

VisionLabs FaceStream

Administrator Manual

v.5.1.2

Contents

Glossary	10
1 Introduction	11
2 System requirements	12
2.1 Minimum requirements	12
3 Software requirements	13
4 Overview	14
4.1 FaceStream workflow with faces and bodies	16
4.1.1 FaceStream workflow with faces	16
4.1.2 FaceStream workflow with bodies	18
4.2 Interaction of FaceStream with LUNA Streams	20
4.2.1 Stream distribution in LUNA Streams	22
4.2.1.1 Statuses transition table	22
4.2.2 Stream processing pipeline	23
4.2.3 LUNA Streams database description	24
4.3 General recommendations for FaceStream configuration	26
4.3.1 Before starting configuration	26
4.3.2 FaceStream performance configuration	27
4.3.2.1 Reduction of face search area	27
4.3.2.2 Frame scaling	29
4.3.3 Defining area with movement	31
4.3.4 Batch processing of frames	32
4.3.4.1 Minimal face size	32
4.3.5 General configuration information	34
4.3.5.1 Working with track	34
4.3.5.2 Bestshot sending	34
4.3.5.3 Frames filtration	35
4.3.5.4 Working with ACMS	36
4.4 Additional information	38
4.4.1 Formats, video compression standards, and protocols	38
4.4.1.1 Video formats	38
4.4.1.2 Encodings	38
4.4.1.3 Protocols	38
4.4.2 Memory consumption when running FaceStream	39
4.4.3 Monitoring	40

4.4.4	InfluxDB OSS 2	40
4.4.4.1	InfluxDB configuration	40
5	Priority parameters list	42
5.1	Parameters for sending faces	43
5.2	Parameters for sending bodies	44
6	FaceStream configuration	46
6.1	Logging section	47
6.1.1	Severity parameter	47
6.1.2	Tags parameter	47
6.1.3	Mode parameter	49
6.2	Sending section	50
6.2.1	Request_type section	50
6.2.2	Portrait_type parameter	51
6.2.3	Send_source_frame parameter	51
6.2.4	Send_detection_path parameter	52
6.2.5	Detection_path_length parameter	52
6.2.6	Minimal_body_track_length_to_send parameter	52
6.2.7	Async_requests parameter	53
6.2.8	Aggregate_attr_requests parameter	53
6.2.9	Jpeg_quality_level parameter	53
6.3	Lunastreams section	55
6.3.1	Origin parameter	55
6.3.2	Api_version parameter	55
6.3.3	Max_number_streams parameter	55
6.3.4	Request_stream_period parameter	56
6.3.5	Send_feedback_period parameter	56
6.3.6	Max_feedback_delay parameter	56
6.4	Performance section	58
6.4.1	Stream_images_buffer_max_size parameter	58
6.4.2	Enable_gpu_processing parameter	58
6.4.3	Convert_full_frame parameter	58
6.5	Debug section	60
6.5.1	Save_debug_info parameter	60
6.5.2	Save_only_jpegs_with_honest_detections parameter	60
6.5.3	Save_jpegs parameter	60
7	Streams management configuration	62
7.1	Account_id parameter	63

7.2	Name parameter	63
7.3	Description parameter	63
7.4	Data section	64
7.4.1	Type parameter	64
7.4.2	Reference parameter	64
7.4.3	Roi parameter	65
7.4.4	Droi parameter	65
7.4.5	Rotation parameter	67
7.4.6	Preferred_program_stream_frame_width parameter	67
7.4.7	Mask parameter	68
7.5	Event_handler section	69
7.5.1	Origin parameter	69
7.5.2	Api_version parameter	69
7.5.3	Handler_id parameter of bestshot_handler section	69
7.5.4	Handler_id parameter of detection_handler section	69
7.5.5	Frame_store parameter	70
7.6	Policies section	71
7.6.1	Sending section	71
7.6.1.1	Time_period_of_searching parameter	71
7.6.1.2	Silent_period parameter	72
7.6.1.3	Type parameter	72
7.6.1.4	Number_of_bestshots_to_send parameter	73
7.6.1.5	Send_only_full_set parameter	73
7.6.1.6	Delete_track_after_sending parameter	73
7.6.2	Primary_track_policy section	74
7.6.2.1	Use_primary_track_policy parameter	74
7.6.2.2	Best_shot_min_size parameter	74
7.6.2.3	Best_shot_proper_size parameter	74
7.6.3	Liveness section	75
7.6.3.1	General recommendations for Liveness usage	75
7.6.3.2	Use_shoulders_liveness_filtration parameter	79
7.6.3.3	Use_mask_liveness_filtration parameter	79
7.6.3.4	Use_flying_faces_liveness_filtration parameter	79
7.6.3.5	Liveness_mode parameter	79
7.6.3.6	Number_of_liveness_checks parameter	79
7.6.3.7	Liveness_threshold parameter	80
7.6.3.8	Liveness_weights parameter	80
7.6.3.9	Mask_backgrounds_count parameter	80

