VisionLabs FaceStream

Administrator manual

v.5.84.0

Contents

Glossary

Term	Meaning
Aspect angle	Head rotation degree (in degrees) on each of the three axes (up/down tilt relative to the horizontal axis; left/right tilt, relative to the vertical axis; a rotation about the vertical axis).
Batch	Group of data processed simultaneously.
Best shot	The frame of the video stream on which the face/body is fixed in the optimal angle for further processing.
Detection	FaceStream entity that contains the coordinates of face or body and the estimated value of the object that determines the best shot.
Descriptor	A set of unique features received from the warp. A descriptor requires much less storage memory in comparison with the sample and is used for comparison of faces.
Event	LUNA PLATFORM entity, which contains information (city, user data, track id, etc.) about one face and/or body. This information is transferred to the LUNA PLATFORM by the FaceStream application. For a complete list of the transferred information, see the OpenAPI LUNA PLATFORM documentation.
LUNA Streams	Service for creating and managing streams that contain policies for processing a video stream/video file/set of images.
Normalized image, warp	Images containing a face or body and corresponding to VisionLabs standard. Used when working with LUNA PLATFORM.
Portrait	Image of face or body that has been transformed to a specific format. The portrait has two types — "warp" (the image is transformed into warp format), "gost" (detection is cut out from the source frame, considering indentation).
Track	Information about object's position (face of a person) in a sequence of frames. If the object leaves the frame zone, the track doesn't discontinue right away. For some time, the system expects the object to return and if it does, the track continues.
Tracking	Object (face) tracking function in the frame sequence.

1 Introduction

This document describes:

- System requirements.
- General description of the application and recommendations for setting up.
- Process of interacting with LUNA Streams.
- List of basic settings required to launch FaceStream.
- Detailed description of FaceStream settings.
- Using FaceStream with LUNA Configurator.
- Using FaceStream with configuration files.
- API errors for FaceStream and LUNA Streams.
- Information about compatibility with camera models.

For more information on launching the application, see the FaceStream installation manual.

2 Overview

FaceStream conducts several functions:

- **Stream reading**

  Web-cameras, USB and IP-cameras (via RTSP protocol), video files and images can act as data sources.

- **Stream processing**

  It searches for faces and bodies in the stream and tracks them until they leave the frame or are blocked.

- **Liveness check**

  Liveness check is performed on one or more frames of the track.

- **Sending face or body best shots as HTTP-requests onto external service**

  VisionLabs Software LUNA PLATFORM 5 acts as an external service.

FaceStream workflow depends on the setting of four configurations.

- Streams management configuration set in LUNA Streams

  Here you can set the settings regarding stream sources such as source type, source address, filtering settings, etc. The settings are set by sending requests with a body in JSON format to the LUNA Streams service. FaceStream takes the settings from LUNA Streams for further processing. A detailed description of how FaceStream works with LUNA Streams is given in the "Interaction of FaceStream with LUNA Streams" section.

- FaceStream settings set in LUNA Configurator

  Here you can set general FaceStream settings, such as logging, setting up sending images from FaceStream to external services, debugging, etc.

- TrackEngine settings set in LUNA Configurator

  Here you can set general TrackEngine settings regarding the face or body detection and tracking.

- LUNA Streams settings set in LUNA Configurator

  Here you can set general settings for the LUNA Streams service, such as logging, database settings, address of the LUNA Licenses service, etc.

- FaceEngine settings set in "faceengine.conf" configuration file and transferred during the launch of the FaceStream container.

  Here you can set the settings for face and body recognition. It is recommended to change the parameters in this configuration only in consultation with VisionLabs employees.

The following features are also available when working with FaceStream:

- Dynamic creation, editing, and deletion of stream sources via API requests.
- Real time video streams preview in a browser for the streams with specified parameters.
- Stream metrics (number of streams, number of errors, number of faces, number of skipped frames, FPS).

FaceStream can be configured to work:

- Only with faces.
- Only with bodies.
- With faces and bodies (see section "Face and body detection collaborative mode").

2.1 FaceStream workflow with faces and bodies

FaceStream can handle both faces and bodies. Each object has its own scheme of operation and its own set of parameters described below.

> The required minimum parameters for working with both objects can be found in the section "Priority parameters list".

2.1.1 FaceStream workflow with faces

FaceStream application workflow with faces is shown in the image below:
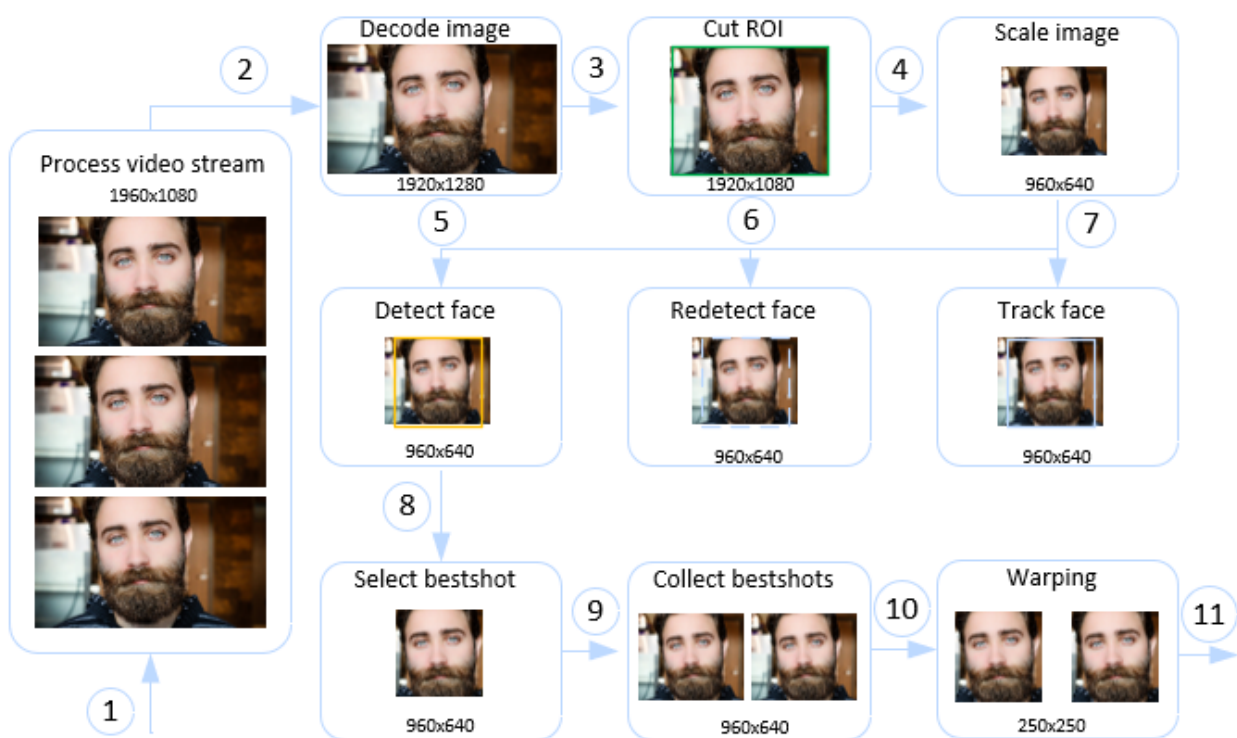
Figure 1: FaceStream workflow with faces

1. FaceStream receives video from a source (IP or USB camera, web-camera, video file) or images. FaceStream can work with several sources of video streams (the number is set by the license). Sources and additional stream management settings are specified in the HTTP-request body to the LUNA Streams service. These settings are then retrieved by the FaceStream application.

2. FaceStream decodes video frames.

3. The ROI area is cut out from the frame if the "roi" (streams management configuration) parameter is specified.

4. The received image is scaled to the "scale-result-size" (TrackEngine configuration) size if the "detector-scaling" (TrackEngine configuration) is set.

5. Faces are detected in the frame.

6. The face is redetected in the frame instead of detection if the "detector-step" parameter (TrackEngine configuration) is set.

7. A track is created for each new face in the stream; then it is reinforced with new detections of this face from the subsequent frames. The track is interrupted if the face disappears from the frame. You can set the "skip-frames" parameter (TrackEngine configuration) so the track will not be interrupted immediately, and the system will wait for the face to appear in the area for several frames.

8. FaceStream filters the frames of low quality and selects best shots. There are several algorithms for choosing the best detection(s) in the track. See the "Frame filtering" section.

9. If the frame is best shot, it is added to the collection of best shots. Depending on the "face_-bestshots_to_send" (streams management configuration) setting one or several best detections are collected from each track.

10. **Optional.** If the "warp" type is set in the "portrait_type" (streams management configuration) parameter, the best shots are normalized to the LUNA PLATFORM standard, and normalized images are created. Normalized image is better for processing using LUNA PLATFORM.

11. The best shots, source images (optional) and additional information from stream management configuration are sent to the LUNA PLATFORM in the form of an HTTP request to the resource "/6/handlers/{handler_id}/stream_events" to generate an event.

    The general parameters of the video stream (data transfer protocol, path to the source, region of interest on the frame, etc.) are set in the "data" (streams management configuration) section.

    The frequency of images sending is specified in the "sending" (streams management configuration) section.

    The LUNA PLATFORM address is specified in the "lunaplatform" (FaceStream configuration) section.

2.1.2 FaceStream workflow with bodies

FaceStream application workflow with bodies is shown in the image below:
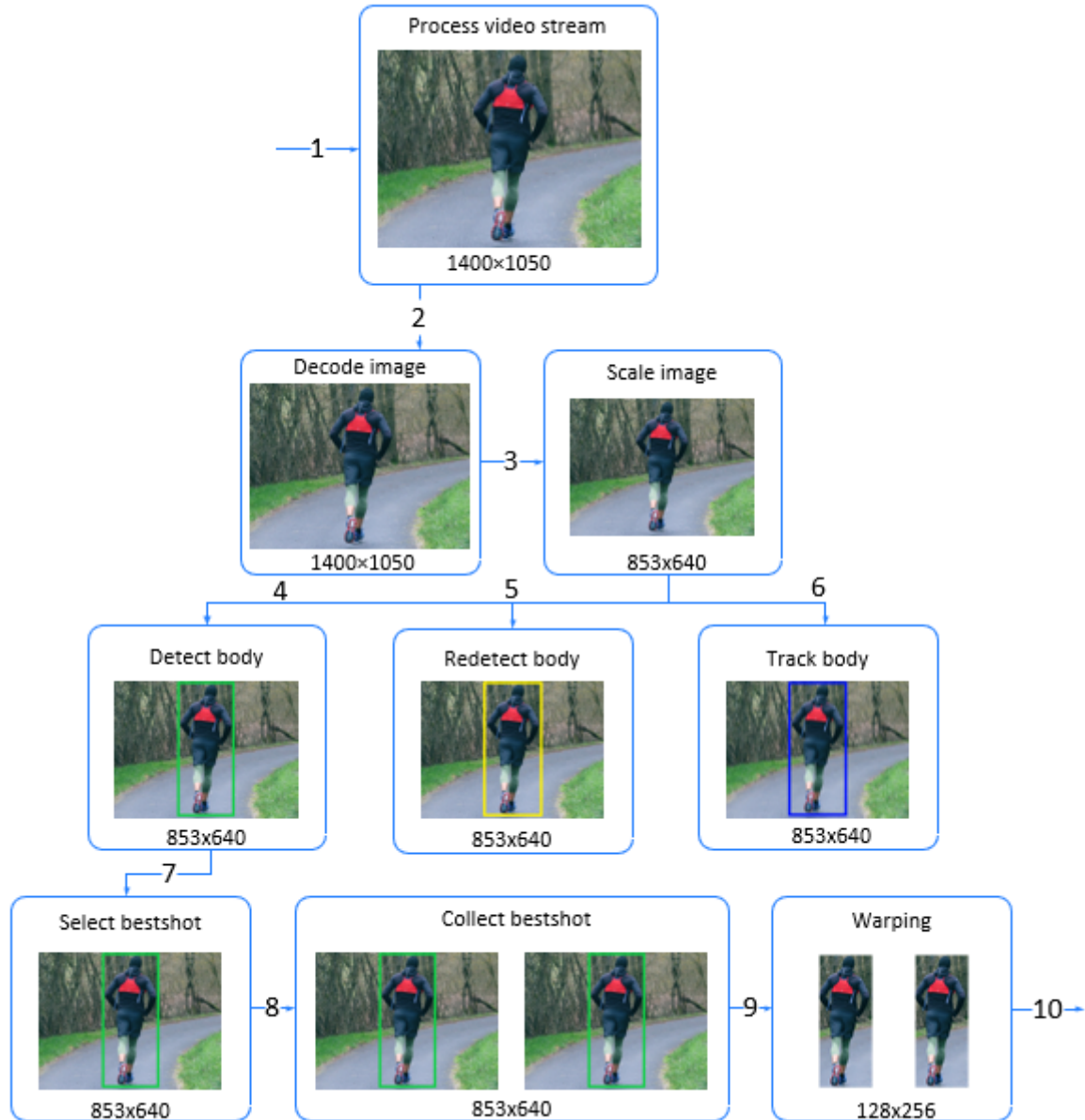
Figure 2: FaceStream workflow with bodies

1. FaceStream receives video from a source (IP or USB camera, web-camera, video file) or images. FaceStream can work with several sources of video streams (the number is set by the license). Sources and additional stream management settings are specified in the HTTP-request body to the LUNA Streams service. These settings are then retrieved by the FaceStream application.

2. FaceStream decodes video frames.

3. The received image is scaled to the "scale-result-size" (TrackEngine configuration) size if the "detector-scaling" (TrackEngine configuration) is set.

4. Bodies are detected in the frame.

5. The body is redetected in the frame instead of detection if the "detector-step" parameter (TrackEngine configuration) is set.

6. A track is created for each new body in the stream. then it is reinforced with new detections of this body from the subsequent frames. The track is interrupted if the body disappears from the frame. You can set the "skip-frames" parameter (TrackEngine configuration) so the track will not be interrupted immediately, and the system will wait for the body to appear in the area for several frames.

7. FaceStream filters low quality frames and selects the best shots. See "min_score_body" (streams management configuration).

8. If the frame is best shot, it is added to the collection of best shots. Depending on the "body_-bestshots_to_send" (streams management configuration) parameter one or several best detections are collected from each track.

9. The best shots are normalized to the LUNA PLATFORM standard, and normalized images are created. Normalized image is better for processing using LUNA PLATFORM.

10. The best shots, source images (optional) and additional information from stream management configuration are sent to the LUNA PLATFORM in the form of an HTTP request to the resource "/6/handlers/{handler_id}/stream_events" to generate an event. Along with the best shots, detections with the coordinates of the human body are sent. The number of detections is set in the "minimal_body_track_length_to_send" parameter (streams management configuration).

    The general parameters of the video stream (data transfer protocol, path to the source, region of interest on the frame, etc.) are set in the "data" (streams management configuration) section.

    The frequency of images sending is specified in the "sending" (streams management configuration) section.

    The LUNA PLATFORM address is specified in the "lunaplatform" (FaceStream configuration) section.

2.2 Interaction of FaceStream with LUNA Streams

To work with FaceStream, you should first launch an additional service — LUNA Streams (the default port is 5160). In the "create stream" request body to the LUNA Streams service, settings for stream management are specified. After sending the request, a stream is created, whose settings are taken by FaceStream for further processing. See the LUNA Streams Open API Specification for request examples.

LUNA Streams has its own user interface designed to work with streams. For more information, see "LUNA Streams user interface".

To use the LUNA Streams service, you should use the LUNA PLATFORM 5 services — LUNA Licenses and LUNA Configurator, as well as PostgreSQL or Oracle and Influx.

> The Influx database is needed for the purposes of monitoring the status of LUNA PLATFORM services. If necessary, monitoring can be disabled.

> The FaceStream documentation does not describe the use of an Oracle database.

> If necessary, you can launch LUNA Streams without LUNA Configurator. This method is not described in the documentation.

FaceStream is licensed using the LUNA PLATFORM 5 key, which contains information about the maximum number of streams that LUNA Streams can process. The license is regulated by the LUNA Licenses service.

> See the FaceStream installation manual for detailed information on activating the LUNA Streams license.

The PostgreSQL/Oracle database stores all the data of LUNA Streams.

2.2.1 LUNA Streams API versions

LUNA Streams has two API versions.

To switch between API versions, you need to update the corresponding value in the "api_version" parameter of the "lunastreams" section in the FaceStream settings.

The first API version is used by default. This documentation describes the second API version. See the description of the first API version in the FaceStream documentation v.5.1.49 and below.

Important: Attempts to execute requests to the second version of the API with the value "api_version" = "1" and vice versa will lead to errors, since for the first version of the API the LUNA PLATFORM address is specified in the body of the request to create a stream, and for the second version the address is specified in the "lunaplatform" parameter group in the FaceStream settings.

2.2.2 Stream creation and processing sequence diagram

The stream creation and processing sequence diagram is shown below:
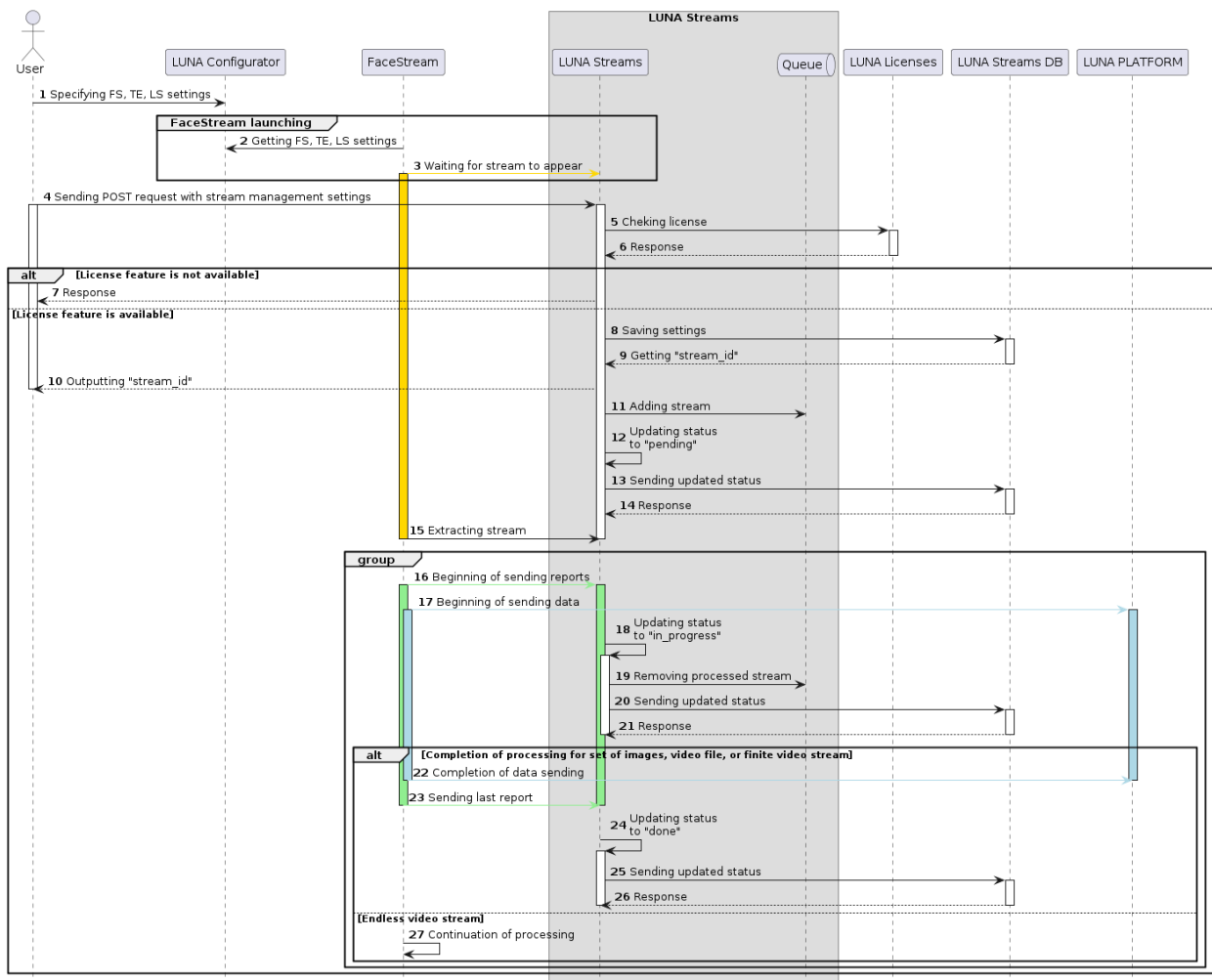
Figure 3: Stream creation and processing sequence diagram

1⬛ The user sets the settings of FaceStream, TrackEngine and the settings of the LUNA Streams service in the LUNA Configurator service.

> LUNA Streams service settings contain only service settings (LUNA Streams database address, InfluxDB address, etc.) and do not contain video stream management settings (parameters for sending the best shots, LP address, handler IDs, etc.). Such settings are set using a separate HTTP request after starting FaceStream (see point 4).

> If necessary, the user can set the FaceEngine settings in a separate configuration file.

2⬛ When starting FaceStream, the application reads the settings from the LUNA Configurator service.

3⬛ FaceStream switches to the waiting mode for the stream to appear.

4⬛ The user sends an HTTP request "create stream" to the LUNA Streams service containing stream management settings in the request body.

5⬛ The LUNA Streams service checks the availability of a license feature that regulates the number of streams for the operation of LUNA Streams in the LUNA Licenses service. If any streams are already being processed at the moment, then the number of streams already being processed at the time of the request is additionally checked using the FaceStream report (not reflected in the diagram, see point 17).

6⬛ The LUNA Licenses service returns a response.

7⬛ If there is no license feature or the maximum number of available streams is being processed at the time of stream creation, the corresponding error is returned to the user.

8⬛ If the license feature is present and the maximum number of available streams is not processed yet at the time of stream creation, the LUNA Streams service creates a stream and records the stream management settings in the LUNA Streams database.

9⬛ The LUNA Streams service receives the "stream_id".

10⬛ The LUNA Streams service returns the unique "stream_id" to the user.

11⬛ The LUNA Streams service adds a stream to the internal queue.

> The queue is implemented in the LUNA Streams service itself and is not external.

12⬛ The LUNA Streams service updates the status of the stream to "pending".

13⬛ The LUNA Streams service updates the status of the stream in the LUNA Streams database.

14⬛ The LUNA Streams service receives a response.

> If FaceStream is disabled at the time of stream creation, then only the number of streams with the "pending" status that is stipulated by the license can be created. After the FaceStream is launched, the streams created in the queue order will be accepted for processing.

> You can view the streams in the queue by filtering them in a certain way using the "streams/processing/queue" GET request.

> Streams can be created with the status "pause". In this case, they will be added to the database and will wait for a manual status update to "pending".

15⬛ FaceStream retrieves the stream(s) from the queue with the status "pending".

16⬛ FaceStream starts sending data to the main LUNA PLATFORM 5 services for further processing of frames according to the specified handler and generating events.

17⬛ FaceStream starts sending stream processing reports to LUNA Streams.

> The time of sending reports is fixed and cannot be changed.

18⬛ The LUNA Streams service updates the status of processed streams to "in_progress".

19⬛ The LUNA Streams service removes the processed stream from the queue.

20⯍ The LUNA Streams service updates the status of the stream in the LUNA Streams database.

21⯍ The LUNA Streams service receives a response.

Completion of processing for a set of images, a video file, or a finite video stream:

22⯍ FaceStream stops sending data to LUNA PLATFORM.

> If it is necessary for the video stream to be perceived as finite, then it is necessary to enable the "endless" parameter.

23⯍ FaceStream sends the latest report to the LUNA Streams service.

