolean"
        },
        "jpeg_quality": {
          "type": "integer"
        },
        "overall_only": {
          "type": "boolean"
```

(continues on next page)

(continued from previous page)

```
        },
        "use_stream_timestamp": {
          "type": "boolean"
        },
        "ffmpeg_params": {
          "items": {
            "type": "string"
          },
          "type": "array"
        },
        "router_timeout_ms": {
          "type": "integer"
        },
        "router_verify_ssl": {
          "type": "boolean"
        },
        "router_headers": {
          "items": {
            "type": "string"
          },
          "type": "array"
        },
        "router_body": {
          "items": {
            "type": "string"
          },
          "type": "array"
        },
        "start_stream_timestamp": {
          "type": "integer"
        },
        "imotion_threshold": {
          "type": "number"
        },
        "rot": {
          "type": "string"
        },
        "roi": {
          "type": "string"
        },
        "realtime_post_interval": {
          "type": "number"
        },
        "realtime_post_every_interval": {
          "type": "boolean"
        },
        "ffmpeg_format": {
          "type": "string"
        },
        "disable_drops": {
          "type": "boolean"
        },
```

(continues on next page)

(continued from previous page)

```
        "router_full_frame_png": {
          "type": "boolean"
        },
        "router_disable_normalized": {
          "type": "boolean"
        },
        "crop_fullframe_rot": {
          "type": "boolean"
        },
        "realtime_post_first_immediately": {
          "type": "boolean"
        },
        "post_first_track_frame": {
          "type": "boolean"
        },
        "post_last_track_frame": {
          "type": "boolean"
        },
        "track_max_duration_frames": {
          "type": "integer"
        },
        "send_track_history": {
          "type": "boolean"
        },
        "stream_data_filter": {
          "type": "string"
        },
        "video_transform": {
          "type": "string"
        }
      },
      "additionalProperties": false,
      "type": "object"
    }
  }
}
```

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