7.6.4	Filtering section	81
7.6.4.1	Min_score parameter	81
7.6.4.2	Detection_yaw_threshold parameter	81
7.6.4.3	Detection_pitch_threshold parameter	81
7.6.4.4	Detection_roll_threshold parameter	82
7.6.4.5	Yaw_number parameter	82
7.6.4.6	Yaw_collection_mode parameter	83
7.6.4.7	Mouth_occlusion_threshold parameter	83
7.6.5	Frame_processing_mode parameter	84
7.6.6	Real_time_mode_fps parameter	84
7.6.7	Ffmpeg_threads_number	85
7.6.8	Health_check section	86
7.6.8.1	Max_error_count parameter	86
7.6.8.2	Period parameter	86
7.6.8.3	Retry_delay parameter	86
7.7	Location section	87
7.8	Autorestart section	87
7.9	Status parameter	88
8	Trackengine configuration	89
8.1	Use-face-detector and use-body-detector	89
8.2	Detector-step	89
8.3	Detector-scaling	89
8.4	Scale-result-size	90
8.5	Frg-subtractor	91
8.6	Frg-regions-alignment	92
8.7	Frg-regions-square-alignment	92
8.8	Batched-processing	92
8.9	Min-frames-batch-size	93
8.10	Max-frames-batch-gather-timeout	93
9	LUNA Streams configuration	95
9.1	LUNA_STREAMS_DB section	95
9.1.1	Db_type parameter	95
9.1.2	Db_user parameter	95
9.1.3	Db_password parameter	95
9.1.4	Db_name parameter	95
9.1.5	Db_host parameter	96
9.1.6	Db_port parameter	96
9.1.7	Connection_pool_size parameter	96

9.2	LUNA_STREAMS_LOGGER section	96
9.2.1	Log_level parameter	96
9.2.2	Folder_with_logs parameter	97
9.2.3	Log_time parameter	97
9.2.4	Log_to_stdout parameter	97
9.2.5	Log_to_file parameter	97
9.2.6	Multiline_stack_trace parameter	98
9.3	LUNA_LICENSES_ADDRESS section	98
9.3.1	Origin parameter	98
9.3.2	Api_version parameter	98
9.4	STREAM_WORKER_ASYNC_LOCK_TIMEOUT parameter	98
9.5	STREAM_STATUS_OBSOLETING_PERIOD parameter	99
9.6	LUNA_STREAMS_ACTIVE_PLUGINS parameter	99
9.7	STORAGE_TIME parameter	99
9.8	INFLUX_MONITORING section	99
9.8.1	Send_data_for_monitoring parameter	99
9.8.2	Use_ssl parameter	99
9.8.3	Flushing_period parameter	100
9.8.4	Host parameter	100
9.8.5	Port parameter	100
9.8.6	Bucket parameter	100
9.8.7	Organization parameter	100
9.8.8	Token parameter	101
10	Use FaceStream with LUNA Configurator	102
10.1	Features of working with Configurator	102
10.2	Parameters in Configurator	102
10.3	Set configurations for several FaceStream instances	103
11	Use FaceStream with configuration files	105
11.1	Launching keys for server mode with configuration files	106
12	Outputting information to logs	107
12.1	FaceStream log output format	107
12.2	Additional services API errors	107
12.2.1	General errors	107
12.2.1.1	Code 0 returned	107
12.2.1.2	Code 1 returned	108
12.2.1.3	Code 3 returned	108
12.2.1.4	Code 4 returned	108

12.2.1.5	Code 5 returned	109
12.2.1.6	Code 6 returned	109
12.2.1.7	Code 7 returned	109
12.2.1.8	Code 8 returned	110
12.2.1.9	Code 9 returned	110
12.2.1.10	Code 10 returned	111
12.2.1.11	Code 11 returned	111
12.2.1.12	Code 12 returned	111
12.2.1.13	Code 13 returned	112
12.2.1.14	Code 14 returned	112
12.2.1.15	Code 15 returned	112
12.2.1.16	Code 16 returned	113
12.2.1.17	Code 17 returned	113
12.2.1.18	Code 18 returned	113
12.2.1.19	Code 19 returned	113
12.2.1.20	Code 21 returned	114
12.2.1.21	Code 22 returned	114
12.2.1.22	Code 23 returned	114
12.2.1.23	Code 24 returned	114
12.2.1.24	Code 11055 returned	115
12.2.2	CORE errors, returned in API responses	115
12.2.2.1	Code 100 returned	115
12.2.2.2	Code 101 returned	115
12.2.2.3	Code 102 returned	116
12.2.2.4	Code 105 returned	116
12.2.2.5	Code 106 returned	116
12.2.2.6	Code 200 returned	117
12.2.3	Database Errors	117
12.2.3.1	Code 10015 returned	117
12.2.3.2	Code 10016 returned	117
12.2.3.3	Code 10017 returned	118
12.2.3.4	Code 10018 returned	118
12.2.4	REST API common errors	118
12.2.4.1	Code 12001 returned	118
12.2.4.2	Code 12002 returned	119
12.2.4.3	Code 12003 returned	119
12.2.4.4	Code 12005 returned	119
12.2.4.5	Code 12012 returned	119
12.2.4.6	Code 12013 returned	120