> If the report says that some stream has been processed, then the FaceStream handler takes the following parameters of the stream with the status "pending" from the LUNA Streams queue, and the service changes the status of the stream from "pending" to "in_progress", removing it from the queue. If, for unknown reasons, the report was not transferred, then the streams are re-queued.

24⯍ The LUNA Streams service updates the status of the stream to "done" in the Streams database.

25⯍ The LUNA Streams service updates the status of the stream in the LUNA Streams database.

26⯍ The LUNA Streams service receives a response.

Endless video stream:

27⯍ FaceStream will send data to LUNA PLATFORM and reports to LUNA Streams until the video stream is interrupted.

For a more detailed description of the processing of the LUNA Streams stream, see "The process of processing streams in LUNA Streams".

For more detailed description of LUNA Streams stream processing, see "Stream processing pipeline".

2.2.3 Requests to LUNA Streams

The following requests to the LUNA Streams service are available for working with streams:

- Set stream management settings ("create stream" request).
- Get existing streams by their "stream_id" with a description of the data of each stream ("get streams" request).
- Get all information about a stream by its "stream_id" ("get stream" request).
- Delete existing streams by their "stream_id" ("delete streams" request).
- Delete stream by its "stream_id" ("remove stream" request).
- Get the number of streams created ("count streams" request).
- Update the "description" and "status" fields of a stream by its "stream_id" ("update stream" request).
- Replace of all stream data with new ones by its "stream_id" ("put stream" request).

- Get link to the last frame ("get last frame preview" request).
- Get link to live stream ("get live preview" request).
- Get streams logs ("get streams logs" request).
- Delete streams logs ("delete streams logs" request).

> A detailed description of requests and example requests can be found in the Open API specification of LUNA Streams service.

2.2.4 Stream distribution in LUNA Streams

As mentioned earlier, the ability to process multiple streams at the same time is available.

For each stream, its current status is assumed:

- "pending" — Stream is waiting for FaceStream worker.
- "in_progress" — Stream processing is in progress.
- "done" — Stream processing is completed (relevant for stream transmission types "videofile"/"images" or "tcp/udp" with "endless" set to "false").
- "pause" — Stream processing is paused by user (not applicable for stream transmission types "videofile" or "images").
- "restart" — Stream processing is restarted by server.
- "cancel" — Stream processing is cancelled by user.
- "failure" — Stream processing is failed by FaceStream worker.
- "handler_lost" — Stream processing worker is lost, needs to be passed to another worker (not applicable for stream transmission types "videofile" or "images").
- "not_found" — Stream was removed during the processing.
- "deleted" — Stream was removed intentionally.

Statuses "pause" and "cancel" can be specified when updating a stream using the "update stream" request.

Statuses "restart", "handler_lost" are transient. With these statuses, it is impossible to receive a stream, however, the transition through these statuses is logged as usual. The "restart" status can only occur when using the "autorestart" section (see the "Streams automatic restart" section below).

The "not_found" status is internal and will be sent back for feedback if the stream was removed during processing. With this status, it is impossible to receive a stream.

The "deleted" status is virtual. Stream with this status cannot exist, but this status can be seen in the stream logs.

2.2.4.1 Statuses transition table

The following table shows statuses that may be received after each listed status.

The "+" symbol means that the status listed in the first row may occur after the status in the first column. An empty field means that there are no cases when the status may occur.

The "-" symbol means that there is no stream in the system (it was not created or it was already deleted).

	None	pending	in_-progress	done	restart	pause	cancel	failure	handler_-lost
None		+				+			
pending	+		+		+	+	+		
in_progress	+	+		+	+	+*	+	+	+
done	+	+			+	+			
restart		+				+			
pause	+	+			+		+		
cancel	+	+			+	+			
failure	+	+			+	+			
handler_-lost					+				

* not supported for stream transmission types "videofile" or "images"

2.2.5 Stream processing pipeline

By default, the new stream is created with the "pending" status and immediately enters the processing queue. Stream processing can be postponed by specifying the pause status when creating.

As soon as a free stream worker appears with a request for a pool from the queue, the stream is accepted for processing and it is assigned the "in_progress" status.

After the stream has been processed by the worker, it is assigned to the status "done" in case of success (relevant for stream transmission types "videofile"/"images" or "tcp/udp" with "endless" set to "false"), or "failure" if any errors have occurred. However, stream processing status may be downgraded from "in_progress" for the following reasons:

- No feedback from stream worker: process will be downgraded by server and record with "handler_-lost" status will be added to the stream logs.

- Replacing the stream by user: record with "restart" status will be added to the stream logs.

> For stream transmission types "tcp" or "udp" with "endless" set to "true", the status cannot change to "done".

During the processing routine, any change in the stream status is logged. Thus, you can restore the stream processing pipeline from the logs.

> The number of simultaneous processing streams (statuses "pending" and "in_progress") is regulated by the license, but the LUNA Streams database can store an infinite number of streams with a different status, for example, "pause".

Streams with "failure" status can be automatically restarted.

2.2.5.1 Streams automatic restart

The ability to automatically restart streams is relevant only for streams with a "failure" status. Automatic restart options (restart possibility, maximum number of restart attempts, delay between attempts) are specified by the user for each stream in the "autorestart" section of stream management settings. The parameters and automatic restart status can be received using the "get stream" request.

The automatic restart statuses are listed below:

- "disabled" — Stream automatic restart is disabled by user ("restart" parameter is disabled).
- "enabled" — Automatic restart is enabled but is not currently active because the stream is not in the "failure" status.
- "in_progress" — Automatic restart in progress.
- "failed" — Allowed number of automatic restart attempts was exceeded and none of the attempts were successful.
- "denied" — Automatic restart is allowed by the user, but not possible due to a fatal error* received in the FaceStream report.

* fatal error is considered a `Failed to authorize in Luna Platform` error.

The process of processing streams with automatic restart enabled is described below.

When an attempt is made to automatically reload the data stream, the following changes occur:

- Stream status changes first to "restart" and then to "pending".
- Counter of automatic restart attempts "current_attempt" increases by 1.
- Time record of the last attempt "last_attempt_time" is updated to reflect the current time.

In order for a restart to occur, the following conditions must be met:

- Status of the stream should be in the "failure" state.
- Automatic restart of the stream must be enabled (the "restart" parameter).
- Value of the current automatic restart attempt "current_attempt" must be equal to "null" or less than the maximum number of attempts "attempt_count".
- Time of the last attempt of automatic restart "last_attempt_time" should be equal to "null" or the difference between the current time and the time of the last attempt should be greater than or equal to the delay.

If the conditions below are met, then the automatic restart of the stream will fail (stopping restart attempts):

- Stream status is in the "failure" state.
- Status of the automatic restart of the stream is in the "in_progress" state.
- Value of the current automatic restart attempt "current_attempt" is equal to the value of the maximum number of attempts "attempt_count".

If the conditions below are met, then the automatic restart of the stream will be completed:

- Status of the stream is not equal to the "failure" status.
- Status of the automatic restart of the stream is in the "in_progress" state.
- Time of the last attempt of the automatic restart "last_attempt_time" is "null" or the difference between the current time and the time of the last attempt is greater than or equal to the delay.

> The completion of the automatic restart of the stream means changing the status of the automatic restart of the stream to "enabled" and resetting the values of the current attempt of automatic restart "current_attempt" and the time of the last attempt of automatic restart "last_attempt_time".

2.2.6 Streams logs automatic deletion

If necessary, you can start automatic deletion of stream logs. Automatic deletion of logs helps to clear the "log" table of the LUNA Streams database from a large number of unnecessary logs.

The most recent entry for each stream will not be deleted.

Automatic deletion of stream logs is configured using the following parameters from the "LUNA_-STREAMS_LOGS_CLEAR_INTERVAL" section:

- "interval" — Sets the interval for deleting logs. Logs older than this interval will be deleted.
- "interval_type" — Sets the interval type (weeks, days, hours, minutes, seconds).
- "check_interval" — Sets the frequency of checking logs for deletion (seconds).
- "active" — Enables/disables automatic deletion of stream logs.

By default, automatic deletion of logs is disabled ("active" = false).

The default settings include automatic deletion of logs ("active" = true) with checking of log streams in the database every 180 seconds ("check_interval" = 180) and delete logs older than 7 days ("interval" = 7 and "interval_type" = days).

Example of checking logs for deletion every 5 minutes and deleting logs older than 4 weeks:

```
{
    "interval": 4,
    "interval_type": "weeks",
    "check_interval": 300,
```

```
        "active": true
  }
```

2.2.7 Streams grouping

Streams can be grouped. Grouping is intended to combine streams with multiple cameras into logical groups. For example, you can group streams by territorial characteristic.

A stream can be linked to several groups.

The group is created using the "create group" request. To create a group, you need to specify the required parameters "account_id" and "group_name". If necessary, you can specify a description of the group.

Stream can be linked to a group in two ways:

- Using the "group_name" or "group_id" parameters during stream creation ("create stream" request).
- Using the "linker" request. In the request, you should specify the streams IDs and the group to which they need to be linked.

> Using the "linker" request you can also unlink streams from a group.

If the stream was linked to a group, then the "get stream" or "get streams" requests will show the group in the "groups" field.

2.2.8 LUNA Streams database description

The LUNA Streams database general scheme is shown below.
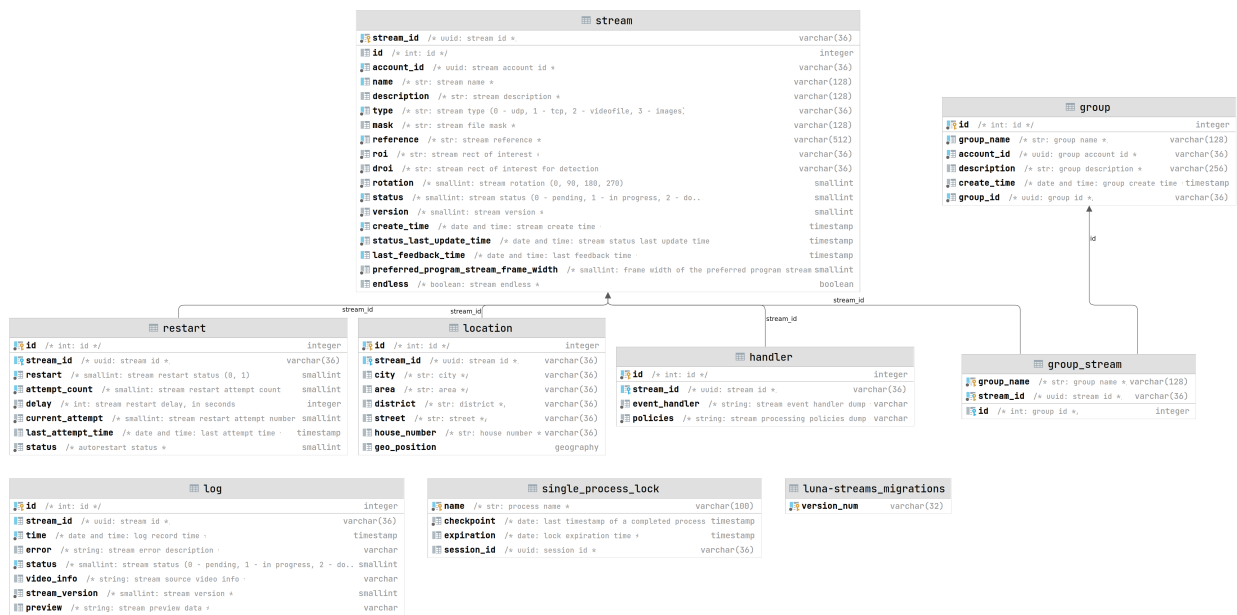
Figure 4: LUNA Streams database

See "Streams management configuration" for a description of the database data.

2.3 Recommendations for FaceStream configuration

This section provides general guidelines for setting up FaceStream.

The names of the configuration, which describes the configured parameters, are mentioned in this section.

2.3.1 Before starting configuration

You should perform the FaceStream configuration for each camera used separately. FaceStream should work with the stream of the camera, located in the standard operating conditions. The following reasons lead to these requirements:

- Frames with different cameras may differ by:
    - Noise level
    - Frame size
    - Light
    - Blurring
    - Etc.
- FaceStream settings depend on the lighting conditions, therefore, will be different for the cameras placed in a dark room and a light.
- FaceStream performance depends on the number of faces or bodies in the frame. Therefore, the settings for the camera, which detects one face every 10 seconds, will be different from the settings for the camera detecting 10 faces per second.
- The number of detected faces and bodies and the quality of these detections depend on correct location of the camera. When the camera is at a wrong angle, faces are not detected in frames. Moreover, head angles can also exceed the acceptable degree hence the frame with the detected face could not be used for further processing.
- Faces and bodies in the zone of camera view can be partially or completely blocked by some objects. There can be background objects that can prevent the proper functioning of recognition algorithms.

The camera can be positioned so that the lighting or shooting conditions change throughout the day. It is recommended to test FaceStream work under different conditions and choose the best mode, providing reliable FaceStream operation under any conditions.

You can specify the FPS for video processing using the "real_time_mode_fps" parameter.

2.3.2 FaceStream performance configuration

The mentioned above parameters have the greatest impact on the FaceStream performance.

2.3.2.1 Reduction of face search area

Not all the areas of the frame contain faces. Besides, not all the faces in the frame have the required size and quality. For example, the sizes of faces in the background may be too small, and the faces near the edge of the frame may have unacceptable pitch, roll, or yaw angles.

The "roi" parameter (streams management configuration, section "data"), enables you to specify a rectangular area to search for faces.

Figure 5: Source frame with DROI area specified

The specified rectangular area is cut out from the frame and FaceStream performs further processing using this image.

Figure 6: Cropped image processed by FaceStream

The smaller the search area, the less resources are required for processing each frame.

> Correct exploitation of the "roi" parameter significantly improves the performance of FaceStream.

> The parameter should be used only when working with faces.

2.3.2.2 Frame scaling

Frame scaling before processing can significantly increase the performance of FaceStream. Frame scaling can be enabled in the following settings:

Operating mode	Parameter name	Default value	Default state
Only with face detector	"minFaceSize" FaceEngine settings	50 (reduce by 2.5 times)	Enabled
Only with body detector	"ImageSize" FaceEngine settings	640 (reduce to 640 pixels on the largest side)	Enabled
With face detector and with body detector	"detector-scaling" and "scale-result-size" settings TrackEngine	640 (reduce to 640)	Only with body detector

2.3.3 Defining area with movement

Three parameters are responsible for determining the area with movement, set in the TrackEngine configuration:

- frg-subtractor — Enable the mode of considering for movements in the frame.
- frg-regions-alignment — Set alignment for areas with motion.
- frg-regions-square-alignment — Set equal width and height of the area with movement.

Below are the recommended values for these settings when using CPU and GPU:

	frg-subtractor	frg-regions-alignment	frg-regions-square-alignment
Recommended value when utilizing CPU	1	0	0
Recommended value when utilizing GPU	1	360	0

2.3.4 Batch processing of frames

Three parameters are responsible for batch processing of frames, set in TrackEngine configuration:

- batched-processing — Enable batch processing of frames.
- min-frames-batch-size — Minimal number of frames collected from all the cameras before processing.
- max-frames-batch-gather-timeout — Time between processing of the batches.

2.3.5 General configuration information

2.3.5.1 Working with track

A new track is created for each detected face or body. Best shots are defined and sent for each track.

In general, the track is interrupted when the face can no longer be found in the frame. If a track was interrupted and the same person appears in the frame, a new track is created.

There can be a situation when two faces or bodies interact in a frame (one person behind the other). In this case, the tracks for both persons are interrupted, and new tracks are created.

There can be a situation when a person turns away, or a face or body is temporarily blocked. In this case, you can specify the "skip-frames" parameter (TrackEngine configuration) instead of interrupting the track immediately. The parameter sets the number of frames during which the system will wait for the face to reappear in the area where it disappeared.

When working with the track, it is also useful to use the "detector-step" parameter, which enables you to specify the number of frames on which face redetection will be performed in the specified area before face detection is performed.

2.3.5.2 Best shot sending

The "sending" parameters group (streams management configuration) enables you to set parameters for the best shot sending. FaceStream sends the received best shots to LUNA PLATFORM (see "Priority parameters list").

You can send several best shots for the same face or body to increase the recognition accuracy. You should enable the "face_bestshots_to_send" or "body_bestshots_to_send" (streams management configuration) parameters in this case.

> LUNA PLATFORM enables you to aggregate the best shots and create a single descriptor of a better quality using them.

If the required number of best shots was not collected during the specified period or when the track was interrupted the collected best shots are sent.

The "time_period_of_searching" and "silent_period" parameters (streams management configuration) can be specified in seconds or in frames. Use the "type" parameter to choose the type.

Note: The "silent_period" parameter only works when the face detection mode is enabled. It will not work when working with bodies or in collaborative mode.

The general options for configuring the "time_period_of_searching" and "silent_period" parameters of the "sending" section from streams management configuration when working with faces are listed below.

- The best shot is sent after the track is interrupted and the person left the video camera zone of view.

  All the frames with the person's face or body are processed and the best shot is selected.

```
    time_period_of_searching = -1
    silent_period  = 0
```

- It is required to quickly receive the best shot and then send best shots with the specified frequency.

  For example, it is required to send a best shot soon after an intruder entered the shop. The intruder will be identified by the blacklist.

  The mode is also used for the demonstration of FaceStream capabilities in real-time.

  > The best shot will be sent after the track is interrupted even if the specified period did not exceed.

```
    time_period_of_searching = 3
    silent_period  = 0
```

- It is required to quickly send the best shot and then send the best shot only if the person is in the frame for a long time.

```
    time_period_of_searching = 3
    silent_period  = 20
```

- It is required to quickly send the best shot and never send the best shot from this track again.

```
    time_period_of_searching = 3
    silent_period  = -1
```

2.3.5.3 Frames filtration

The filtration of face frames is performed by three main criteria (they are all set in the streams management configuration):

- Head pose ("detection_yaw_threshold", "detection_pitch_threshold", "detection_roll_-threshold").

  The "yaw_number" and "yaw_collection_mode" parameters are additionally set for the yaw angle. The parameters reduce the possibility of the error occurrence when the "0" angle is returned instead of a large angle.

- Frame quality for further processing ("min_score_face").

- Mouth occlusion ("mouth_occlusion_threshold").

If a frame did not pass at least one of the specified filters, it cannot be selected as a best shot.

If the "face_bestshots_to_send" or "body_bestshots_to_send" parameter is set, the frame is added to the array of best shots to send. If the required number of best shots to send was already collected, the one with the lowest frame quality score is replaced with the new best shot if its quality is higher.

The filtration of body frames is performed only by one criterion - "min_score_body".

2.3.5.4 Working with ACMS
Work with ACMS is performed only with faces.

Use the "primary_track_policy" settings when working with ACMS. The settings enables you to activate the mode for working with a single face, which has the largest size. It is considered, that the face of interest is close to the camera.

The track of the largest face in the frame becomes primary. Other faces in the frame are detected but they are not processed. Best shots are not sent for these faces.

As soon as another face reaches a larger size than the face from the primary track, this face track becomes primary and the processing is performed for it.

The mode is enabled using the "use_primary_track_policy" parameter.

The definition of the best shots is performed only after the size (vertical) of the face reaches the value specified in the "best_shot_min_size" parameter. Frames with smaller faces can't be the best shots. When the face detection vertical size reached the value set in the "best_shot_proper_size" parameter the best shot is sent as a best shot at once.

> The "best_shot_min_size" and "best_shot_proper_size" are set depending on the video camera used and its location.

The examples below show configuration of the "sending" group parameters from streams management configuration for working with ACMS.

- The turnstile will only open once. To re-open the turnstile you should interrupt the track (move away from the video camera zone of view).

```
time_period_of_searching = -1
silent_period  = 0
```

- The turnstile will open at certain intervals (in this case, every three seconds) if a person stands directly in front of it.

```
time_period_of_searching = 3
silent_period  = 0
```

If the "use_primary_track_policy" parameter is enabled, the best shot is never sent when the track is interrupted.

2.4 Formats, video compression standards, and protocols

FaceStream utilizes the FFMPEG library to convert videos and get a stream using various protocols. All the main formats, video compression standards, and protocols that were tested when working with FaceStream are listed in this section.

> FFMPEG supports more formats and video compression standards. They are not listed in this section, because they are rarely used when working with FaceStream.

2.4.1 Video formats

Video formats that are processed using FaceStream:

- AVI
- MP4
- MOV
- MKV
- FLV

2.4.2 Encodings (codecs)

Basic video compression standards that FaceStream works with:

- MPEG1, MPEG2, MPEG3, MPEG4
- MS MPEG4
- MS MPEG4v2
- MJPEG
- H.264
- H.265
- VC1
- HEVC
- VP8
- VP9
- AV1
- Other

2.4.3 Protocols

Basic transport layer protocols used by FaceStream to receive data:

- TCP
- UDP

Basic application layer protocols used by FaceStream to receive data:

- HTTP (based on the transport layer protocol **TCP**)
- HTTPS (based on the transport layer protocol **TCP**)
- RTP (based on the transport layer protocol **UDP**)
- RTSP (based on transport layer protocols **TCP** or **UDP**)
- RTMP (based on the transport layer protocol **TCP**)
- HLS (based on the transport layer protocol **TCP**)

To use application layer protocols, you must specify the appropriate transport layer protocol in the "type" parameter of the streams management settings.

2.5 Memory consumption when running FaceStream

This section lists the reasons for increasing RAM consumption when running FaceStream.

- Each stream increases memory consumption. The amount of the consumed memory depends on the settings set for FaceStream:

    - Number of Ffmpeg threads in the "ffmpeg_threads_number" parameter (streams management configuration).

    - Image cache size in the "stream_images_buffer_max_size" parameter (FaceStream configuration).

    - Set buffer sizes in the "frames-buffer-size" parameter (TrackEngine configuration).

- If the number of threads specified in the "ffmpeg_threads_number" parameter is greater than "1" (streams management configuration), the memory consumption increases significantly. At the same time, the increase in consumption is extremely slow and can be noticed after several hours of operation only.

    > For RTSP streams, you can set the "ffmpeg_threads_number" parameter to "0" or "1" (streams management configuration). In this case, memory growth is not noticed.

- Memory consumption increases after FaceStream starts. Growth occurs within 1-2 hours. This is related to caches filling (see point 1). If no new streams are created and step 2 is not executed, the memory consumption stops growing.

- Memory consumption increases when settings in the Debug section are enabled (FaceStream and TrackEngine configurations).

2.6 Stream playback interface

FaceStream has the ability to view the stream in real time. To view the stream, you should enter the following address in the browser bar after the FaceStream starts processing the stream:

```
http://127.0.0.1:34569/api/1/streams/preview/<stream_id>.
```

When objects appear in the camera's field of view, FaceStream displays them in a certain way.

Yellow bounding box occurs if a detection fails at least one of the "detection_yaw_threshold", "detection_pitch_threshold" or "detection_roll_threshold" parameters.

Red color bounding box occurs if the detection acceptance score is lower than the value specified in the "min_score_face" or "min_score_body" parameters.

Blue color bounding box occurs when an object is detected (redetected) or tracked.

Green bounding box occurs in all other cases when all conditions are met.

Orange bounding box occurs when using ROI.

2.7 Face and body detection collaborative mode

FaceStream has a face and body detection collaborative mode available.

Important: The functionality is in beta testing. Some functions may not work.

Collaborative mode is enabled by simultaneously enabling the "use-face-detector" and "use-body-detector" settings. The use of collaborative mode is controlled by the "data" > "analytics" > "mode" setting in the stream management settings.

When you enable collaborative mode, FaceStream sends the following data to the `handlers/{handler_id}/stream_events` LUNA PLATFORM resource:

- Face and body best shots
- Detection time
- Time relative to the beginning of the video file
- Coordinates of faces and bodies
- Source images

In collaborative mode, the threshold "min_score_body" is used.

Periodic sending of best shots in collaborative mode works similarly to working with bodies. This means that the "silent_period" parameter will not work.

2.7.1 Unsupported features

The following features are not currently used in collaboration mode:

- Primary Track Policy.
- Parameter "data" > "analytics" > "send" > "full_frame_settings" that determines which source frames will be sent (face, body, or face and body). Currently, only source face and body frames are sent.
- Periodic sending of faces. At the moment, the scenario of periodic sending of bodies is working (see above).
- Parameter "data" > "analytics" > "sending" > "bestshot_settings" > "type" that determines which images will be sent best (face, body, or face and body). At this time, only the best shots of faces and bodies will be featured.

3 Priority parameters list

To send photo images to the LUNA PLATFORM, first of all, you need to configure FaceStream to work with faces or bodies (see parameters for switching detection mode below), as well as configure the basic parameters necessary for the correct operation of the application. All parameters are separated as follows:

- FaceStream parameters are set in the in the "FACE_STREAM_CONFIG" section in the Configurator or in the "fs3config.conf" configuration file.

- Streams management parameters are set in a request with a body in JSON format to the "/streams" resource.

- TrackEngine parameters are set in the in the "TRACK_ENGINE_CONFIG" section in the Configurator or in the "trackengine.conf" configuration file.

See the detailed description of the parameters listed below in the relevant sections.

The following common parameters are available for sending both faces and bodies:

Table 5: FaceStream parameters

Parameter	Description
sending > async_requests	Enables you to switch between asynchronous and synchronous request sending modes in LUNA PLATFORM
sending > send_source_frame	Enables sending the source frame to LUNA PLATFORM
sending > jpeg_quality_level	Enables you to set the compression ratio of the source frame

Table 6: Streams management parameters

Parameter	Description
data > analytics > event_handler > frame_store	Enables you to set the URL of the Image Store service to send the source frame

3.1 Parameters for sending faces

The parameters for sending face images to LUNA PLATFORM 5 are listed below.

Table 7: TrackEngine parameters

Parameter	Description
use-face-detector	Enables face detection — `1`
use-body-detector	Enables body detection — `0`

Table 8: FaceStream parameters

Parameter	Description
lunastreams > api_version	The version of the API of the LUNA Streams service — `1`
lunastreams > origin	Full network path to LUNA Streams service
lunaplatform > api_version	The version of the API of the LUNA API service — `6`
lunaplatform > origin	Full network path to LUNA API service
sending > request_type	Request type for sending images to LP — `jpeg`
sending > portrait_type	Image transfer format — `warp`
sending > aggregate_attr_requests	Enables aggregation of the best shots to get a single descriptor in LUNA PLATFORM — `true` or `false`

Table 9: Streams management parameters

Parameter	Description
data > type	Type of signal source (tcp, udp, videofile, images) — `string`
account_id	"Luna_account_id", to which the request is related — `string in UUID format`

Parameter	Description
data > analytics > event_handler > handler_id	LP handler that enables you to flexibly configure the faces processing — `string in UUID format`

> For detailed information about the handlers, see the documentation APIReferenceManual.html included in the LUNA PLATFORM 5 distribution package.

3.2 Parameters for sending bodies

The parameters for sending body images to LUNA PLATFORM 5 are listed below.

Table 10: TrackEngine parameters

Parameter	Description
use-face-detector	Enables face detection — `0`
use-body-detector	Enables body detection — `1`

Table 11: FaceStream parameters

Parameter	Description
lunastreams > api_version	The version of the API of the LUNA Streams service — 1
lunastreams > origin	Full network path to LUNA Streams service
sending > aggregate_attr_requests	Enables aggregation of the best shots to get a single descriptor in LUNA PLATFORM — `true` or `false`
sending > minimal_body_track_length_to_send	This parameter enables the sending of detections with the coordinates of the human body — x, y, width and height and sets the number of detections, less than the value of which they will not be sent — 3

Parameter	Description
sending > detection_path_length	This parameter sets the maximum number of detections for the "minimal_body_track_length_to_send" parameter. — 100

Table 12: Streams management parameters

Parameter	Description
data > type	Type of signal source (tcp, udp, videofile, images) — `string`
account_id	"Luna_account_id", to which the request is related — `string in UUID format`
data > analytics > event_handler > handler_id	LP handler that enables you to flexibly configure the processing of bodies — `string in UUID format`

For detailed information about the handlers, see the documentation APIReferenceManual.html included in the LUNA PLATFORM 5 distribution package.

4 Streams management configuration

Parameters for stream management are set in the LUNA Streams service. The service enables you to create and store streams in the LUNA Streams database.

Important: LUNA Streams has two API versions. This documentation describes the second version of the API. See the description of the first version of the API in the documentation of FaceStream v.5.1.49 and below.

Important: Streams management settings are not stored in the LUNA Configurator service and can only be set using HTTP requests to the LUNA Streams service. The LUNA Streams settings set in LUNA Configurator are described in the section "LUNA Streams settings".

A basic description of the parameters is given in the LUNA Streams OpenAPI specification. This section provides extended descriptions for some parameters.

In fact, all stream management settings can be divided into the following categories:

- General stream settings — Stream name, description, restart policy on error, etc.
- Stream processing settings — Information about the type of stream, its address and some additional information.
- Analytics settings — Analysis of the content of the stream and setting up filtering for sending to an external system.
- Liveness settings
- Primary track policy settings

4.1 General stream settings

General settings enables you to define key parameters necessary for authorization, control and characterization of the stream.

General stream settings:

- "account_id" — Sets the value of the required field "account_id", which is sent in the request header in LUNA PLATFORM 5 to the LUNA API service. The parameter is used to bind the received data to a specific user.
- "name" — Stream name. Serves to identify the source of sent frames.
- "description" — Custom description of the stream.
- "location" — Sets information about the location of the video source (city, region, district, etc.).
- "autorestart" — Setting for automatically restarting the stream. See "Streams automatic restart" for more information.
- "status" — Stream status at the start of processing.
- "group_name" and "group_id" - Parameters for linking a stream to group.

The "data" section is also available, which specifies the main settings for processing the stream. See "Stream processing settings" for more details.

4.2 Stream processing settings

To begin processing, FaceStream must receive information about the type of stream, its address and some additional information. The corresponding parameters are set in the "data" section.

Basic settings in the "data" section:

- "type" — Stream transmission type:
  - TCP network protocol
  - UDP network protocol
  - set of images
  - video file
- "reference" — Source of the stream (link, path to a video file or set of images, etc.)
- "roi" — Region of interest in which the detection and tracking of a face in the frame occurs.
- "ffmpeg_threads_number" — Number of threads for video decoding using FFMPEG.
- "real_time_mode_fps" — Number of FPS with which the video file will be processed.
- "frame_processing_mode" — Parameter that determines whether the full or scaled frame will be processed.
- "rotation" — Angle of rotation of the image from the source. Used if the incoming stream is rotated, for example if the camera is installed on the ceiling. The rotation is performed clockwise.
- "preferred_program_stream_frame_width" — Enables you to automatically select the optimal channel from several in the stream.
- "endless" — Control of stream restart when receiving a network error.
- "mask" — Mask of file names in the directory with images.

Analytics settings are also specified in the "data" section. See "Analytics settings" for more details.

4.2.1 type

Stream transfer type. After selecting the stream transfer type, you must specify the path to the source/images/USB device, etc. in setting "reference".

FaceStream can use one of the following stream transfer types:

- **tcp** — Transport layer network protocol to receive video data.
- **udp** — Transport layer network protocol to receive video data.
- **images** — Set of frames as separate image files.
- **videofile** — Video file.

> Only transport layer protocols (TCP or UDP) are specified in FaceStream. It is necessary to understand on which transport layer protocol the application layer protocol is based (HTTP, RTSP, HLS, etc.). See "Protocols" for details.

TCP Protocol implements an error control mechanism that minimizes the loss of information and the skip of the reference frames at the cost of increasing the network delay. **Key frames** are the basis of various compression algorithms used in video codecs (for example, h264). Only the reference frames contain enough information to restore (decode) the image completely, while the **intermediate frames** contain only differences between adjacent reference frames.

In terms of broadcasting on the network, there is a risk of package loss due to imperfect communication channels. In case of loss of the package containing the data keyframe, the stream fragment cannot be correctly decoded. Consequently, distinctive artifacts appear, that are easily and visually distinguishable. These artifacts do not allow the face detector to operate in normal mode.

The **UDP protocol** does not implement an error control mechanism, so the stream is not protected from damage. The use of this protocol is recommended only, if there is a high-quality network infrastructure.

> With a large number of streams (10 or more), it is strongly recommended to use the **UDP protocol**. When using the **TCP protocol**, there may be problems with reading streams.

FaceStream processes data from the **images** and **videofile** types only **once**. After processing all images or video files, a message indicating the completion of processing will be displayed in the FaceStream logs. If a set of images or video files have been changed, then you need to restart the stream processing, after which FaceStream will again process all the images or video file once.

4.2.2 reference

Full path to the source (for "tcp"/"udp" type):

```
"reference": "rtsp://some_stream_address"
```

USB device number (for "tcp"/"udp" type):

```
"reference": "/dev/video0"
```

> To use USB device, you should specify the `--device` flag with the address of the USB device when launching the FaceStream Docker container. See the "Launching keys" section of the FaceStream installation manual.

Full path to the video file (for "videofile" type):

```
"reference": "/example/path/to/video/video.mp4"
```

Full path to the directory with the images (for "images" type):

```
"reference": "/example/path/to/images/"
```

> To use video files and images, you should first move them to a docker container.

4.2.3 roi

This parameter is used only for working with faces.

ROI specifies the region of interest in which the face detection and tracking are performed.

The specified rectangular area is cut out from the frame and FaceStream performs further processing using this image.

> Correct exploitation of the "roi" parameter significantly improves the performance of FaceStream.

ROI on the source frame is specified by the "x", "y", "width", "height" and "mode" parameters, where:

- "x", "y" – Coordinates of the upper left point of the ROI area of interest.
- "width" and "height" – Width and height of the processed area of the frame.
- "mode" – Mode for specifying "x", "y", "width" and "height". Two modes are available:
    - "abs" – Parameters "x", "y", "width" and "height" are set in pixels.
    - "percent" – Parameters "x", "y", "width" and "height" are set as percentages of the current frame size.

    If the "mode" field is not specified in the request body, then the value "abs" will be used.

With width and height values of "0", the entire frame is considered the region of interest.

The coordinate system on the image is set similarly to the figure below.
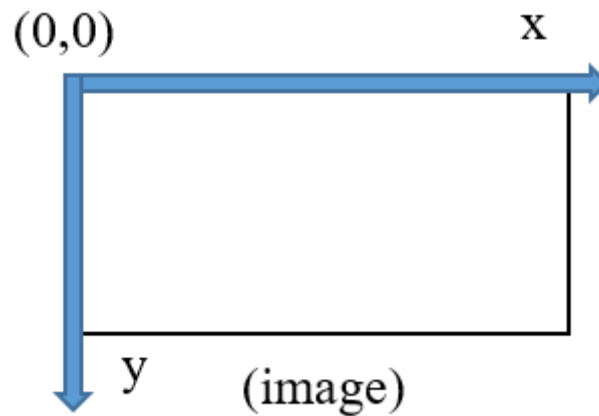
Figure 7: ROI coordinate system

Below is an example of calculating ROI as a percentage:
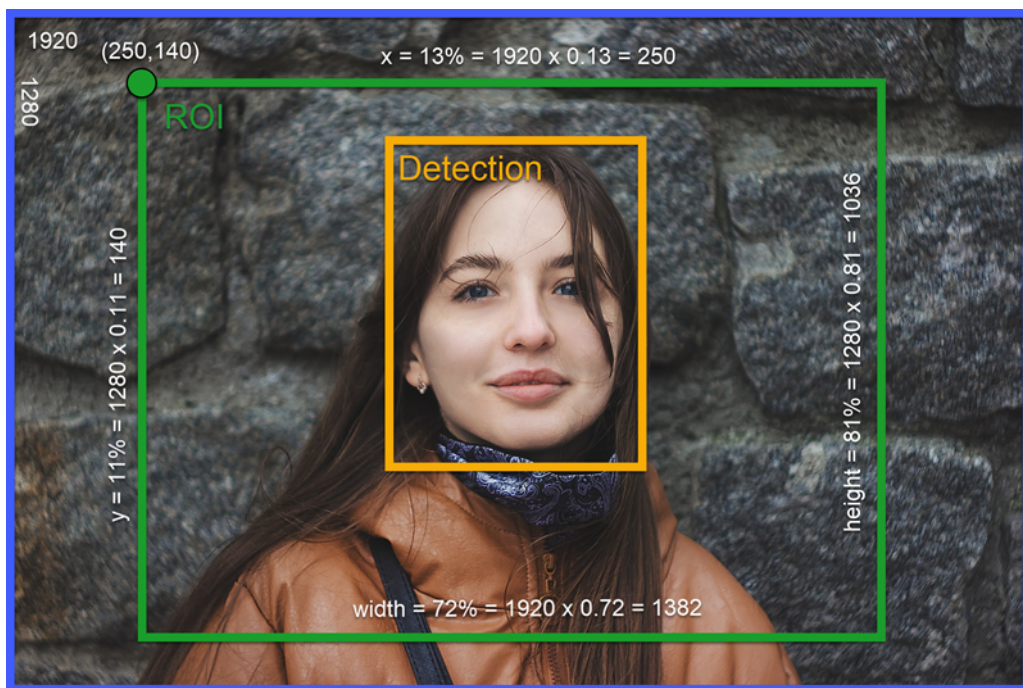
Figure 8: Example of calculating ROI as a percentage

4.2.4 frame_processing_mode

This parameter is used for "tcp", "udp" and "videofile" types only.

This parameter is similar to convert_full_frame, but is set for a specific FaceStream instance.

If the value is set to "full", the frame is immediately converted to RGB image of the required size after decoding. This results in a better image quality and reduces the speed of frames processing.

When set to "scale", the image is scaled according to the settings in the TrackEngine configuration (standard behavior for releases 3.2.4 and earlier).

The default value is "auto". In this case, one of the two modes is selected automatically.

4.2.5 real_time_mode_fps

This parameter is used for "videofile" type only.

This parameter enables you to set the number of FPS with which the video stream will be processed.

If a video has high FPS value and FaceStream cannot work with the specified number of frames per second, frames are skipped.

Thus, the video file emits a stream from a real video camera. It can be useful for performance tuning. The video will be played at the specified speed, which is convenient for load testing and subsequent analysis.

This parameter is disabled when set to "0".

4.2.6 ffmpeg_threads_number

The parameter enables to specify the number of threads for decoding video using FFMpeg.

The number of processor cores involved in decoding process increases according to the number of threads. An increase in the number of threads is recommended when processing high-resolution video (4K or higher).

4.2.7 preferred_program_stream_frame_width

This parameter is used for tcp** or **udp** types only.**

This parameter is intended to work with protocols that imply the presence of several channels with different bitrates and resolutions (for example, HLS).

If the stream has several such channels, then this parameter will enable you to select from all the channels of the whole stream the channel whose frame width is closer to the value specified in this parameter.

For example, there are 4 channels whose frame widths are 100, 500, 1000 and 1400. If the parameter "preferred_program_stream_frame_width" is equal to "800", then a channel with a frame width of 1000 will be selected.

If the stream has only one channel, this parameter will be ignored. You can set the value very large to deliberately select the largest channel.

In FaceStream logs (tag "ffmpeg") the selected value will be indicated as a message:

```
Url: [url] selected preferred stream #[res], [width]x[height] preferred
    width: [preferredProgramStreamFrameWidth]
```

The default value is 800.

4.2.8 endless

This parameter enables you to control the restart of the stream when a network error is received (the error is determined by the system as an "eof" (end-of-file) marker).

The parameter is available only for the "udp" and "tcp" source types.

If the "endless" parameter takes the value "true", then in case of receiving "eof" and successful reconnection, the processing of the stream will continue. If all reconnection attempts failed (see the "healthcheck" section), then the stream will take the "failure" status. If the parameter takes the value "false", then the processing of the stream will not continue and the status of the stream will take the "done" status.

When using a video file as a "tcp" or "udp" source, it is assumed to use the value "false". This will avoid re-processing an already processed fragment of the video file when receiving "eof". If, when using a video file, the value of the "endless" parameter is "true", then after the processing is completed, the video file will be processed from the beginning.

4.2.9 mask

This parameter is used for images** type only.**

A mask of file names in the directory with images. The mask allows FaceStream to understand which files from the specified folder should be used and in what order.

If you set the mask "Img_%02d.jpg", then FaceStream will take from the folder files which names consist of: Prefix (Img_) + two-digit number (%02d) + format (.jpg)

The following images will be taken in turn:

- Img_00.jpg
- Img_01.jpg
- Img_02.jpg
- Img_03.jpg

Another example of a mask is Photo-%09d.jpg. The following images will be taken:

- Photo-000000000.jpg
- Photo-000000001.jpg
- Photo-000000002.jpg

- Photo-000000003.jpg

FaceStream processes files in numerical order and does not skip nonexistent files. If there is a missing file in the file sequence FaceStream stops files processing.

The specified mask "example1_%04d.jpg" in the example will result in image processing, which name is composed of an "example1_" prefix and of a sequential frame number, of 4 characters size (for example: example1_0001.jpg, example1_0002.jpg, etc.).

```
"mask": "example1_%04d.jpg"
```

4.3 Analytics settings

Analytics refers to analyzing the content of a video stream to extract key data and characteristics. Setting analytics settings also provides the ability to configure data filtering and send it to an external system. The corresponding parameters are set in the "analytics" section.

General settings in the "analytics" section:

- "enabled" — Enable analytics.
- "mode" — Analytics mode (1 — only faces, 2 — only bodies, 3 — bodies and faces).
- "droi" — Source frame selection area.
- "filtering" — Objects for filtering images and sending the resulting best shots.
- "sending" — Setting up parameters related to compiling a collection of the best shots, as well as the period during which the frames will be analyzed to select the best shot.
- "event_handler" — Setting parameters related to integration with an external system for subsequent frame processing.
- "healthcheck" — Setting parameters responsible for reconnecting to the stream if errors occur during video streaming.

Also in the "analytics" section, settings for the Liveness check and the Primary Track policy are specified. As a rule, these parameters are rarely used in cooperative mode. See sections "Liveness Settings" and "Primary Track Policy Settings" for more information

4.3.1 Best shot selection region

This parameter is used only for working with faces.

The region of interest within the source frame or ROI is specified using the "droi" parameter. If an ROI region is used, then face detection is performed in the ROI region, but the best shot is selected only in the DROI region. Face detection must be completely within the DROI zone for the frame to be considered as a best shot. Neither detection side should protrude from the DROI area even by a millimeter.

DROI is recommended to use when working with Access Control Systems and when the "use_mask_-liveness_filtration" mode is enabled.

For example, it can be used, if there are several turnstiles close to each other and their cameras should find faces only in a small area and simultaneously perform Liveness check. Using DROI enables to limit the area of the best shot selection without losing information about the background.

DROI on the source frame is specified by the "x", "y", "width", "height" and "mode" parameters, where:

- "x", "y" – Coordinates of the upper left point of the DROI.
- "width" and "height" – Width and height of the processed area of the frame.
- "mode" – Mode for specifying "x", "y", "width" and "height". Two modes are available:
    - "abs" – Parameters "x", "y", "width" and "height" are set in pixels.
    - "percent" – Parameters "x", "y", "width" and "height" are set as percentages of the current frame size.
  If the "mode" field is not specified in the request body, then the value "abs" will be used.

> When calculating DROI, one must take into account that this region of interest is calculated relative to the original frame, and not relative to ROI.
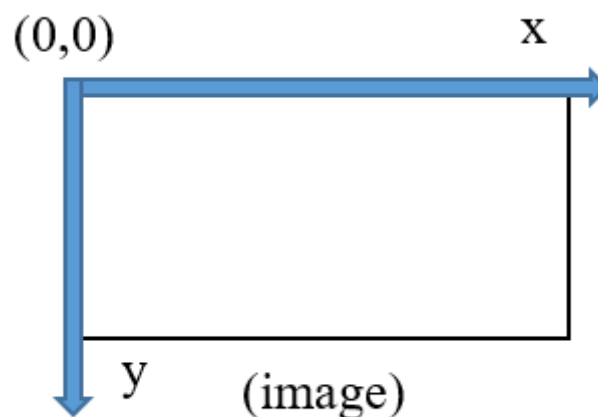
Figure 9: DROI coordinate system

When the ROI size is changed and the DROI size remains the default (0, 0, 0, 0), the DROI is not considered. If you change the size of the DROI, it will be considered when choosing the best shot.

Below is an example of calculating DROI as a percentage:
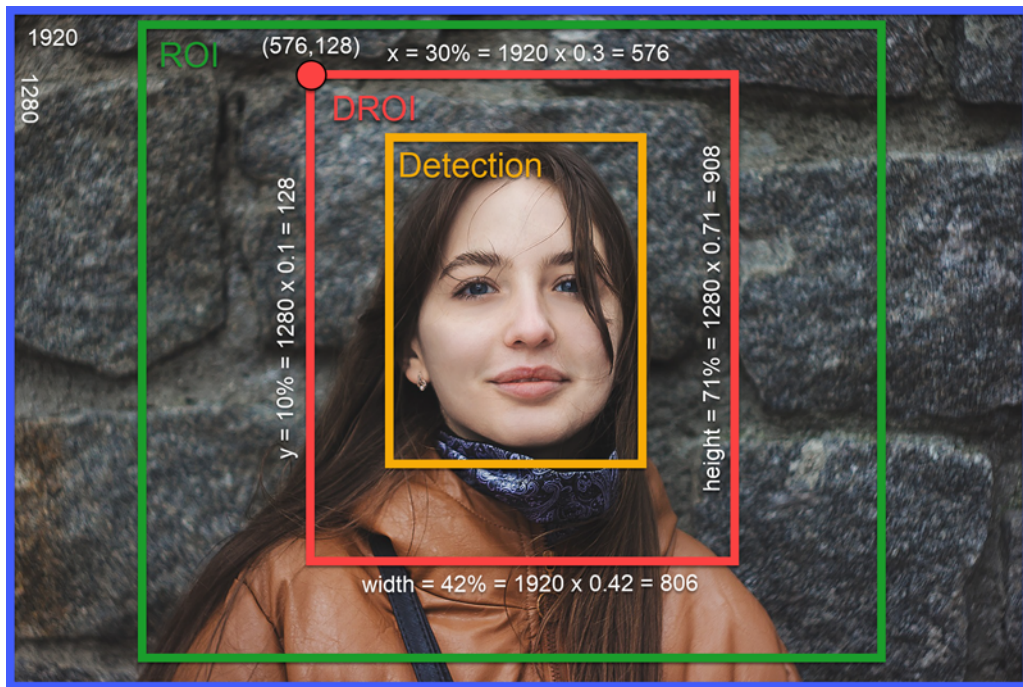
Figure 10: Example of calculating DROI as a percentage

4.3.2 Frame filtering

The "filtering" section describes objects for filtering images and sending the resulting best shots.

4.3.2.1 min_score_face

This parameter sets the threshold for filtering face detections sent to the server.

For each detection, the AGS (Approximate Garbage Score) and the general detection quality are calculated.

Detections whose AGS value is less than the value specified in the "min_score_face" threshold will not be considered acceptable for further work. Next, the best shots will be selected according to the filtered detections in accordance with the general detection quality.

If the "min_score_face" parameter is set to "0", then the best shot will be determined by the general detections quality.

Default value is **0.5187**.

> Recommended value was established through research and analysis of detections on various face and body images.

4.3.2.2 min_score_body

This parameter sets the threshold for filtering body detections sent to the server.

For each detection, the general detection quality is calculated.

Detections whose overall quality is less than the value specified in the "min_score_body" threshold will not be considered acceptable for further work. Next, the best shots will be selected based will be selected according to the filtered detections.