12.2.4.7	Code 12014 returned	120
12.2.4.8	Code 12016 returned	120
12.2.4.9	Code 12017 returned	121
12.2.4.10	Code 12018 returned	121
12.2.4.11	Code 12021 returned	121
12.2.4.12	Code 12022 returned	121
12.2.4.13	Code 12023 returned	122
12.2.4.14	Code 12024 returned	122
12.2.4.15	Code 12025 returned	122
12.2.4.16	Code 12027 returned	123
12.2.4.17	Code 12028 returned	123
12.2.4.18	Code 12029 returned	123
12.2.4.19	Code 12030 returned	123
12.2.4.20	Code 12031 returned	124
12.2.4.21	Code 12032 returned	124
12.2.4.22	Code 12033 returned	124
12.2.4.23	Code 12034 returned	125
12.2.4.24	Code 12035 returned	125
12.2.4.25	Code 12036 returned	125
12.2.4.26	Code 12037 returned	126
12.2.4.27	Code 12038 returned	126
12.2.4.28	Code 12039 returned	126
12.2.4.29	Code 12040 returned	126
12.2.4.30	Code 12041 returned	127
12.2.4.31	Code 12042 returned	127
12.2.4.32	Code 12043 returned	127
12.2.4.33	Code 12044 returned	128
12.2.4.34	Code 12045 returned	128
12.2.5	LUNA Licenses service errors	128
12.2.5.1	Code 33001 returned	128
12.2.5.2	Code 33002 returned	128
12.2.5.3	Code 33003 returned	129
12.2.5.4	Code 33004 returned	129
12.2.5.5	Code 33005 returned	129
12.2.6	LUNA Streams service errors	130
12.2.6.1	Code 39001 returned	130
12.2.6.2	Code 39002 returned	130
12.2.6.3	Code 39003 returned	130
12.2.6.4	Code 39004 returned	131

12.2.6.5	Code 39005 returned	131
----------	-------------------------------	-----

13	Cameras Compatibility	132
-----------	------------------------------	------------

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).
Bestshot, best detection	Best shot is selected from all frames of the track. The main conditions for best shot selection are appropriate quality and the presence of a face with best aspect angle. Such conditions are set through FaceStream configuration.
Detection	Face detection in the frame.
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 person 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.
Normalized image, warp	Certain image format with horizontally aligned face. Used when working with LUNA PLATFORM. Such image contains all necessary information about the face and allows for faster process by the system.
Portrait	Face image fragment from a frame, selected in accordance with algorithm configuration and GOST 19794-5-2006 / ISO IEC 19794-5 2005(E) standard requirements.
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 and software requirements
- general description of the application and recommendations for setting up
- the 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 System requirements

2.1 Minimum requirements

The following minimum requirements are given per FaceStream instance.

For the application to work correctly, the hardware must meet the following minimum requirements:

- 2 GHz or faster processor;
- 4 Gb RAM or higher;
- 10 Gb available hard disk space.
- Access to the Internet (for containers and additional software download).

Hardware requirements can be affected by several factors:

- Number of video streams;
- Frame frequency and resolution of video streams;
- FaceStream settings. The default settings are the most versatile. Depending on the operating conditions of the application, using their values can affect the quality, or performance.

Hardware should be selected based on the above factors.

FaceStream can also work in the computation speedup mode due to:

Video card resources usage

GPU calculations are supported for FaceDetV3 only. See “defaultDetectorType” parameter in the Faceengine configuration (“faceengine.conf”).

A minimum of 6GB or dedicated video RAM is required. 8 GB or more VRAM recommended.

Pascal, Volta, Turing architectures are supported.

Compute Capability 6.1 or higher and CUDA 11.2.1 are required.

The recommended NVIDIA drivers are r450, r455.

Now only one video card is supported per FaceStream instance.

AVX2 instructions usage

CPU with AVX2 support is required.

The system automatically detects available instructions and runs best performance.

3 Software requirements

FaceStream and LUNA Streams containers launch were tested on the following operating systems:

- CentOS Linux release 7.8.2003 (Core)

The following OS is used inside the FaceStream container:

- CentOS Linux release 8.3.2011

Docker should be installed for containers launch.

4 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.

- **Sending face images 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.

- [FaceStream configuration](#)

In this configuration, general FaceStream settings are set, such as logging, setting up sending images from FaceStream to external services, debugging, etc.

- [Streams management configuration](#)

In this configuration, settings concerning stream sources are set, 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.

- [Trackengine configuration](#)

In this configuration, settings are set regarding the face or body detection and tracking.

- [Faceengine configuration](#)

In this configuration, the settings for face recognition are set. 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:

- Using the LUNA Configurator service, which stores FaceStream startup parameters and enables you to continue processing the current video even after restarting FaceStream in case of an emergency shutdown
- 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 with either faces or bodies. Simultaneous processing of faces and bodies is not possible.

4.1 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 “Settings for sending images to LUNA PLATFORM”.