If the "min_score_body" parameter is set to "0", then the best shots will be determined by the general detections quality.

Default value is **0.5**.

> Recommended value was established through research and analysis of detections on various face and body images.

4.3.2.3 detection_yaw_threshold

This parameter is used only for working with faces.

This parameter sets the maximum value of head yaw angle in relation to camera.

If, in a frame, head yaw angle is above the value of this parameter, the frame is considered as **not** appropriate for further analysis.

To disable this filtering, you must set the value "180".

4.3.2.4 detection_pitch_threshold

This parameter is used only for working with faces.

This parameter sets the maximum value of head pitch angle in relation to camera.

If, in a frame, head pitch angle is above the value of this parameter, the frame is considered as **not** appropriate for further analysis.

To disable this filtering, you must set the value "180".

4.3.2.5 detection_roll_threshold

This parameter is used only for working with faces.

This parameter sets the maximum value of head yaw angle in relation to camera.

If, in a frame, head roll angle is above the value of this parameter, the frame is considered as not appropriate for further analysis.
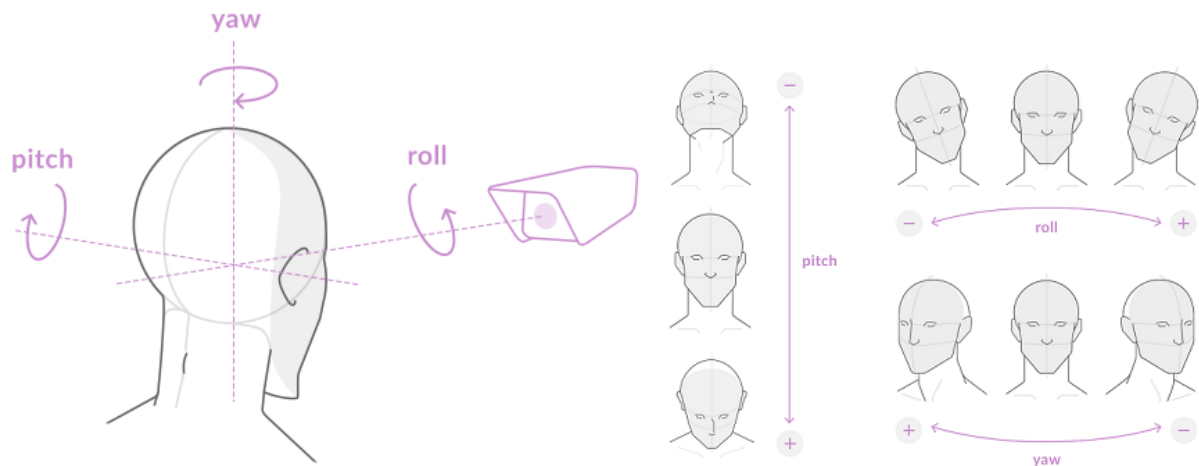
Figure 11: Head pose

To disable this filtering, you must set the value "180".

4.3.2.6 yaw_number

This parameter is used only for working with faces.

This parameter defines the number of frames for image filtration based on head yaw angle. This filter removes images where head's yaw angle is too high.

How it works:

Parameter specifies the number of frames to analyze. A special algorithm analyzes head yaw angles on each of those frames. If on one of them the angle is significantly different from the average value of angles, the frame will not be considered as a candidate for best shot.

Example. Parameter value is set "7", meaning 7 frames will be analyzed. If on six of the frames the rotation angle is in the range between 50-60 degrees and the angle on the seventh frame is estimated at 0, the angle on the seventh frame is, most likely, estimated incorrectly. Reason is: a person cannot turn his head so abruptly in such short period of time. The seventh frame will not be considered for best shot.

By default, the parameter is disabled, the value is "1". The recommended value is "7".

4.3.2.7 yaw_collection_mode

This parameter is used only for working with faces.

This parameter sets the number of frames the system must collect the number of frames specified in the "yaw_number" parameter to analyze the head yaw angle.

If "yaw_collection_mode" parameter is disabled, the system will analyze the frames sequentially, meaning it analyzes one frame, then two, then three and so on. Maximum number of frames to analyze is set in "yaw_number" parameter.

Parameter is disabled by default.

The purpose of utilizing "yaw_number" and "yaw_collection_mode" parameters is to increase the accuracy of best shot selection from a track.

4.3.2.8 mouth_occlusion_threshold

This parameter is used only for working with faces.

This parameter determines how much the mouth can be obscured in the frame.

I.e. when the value is equal to "0.5", 50% of the mouth can be occluded.

If mouth occlusion of a face in a frame exceeds the value of this threshold, the frame is considered as **not** appropriate for further analysis.

> The filtration is performed when the set value is "0.3" or higher. When the value is lover, the filtration is **disabled**.

4.3.2.9 min_body_size_threshold

The parameter sets the body detection size, less than which it will not be considered as the best shot. It is calculated as the square root of the product of the body detection height (in pixels) by its width (in pixels).

Example: `min_body_size_threshold = sqrt (64*128)= 90.5`

If the value is "0", then filtering of body detection by size will not be performed.

4.3.3 Sending to external system

In the "sending" section, the period during which the frames will be analyzed to select the best shot is determined, and all parameters associated with compiling a collection of the best shots are also defined.
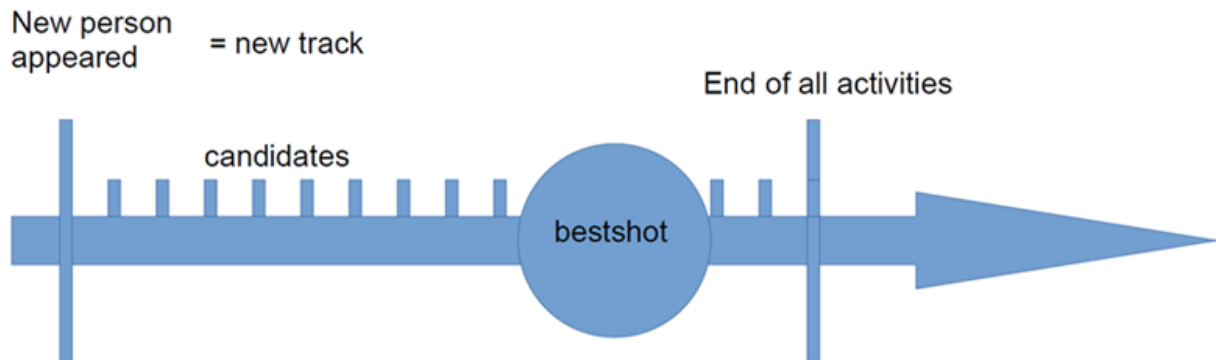
Figure 12: Best shots

4.3.3.1 Sending body data settings

Along with the initial frames of bodies, detections with the coordinates of the person's body can be sent so that the person's path can be tracked. Settings for sending body data are regulated in the "body" section.

Two parameters are available:

- "delete_track_after_sending" — Enable deletion of the best shots and detections after sending data. If the value is "false" (default), then the data will remain in memory.
- "send_only_full_set" — Enable sending data to an external system only if a certain number of best shots is collected (parameter "body_bestshots_to_send") and only if a certain number of detections with the coordinates of the human body is collected (parameter "minimal_body_-track_length_to_send" of the FaceStream settings).

4.3.3.2 Sending best shots settings

The settings for sending the best shots are regulated in the "bestshot_settings" section.

Following options are available:

- "type" — Mode for sending the best shots:
    - Only the best facial shots
    - Only the best body shots
    - Best shots of faces and bodies in one request
    - Best shots of faces and bodies in different requests

- "face_bestshots_to_send" and "body_bestshots_to_send" - Number of best shots that the user wants to get from a track or from a certain period of time on this track.

  Using this parameter involves creating a collection of the best shots of a track or a time period of a track specified in the "time_period_of_searching" parameter. This collection will be sent to the external system.

  Increasing the value increases the probability of correct object recognition, but affects the network load.

4.3.3.3 full_frame_settings

Important: Currently, face and body detection collaborative mode only sends raw face and body frames.

This parameter specifies the mode for sending source frames:

- Only source frames of faces
- Only source frames of bodies
- Source frames of faces and bodies in one request
- Source frames of faces and bodies in different requests

Important: Sending source frames must be enabled in the "send_source_frame" parameter in FaceStream settings.

4.3.3.4 time_period_of_searching

Interval in track after the end of which a best shot is sent to the server (period starts with the first detection – person appears in the frame). Lowering this parameter speeds up recognition but decreases precision.
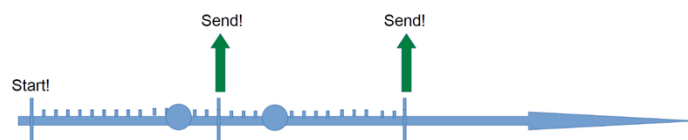
Figure 13: Sending period

The measurement type is set in the "type" parameter (see below). If the value equals "-1" (by default), analysis is conducted on all frames until the end of track. Once the track is over (person leaves the frame), best shot is sent to an external service.

4.3.3.5 silent_period

This parameter is used only for working with faces.

Interval between period. Once the analysis period is over, the system holds this silent_period before starting next period of frame analysis.
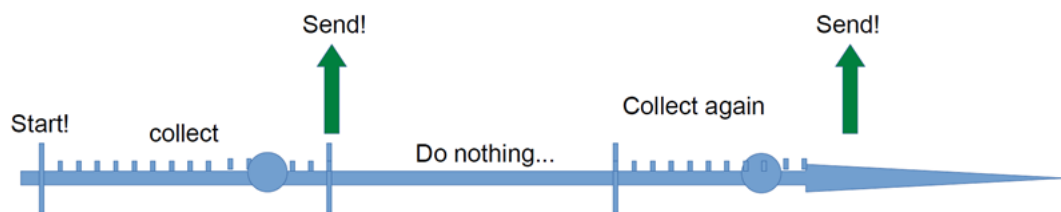
Figure 14: Silent period

The measurement type is set in the "type" parameter (see below). If the value equals "-1", system holds silent period indefinitely.
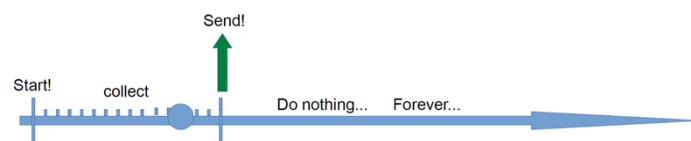
Figure 15: Endless waiting period

4.3.3.6 type

The parameter specifies the measurement type for the "silent_period" and "time_period_of_searing" parameters. It can take two values - "frames" or "sec".

4.3.4 Integration with external system settings

Frames sent by FaceStream can be processed by an external system. To do this you need to specify:

- Handler ID
- Address for saving source frames
- Authorization settings via token

4.3.4.1 frame_store

This parameter sets a URL for saving the source frames of faces or bodies in LUNA PLATFORM 5.

As the URL, you can specify either the address to the LUNA Image Store service bucket, or the address to the "/images" resource of the LUNA API service. When specifying the address to the "/images" resource, the source frame will be saved under the "image_id".

The "send_source_frame" parameter should be enabled for sending source frames.

Example of address to LUNA Image Store bucket:

```
"frame_store": "http://127.0.0.1:5020/1/buckets/<frames>/images"
```

Here:

- `127.0.0.1` - IP address where the LUNA Image Store service is deployed.
- `5020` - Default Image Store service port.
- `1` - API version of the LUNA Image Store service.
- `<frames>` - Name of the LUNA Image Store bucket where the source image of face or body should be saved. The bucket should be created in advance.

An example of the "source-images" bucket creation:

```
curl -X POST http://127.0.0.1:5020/1/buckets?bucket=source-images
```

Example of address to "/images" resource of LUNA API service:

```
"frame_store": "http://127.0.0.1:5000/6/images"
```

Here:

- `127.0.0.1` - IP address where the LUNA API service is deployed.
- `6` - API version of the LUNA API service.
- `5000` - Default port of the API service.

> See the LUNA PLATFORM 5 administrator's manual for more information about buckets and the "/images" resource.

4.3.4.2 authorization

In this section, either token or account_id are set to make requests to the LUNA API service.

The "event_handler" > "authorization" > "account_id" parameter must match the "account_id" parameter specified in the request. If the authorization field is not filled in, then the "account_id" specified when the stream was created will be used.

See the LUNA PLATFORM 5 administrator manual for details on LUNA PLATFORM authorization.

4.3.5 Healthcheck settings

The section is used only for the "tcp", "udp" and "videofile" types.

In this section, you can set the parameters for reconnecting to the stream when errors occur while the video is streamed.

4.3.5.1 max_error_count

The maximum number of errors when playing the stream.

The parameter works in conjunction with the "period" and "retry_delay" parameters. After receiving the first error, the wait specified in the "retry_delay" parameter is performed, and then the connection to the stream is retried. If during the time specified in the "period" parameter, the number of errors greater than or equal to the number specified in "max_error_count" was accumulated, then the processing of the stream will be terminated and its status will change to "failure".

For example, when it is unable to retrieve or decode a frame. Network problems or inaccessibility of a video can cause the errors.

4.3.5.2 period

The parameter represents the period during which the number of errors is calculated. The value is set in seconds.

The parameter works in conjunction with the "retry_delay" and "max_error_count" parameters. See the description of working with this parameter in the "max_error_count" section.

4.3.5.3 retry_delay

The parameter specifies the period after which the reconnection attempt is performed. The value is set in seconds.

The parameter works in conjunction with the "period" and "max_error_count" parameters. See the description of working with this parameter in the "max_error_count" section.

4.3.5.4 timeout

The parameter specifies the timeout in milliseconds for reading the encoded packet.

Using this parameter, it is possible to provide more flexible processing of the video stream, control the speed of reading video packets and prevent data buffer overflow.

4.4 Liveness settings

Liveness is not used for the "images" type.

Liveness is used only for working with faces.

Liveness is used to check whether a person in the frame is real and prevents fraud when printed photos or images on the phone are used to pass the Liveness check.

Liveness settings are set in the "data" > "analytics" > "liveness" section.

> It is recommended to use this functionality only after discussing it with the VisionLabs team.

4.4.1 General recommendations for Liveness usage

Liveness can be used at access control checkpoints only. This is a case when a person does not stay in front of the camera for more than ten seconds.

Liveness is used to minimize the risk of fraud when someone is trying to enter a secured area using a printed photo or a photo on a phone of someone who has the access rights.

Liveness returns a value, which defines the degree of the system certainty on whether the person in the frame is real. The value is in the range of 0 to 1.

4.4.1.1 Camera placement requirements

The following conditions must be met for Liveness check set up:

- Face should remain within a frame. The distance from left and right edges of the frame should be greater than or equal to the width of the face, the distance from the top and bottom edges of the frame should be greater than or equal to the height of the face;

- The frame should include the chest region;

- A camera should be located about waist height and should look upwards capturing the body and head;

- The frame should not include rectangular elements framing the face area from all four sides (such as doorways or windows).

An example of the correct camera location is given in the image below.
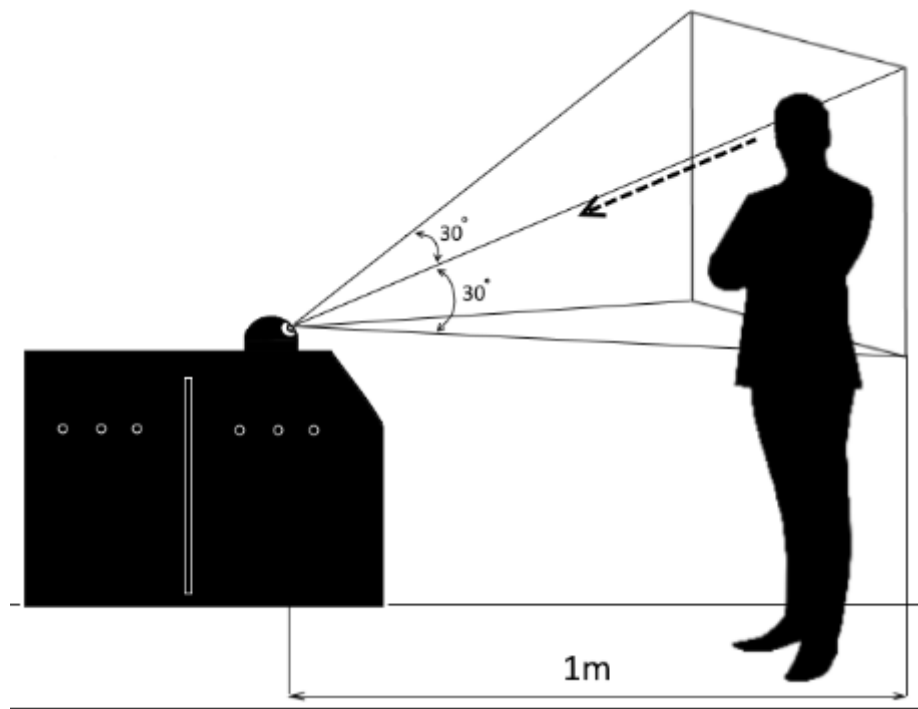
Figure 16: Proper camera placement for Liveness

FS starts collecting frames and selecting the best shot at a distance of 3-4 meters when a camera is placed properly.

Foreign objects and people who do not pass through the turnstile do not get into the camera view zone.

FS sends the best shot when a person is at a distance of 1 meter from the camera. At this distance, the face reaches the size required for sending.

An example of inappropriate camera placement is given in the image below.
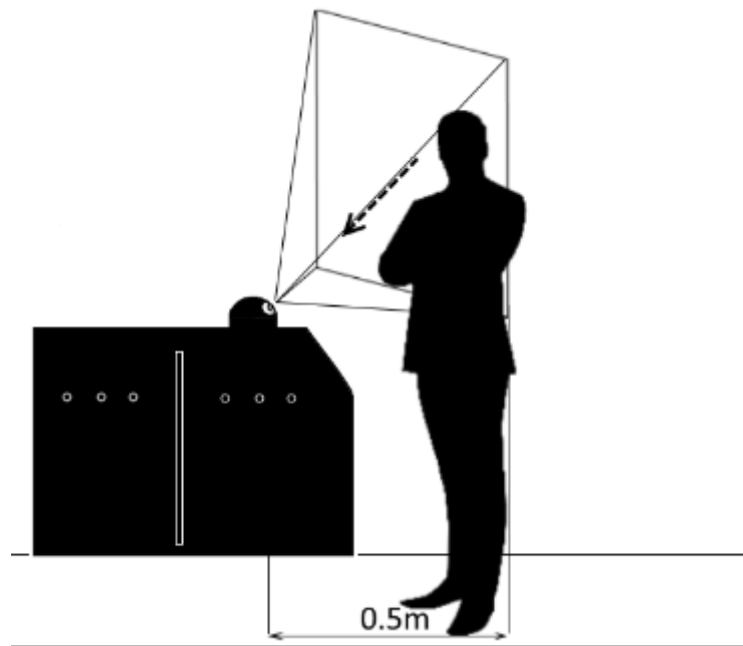
Figure 17: Inappropriate camera placement for Liveness

If the camera is not configured correctly:

- The person gets into the frame too late. FS does not have time to get the required number of frames for processing.
- The person looks upside-down at the camera. This degrades the quality of the frame for subsequent processing.
- The camera field of view covers the area outside the area of interest. This space may contain people or objects that interfere with the correct operation of the FS.

4.4.1.2 Recommendations for configuring FS

The recommended values for the "Liveness" section parameters are given below.

```
"use_mask_liveness_filtration": true,
"use_flying_faces_liveness_filtration": true,
"liveness_mode": 1,
"number_of_liveness_checks": 10,
"liveness_threshold": 0.8,
"livenesses_weights": [0.0, 0.25, 0.75],
"mask_backgrounds_count": 300
```

We do not recommend changing these settings.

The "best_shot_min_size" parameter should be set based on the fact that the person is at a distance of 3-4 meters from the turnstile.

The "best_shot_proper_size" parameter should be set based on the fact that the person is at a distance of 1 meter from the turnstile.

To control the selection of the right person, use the "droi" parameter. The rectangle is selected so that people who have the intention to approach this turnstile appear in the rectangle as early as possible. This is true for turnstiles located close to each other. People from neighboring queues can get into the view zone of the cameras of such turnstiles.

4.4.1.3 FAQ Liveness

Stream processing is slow when using Liveness

When the camera resolution is 1920 x 1080 and higher, Mask Liveness is working slowly.

To solve the problem, you should manually reduce the resolution in the camera to 720p. This will not affect the quality of recognition and the work of Liveness, because they work without loss of quality with faces that are approximately 100 pixels in size.

People cannot pass the Liveness check under the default FS settings

Possible causes:

- The default settings in the Liveness section have been changed.

  Do not change the settings in the Liveness section, except for the "liveness_threshold" setting.

  The value of the "liveness_threshold" parameter can be reduced, but it should not be lower than "0.6".

- Liveness is not applied to the target case.

  FS Liveness is not intended for authorization processes and cases of a long stay in front of the camera.

- Unacceptable objects fall into the camera's view zone.

  For example, if there is a screen broadcasting a video in the background, Liveness will not work.

- The camera is set to the wrong resolution.

  Check the camera resolution. See "Stream processing is slow when using Liveness".

- There is a delay in the transmission of frames.

  If the camera does not transmit frames in real-time, then the frames may arrive with a delay.

- The value "best_shot_min_size" is set incorrectly.

  If the "best_shot_min_size" parameter is too high, Liveness does not have time to accumulate the required number of different frames.

The Primary Track Policy is often used together with Liveness. See FaceStream activity diagram when using Liveness and Primary Track Policy.

4.4.2 use_mask_liveness_filtration

The parameter enables checking the presence of a real person in the frame based on backgrounds.

The check performance depends on the size of the video frames. If the processing speed decreases when the parameter is enabled, it is necessary to reduce the video resolution in the camera settings (e.g., up to 1280x720).

4.4.3 use_flying_faces_liveness_filtration

The parameter enables checking the presence of a real person in the frame based on the facial surrounding.

4.4.4 liveness_mode

This parameter enables to specify which frames from a track will undergo Liveness check. There are three options for selecting a frame:

- **0** — First N frames.
- **1** — Last N frames before the best shot sending (recommended value).
- **2** — All frames in a track.

N value is specified in the "number_of_liveness_checks" parameter.

4.4.5 number_of_liveness_checks

The parameter enables to specify the number of frames to check fo Liveness. The specified value is used in the "liveness_mode" parameter.

It is not recommended to set a value less than 10.

4.4.6 liveness_threshold

The "liveness_threshold" parameter value is used to define the presence of a real person in a frame. The system confirms that it is a real person in the frame, and not a photo, only if Liveness returned a value higher than the one specified in the parameter.

The recommended value is "0.8". It is not recommended to set a value lower than "0.6".

4.4.7 livenesses_weights

The parameter determines the involvement of each liveness check type (mask, and flying_faces) in the resulting estimation of the presence of a human face in the frame.

User must specify two values assigned to different types of liveness. Values are specified in decimals in the following order:

- "use_mask_liveness_filtration"
- "use_flying_faces_liveness_filtration"

Values are indicated in fractions of a unit. In the example below 0.25 - 25% on "mask_liveness" and 0.75 - 75% on "flying_faces_liveness". The ratio is always scaled based on the given numbers, regardless of whether they add up to one or which liveness methods are enabled.

```
"livenesses_weights": [0.25, 0.75]
```

4.4.8 mask_backgrounds_count

The number of background frames that are used for the corresponding checks.

> Do not change this parameter.

4.5 Primary Track policy settings

Primary Track policy is used only for working with faces.

Primary Track policy is not used for the "images" type. Important: Primary Track policy is currently not supported for face and body detection collaborative mode.

This section is designed to work with Access Control Systems (ACS, turnstiles at the entrances to banks/office buildings) to simplify the control and the introduction of facial recognition technology at the entrance to a protected area.

The Primary Track policy settings are set in the "data" > "analytics" > "primary_track_policy" section.

Liveness is often used together with the Primary Track policy. See FaceStream activity diagram when using Liveness and Primary Track Policy.

4.5.1 use_primary_track_policy

If the parameter value is "true", the primary track implementation mode is enabled.

Out of all detections, one of the biggest sizes is selected and its track becomes the primary one. Further analysis is conducted on this track. The best shot from this track is then sent to the server.

All other tracks are processed in regular mode. However, the best shot is sent only from the primary track.

As soon as another face reaches a larger size than the face from the primary track, this face track becomes primary and the processing is performed for it.

While using this parameter at the access control checkpoint, only the best shots of the person who is the closest to the turnstiles will be sent to the server (here the biggest detection size condition is held)

4.5.2 best_shot_min_size

The parameter is used when "use_primary_track_policy" parameter is enabled.

The "best_shot_min_size" parameter sets the minimal height of detection in pixels at which the analysis of frames and best shot definition begins.

4.5.3 best_shot_proper_size

The parameter is used when "use_primary_track_policy" parameter is enabled.

The "best_shot_proper_size" parameter sets the height of detection in pixels for Primary track policy. When the detection size reaches the specified value, FaceStream determines the best shots before the end of the track, and then sends them to the LP.

5 Settings in LUNA Configurator service

The LUNA Configurator service stores general settings for:

- FaceStream
- TrackEngine
- LUNA Streams

See "Use FaceStream with LUNA Configurator" for details on how the LUNA Configurator service interacts with FaceStream.

5.1 FaceStream settings

Settings configuration is performed by editing parameters in the "FACE_STREAM_CONFIG" section in the Configurator service (see "Use FaceStream with Configurator");

> You can also setting up FaceStream by editing the configuration file "fs3Config.conf" in the mode of working without the Configurator service (see "Use FaceStream with configuration files").

Below are the settings divided into logical blocks depending on the main functions performed by the block.

5.1.1 logging

Settings section of the application logging process. It is responsible for message output on errors and/or current state of the application.

5.1.1.1 severity

Severity parameter defines which information the user receives in logs. There are three information filter options:

0 — Outputs all the information **1** — Outputs system warnings only **2** — Outputs errors only

5.1.1.2 tags

Tags enable you to get information about the processing of frames and errors that occur only for FaceStream processes of interest.

This parameter enables you to list tags that are associated with logging of relevant information. If a corresponding tag is not specified, the information is not logged.

Information on specified tags is displayed according to the *severity* parameter value.

Logs text includes the corresponding tag. It can be used for logs filtration.

Errors are always written in the log. They do not require additional tags.

Tag	Description
streams	Information about LUNA Streams operation.
common	General information.
ffmpeg	Information about FFMPEG library operation.
liveness	Information about the presence of a living person in the frame ("liveness" section): is there enough information for liveness check, and does the frame pass the liveness check.
primary-track	Information about the primary track ("primary_track_policy" section): the frame passed the specified thresholds and what track is selected as primary.
bestshot	Information about the best shot selection: best shot occurrence, its change and sending to external service.
angles	Information about filtration by head pose.
ags	Information corresponding to the frames quality. The information is used for further processing using LUNA PLATFORM.
mouth-occlusion	Information about mouth occlusion is recorded to the log file.
statistics	Information about performance, the number of frames processed per second, and the number of frames skipped.
http_-api	Information about API requests sent to FaceStream in server mode and received responses.
client	Information about sending messages to LUNA PLATFORM and the responses received.
json	Information about processing parameters from configuration files and the Configurator service.
debug	Debug information. It is recommended to use this tag when debugging only and not during FS operation. It provides a large amount of debugging information.
estimator	Information about the statistics of completed estimations.

Tag "estimator"

When the "estimator" tag is enabled, FaceStream logs return information on the size of the butches with which the estimators were called. Example of log content with the "estimator" tag enabled:

```
[I0609 15:48:03.779697    65 EstimatorStatistic.cpp:85] [estimator] Batch
    statistic for estimator HeadPoseEstimator
```

```
Total calls: 1311 total time: 191 ms.
sz  cnt
1   1311 (100.00%)
2   0 (0.00%)
3   0 (0.00%)
...
16  0 (0.00%)
```

In this case, the statistics show the following:

- Total of 1311 calls were made to the "HeadPoseEstimator" estimator, which took 191 milliseconds.
- All calls (1311) were processed individually, each in its own batch of size 1.
- Batch sizes from 2 to 16 were not used.

If the estimators were not called, then no information will be output in the FaceStream logs.

5.1.1.3 mode

Mode parameter sets the logging mode of the application: file or console. There are three modes available:

- "l2c" – Output information to console only
- "l2f" – Output information to file only
- "l2b" – Output to both, console and file

In the FaceStream mode of working with configuration files, you can configure the directory to save logs when information is output to a file using the `--log-dir` launching parameter.

5.1.2 sending

This section is used to send portraits in the form of HTTP-requests from FaceStream to external services.

5.1.2.1 request_type

This parameter is used only for working with faces.

This parameter defines a type of query that is used to send portraits to external services. There are 2 supported types (to work with different versions of LUNA):

- "jpeg" is used to send normalized images to VisionLabs LUNA PLATFORM.
- "json" may be used to send portraits to custom user services for further image processing.

> For a detailed description of the requests, see the table below.

Format	Request type	Authorization headers	Body
JSON	PUT	Authorization: Basic, login/password(Base64)	Media type: `application/json`; frame – the original frame in Base64 (if send_source_frame option is on); data – a portrait in Base64; identification – Cid parameter value. JSON example: `{frame":"","data":"image_in_base_64","identification":"camera_1"}`
JPEG	POST	Authorization: Basic, login/password(Base64) or X-Auth-Token: 11c59254-e83f-41a3-b0eb-28fae998f271(UUID4)	Media type: `image/jpeg`

5.1.2.2 request_timeout_ms

This parameter sets the timeout for sending and receiving events from/to the LUNA PLATFORM in milliseconds.

The default value is 30 seconds (30,000 milliseconds).

5.1.2.3 portrait_type

This parameter is used only for working with faces.

This parameter defines the format of a detected face to send to an external service. Possible values:

- "warp" — Use a normalized image.
- "gost" — Do not use the transformation, cut the detected area from the source frame, considering indentation.
- "crop" – Use an area expanded and cropped around the detected object.

Warp format

Properties of the normalized image (warp):

- Size of 250x250 pixels.
- Face is centered.

- Face should be aligned in the way that, if you draw an imaginary line connecting the corners of the eyes, it is close to horizontal.

Such image format when working with LUNA PLATFORM offers the following advantages:

- Constant minimal predictable amount of data for network transfer;
- Face detecting phases in LUNA PLATFORM automatically turn off for such images, which leads to the significant reduction of interaction time.

Crop format

A Crop image is a fragment of an image obtained by expanding and cropping the area around the detected object. The expansion of the area is determined by the new "sending" > "crop_factor" parameter, which increases the dimensions of the original rectangle containing the object. The maximum size of the cropped image is controlled by the new "sending" > "max_crop_size" parameter, which limits the largest side of the area. If the size of the fragment exceeds the specified parameter, the fragment will be scaled proportionally along both coordinates to the desired size. A Crop image allows capturing more context and detail around the object compared to a sample.

This allows for more flexible image processing. Sending Crop images has several key advantages:

- Along with the Crop image, the coordinates of the face on both the Crop image and the source image are transmitted. This avoids the need to use redetetor, ensuring the presence of the face in the image and saving computational resources.
- A Crop image contains more context and details around the face, allowing for the estimation of various parameters that cannot be estimated on a biometric. For example, Deepfake estimation, Liveness estimation, etc., can be performed with greater accuracy.
- External services and face recognition algorithms often work better with Crop images than with samples. This is because Crop images contain more information and context, improving the accuracy of recognition and analysis.

5.1.2.4 crop_factor

The factor to increase the size of the original rectangle containing the object.

This parameter determines how much wider and taller the crop area will be compared to the original size. For example, with a "crop_factor" of 2.2, the crop area will be enlarged to 2.2 times its original size. This allows you to capture more context around the detected object.

5.1.2.5 max_crop_size

Maximum size of cropped image.

This parameter limits the largest side of the crop area to no larger than a specified value. If the cropped image exceeds "max_crop_size", it will be scaled to a valid size while maintaining aspect ratio. This ensures that the cropped images are not too large and remain within the specified dimensions.

5.1.2.6 send_source_frame

This parameter enables to send a full source frame where the face was detected.

The source frame can be saved both in the bucket of the LUNA Image Store service and in the "/images" resource of the LUNA API service. The method of saving the source frame is set by specifying the appropriate address in the "frame_store" parameter.

For the collection of best shots ("face_bestshots_to_send" > 1), only one source frame is sent, which is determined by FaceStream to be the best of all best shots. For example, if the setting value "number_of_-bestshots_to_send" is 3, then three best shots will be sent to LP and only one source frame, which will be automatically selected from the three best shots.

If the source frame is supposed to be saved to the LUNA Image Store service bucket, then the frame will be sent to the LP some time before the best shot is sent. First, it is saved in the LUNA Events database and assigned a unique ID, which is stored in the "image_origin" field of the "face_detect_result"/"body_-detect_result" table. Next, the best shot is sent to the LP and an event is generated, where the "image_-origin" field indicates the ID of the source frame taken from the LUNA Events database.

Example from the response body of the event about successful saving to the bucket:

```
"events": [
    detections": [
        "image_origin": "/1/buckets/source-images/images/83c383d8-8af4-406c
            -8ff8-76eb389e61c8"
```

If the source frame is supposed to be saved to the "/images" resource, then the frame will be sent along with the best shot using the "multipart/form-data" scheme of the request to generate the LUNA PLATFORM event.

Example from the response body of the event about successful saving to the "/images" resource of the LUNA API service:

```
"events": [
    detections": [
        "image_origin": "/6/images/19c5c40a-04e4-4f2b-811c-0f3bb0003359"
```

If the "image_origin" field contains an address like "/6/samples/faces/c93f39f1-809c-4583-b386-c166cecac333", it signifies the location for storing the best shot rather than the source image.

5.1.2.7 size_source_frame

Parameter changes the width of the source frame to the specified value. Valid range is [0…1024]. The value "0" means that the width will not change. The fractional part of the input is affected.

FaceStream sends the "X-Luna-Meta-rescale" header to the LUNA PLATFORM, which contains the scaling factor of the source image in the form of a floating-point number.

The scaling factor enables calculating the real coordinates of the scaled source image. To do this, divide the size of the obtained image by the obtained scaling factor.

Note. The real size is calculated with an absolute error of less than 1 / (2 * scaling factor).

For example, if the obtained image has a size of 200x113, and the scaling factor is 0.104399, then the real size of the source image is 1916x1082 (200/0.104399=1916, 113/0.104399=1082).

The scaling factor can be found in LUNA PLATFORM:

- In the response header to the "get image" request to the LUNA API service. The request must be made with the "with_meta = 1" query parameter.
- In the "image_id.meta.json" file located next to the source image in the LUNA Image Store bucket of source images.

The "X-Luna-Meta-rescale" header is only sent if sending the source frame is enabled (parameter "send_-source_frame = 1"). The LUNA PLATFORM handler should also have a policy of saving the source image.

If scaling is not set for the source frame (parameter "size_source_frame = 0"), then the value of the "X-Luna-Meta-rescale" response header will be equal to 1.

5.1.2.8 detection_path_length

This parameter is used only for working with bodies.

Parameter sets the maximum number of detections for the "minimal_body_track_length_to_send" parameter. Values from 1 to 100 inclusive are available.

> The maximum number of detections cannot exceed 100. If there are more than 100 detections, then the FaceStream algorithm will remove unnecessary detections with some step.

5.1.2.9 minimal_body_track_length_to_send

This parameter is used only for working with bodies.

This parameter enables you to send a specified number of detections with the coordinates of the human body — x, y, width and height along with the best shots (see the "save event" request in the LUNA PLATFORM OpenAPI document). According to the sets of detections of the human body, it is possible to determine its path.

The detection with the best shot is mandatory to be sent.
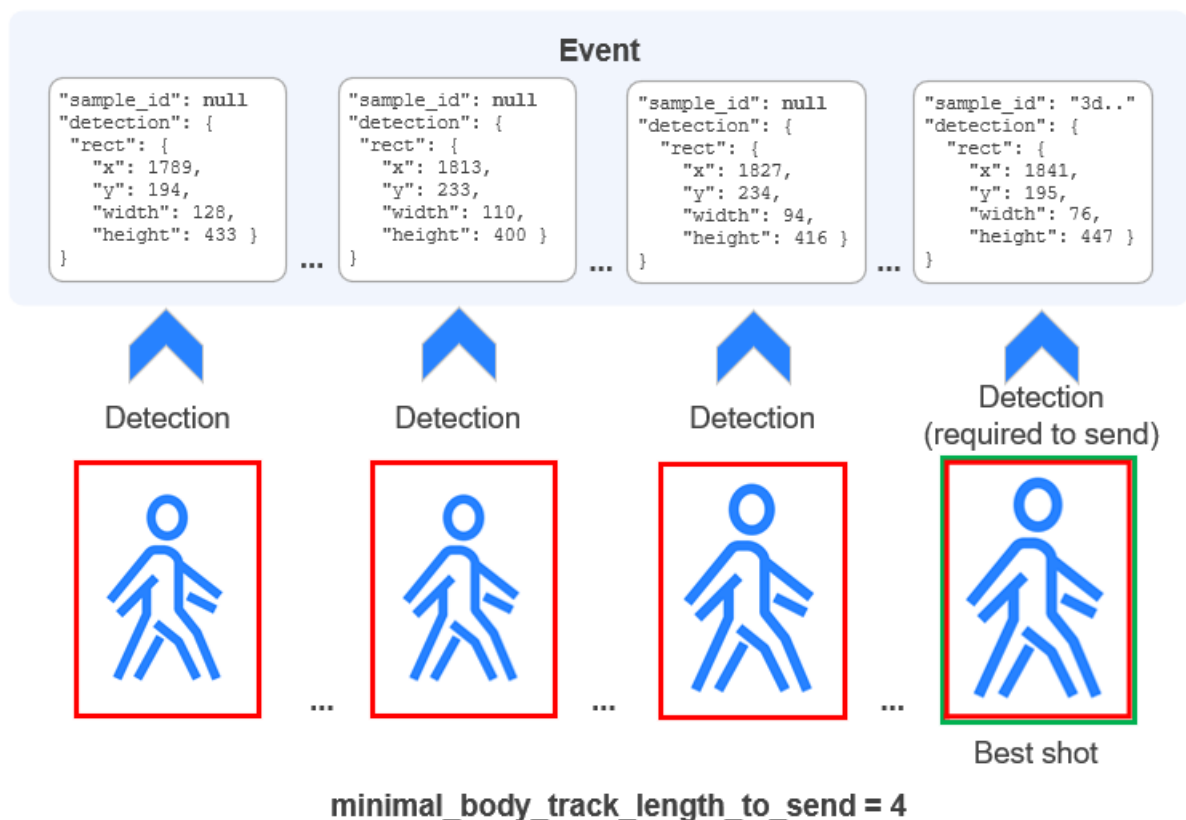
Figure 18: Sending detections

If the number of detections is less than the specified value, then these detections will not be sent. For example, if the number of detections is "3" and the value of "minimal_body_track_length_to_send" is "4", then no detections will be sent and the following message will be displayed in the FaceStream logs:

```
Track is too short and will not be sent. Length 3 < 4
```

If the parameter value is "0", then the number of detections will be equal to the number of best shots.

5.1.2.10 async_requests

The parameter specifies whether to execute requests asynchronously or synchronously in LUNA PLATFORM.

By default, the asynchronous mode is set, in which all requests to the LUNA PLATFORM are performed in parallel.

5.1.2.11 aggregate_attr_requests

This parameter enables the best shots aggregation to receive a single descriptor in LUNA PLATFORM.

Aggregation is performed if there is more than one best shot sent. The number of frames to send is set by the "face_bestshots_to_send" or "body_bestshots_to_send" parameter.

The accuracy of face and body recognition is higher when using an aggregated descriptor.

5.1.2.12 jpeg_quality_level

JPEG quality for source frames sending:

- "best" — Compression is not performed.
- "good" — 75% of source quality.
- "normal" — 50% of source quality.
- "average" — 25% of source quality.
- "bad" — 10% of source quality.

The "best" quality is set by default.

> High quality images sending can affect the frames processing speed.

5.1.3 lunaplatform

This section describes the settings for connecting to LUNA PLATFORM (API service) for subsequent processing of data sent by FaceStream.

5.1.3.1 origin

The address and port of the server where the LUNA API service is running.

5.1.3.2 api_version

The parameter specifies the API version of the LUNA API service. At the moment, the API version "6" is supported.

5.1.4 lunastreams

This section describes how to send ready-made images as HTTP requests from FaceStream to the LUNA Streams service.

See "Interaction of FaceStream with LUNA Streams" section for details on how LUNA Streams works with FaceStream.

5.1.4.1 origin

The address and port of the server where the LUNA Streams service is running.

5.1.4.2 api_version

The parameter specifies the version of the LUNA Streams service API. Currently, two API versions are supported - "1" and "2".

The default version is "1".

> The current version of the API can always be found in the LUNA Streams service documentation.

5.1.4.3 max_number_streams

The parameter sets the upper bound on the number of streams. The value must be greater than 0.

5.1.4.4 request_stream_period

The parameter sets the time period between requests to receive new streams from LUNA Streams in the range from 0.1 to 3600 seconds.

The default value is 1 second.

5.1.4.5 send_feedback_period

The parameter sets the time period between sending reports on processed streams to LUNA Streams in the range from 1.0 to 3600 seconds.

The default value is 5 seconds.

The value of this parameter should not exceed the value of the "STREAM_STATUS_OBSOLETING_PERIOD" parameter, set in the LUNA Streams service settings.

5.1.4.6 max_feedback_delay

The parameter sets the maximum report sending delay in the range from 1.0 to 3600 seconds. If the report has not been sent within the given time, then FaceStream will stop processing the current streams.

The default value is 10 seconds.

The value of this parameter should not be less than the value of the parameter "send_feedback_period" and should not exceed the value of the parameter "STREAM_STATUS_OBSOLETING_PERIOD", set in the LUNA Streams service settings.

5.1.5 performance

5.1.5.1 stream_images_buffer_max_size

The parameter specifies the maximum size of buffer with images for a single stream.

When you increase the parameter value, the FaceStream performance increases. The higher is the value, the more memory is required.

We recommend setting this parameter to 40 when working with GPU, if there is enough GPU memory.

5.1.5.2 enable_gpu_processing

This parameter enables you to utilize GPU instead of CPU for calculations.

GPU enables you to speed up calculations, but it increases the consumption of RAM.

> GPU calculations are supported for FaceDetV3 only. See "defaultDetectorType" parameter in the FaceEngine configuration ("faceengine.conf").

5.1.5.3 convert_full_frame

If this parameter is enabled, the frame is immediately converted to an RGB image of the required size after decoding. This results in a better image quality but reduces the speed of frames processing.

If this parameter is disabled, the image is scaled according to the settings in the TrackEngine configuration (standard behavior for releases 3.2.4 and earlier).

This parameter is similar to frame_processing_mode parameter, but it is set for all FaceStream instances at once.

5.1.6 fps_floor

This parameter is intended to control the number of frames analyzed in the video stream, which enables you to reduce the load on the system by skipping some frames. This can be useful when you need to reduce the requirements for computing resources while maintaining acceptable analysis accuracy.

The number of skipped frames is calculated as the difference between the frame rate of the video stream and the "fps_floor" value. The skipped frames are distributed evenly over each second.

If the "fps_floor" value is zero or negative, FaceStream will ignore the setting, and the number of frames processed will remain unchanged.

For example, with "fps_floor" = 15 for a video stream with a frequency of 25 fps, FaceStream will skip 10 frames per second (25 - 15). These 10 frames will be evenly distributed over time to maintain the integrity of video playback and avoid sharp jumps.

5.1.7 monitoring

Monitoring settings are set in this section.

5.1.7.1 storage_type

This parameter enables to select the type of data storage for monitoring. For now, only the Influx database can be used.

5.1.7.2 send_data

This parameter enables sending monitoring. Disabled by default.

5.1.7.3 organization

This parameter specifies the InfluxDB 2.x workspace.

The workspace is created by passing the corresponding argument at the start of the InfluxDB container.

5.1.7.4 bucket

This parameter sets the InfluxDB 2.x bucket.

The bucket is created by passing the corresponding argument at the start of the InfluxDB container. It can be also created using the Influx UI.

5.1.7.5 token

This parameter specifies the InfluxDB 2.x authentication token.

The token is created by passing the corresponding argument at the start of the InfluxDB container.

5.1.7.6 origin

This parameter specifies the IP address and port of the server with InfluxDB 2.x.

The default address is "127.0.0.1". Such an address means that InfluxDB 2.x, located on a server with LUNA Configurator, will be used. If InfluxDB 2.x is located on a different server, then in this parameter you need to specify the correct IP address of InfluxDB 2.x.

```
"origin": "http://127.0.0.1:8086/"
```

5.1.7.7 flushing_period

This parameter sets frequency of sending monitoring data to InfluxDB (in seconds).

The default is 1 second.

5.2 TrackEngine settings

This section describes the parameters of the TrackEngine configuration file that are used to configure FaceStream.

Settings configuration is performed by editing parameters in the "TRACK_ENGINE_CONFIG" configuration in the Configurator service (see "Use FaceStream with Configurator");

> You can also setting up TrackEngine by editing the configuration file "trackengine.conf" in the mode of operation without the Configurator service (see "Use FaceStream with configuration files").

5.2.1 use-face-detector and use-body-detector

These parameters allow you to enable or disable detection of faces and bodies. You can enable simultaneous detection.

> See "Face and body detection collaborative mode" for details.

> It should be remembered that in order to successfully change the detection, it is necessary to set the appropriate FaceStream settings and settings of stream sources.

```
<!-- use-face-detector: Flag to use or not face detection -->
<param name="use-face-detector" type="Value::Int1" x="1" />

<!-- use-body-detector: Flag to use or not body detection -->
<param name="use-body-detector" type="Value::Int1" x="0" />
```

5.2.2 detector-step

The "detector-step" parameter enables you to specify the number of frames on which face redetection will be performed in the specified area before face detection is performed. Redetection requires fewer resources, but the face may be lost if you set a large number of frames for redetection.

```
<!-- detector-step: The count of frames between frames with full detection,
    [0 .. 30] ('7' by default). -->
<param name="detector-step" type="Value::Int1" x="7" />
```

5.2.3 detector-scaling

The "detector-scaling" parameter enables you to scale the frame before processing. The frame size for scaling is set in the parameter "scale-result-size".

For more information about scaling, see "Frame scaling".

```
<!-- detector-scaling: Scale frame before detection for performance reasons,
    [0, 1] ('0' by default). -->
<param name="detector-scaling" type="Value::Int1" x="0" />
```

5.2.4 scale-result-size

The "scale-result-size" parameter sets the maximum frame size after scaling the largest side of the frame.

To use this parameter, you must enable the "detector-scaling" parameter.

If the source frame had a size of 1920x1080 and the value of "scale-result-size" is equal to 640, then FaceStream will process the frame of 640x360 size.

> If the frame was cut out using the "roi" parameter, the scaling will be applied to this cropped frame. In this case, you should specify the "scale-result-size" parameter value according to the greater ROI side.

You should gradually scale the frame and check whether face or body detection occurs on the frame, to select the optimal "scale-result-size" value. You should set the minimum image size at which all objects in the area of interest are detected.

Further extending our example, images below depict a video frame without resize (at original 1920x1080 resolution) and after resize to 960x640 with face detections visualized as bounding boxes.

Six faces can be detected when the source image resolution is 1920x1080.