4.1.1 FaceStream workflow with faces

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

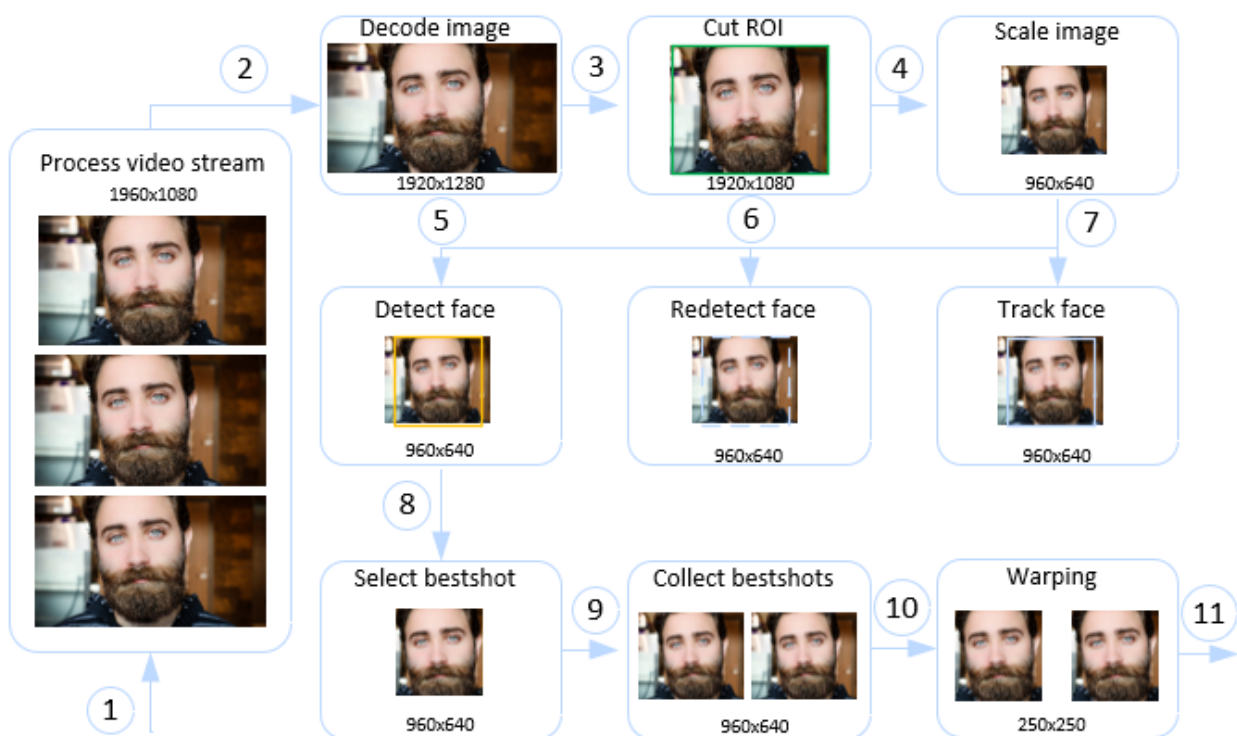


Figure 1: FaceStream workflow with faces

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 are set by sending requests with the necessary parameters to the [LUNA Streams](#) service;
2. FaceStream decodes video frames;
3. The ROI area is cut out from the frame if the “roi” parameter is specified;
4. The received image is scaled to the “scale-result-size” size if the “detector-scaling” is set in the trackengine configuration;
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 bestshots. There are several algorithms for choosing the best detection(s) in the track. See the [“Filtering section”](#);
9. If the frame is bestshot, it is added to the collection of bestshots.

Depending on the [“number_of_bestshots_to_send”](#) setting one or several best detections are collected from each track;

10. **Optional.** If the [“warp”](#) type is set in the [“portrait_type”](#) parameter, the bestshots are normalized to the LUNA PLATFORM standard, and normalized images are created. Normalized image is better for processing using LUNA PLATFORM;
11. The bestshots are sent to an external service via HTTP-request. The image may be sent as it is or transformed into the normalized image.

The frequency of images sending is specified in the [“sending”](#) (streams management configuration) section.

The sending parameters and external service address are specified in sections [“data”](#) (streams management configuration) and [“sending”](#) (FaceStream configuration).