Figure 19: Detections in image 1920X1080

Three faces are detected after the image is scaled to the 960x640 resolution. The faces in the background are smaller in size and are of poor quality.
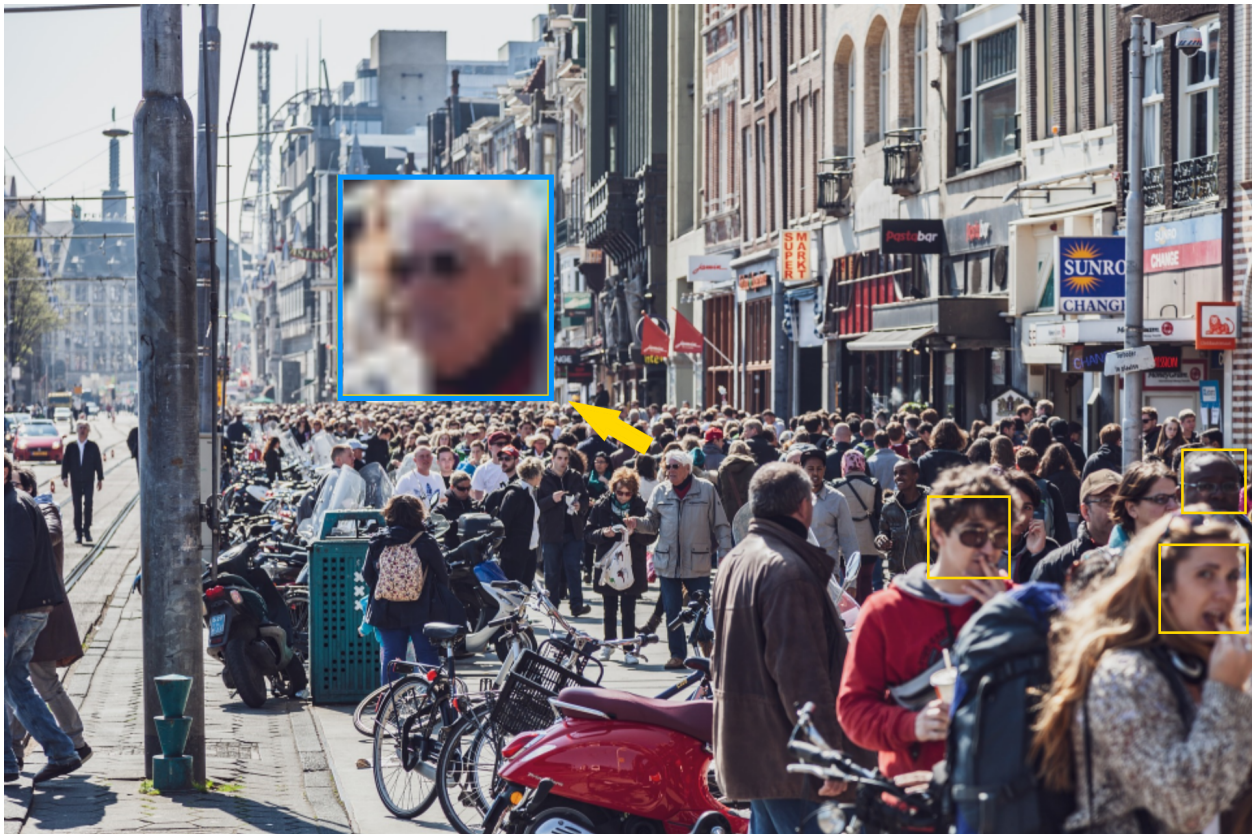
Figure 20: Detections in image 960X640

The smaller the frame resolution, the less resources are consumed.

> When working with bodies, this parameter works the same way.

```
<!-- scale-result-size: If scaling is enable, frame will be scaled to this
    size in pixels (by the max dimension — width or height).
Upper scaling is not possible. ('640 by default') -->
<param name="scale-result-size" type="Value::Int1" x="640" />
```

5.2.5 skip-frames

This parameter is used only for working with faces.

The "skip-frames" parameter sets the number of frames during which the system will wait for the object to reappear in the area where it disappeared.

Note that very high values may lead to performance degradation.

```
<!-- skip-frames: If track wasn't updated by detect/redetect for this number
     of frames, then track is finished ('36' by default). -->
<!-- note: very high values may lead to performance degradation. Parameter
     doesn't affect on human tracking. -->
<param name="skip-frames" type="Value::Int1" x="36" />
```

5.2.6 frg-subtractor

When the "frg-subtractor" parameter is enabled, motion in the frame is considered. The following face and body detection will be performed in the area with motion, not in the entire frame.

> The areas with motion are determined after the frame is scaled.

When the "frg-subtractor" is enabled, the performance of FaceStream is increased.

```
<!-- frg-subtractor: Use foreground subtractor for filter of frames, [0, 1]
     ('1' by default). -->
<param name="frg-subtractor" type="Value::Int1" x="1" />
```

5.2.7 frg-regions-alignment

The "frg-regions-alignment" parameter enables you to set the alignment for the area with motion.

```
<!-- frg-regions-alignment: frg regions alignment. Useful for detector
     better batching utilization. -->
<!-- 0 or negative values mean using non aligned regions, (0 by default).
     -->
<param name="frg-regions-alignment" type="Value::Int1" x="0" />
```

5.2.8 frg-regions-square-alignment

When the "frg-regions-square-alignment" parameter is enabled, the width and height of the area with motion will always be equal.

```
<!-- align frg regions to rect with equal sides (max side choosen). See frg-
     regions-alignment, [0, 1] ('1' by default). -->
<param name="frg-regions-square-alignment" type="Value::Int1" x="1" />
```

5.2.9 batched-processing

The "batched-processing" parameter enables batch processing of frames.

When working with several video cameras, a frame is collected from each frame. Then the batch of frames is processed.

When the parameter is disabled, the frames are processed one by one.

> When using batch processing mode, the delay before processing increases, but the processing itself is faster.

> It is recommended to enable the parameter both when using the GPU and when using the CPU.

```xml
<!-- batched-processing: Process streams frames in batch or separately, [0,
    1] ('1' by default). -->
<param name="batched-processing" type="Value::Int1" x="1" />
```

5.2.10 min-frames-batch-size

The "min-frames-batch-size" parameter sets the minimal number of frames collected from all the cameras before processing.

> It is recommended to set the "min-frames-batch-size" parameter value equal to the number of streams when using the GPU.

> It is recommended to set the "min-frames-batch-size" parameter value equal to "2" when using the CPU.

```xml
<!-- min-frames-batch-size: stream frames min batch size value to process, (
    '0' by default). -->
<!-- higher values lead to higher processing latency but increase throughput
    and device utilization. -->
<!-- zero/negative values disable this feature, so any stream frames will be
    processed if they are available -->
<!-- note: this parameter should be regulated with 'max-frames-batch-gather-
    timeout' (see below) -->
<param name="min-frames-batch-size" type="Value::Int1" x="0" />
```

5.2.11 max-frames-batch-gather-timeout

The "max-frames-batch-gather-timeout" parameter specifies the time between processing of the batches.

If a single frame is processed within the specified time and there is an additional time margin, FaceStream waits for additional frames to increase GPU utilization.

If the "max-frames-batch-gather-timeout" parameter is set to "20", this time is used to process the previous batch and collect a new one. After 20 seconds, the processing begins even if the number of frames equal to "min-frames-batch-size" was not collected. Processing of the next batch cannot begin before the processing of the previous one is finished.

There is no timeout for collecting frames to the batch if the parameter is set to "0" and "min-frames-batch-size" is ignored.

> It is recommended to set the "max-frames-batch-gather-timeout" parameter value equal to "0" both when using the GPU and when using the CPU.

```
<!-- max-frames-batch-gather-timeout: max available timeout to gather next
    stream frames batch (see 'min-frames-batch-size') from last processing
    begin time point (measured in ms), ('-1' by default). -->
<!-- negative values disable this feature (no timeout, so only stream frames
     batches with size no less than 'min-frames-batch-size' value will be
    processed) -->
<!-- note: this parameter is complementary to 'min-frames-batch-size' and
    controls min average fps of stream frames batches processing -->
<param name="max-frames-batch-gather-timeout" type="Value::Int1" x="-1" />
```

5.3 LUNA Streams settings

This section describes the parameters of the LUNA Streams service that are set in the LUNA Configurator service.

5.3.1 LUNA_STREAMS_DB

5.3.1.1 db_type

The parameter sets the type of database used. The following types are available:

- "postgres" — PostgreSQL database type
- "oracle" — Oracle database type

Setting format: `string`.

Default value: `postgres`.

5.3.1.2 db_host

The parameter sets the IP address of the server with the LUNA Streams database.

The default address is "127.0.0.1". This address means that the LUNA Streams database located on the server with LUNA Configurator will be used. If the database is located on another server, then in this parameter you should specify the correct IP address of the server with the database.

Setting format: `string`.

Default value: `127.0.0.1`.

5.3.1.3 db_port

The parameter sets LUNA Streams database listener port.

The default port for "postgres" type is "5432".

The default port for "oracle" type is "1521".

Setting format: `string`.

Default value: `5432`.

5.3.1.4 db_user

The parameter sets the database username.

Setting format: `string`.

Default value: `luna`.

5.3.1.5 db_password

The parameter sets the database password.

Setting format: `string`.

Default value: `luna`.

5.3.1.6 db_name

The parameter sets the database name for "postgres" and the name of the SID for "oracle" type to connect to.

Setting format: `string`.

Default value: `luna_streams`.

5.3.1.7 db_settings > connection_pool_size

The parameter sets the database connection pool size.

Setting format: `string`.

Default value: `10`.

5.3.1.8 dsn

The parameter sets the DSN connection string for connecting to the database.

Setting format: `string`.

> DSN is a connection string that identifies and points to the data source (database) to which you want to establish a connection.

Settings such as multiple hosts, authentication data, port, and other parameters can be set in the DSN string.

> The settings depend on the type of database. Multiple hosts are supported only with PostgreSQL.

> By default, the "dsn" parameter is not displayed in the "Settings" tab in the Configurator. You can check the list of all available parameters for a section on the "Limitations" tab.

Below is an example of specifying the "dsn" parameter:

```
{
"dsn": "luna:luna@postgres01:5432,postgres02:5432/luna_streams?some_option=
    some_value"
"db_settings": {
    "connection_pool_size": 5
    }
```

```
    }
```

Here:

- "luna:luna" — Username and password for connecting to PostgreSQL.
- "@postgres01:5432,postgres02:5432" — Comma-separated list of hosts and ports. This means that the service will try to connect to the first host ("postgres01") on port 5432. If this fails, it will try to connect to the second host ("postgres02") also on port 5432.
- "/luna_streams" — Database of the Streams service.
- "?some_option=some_value" — Optional parameters for connection.

If necessary, you can combine the DSN string and the classic settings, but the DSN string is a higher priority. You can partially fill in the DSN string (for example, "postgres 01,postgres02/luna_streams"), and then the missing parameters will be filled in from the values of the parameters "db_host", "db_port", "db_name", "db_user" and "db_password".

At startup, the service will create a pool of connections to one of the available DSN hosts. In case of problems with establishing a connection after several unsuccessful attempts, the service will again try to set up a connection pool to any of the available DSN hosts.

5.3.2 LUNA_LICENSES_ADDRESS

This section sets the settings for connecting to the LUNA Licenses service.

5.3.2.1 origin

The parameter sets the protocol, IP address and port of the LUNA Licenses service.

The IP address "127.0.0.1" means that the LUNA Licenses service located on the server with LUNA Configurator will be used. If the service is located on another server, then in this parameter you need to specify the correct IP address of the server running the LUNA Licenses service.

Setting format: `string`.

Default value: `http://127.0.0.1:5120`.

5.3.2.2 api_version

This parameter sets the API version of the LUNA Licenses service. The available API version is "1".

Setting format: `integer`.

Default value: `1`.

5.3.3 LUNA_STREAMS_LOGGER

This section sets the logging settings for the LUNA Streams service.

5.3.3.1 log_level

The parameter sets the level of debug printing, by priority: "ERROR", "WARNING", "INFO", "DEBUG".

Setting format: `string`.

Default value: `INFO`.

5.3.3.2 log_time

The parameter sets the time format used in log entries. The following values are available:

- "LOCAL" — Displays the local time of the system on which the logs are being recorded.
- "UTC" — Displays Coordinated Universal Time, which is a time standard and is not affected by local time zones or seasonal time changes.

Setting format: `string`.

Default value: `LOCAL`.

5.3.3.3 log_to_stdout

The parameter enables you to send logs to standard output (stdout).

Setting format: `boolean`.

Default value: `true`

5.3.3.4 log_to_file

The parameter enables you to save logs to a file. The directory with log files is specified in the "folder_-with_logs" parameter.

Setting format: `boolean`.

Default value: `false`.

5.3.3.5 folder_with_logs

The parameter sets the directory in which logs are stored. The relative path starts from the application directory.

To use this parameter, you must enable the "log_to_file" parameter.

Setting format: `string`.

Default value: `./`

Example:

```
"folder_with_logs": "/srv/logs"
```

5.3.3.6 max_log_file_size

The parameter sets the maximum size of the log file in MB before performing its rotation (0 — do not use rotation).

To use this parameter, you must enable the "log_to_file" parameter.

Setting format: `integer`.

Default value: `1024`

5.3.3.7 multiline_stack_trace

The parameter enables multi-line stack tracing in logs. When the parameter is enabled, information about the call stack is recorded in the logs so that each stack frame is placed on a separate line, which improves readability. If the parameter is disabled, information about the call stack is recorded on one line, which may make logs less convenient for analysis.

Setting format: `boolean`.

Default value: `true`.

5.3.3.8 format

The parameter defines the format of the output logs. The following values are available:

- "default" — Standard output format of the LUNA PLATFORM logs.
- "json" — Output of logs in json format.
- "ecs" — Output of logs in ECS format (Elastic Common Schema).

When using the "ecs" value, the following fields will be used:

- "http.response.status_code" — Contains the HTTP response status code (e.g., 200, 404, 500, etc.).
- "http.response.execution_time" — Contains information about the time taken to execute the request and receive the response.
- "http.request.method" — Contains the HTTP request method (GET, POST, PUT, etc.).
- "url.path" — Contains the path in the request's URL.
- "error.code" — Contains the error code if the request results in an error.

Setting format: `string`.

Default value: `default`.

5.3.4 LUNA_MONITORING

In this section, settings for monitoring LUNA PLATFORM services are set.

For more information about monitoring, see "Monitoring" section.

5.3.4.1 storage_type

The storage type for storing monitoring data.

Currently, only the Influx DB is available.

Setting format: `string`.

Default value: `influx`.

5.3.4.2 send_data_for_monitoring

The parameter enables you to enable or disable sending monitoring data to InfluxDB.

Setting format: `integer`.

Default value: `1`.

5.3.4.3 use_ssl

The parameter enables you to use HTTPS to connect to InfluxDB.

Setting format: `integer`.

Default value: `0`.

5.3.4.4 organization

The parameter sets InfluxDB 2.x workspace.

Setting format: `string`.

Default value: `luna`.

5.3.4.5 token

The parameter sets InfluxDB 2.x authentication token.

Setting format: `string`.

5.3.4.6 bucket

The parameter sets InfluxDB 2.x bucket name.

Setting format: `string`.

Default value: `luna_monitoring`.

5.3.4.7 host

The parameter sets IP address of server with InfluxDB 2.x.

The default address is "127.0.0.1". This address means that InfluxDB 2.x will be used, located on the server with LUNA Configurator. If InfluxDB 2.x is located on a different server, then you should specify the correct InfluxDB 2.x IP address in this parameter.

Setting format: `string`.

Default value: `127.0.0.1`.

5.3.4.8 port

The parameter sets InfluxDB 2.x port.

Setting format: `string`.

Default value: `8086`.

5.3.4.9 flushing_period

The parameter sets frequency of sending monitoring data to InfluxDB.

Setting format: `integer` (seconds).

Default value: `1`.

5.3.5 LUNA_SERVICE_METRICS section

This section enables and configures the collection of metrics in the Prometheus format.

For more information about monitoring, see "Monitoring" section.

5.3.5.1 enabled

The parameter enables metrics collection.

If metrics collection is disabled, a request to the `/metrics` resource will return an appropriate message.

Setting format: `boolean`.

Default value: `false`.

5.3.5.2 metrics_format

The parameter sets the metrics format.

Currently only the Prometheus format is supported.

See the official Prometheus documentation for more details.

Setting format: `string`.

Default value: `prometheus`.

5.3.5.3 extra_labels

The parameter specifies custom label types.

Setting format: `label_name=label_value`.

Default value is not set.

5.3.6 LUNA_STREAMS_HTTP_SETTINGS

This section contains parameters responsible for process HTTP connections. More detail see here.

5.3.6.1 request_timeout

The parameter sets the duration of time between the instant when a new open TCP connection is passed to the server, and the instant when the whole HTTP request is received.

Setting format: `integer` (seconds).

Default value: `60`.

5.3.6.2 response_timeout

The parameter sets the duration of time between the instant the server passes the HTTP request to the app, and the instant a HTTP response is sent to the client.

Setting format: `integer` (seconds).

Default value: `600`.

5.3.6.3 request_max_size

The parameter sets the maximum size of the request.

Setting format: `integer` (bytes).

Default value: `1073741824`.

5.3.6.4 keep_alive_timeout

The parameter sets the timeout for maintaining HTTP activity.

Setting format: `integer` (seconds).

Default value: `15`.

5.3.7 LUNA_STREAMS_LOGS_CLEAR_INTERVAL

This section contains settings for automatic deletion of stream logs.

Automatic log deletion helps clear the "log" table of the LUNA Streams database from a large number of unnecessary logs.

See the detailed description with examples in the section "Streams logs automatic deletion".

5.3.7.1 active
The parameter enables or disables automatic deletion of logs.

Setting format: `boolean`.

Default value: `0`.

5.3.7.2 interval
The parameter specifies the interval for deleting logs. Logs older than this interval will be deleted.

The interval must be specified in conjunction with interval type. For example, "3" (interval) + day (interval type).

Setting format: `integer`.

Default value: `7`.

5.3.7.3 interval_type
The parameter specifies one of the following interval types:

- "weeks"
- "days"
- "hours"
- "minutes"
- "seconds"

The interval type must be specified in conjunction with the interval. For example, "3" (interval) + day (interval type).

Setting format: `string`.

Default value: `days`.

5.3.7.4 check_interval
The parameter specifies the frequency of checking logs for deletion in seconds.

Setting format: `integer`.

Default value: `180`.

5.3.8 Other

5.3.8.1 stream_worker_async_lock_timeout

The parameter sets timeout of the LUNA Streams instance to lock a row in a database table. The value of this setting should be increased if the stream statuses are not updated, which may be due to a slow connection from the service to the database.

Setting format: `number` (seconds in range (0, 1]).

Default value: `0.1`.

5.3.8.2 stream_status_obsoleting_period

The parameter sets stream status obsolescence period. For this period of time, the FaceStream worker should transfer the LUNA Streams report. Otherwise, the status of the stream will be changed to "restart", and the belated report will be rejected.

Setting format: `number` (seconds in range (0, 86400]).

Default value: `20`.

5.3.8.3 luna_streams_active_plugins

The parameter sets the list of active plugins (see the information about plugins workflow in the LUNA PLATFORM 5 administrator manual).

Setting format: `array > string`.

Default value: `[]`.

5.3.8.4 storage_time

The parameter specifies the time format used for records in the database. The following values are available:

- "LOCAL" — Displays the local time of the system on which the logs are being recorded.
- "UTC" — Displays Coordinated Universal Time, which is a time standard and is not affected by local time zones or seasonal time changes.

Setting format: `string`.

Default value: `LOCAL`.

6 FaceEngine settings

This section describes the FaceEngine parameters that are used to configure FaceStream.

Configuration of FaceEngine parameters is performed only by editing the configuration file "faceengine.conf". The configuration file is mounted into the FaceStream container when it is launched (see the FaceStream launch command in the installation manual).

In this section you can set the settings for face and body recognition. It is recommended to change the parameters in this configuration only in consultation with VisionLabs employees.

6.1 FaceDetV3

This parameter sets the settings for the face detector.

Note. This section is used only when the "use-face-detector" parameter is enabled. When working with bodies, parameters from the "HumanDetector" section will be used.

6.1.1 minFaceSize

The parameter sets the scaling factor of the source frame before processing.

Images are resized by minFaceSize/20 times. For example, if the value is minFaceSize=50, then the image size will be resized by 2.5 times (minFaceSize=50/20).

For more information about scaling, see "Frame scaling".

6.2 HumanDetector

This section sets the settings for the face and body association detector.

Note. This section is used only when the "use-body-detector" parameter is enabled. When working with faces, parameters from the "FaceDetV3" section will be used.

6.2.1 ImageSize

The parameter sets the maximum frame size after scaling along the largest side of the frame.

This parameter works similarly to the "scale-result-size" parameter from the TrackEngine settings.

For more information about scaling, see "Frame scaling".

7 Use FaceStream with LUNA Configurator

LUNA Configurator service enables you to store FaceStream, TrackEngine, and LUNA Streams settings and pass them to running FaceStream instances.

> LUNA Configurator service also enables you to store the settings all the LUNA PLATFORM services necessary to launch FaceStream. See the LUNA PLATFORM settings in the LUNA PLATFORM 5 administrator manual.

After FaceStream is launched it uses the parameters specified in the LUNA Configurator service and does not request them until restart by default. If necessary, you can enable checking for changes in LUNA Configurator configurations and automatically restarting the FaceStream instance(s) using the "CONFIG_- RELOAD" launch key, and set the period for receiving parameters using the "PULLING_TIME" launch key.

> See description of the launch keys in the "Launch FaceStream manually" > "Commands for launching FaceStream container" > "Launching keys" section FaceStream installation manual.

If checking for changes in the settings is disabled, then to apply the FaceStream instance settings changed in LUNA Configurator, you should manually restart this FaceStream instance.

7.1 Features of working with Configurator

If the use of the LUNA Configurator service is specified in the FaceStream startup line, but the service is unavailable, or the specified parameters are incorrect, then FaceStream will issue an error in the log and will not be started. In this case, the FaceStream and TrackEngine settings will be taken from the local configuration files, if they were uploaded to the container.

> LUNA Streams is not intended to use a configuration file.

7.2 Parameters in Configurator

LUNA Configurator includes records with the specified parameters.

Each of the LUNA Configurator records contains a name, a tag, and a configuration body. A record corresponds to one of the configuration files.

Parameters in the LUNA Configurator services have the same names as in configuration files (fs3Config.conf, trackengine.conf) and documentation.

Table 15: Correspondence of the LUNA Configurator data and distribution configuration files

Record name in LUNA Configurator	Corresponding configuration file	Description
FACE_STREAM_CONFIG	fs3Config.conf	FaceStream configuration

Record name in LUNA Configurator	Corresponding configuration file	Description
TRACK_ENGINE_CONFIG	trackengine.conf	Detection and tracking parameters face or body

7.3 Set configurations for several FaceStream instances

If a single FaceStream instance is working with the Configurator service, it uses the settings that are loaded by default.

If you want to use multiple FaceStream instances with different settings, create a separate record with a unique tag for each of these settings.

The tag is a unique identifier for the record and is specified in the launching keys described in the installation manual. Thus a specific FaceStream instance can get its own unique settings.

Follow these steps:

- Duplicate the record, for example, "FACE_STREAM_CONFIG", by pressing the **Duplicate** button.

Figure 21: Duplicate record

- Set a tag and specify parameters values.

Figure 22: Change tag

Tags are not created for the default records.

8 Use FaceStream with configuration files

If necessary, you can launch FaceStream independently of the "FACE_STREAM_CONFIG" and "TRACK_-ENGINE_CONFIG" settings of the LUNA Configurator service using the settings from the configuration files.

> With this launch option, it is assumed that the dependent LUNA PLATFORM services will also be launched with configuration files. The description of launching LUNA PLATFORM services with configuration files is not given in this documentation.