4.1.2 FaceStream workflow with bodies

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

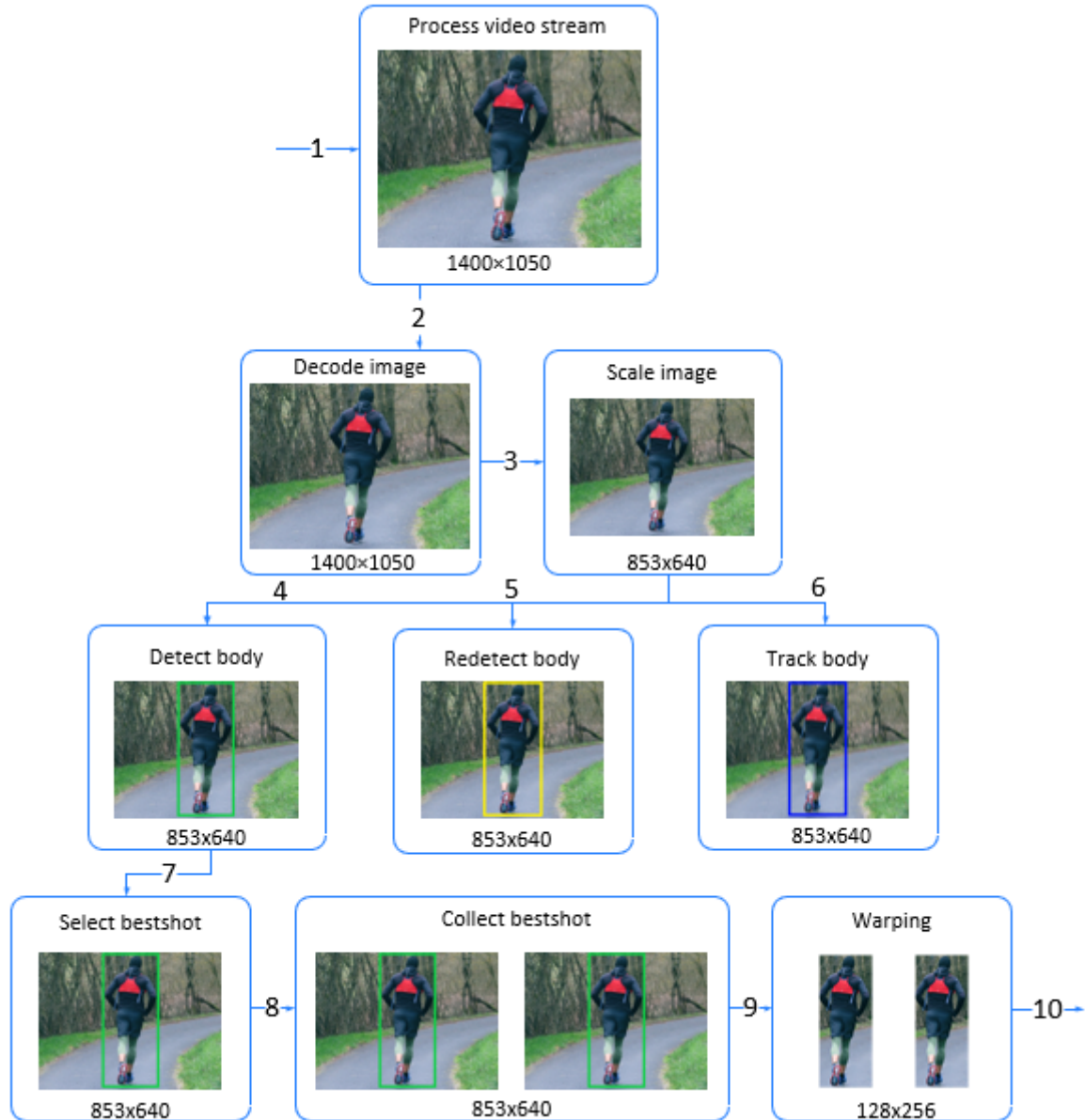


Figure 2: FaceStream workflow with bodies

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 are set by sending requests with the necessary parameters to the [LUNA Streams](#) service;
2. FaceStream decodes video frames;
3. The received image is scaled to the “scale-result-size” size if the “detector-scaling” is set in the

Trackengine configuration;

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 bestshots. See “[Min-score](#)”;
8. If the frame is bestshot, it is added to the collection of bestshots.

Depending on the “[number_of_bestshots_to_send](#)” setting one or several best detections are collected from each track;

9. The bestshots are normalized to the LUNA PLATFORM standard, and normalized images are created. Normalized image is better for processing using LUNA PLATFORM;
10. The bestshots are sent to an external service via HTTP-request. Events can be generated in an external service according to the specified handler (see the description of the event in the LUNA PLATFORM administrator manual). The bestshots are transformed into warps. Along with the bestshots, the coordinates of the human body can be sent if parameter “[send_detection_path](#)” is enabled.

The frequency of images sending is specified in the “[sending](#)” (stream management configuration) section.

The sending parameters and external service address are specified in sections “[data](#)” (stream management configuration) and “[sending](#)” (FaceStream configuration).

4.2 Interaction of FaceStream with LUNA Streams

To work with FaceStream, you should first launch an additional service - LUNA Streams. The service sets [settings for stream management](#) and passes it to FaceStream for further processing. The default service port is 5160.

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.

The general process of interaction between FaceStream and LUNA Streams is presented below:

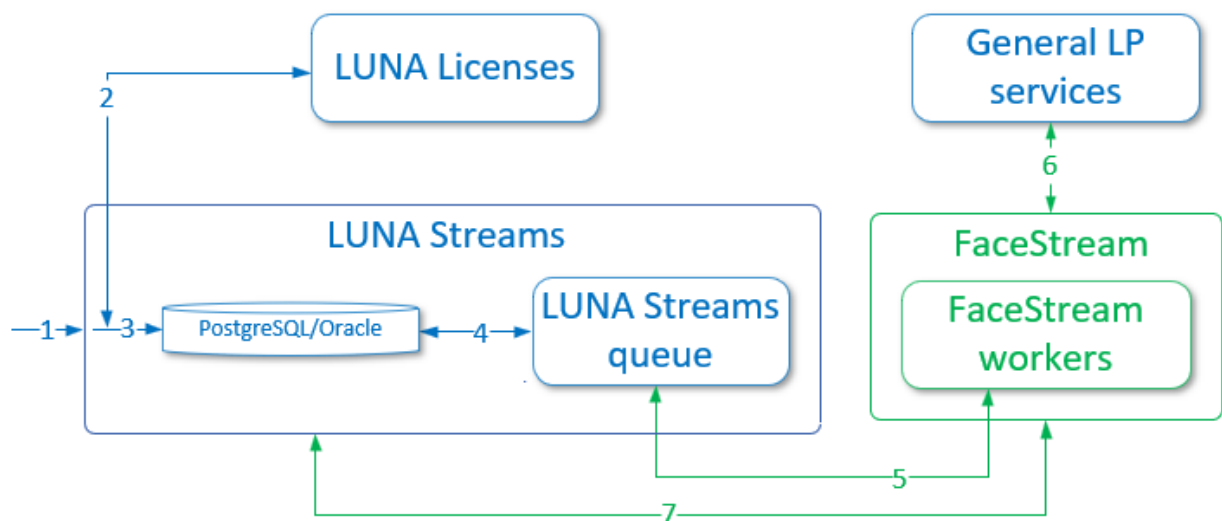


Figure 3: Interaction between FaceStream and LUNA Streams

After sending an HTTP request with the specified parameters to the LUNA Streams **(1)** service, the presence of a parameter regulating the number of streams for LUNA Streams operation is checked at

the LUNA PLATFORM key using the LUNA Licenses **(2)** service. The number of streams already being processed at the time of the request is also checked using the FaceStream report **(7)** (see below).

If the key parameter is missing, a license error will be issued.

If at the time of stream creation the maximum number of available streams is not processed yet, the parameters are added to the LUNA Streams database **(3)** under the unique identifier `stream_id`. The stream with the parameters gets into the queue **(4)**, where it is in the [status](#) “pending” until a special FaceStream worker picks up the stream from the queue for subsequent processing.

If the maximum amount is already being processed at the time of stream creation, LUNA Streams will not be able to add parameters to the database and a license error will be issued.

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.

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”.

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

Next, FaceStream workers take the parameters of the stream(s) from the queue **(5)** with the status “pending” and begin processing. In this case, the status of the processed streams is changed to “in_progress” and the stream is removed from the queue.

During processing, data is regularly sent to the main services of LUNA PLATFORM 5 for further processing of frames according to the specified [handler_id](#) and for creating events **(6)**, and a report on processing streams in LUNA Streams **(7)** is regularly sent.

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

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 **(5)**, 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.

For more detailed description of LUNA Streams stream processing, see [“Stream processing pipeline”](#).

[Settings for stream management](#) are set using a POST request to the [“/streams”](#) resource.

In addition, the following actions are available for a stream:

- getting existing streams by their “stream_id” with a description of the data of each stream ([“get streams”](#) request)

- getting all information about a stream by its “stream_id”, incl. sizes and frame rate, bitrate, group of frames (gop), creation time, stream processing start time, last processing error, etc. (“[get stream](#)” request)
- deleting existing streams by their “stream_id” (“[delete streams](#)” request)
- deleting stream by its “stream_id” (“[remove stream](#)” request)
- getting the number of streams created (“[count streams](#)” request)
- updating the “description” and “status” fields of a stream by its “stream_id” (“[update stream](#)” request)
- replacement of all stream data with new ones by its “stream_id” (“[put stream](#)” request)

A detailed description of requests and example requests can be found in the Open API document “[StreamsReferenceManual.html](#)”.

4.2.1 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 handler
- in_progress - stream processing is in progress
- done - stream processing is completed (relevant for video files)
- pause - stream processing is paused by user (not applicable for video files)
- restart - stream processing is restarted by server
- cancel - stream processing is cancelled by user (relevant for video files, but it can also be used for other sources)
- failure - stream processing is failed by handler
- handler_lost - stream processing handler is lost, needs to be passed to another handler (not applicable for video files)
- not_found - stream was removed during the processing
- deleted - stream was removed intentionally

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 “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.

4.2.1.1 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).

	in_-								handler_-
-	pending	progress	done	restart	pause	cancel	failure	lost	
-	+				+				
pending	+	+		+	+	+			
in_- progress	+	+	+	+	+	+	+	+	
done	+	+		+	+				
restart		+			+				
pause	+	+		+		+			
cancel	+	+		+	+				
failure	+	+		+	+				
handler_- lost				+					

* not supported for video files

4.2.2 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 handler 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 handler, it is assigned to the status “done” in case of success, 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 handler: 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

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

For streams with a “failure” status that allows **automatic restart**, several restart attempts will be made:

- its status will be automatically changed to “restart” and then, to “pending”
- “current_attempt” parameter will be increased by 1
- “last_attempt_time” parameter will be actualized

The possibility of autorestart, the maximum number of restart attempts, the delay between attempts are specified by the user for each stream at “**autorestart**” section.

In this case, a successfully restarted stream will be considered a stream that, after a specified period of time (delay), will have a status other than “failure”.

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”.

4.2.3 LUNA Streams database description

The LUNA Streams database general schem is shown below.

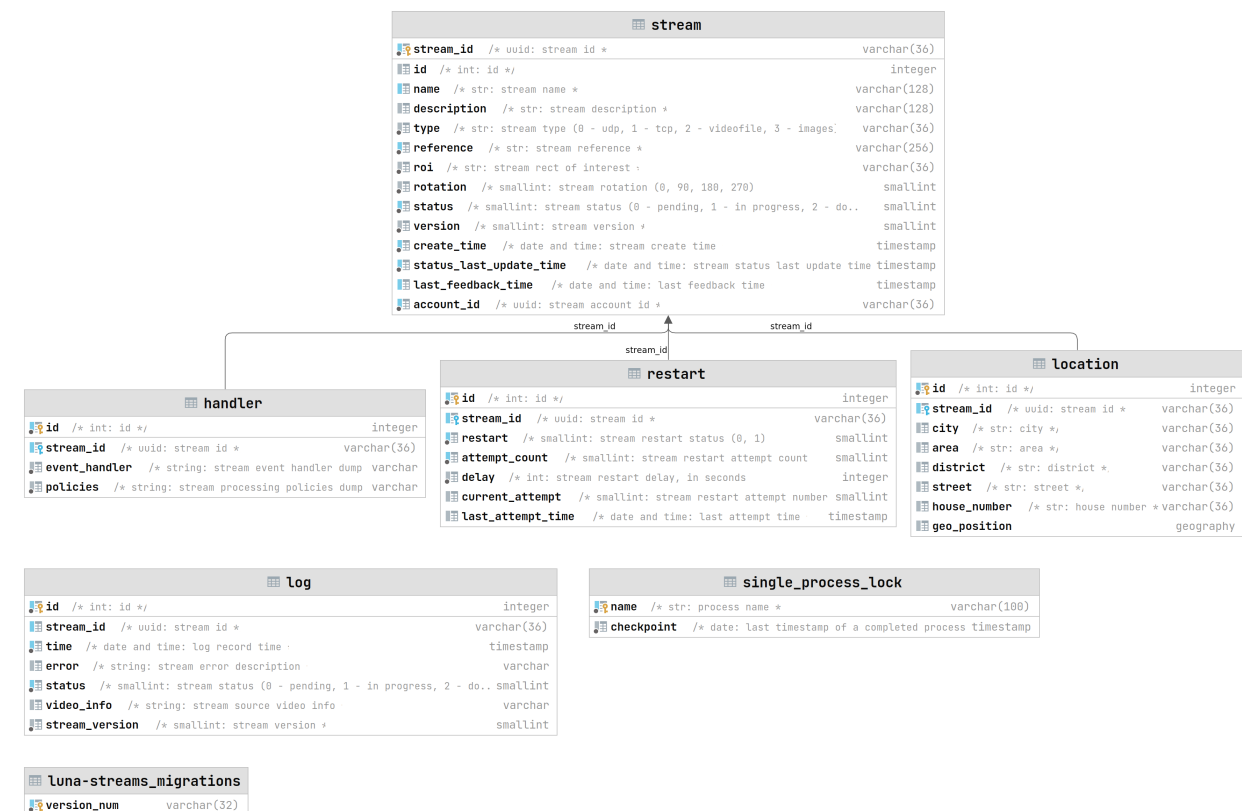


Figure 4: LUNA Streams database

See “[Streams management configuration](#)” for a description of the database data.

4.3 General 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.

4.3.1 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.

The video cameras tested with FaceStream are listed in section “[Appendix A: Cameras Compatibility](#)”.

4.3.2 FaceStream performance configuration

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

4.3.2.1 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 (stream 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.

4.3.2.2 Frame scaling

The “detector-scaling” option (Trackengine configuration) enables you to scale the frame before processing.

The appropriate frame size should be selected using the “scale-result-size” parameter (Trackengine configuration file). This parameter sets the maximum frame size after scaling the largest side of the frame. 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 7: Detections in image 1960X1080

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.