FaceStream can be launched with settings from configuration files using the following configuration files:

- fs3Config.conf (settings are similar to "FACE_STREAM_CONFIG" section in LUNA Configurator)
- trackengine.conf (settings are similar to "TRACK_ENGINE_CONFIG" section in LUNA Configurator)
- faceengine.conf

You should first set all the necessary parameters in these files before launching FaceStream.

The command for manually launching a container using configuration files will differ from the command for launching with Configurator and will look as follows:

```
docker run \
--env=CONFIGURATION_PATH=/srv/facestream/data/fs3Config.conf
-v /var/lib/fs/fs-current/extras/conf/configs/fs3Config.conf:/srv/facestream
    /data/fs3Config.conf \
-v /var/lib/fs/fs-current/extras/conf/configs/faceengine.conf:/srv/
    facestream/data/faceengine.conf \
-v /var/lib/fs/fs-current/extras/conf/configs/trackengine.conf:/srv/
    facestream/data/trackengine.conf \
-v /etc/localtime:/etc/localtime:ro \
--restart=always \
--detach=true \
--name=facestream \
--network=host \
--env=PORT=34569 \
--entrypoint /srv/facestream/FaceStream \
dockerhub.visionlabs.ru/luna/facestream:v.5.2.8 \
--config-path /srv/facestream/data/fs3Config.conf \
--data-dir /srv/facestream/data \
--log-dir /srv/facestream/logs \
--http-address http://0.0.0.0:34569
```

> Note that if the "CONFIGURATOR_HOST" and "CONFIGURATOR_PORT" environment variables are specified, the "CONFIGURATION_PATH" flag will be ignored.

The configuration files are included in the FaceStream package in the "conf/config/" directory and are added to the container at launch with the following commands:

```
-v /var/lib/fs/fs-current/extras/conf/configs/fs3Config.conf:/srv/facestream
    /data/fs3Config.conf \
-v /var/lib/fs/fs-current/extras/conf/configs/faceengine.conf:/srv/
    facestream/data/faceengine.conf \
-v /var/lib/fs/fs-current/extras/conf/configs/trackengine.conf:/srv/
    facestream/data/trackengine.conf \
```

8.1 Launching keys for server mode with configuration files

To launch FaceStream in server mode with configuration files inside the container, the following command is used, which enables you to specify the correct paths to directories inside the container.

```
--config-path /srv/facestream/data/fs3Config.conf \
--data-dir /srv/facestream/data \
--log-dir /srv/facestream/logs \
--streams-id 426542d6-5509-4e5b-8a01-e2abd5c0a8c6 ee4c42b6-23ae-410e-a2aa-
    a4220e64ba4b
--groups-name stream_group-1 stream_group-2
--config-reload 1
--pulling-time 1800
--http-address http://0.0.0.0:34569
```

The following keys are available:

- --help – Gives a list of available keys and their description.

- --config-path – Full path to the configuration file "fs3Config.conf" of the application. If this parameter is defined, the path to data is ignored when searching for the configuration file.

- --data-dir – Path to the directory with detectors and settings data.

- --log-dir – Directory to record logging files.

- --streams-id – Tag specifies a list of stream IDs that will be requested from LUNA Streams for processing. Other streams will be filtered. The "stream_id" parameter is given in response to the "create stream" request.

  If the --streams-id tag is not set, then FaceStream will take all existing "stream_id" from the queue.

  If a non-existent value is set, an error about an incorrect UUID will be indicated when launching FaceStream.

- `--streams-name` – List of streams names sets in this tag. Streams names are set using the "name" parameter at the time of their creation ("create streams" request). Streams with these names will be requested from LUNA Streams for processing. Other streams will be filtered.

  Otherwise, the principle of operation is similar to the `--streams-id` tag.

- `--groups-id` and `--groups-name` – Tags specify a list of group IDs or a list of group names. The parameters "group_id" or "group_name" are set during stream creation ("create stream" request). Streams with these parameters will be requested from LUNA Streams for processing. Other streams will be filtered.

  If the `--groups-id`/`--groups-name` tags are not set, then FaceStream will not filter streams by groups.

  If a non-existent value is set, an error about an incorrect UUID will be indicated when launching FaceStream.

- `--config-reload` – Tag that enables checking for changes in the "fs3Config.conf" file and takes the following values:

  - `1` — Change tracking is enabled, if there are changes in the configuration, all FaceStream containers will be automatically restarted.
  - `0` — Change tracking is disabled.

  By default, the value equals `1`.

- `--pulling-time` – tag that sets the period for receiving new parameters from the "fs3Config.conf" file in the range [1…3600] sec. Used in conjunction with the `CONFIG-RELOAD` tag.

  By default, the value equals `10`.

- `--http-address` – HTTP address that FaceStream will listen to. It is set in the format "address:port" (used only for FaceStream in server mode). The user will send requests to this address.

  You should set the external IP of the FaceStream server. By default, it is set to "http://0.0.0.0:34569".

9 LUNA Streams user interface

A user interface is available for the LUNA Streams service.

The interface can be opened in the browser by specifying the address and port of the LUNA Streams service:

```
<streams_server_address>:<streams_server_port>
```

> The default port of the LUNA Streams service is 5160.

It is recommended to read the section "Interaction of FaceStream with LUNA Streams" before introducing this section.

The user interface of the service contains three tabs — "Streams", "Groups" and "Queue".

- "Streams" is a tab where streams statuses and their preview are displayed, it is possible to configure video stream parameters for each of the sources.
- "Groups" is a tab where streams groups are displayed.
- "Queue" is a tab where the streams that are in the processing queue are displayed.

The general view of the available tabs is shown in the figure below:
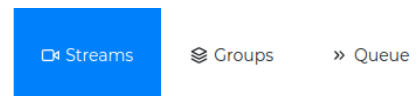
Figure 23: Available tabs

9.1 Streams tab

The "Streams" tab is designed to display all streams, their preview, ID, name, status, description, group and settings of video stream parameters for each source.

The general view of the "Streams" tab is shown in the figure below:
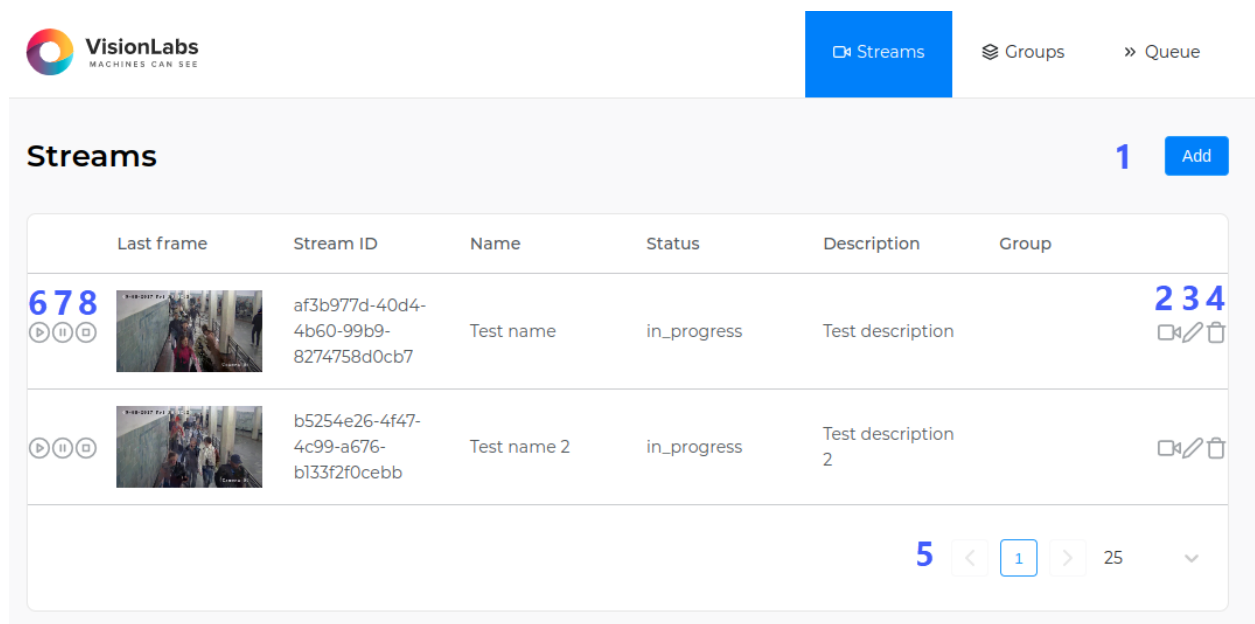
Figure 24: General view of the "Streams" tab

The "Streams" tab contains the following elements:

- "Add" — Button to add a stream (1).
- List of streams:
    - "Last frame" — Last frame of the video stream (preview).
    - "Stream ID" — ID of the video stream.
    - "Name" — Name of the video stream.
    - "Status" — Current status of the video stream.
    - "Description" — Additional information about the video stream.
    - "Group" — Name of the group to which the video stream is linked. You can link a video stream to a group or untie it from it in the "Groups" section.
    - Button for viewing the video stream (2).
    - Button for editing the video stream parameters (3).
    - Button for deleting the video stream (4).
    - Number of streams displayed on the page is set by a switch in the lower right corner of the page. In total there can be 10, 25, 50 or 100 video streams on one page (5).
    - Buttons to control the processing of the stream:
        * "Play" button to start processing the stream (6) (sends the request for processing the stream, the stream is distributed to a certain FaceStream instance and that starts processing it).
        * "Pause" button to pause the processing of the stream, for example, to save resources (7) (suspends the processing of the stream, but the stream remains assigned to the same FaceStream instance).

* "Stop" button to stop processing the stream (8) (stops the process of processing the stream, the stream is no longer assigned to the same FaceStream instance).

9.1.1 Stream creating

To create a stream, click on the "Add" button, after which the "Create stream" form opens to specify parameters.

Figure 25: Stream creating

You need to specify the values of the stream parameters and click the "Create" button in the upper right corner of the screen.

The tables below show the correspondence between the description in the user interface and the name of the corresponding parameters.

9.1.1.1 General stream parameters group

This group reflects the general parameters for creating a stream.

Table 16: "General stream parameters" group

Parameter description	Name in settings
Account ID	Parameter "account_id"
Stream name	Parameter "name"
Description	Parameter "description"
Group	Parameter "group"
Status	Stream status

9.1.1.2 Stream data group

This group reflects the main parameters for working with a video stream/video file/images.

Table 17: "Stream data" group

Parameter description	Name in settings
Type	Parameter "type"
Full path to the source	Parameter "reference"
ROI coordinates	Parameter "roi"
DROI coordinates	Parameter "droi"
Rotation angle of the image from the source	Parameter "rotation"
Frame width	Parameter "preferred_program_stream_frame_width"
Endless	Parameter "endless"
Mask	Parameter "mask"

9.1.1.3 Stream handler parameters group

This group of parameters defines the parameters of the handler created in the LUNA PLATFORM, with which the streams will be processed. Different handlers should be used for the face and body. The handler must be created in LP 5 in advance.

Table 18: "Stream handler parameters" group

Parameter description	Name in settings
Handler URL	Parameter "origin"
API version	Parameter "api_version"
ID of the handler for the best frames	Parameter "handler_id" of the "bestshot_handler" secion
Authorization (Token)	Token from "authorization" section

9.1.1.4 Geoposition group

This group of parameters includes information about the location of the video stream source.

Table 19: Geoposition group

Parameters	Name in settings
City, Area, District, Street, House number, Longitude (in degrees), Latitude (in degrees)	Section "location"

9.1.1.5 Autorestart group

This group of parameters enables you to configure the automatic restart of the stream.

Table 20: Geoposition group

Parameter description	Name in settings
Autorestart	Parameter "restart"
Attempt count	Parameter "attempt_count"
Autorestart delay (in seconds)	Parameter "delay"

9.1.1.6 Sending parameters group

In this group of parameters, the period during which the frame analysis will be carried out to select the best shot is determined, as well as all parameters associated with compiling a collection of the best shots are determined.

Table 21: Sending parameters group

Parameter description	Name in settings
Frame analysis period after which the best shot will be sent	Parameter "time_period_of_searching"
Wait duration between track analysis periods	Parameter "silent_period"
Track analysis and waiting period duration measure	Parameter "type"
Number of frames that the user sets to receive from the track or certain periods of this track	Parameter "number_of_bestshots_to_send"
Send only full set	Parameter "send_only_full_set"
Delete best shot and detection data	Parameter "delete_track_after_sending"

9.1.1.7 Use Primary Track group

This group of parameters is designed to work with access control systems (ACS, turnstiles at the entrances) to simplify the control and implementation of facial recognition technology at the entrance to the protected area. This group of parameters is used only for working with faces.

Table 22: Use Primary Track group

Parameter description	Name in settings
Use Primary Track	Parameter "use_primary_track_policy"
Minimum detection size for Primary Track mode	Parameter "best_shot_min_size"
Size of detection for the main track	Parameter "best_shot_proper_size"

9.1.1.8 Healthcheck parameters group

In this group, you can set parameters for reconnecting to the stream if there are errors in playing the stream.

Parameter description	Name in settings
Maximum number of stream errors to reconnect to the stream	Parameter "max_error_count"
Error count period duration (in seconds)	Parameter "period"
Time between reconnection attempts (in seconds)	Parameter "retry_delay"

9.1.1.9 Liveness parameters group

This group of parameters is used to check for Liveness, i.e. to check whether a person in the frame is real and prevents fraud when printed photos or images on the phone are used to pass the Liveness check.

Table 24: Liveness parameters group

Parameter description	Name in settings
Check RGB ACS Liveness	Parameter "use_mask_liveness_filtration"
Check FlyingFaces Liveness	Parameter "use_flying_faces_liveness_filtration"
Which track frames will be checked for Liveness	Parameter "liveness_mode"
Number of frames in the track for Liveness check when liveness-mode is enabled	Parameter "number_of_liveness_checks"
Threshold value at which the system will consider that there is a real person in the frame	Parameter "liveness_threshold"
Livenesses weights (RGB ACS, FlyingFaces)	Parameter "livenesses_weights"
Number of background frames that are used for the corresponding checks	Parameter "mask_backgrounds_count"

9.1.1.10 Filtering parameters group

This group of parameters describes the objects of image filtering and sending the resulting best shots.

Table 25: Liveness parameters group

Parameter description	Name in settings
Threshold value to filter detections	Parameter "min_score"

Parameter description	Name in settings
Head rotation angle threshold (to the left or right, yaw)	Parameter "detection_yaw_threshold"
Head tilt angle threshold (up or down, pitch)	Parameter "detection_pitch_threshold"
Head tilt angle threshold (to the left or right, roll)	Parameter "detection_roll_threshold"
Number of frames used to filter photo images by the angle of tilt of the head	Parameter "yaw_number"
Number of frames the system must collect to analyze head yaw angle	Parameter "yaw_collection_mode"
Mouth overlap threshold (minimum mouth visibility)	Parameter "mouth_occlusion_threshold"
Minimum body detection size	Parameter "min_body_size_threshold"

9.1.1.11 Additional parameters group

This group of parameters is intended for detailed configuration of video stream processing.

Table 26: Liveness parameters group

Parameter description	Name in settings
Frame processing	Parameter "frame_processing_mode"
Number of threads for video decoding	Parameter "ffmpeg_threads_number"
Maximum FPS for video processing	Parameter "real_time_mode_fps"

After saving the settings for the newly created stream, the message `Source <stream_id> was successfully created` will appear.

9.1.2 Stream editing

To edit the stream parameters, click the "Edit" button.

The form of editing the stream is similar to the form of creating the stream.

Change the necessary parameters and click the "Save" button in the upper right corner of the form.

After saving the settings of the stream parameters, the message `Source <stream_id> was successfully updated` will appear.

9.1.3 Stream deleting

To delete the stream, click the "Delete" button or in the form of editing the stream, click the "Delete" button in the "Streams" tab.

In the pop—up window, you need to confirm the action — click the "Delete" button or cancel the action via the "Cancel" button.

After clicking the "Delete" button, a message about deleting the stream will appear.

9.2 Groups tab

Streams can be grouped. Grouping is designed to combine streams with multiple cameras into logical groups (for example, by territorial feature).

> For more information, see "Streams grouping".

When creating a stream, it can only be added to one group.

The "Groups" tab contains a table that displays the name of groups, group IDs, descriptions, creation dates, account ID in LUNA PLATFORM and group settings.

The "Groups" tab is shown in the figure below:
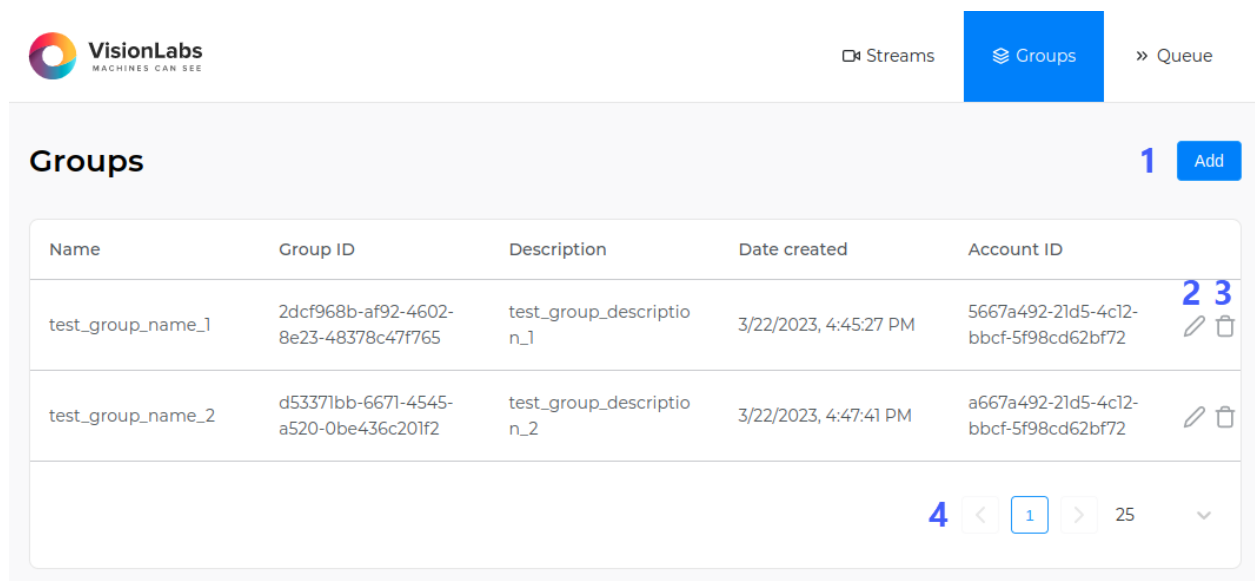
Figure 26: "Groups" tab

The "Groups" tab contains the following elements and group parameters:

- "Add" — Button to create a group (1).
- List of groups and their parameters:
  - "Name" — Name of the group.
  - "Group ID" — Group ID assigned by LUNA Streams.
  - "Description" — Additional user information about the group.
  - "Date created" — Date and time of the group creation.
  - "Account ID" — Account ID in LUNA PLATFORM 5.
- Group edit button (2).
- Group delete button (3).
- Number of groups displayed on the page is set by the radio button in the lower right corner of the page. There can be 10, 25, 50 or 100 groups on one page (4).

The process of creating, editing, and deleting a group is similar to the processes described above for a stream.

9.2.1 Linking stream to group
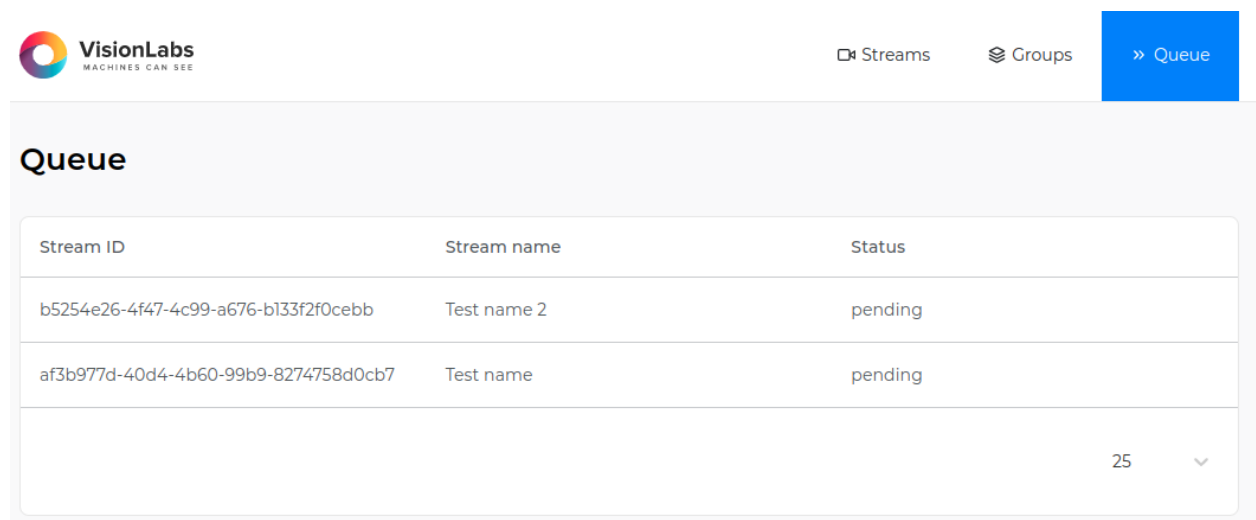
To link a stream to a group, in the "Streams" section in the stream line, click the button to edit the parameters. In the edit form, in the main stream parameters, specify a value for the "Group" parameter from the available drop-down list. The stream will be linked to the selected group. Click the "Save" button in the upper right corner of the form.

9.3 Queue tab

By default, a new stream is created with the status "pending" and immediately enters the processing queue. Processing of the stream can be postponed by specifying the status "pause" when creating. As soon as a free FaceStream worker of a stream with a pool request from the queue appears, the stream is accepted for processing and the status "in_progress" is assigned to it.

> For more information, see "Stream processing queue" section.

The "Queue" tab is informational and is designed to display all the streams in the queue for processing, their IDs and statuses.

Figure 27: Queue tab

The "Queue" section contains the following elements and stream parameters:

- List of streams:
  - "Stream ID" — ID of the video stream in LUNA Streams, generated when creating the stream.
  - "Stream name" — Name of the video stream.
  - "Status" — Current status of the video stream.
- Number of streams displayed on the page is set by a switch in the lower right corner of the page. There can be 10, 25, 50 or 100 threads on a single page.

10 Monitoring

Monitoring in FaceStream is implemented as sending data to InfluxDB and is disabled by default.

LUNA Streams has several monitoring methods:

- Sending data to InfluxDB (enabled by default)
- Exporting metrics in Prometheus format via the /metrics resource (disabled by default)

10.1 InfluxDB

To work with InfluxDB, you need to register with a username and password and specify the bucket name, organization name and token. All this data is set when starting the InfluxDB container using environment variables.

In order to use FaceStream or LUNA Streams monitoring, it is necessary in FaceStream settings or LUNA Streams settings to set for the "bucket", "organization", "token" fields exactly the same data specified when launching the InfluxDB container. So, for example, if the following settings were used when starting the InfluxDB container…:

```
-e DOCKER_INFLUXDB_INIT_BUCKET=luna_monitoring \
-e DOCKER_INFLUXDB_INIT_USERNAME=luna \
-e DOCKER_INFLUXDB_INIT_PASSWORD=password \
-e DOCKER_INFLUXDB_INIT_ORG=luna \
-e DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=kofqt4Pfqjn6o \
```

… then the following parameters should be specified in the FaceStream or LUNA Streams settings:

```
"influxdb": {
    "organization": "luna",
    "token": "kofqt4Pfqjn6o",
    "bucket": "luna_monitoring",
```

> Login and password are used to access the InfluxDB user interface.

> FaceStream and LUNA Streams settings contain different data of the "bucket", "organization" and "token" fields by default. If you need to use monitoring for both services, then you need to set the same settings. If necessary, you can save FaceStream and LUNA Streams data to different buckets (see below).

In order to separate FaceStream and LUNA Streams monitoring data, you can create separate buckets after launching the InfluxDB container. This can be done using one of the following methods:

- Using the InfluxDB user interface (Explore tab > Create bucket) after launching the InfluxDB container.
- Using the command `influx bucket create -n <bucket_name> -o <organization_name>` in InfluxCLI after launching the InfluxDB container.

> The organization name must be the same as when creating the InfluxDB container.

The data sent to InfluxDB differs for LUNA Streams and FaceStream. See the relevant sections below for more details on the data sent.

10.2 FaceStream monitoring

10.2.1 Enable monitoring

To enable FaceStream monitoring, follow these steps:

- Go to the Configurator user interface: `http://<configurator_server_ip>:5070/`.

- Enter "FACE_STREAM_CONFIG" in the "Setting name" field and click "Apply Filters".

- Enable the "send_data" setting in the "monitoring" section.

- Depending on the values of the parameters "DOCKER_INFLUXDB_INIT_BUCKET", "DOCKER_INFLUXDB_INIT_ORG", "DOCKER_INFLUXDB_INIT_ADMIN_TOKEN" set when launching the Influx container, specify the corresponding values in the fields "bucket", "organization" and "token" in the section "monitoring".

- Restart the FaceStream container: `docker restart facestream`.

10.2.2 Data being sent

The following data is sent to InfluxDB:

- "measurement" element. It is equal to the value of "fs-requests".

- Tag set:
  - "fs_ip" — IP address where FaceStream is deployed.
  - "source" — The "name" field set when creating a stream in LUNA Streams (optional).
  - "stream_id"

- Field set:
  - "track_id"
  - "event_id"
  - "request_id" — External ID for communication with monitoring of LUNA PLATFORM services.
  - "track_start_time"

– "track_best_shot_time" — Time when the frame with the best shot being sent appeared in the system.
– "track_best_shot_min_size_time" (optional) — Time when the detection size reached the value specified in the "best_shot_min_size" parameter.
– "track_best_shot_proper_size_time" (optional) — Time when the detection size reached the value specified in the "best_shot_proper_size" parameter.
– "liveness_start_time" (optional) — Liveness start time.
– "liveness_end_time" (optional) — Liveness end time.
– "bestshot_count" — Number of best shots sent in one request to LP along with the current best shot. So, for example, if 2 sends of 10 best shots were made, then the value of this parameter will be 10, and the value of the 'track_send_count' parameter will be 2.
– "time_from_first_frame_to_send" — Time that passed from the appearance of the first frame in FS to sending to LP.
– "track_send_count" — Sequence number of sending data from the track

Tags containing time are sent as UTC with microsecond precision.

- "timestamp" element. Is the time the best shot(s) was(were) sent in microseconds.

The frequency of sending data to InfluxDB is controlled by the "flashing_period" parameter of the FaceStream settings.

There may be several best shots, because sending from one track at a time counts as one measurement. To save this measurement, InfluxDB uses the last best shot data from the best shots group. Data that is unique for each best shot (`track_best_shot_time`, `liveness_start_time`, `liveness_end_time`) will be lost for all best shots except the last one if sent this way.

If there are no optional fields, the data of these fields will not be sent to the Influxdb.

During normal monitoring operation, no additional information is output to the FaceStream logs. If an error is detected during monitoring, the corresponding message will appear in the FaceStream logs.

10.3 LUNA Streams monitoring

10.3.1 Data being sent to InfluxDB

There are two types of events that are monitored:

- Request (all requests)
- Error (failed requests only)

Every event is a point in the time series. For the API service, the point is represented using the following data:

- Series name (requests or errors)

- Timestamp of the request start
- Tags
- Fields

> For other services, the set of event types may differ. For example, the Handlers service also collects data on SDK usage, estimations, and licensing.

The tag is an indexed data in storage. It is represented as a dictionary, where:

- Keys — String tag names.
- Values — String, integer or float.

The field is a non-indexed data in storage. It is represented as a dictionary, where:

- Keys — String field names.
- Values — String, integer or float.

Requests series. Triggered on every request. Each point contains a data about corresponding request (execution time and etc).

- Tags

Tag name	Description
service	Always "luna-streams"
route	Concatenation of a request method and a request resource (POST:/streams)
status_code	HTTP status code of response

- Fields

Field name	Description
request_id	Request ID
execution_time	Request execution time

Errors series. Triggered on failed request. Each point contains *error_code* of luna error.

- Tags

Tag name	Description
service	Always "luna-streams"
route	Concatenation of a request method and a request resource (POST:/streams)

Tag name	Description
status_code	HTTP status code of response
error_code	LUNA PLATFORM error code

- Fields

Field name	Description
request_id	Request ID

Licensing series. Triggered at service start and every 60 seconds. Each dot contains license verification data.

- Tags

Tag name	Description
service	Always "luna-streams"
license_status	License status ("ok", "warning", "error", "exception")

- Fields

Field name	Description
license_streams_limit_rate	Percentage of used streams
warnings	License warning messages
errors	License error messages

10.4 View InfluxDB data

You can use the InfluxDB GUI to view monitoring data.

- Go to the InfluxDB GUI `<server_ip>:<influx_port>`. The default port is 8086. The default login data is luna/password.

- Select the "Explore" tab.

- Select a way to display information in the drop-down list (graph, histogram, table, etc.).

- Select a bucket at the bottom of the page.

- Filter the necessary data.

- Click "Submit".

10.4.1 Export metrics in Prometheus format

LUNA Streams service can collect and save metrics in Prometheus format in the form of time series data that can be used to track the behavior of the service. Metrics can be integrated into the Prometheus monitoring system to track performance. See Prometheus official documentation for more information.

By default, the collection of metrics is disabled. The collection of metrics is enabled in the "LUNA_-SERVICE_METRICS" section.

> Note that all metric data is reset when the service is shut down.

10.4.1.1 Type of metrics

Two types of metrics are available:

- **Counters**, which increase with each event.
- **Cumulative histograms**, which are used to measure the distribution of duration or size of events.

> A cumulative histogram is a mapping that counts the cumulative number of observations in all of the bins up to the specified bin. See description in Wikipedia.

The following metrics of type **counters** are available:

- `request_count_total` — Total number of requests
- `errors_count_total` — Total number of errors

Each of them has at least two labels for sorting:

- `status_code` (or `error_code` for error metrics)
- `path` — Path consisting of a request method and an endpoint route.

> Labels are key pairs consisting of a name and a value that are assigned to metrics.

If necessary, you can add custom label types by specifying the pair `tag_name=tag_value` in the "extra_-labels" parameter.

> Note that the pair `tag_name=tag_value` will be added to each metric of the LUNA PLATFORM service.

A special manager distributes all requests passing through the service among the counters using these tags. This ensures that two successful requests sent to different endpoints or to the same endpoint, but with different status codes, will be delivered to different metrics.

> Unsuccessful requests are distributed according to the metrics `request_count_total` and `request_errors_total`.

The `requests` metric of **cumulative histogram** type tracks the duration of requests to the service. The following intervals (bucket) are defined for the histogram, in which the measurements fall:

- 0.0001
- 0.00025
- 0.0005
- 0.001
- 0.0025
- 0.005
- 0.01
- 0.025
- 0.05
- 0.075
- 0.1
- 0.25
- 0.5
- 0.75
- 1.0
- 2.5
- 5.0
- 7.5
- 10.0
- Inf

In this way the range of request times can be broken down into several intervals, ranging from very fast requests (0.0001 seconds) to very long requests (Inf - infinity). Histograms also have labels to categorize the data, such as `status_code` for the status of a request or `route` to indicate the route of a request.

Examples

If you send one request to the `/healthcheck` resource, followed by three requests to the `/docs/spec` resource, one of which will be redirected (response status code 301), then when executing the request to the `/metrics` resource, the following result will be displayed in the response body:

```
# HELP request_count_total Counter of requests
# TYPE request_count_total counter
request_count_total{path="GET:/docs/spec",status_code="200"} 2.0
request_count_total{path="GET:/docs/spec",status_code="301"} 1.0
request_count_total{path="GET:/healthcheck",status_code="200"} 1.0
```

If you send one invalid POST request to the /streams resource, then when executing the request to the /metrics resource, the following result will be displayed in the response body:

```
# HELP request_count_total Counter of requests
# TYPE request_count_total counter
request_count_total{path="POST:/streams",status_code="401"} 1.0
# HELP request_errors_total Counter of request errors
# TYPE request_errors_total counter
request_errors_total{error_code="12010",path="POST:/streams"} 1.0
# HELP requests Histogram of request time metrics
# TYPE requests histogram
requests_sum{route="GET:/docs/spec",status_code="200"} 0.003174567842297907
requests_bucket{le="0.0001",route="GET:/docs/spec",status_code="200"} 0.0
requests_bucket{le="0.00025",route="GET:/docs/spec",status_code="200"} 0.0
requests_bucket{le="0.0005",route="GET:/docs/spec",status_code="200"} 0.0
requests_bucket{le="0.001",route="GET:/docs/spec",status_code="200"} 1.0
...
requests_count{route="GET:/docs/spec",status_code="200"} 2.0
requests_sum{route="GET:/docs/spec",status_code="301"} 0.002381476051209132
```

10.4.2 Configuring metrics collection for Prometheus

Prometheus must be configured to collect LUNA PLATFORM metrics.

Example Prometheus configuration for collecting LP service metrics:

```
- job_name: "luna-streams"
  static_configs:
    - targets: ["127.0.0.1:5160"]
...

- job_name: "luna-configurator"
  static_configs:
    - targets: ["127.0.0.1:5070"]
```

See the official documentation for an example of running Prometheus.

11 Outputting information to logs

This section describes additional information that can help you when working with FaceStream logs or the logs of the services required for it.

11.1 FaceStream log output format

FaceStream logs have the following format:

```
[I0317 16:27:07.375125    57 LunaBaseClient.cpp:45] [client] Request
```

Where:

- `I0317`:

    - `I` — Logging level. 4 levels can be displayed in the logs — I (Info), W (Warning), E (Error), F (Fatal). If necessary, you can set the logging level (see "severity" parameter);
    - `0317` — Day and month, i.e. March 17.

- `16:27:07.375125` — Timestamp.

- `57` — Process PID ID.

- `LunaBaseClient.cpp` — File name that caused this log line to occur.

- `45` — Log string.

- `[client]` — Tag associated with logging the relevant information (see "tags" parameter).

- `Request` — Description of the log string.

FaceStream errors are not covered in this section.

11.2 LUNA Streams service errors

This section describes the errors returned by the LUNA Streams service. Each of the errors has a unique code. It is convenient to use it to find an error.

The errors can have different reasons.

In case of "Internal server error" or any other unexpected error occurrence, it is recommended to check service logs to find out more information about the error.

When using LUNA Streams together with LUNA PLATFORM services, other LP services may experience errors. In this case, please refer to the LUNA PLATFORM documentation or visit the online documentation site for a complete list of errors returned by LUNA PLATFORM 5 services.

11.2.1 Code 39001 returned

Error Message:

Object not found Stream with id {value} not found

Error Source:

LUNA Streams service errors

Error Description:

The stream with the specified ID was not found. Make sure that the existing "steam_id" is set. You can get a list of existing "stream_id" by using a GET request to the "/streams" resource.

11.2.2 Code 39002 returned

Error Message:

Bad input data "{value}" is not valid stream status; permitted: {value}.

Error Source:

LUNA Streams service errors

Error Description:

When creating the stream, an incorrect status was entered in the "status" field. You can set only two statuses — "pause" and "pending". The rest of the statuses can be obtained only at a certain point in time. The error description shows the expected status.

11.2.3 Code 39003 returned

Error Message:

Unable to stop processing Processing of stream with id "{value}" is already in progress and cannot be stopped.

Error Source:

LUNA Streams service errors

Error Description:

It is not possible to set the "pause" status for the specified "stream_id", since processing has already started (relevant only for video files).

11.2.4 Code 39004 returned

Error Message:

Bad input data "{value}" is not valid stream log target; permitted: {}.

Error Source:

LUNA Streams service errors

Error Description:

A non-existent value of the "targets" parameter is specified in the "/streams/logs" request to receive logs.

11.2.5 Code 39005 returned

Error Message:

Unable to cancel processing Processing of stream with id "{value}" is finished and cannot be cancelled

Error Source:

LUNA Streams service errors

Error Description:

It is not possible to set the "cancel" status for the specified "stream_id", because processing has already been finished.

11.2.6 Code 39006 returned

Error Message:

Unique constraint error Group named {value} already exists

Error Source:

LUNA Streams service errors

Error Description:

Group with specified name already exist.

Enter another name or delete the existing group using the "remove group" request.

11.2.7 Code 39007 returned

Error Message:

Object not found Group named {value} not found

Error Source:

LUNA Streams service errors

Error Description:

Group with specified name was not found.

Check the entered group name.

You can get the list of all existing groups with their parameters using the "get groups" request.

11.2.8 Code 39008 returned

Error Message:

Object not found Group with id {value} not found"

Error Source:

LUNA Streams service errors

Error Description:

Group with specified ID was not found.

Check the entered group ID.

You can get the list of all existing groups with their parameters using the "get groups" request.

11.2.9 Code 39009 returned

Error Message:

Object not found. Not found "{value}" preview url for stream with id "{value}".

Error Source:

LUNA Streams service errors

Error Description:

The specified preview url (live or last_frame) was not found for the specified "stream_id".

Make sure the report contains the required url.

11.2.10 Code 39010 returned

Error Message:

Preview processing error, {value}

Error Source:

LUNA Streams service errors
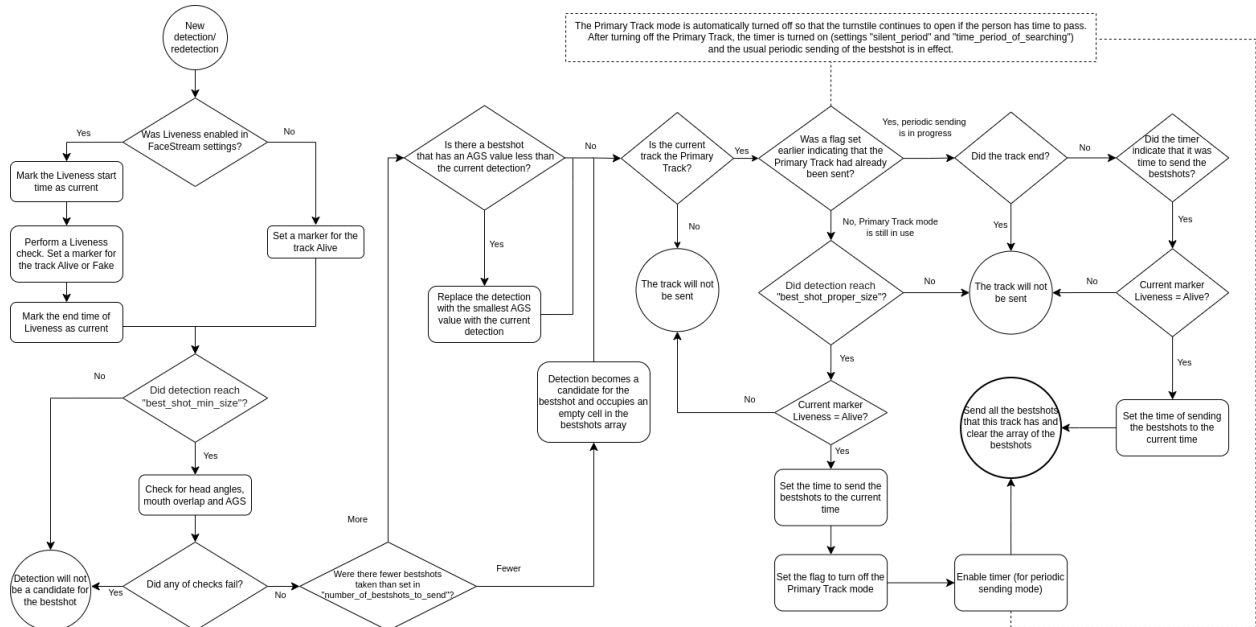
Error Description:

Preview processing error.

The brackets indicate the error being reported from the stream source.

12 Additional information

12.1 Activity diagram for Liveness and Primary Track Policy

Below is a diagram of FaceStream activity when using Liveness and Primary Track Policy.

Figure 28: Activity diagram for Liveness and Primary Track Policy

12.2 Nuances of working with stream preview

In order to write the preview address of the stream (the "preview" > "live" > "url" field) to the LUNA Streams stream, FaceStream must determine its IP address. It determines its IP address in one of two ways:

- Via connecting to the LUNA Streams service.
- Via the system environment variable "VL_FACE_HOST". The environment variable can be set by executing the following command in the server terminal: `export VL_FACE_HOST=< your_ip_address>`.

> If the IP address of the "VL_FACE_HOST" variable is set incorrectly, then LUNA Streams will delete streams with an incorrect address, and FaceStream will terminate with the following error `Failed to validate input json`.

Using an environment variable overrides the way you connect to the LUNA Streams service.

If the IP address of FaceStream is determined by connecting to the LUNA Streams service, then to determine its IP address, FaceStream must always start after LUNA Streams. If for some reason FaceStream started before LUNA Streams (for example, the server was restarted, where both FaceStream and LUNA Streams are running), then FaceStream may terminate with the error `Failed to get local IP address. Reason`. If the FaceStream IP address is determined through the system environment variable "VL_FACE_HOST", then the above error will not occur.

12.3 Proxying requests to LUNA Streams using LUNA API

If FaceStream is used in conjunction with LUNA PLATFORM, then it is possible to send part of the requests to the LUNA Streams service via the LUNA API service.

To do this, you need to enable and configure the special built-in plugin "luna-streams.py" in the API service. The plugin is located in the API service container at the path `/srv/luna_api/plugins` and is enabled using the "LUNA_API_ACTIVE_PLUGINS" setting.

> See more information about the plugin mechanism in the "Plugins" section of the LUNA PLATFORM administrator manual.

The following settings are available for the plugin, specified in the "LUNA_API_PLUGINS_SETTINGS" configuration of the API service:

- "luna-streams-address" > "origin" — Protocol, IP address and port of the LUNA Streams service.
- "luna-streams-address" > "api_version" — API version of the LUNA Streams service.
- "luna-streams-timeouts" > "connect" — Timeout for establishing a connection when sending an HTTP request to the LUNA Streams service.
- "luna-streams-timeouts" > "request" — General timeout for completing the entire HTTP request.
- "luna-streams-timeouts" > "sock_connect" — Timeout for establishing a connection at the socket level.
- "luna-streams-timeouts" > "sock_read" — Timeout for reading data from the socket after a successful connection.

If the "LUNA_API_PLUGINS_SETTINGS" configuration or the settings described above are not specified, the default values will be applied. See the "LUNA_API_PLUGINS_SETTINGS" section of the LUNA PLATFORM administrator manual for details of the above settings and their default values.

Example of setting the "LUNA_API_PLUGINS_SETTINGS" setting:

```
{
    "luna-streams": {
        "luna-streams-address": {
            "origin": "http://127.0.0.1:5160",
            "api_version": 2
        },
        "luna-streams-timeouts": {
            "request": 60,
            "connect": 20,
            "sock_connect": 10,
            "sock_read": 60
        }
    }
}
```

If necessary, you can differentiate access rights for requests to LUNA Streams using token permissions. To do this, you need to create a new or update an existing token by passing a custom "streams" key with permissions for the token.

> To proxy requests from the LUNA API service to the LUNA Streams service, you need to specify a permission named "streams". Other custom key names will not work.

You can set the following permissions:

- "creation" (POST requests)
- "view" (GET requests)
- "modification" (PUT requests and PATCH requests)
- "deletion" (DELETE requests)

Example of specifying a custom key "streams" with all possible permissions in a token creation request body:

```
{
    "permissions": {
        "streams": [
            "creation",
            "view",
            "modification",
            "deletion"
        ]
    }
}
```

> See detailed information about tokens in the "Accounts, tokens and authorization types" section of the LUNA PLATFORM administrator manual.

The table below shows the ratio of routes in the LUNA API and LUNA Streams, as well as links to requests and possible token resolutions.

LUNA API route	LUNA Streams route	Requests	Token permission
prefix/version	/version	get version	None
prefix/docs/spec	/{api_version}/docs/spec	get openapi documentation	None
prefix/plugins	/{api_version}/plugins	get list of plugins	None
prefix/streams	/{api_version}/streams	create stream	Custom key
		get streams	

LUNA API route	LUNA Streams route	Requests	Token permission
		delete streams	
`prefix/streams/count`	`/{api_version}/streams/count`	delete streams	Custom key
`prefix/streams/{stream_id}`	`/{api_version}/streams/{stream_id}`	get stream	Custom key
		update stream	
		put stream	
		remove stream	
`prefix/streams/logs`	`/{api_version}/streams/logs`	get streams logs	Custom key
		delete streams logs	
`prefix/streams/{stream_id}/preview/handler/live`	`/{api_version}/streams/{stream_id}/preview/handler/live`	get live preview	Custom key
`prefix/streams/{stream_id}/preview/handler/frame`	`/{api_version}/streams/{stream_id}/preview/handler/frame`	get last frame preview	Custom key

Prefix is `/6/plugins/luna-streams`.

You can send requests to both the first version of the LUNA Streams API and the second. To do this, you need to use the `LUNA-STREAMS-API-VERSION` header as follows:

- If the `LUNA-STREAMS-API-VERSION` header is not specified, the first API version will be automatically selected.
- If the value of the `LUNA-STREAMS-API-VERSION` header is "1", the first version will be selected.
- If the value of the `LUNA-STREAMS-API-VERSION` header is "2", the second version of the API will be selected.
- Otherwise an error will be returned.

12.4 Event data generated by FaceStream

By processing streams, FaceStream passes requests to generate events to LUNA PLATFORM. To generate an event, when creating a stream, you must fill in the "event_handler" field, indicating the handler ID in the "bestshot_handler" field, the address and API version of the LUNA API service, and the address where the source frame will be saved (if necessary).

Below is a table explaining what data processed by FaceStream is recorded in the LUNA PLATFORM event.

Parameter name	Description
account_id	Account ID set during the creation of a stream in LUNA Streams.
create_time	Start time of the track (time of event occurrence in the video stream) relative to the server FS time.
end_time	End time of the track (time of event completion in the video stream) relative to the server FS time.
handler_id	Handler ID set during the creation of a stream in LUNA Streams.
stream_id	Stream ID in LUNA Streams.
source	Field "name" set during the creation of a stream in LUNA Streams.
track_id	FaceStream track ID.
face_detections:	Face detection data:
	* **sample_id**: Sample ID, created if the "face_sample_policy" is enabled.
	* **detect_time**: Time of detection of the face best shot relative to the server FS time.
	* **image_origin**: Source frame.
	* **detect_ts**: Time of detection of the face best shot relative to the start of the video file.
	* **detection**: Bounding box coordinates with face detection ("x", "y", width, height).
body_detections:	Body detection data:
	* **sample_id**: Sample ID, created if the "body_sample_policy" is enabled.
	* **detect_time**: Time of detection of the body best shot relative to the server FS time.
	* **image_origin**: Source frame.
	* **detect_ts**: Time of detection of the body best shot relative to the start of the video file.

Parameter name	Description
	* **detection**: Bounding box coordinates with body detection ("x", "y", width, height).
location	Location data set during the creation of a stream in LUNA Streams.

In addition to the above parameters, the event will be supplemented with parameters generated by the specified handler (for example, "face_id", "match_result", "attach_result", "gender", etc.). The fields "external_id" and "user_data" will always remain empty when generating an event based on a sample transferred from FaceStream.