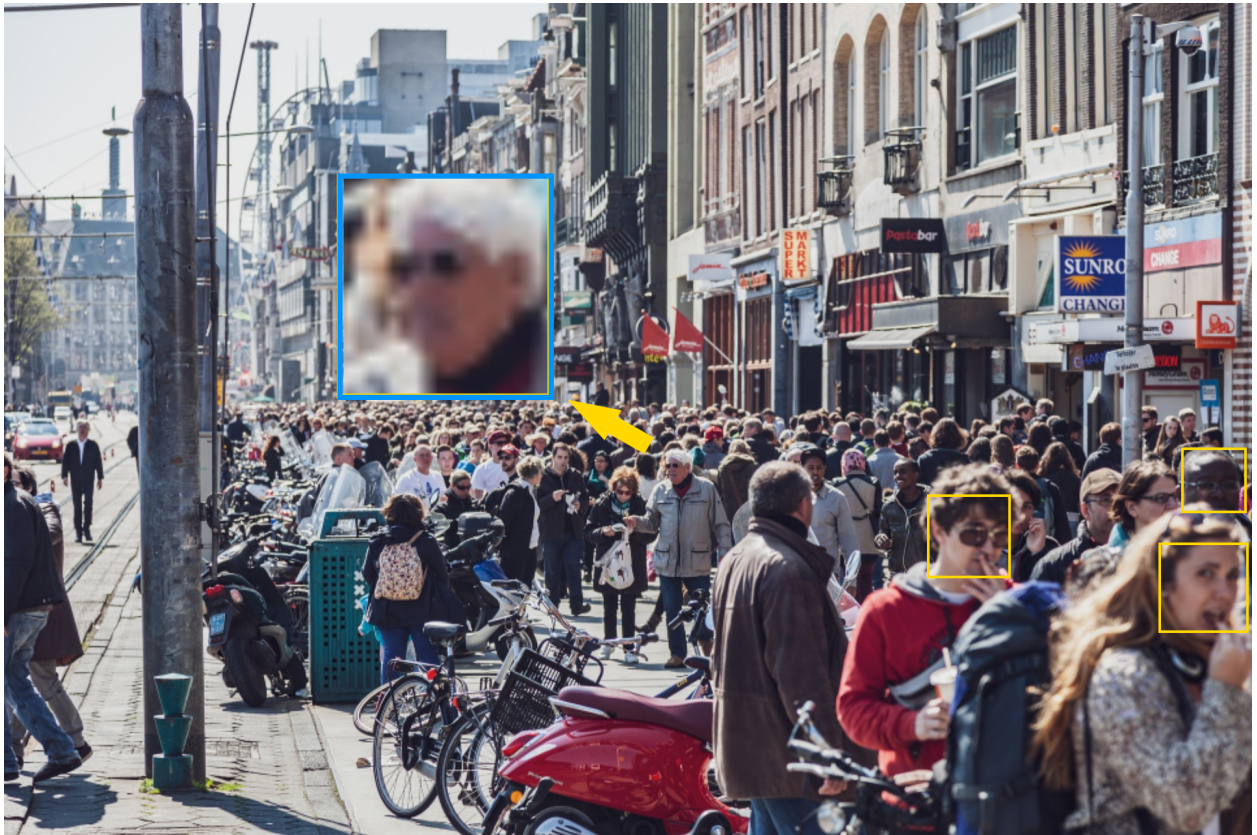


Figure 8: Detections in image 960X640

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

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

4.3.3 Defining area with movement

	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

When the “frg-subtractor” parameter (Trackengine configuration) 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.

The “frg-regions-alignment” parameter (trackengine.conf) enables you to set the alignment for the area with motion.

When the “frg-regions-square-alignment” parameter (Trackengine configuration) is enabled, the width and height of the area with motion will always be equal.

4.3.4 Batch processing of frames

The following parameters configure frames batches processing. The parameters are set in Trackengine settings.

The “batched-processing” 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.

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.

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.

4.3.4.1 Minimal face size

You should configure the “minFaceSize” parameter in the Faceengine configuration file to specify the minimal face size for detection.

You should set the maximum possible face size. The larger the face, the fewer resources are required to perform detections.

Note that the face size will depend on the actual frame size set by the “scale-result-size” parameter (Trackengine configuration). A face with a size equal to 100 pixels on a 1280x760 frame will have a size equal to 50 pixels on a 640x480 frame.

4.3.5 General configuration information

4.3.5.1 Working with track

A new track is created for each detected face or body. Bestshots 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.

The “detector-step” parameter in “trackengine.conf” 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.

4.3.5.2 Bestshot sending

The “sending” parameters group (stream management configuration) enables you to set parameters for the bestshot sending. FaceStream sends the received bestshots to LUNA PLATFORM (see “[Settings for sending images to LUNA PLATFORM](#)”).

You can send several bestshots for the same face or body to increase the recognition accuracy. You should enable the “[number_of_bestshots_to_send](#)” (stream management configuration) parameters in this case.

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

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

The “[time_period_of_searching](#)” and “[silent_period](#)” parameters can be specified in seconds or in frames. Use the “[type](#)” parameter to choose the type.

The general options for configuring the “[time_period_of_searching](#)” and “[silent_period](#)” parameters of the “sending” group from streams management configuration are listed below.

1. The bestshot 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 bestshot is selected.

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

2. It is required to quickly receive the bestshot and then send bestshots with the specified frequency.

For example, it is required to send a bestshot 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 bestshot will be sent after the track is interrupted even if the specified period did not exceed.

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

3. It is required to quickly send the bestshot and then send the bestshot only if the person is in the frame for a long time.

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

4. It is required to quickly send the bestshot and never send the bestshot from this track again.

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

4.3.5.3 Frames filtration

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

- Head angles (“[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](#)”);
- Mouth occlusion (“[mouth_occlusion_threshold](#)”).

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

If the “[number_of_bestshots_to_send](#)” parameter is set, the frame is added to the array of bestshots to send. If the required number of bestshots to send was already collected, the one with the lowest frame quality score is replaced with the new bestshot if its quality is higher.

The filtration of body frames is performed only by one criterion - “[min_score](#)”.

4.3.5.4 Working with ACMS

Work with ACS 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. Bestshots 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 bestshots 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 bestshots. When the face detection vertical size reached the value set in the “[best_shot_proper_size](#)” parameter the bestshot is sent as a bestshot 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.

1. 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
```

2. 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 bestshot is never sent when the track is interrupted.

4.4 Additional information

4.4.1 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.

4.4.1.1 Video formats

Video formats that are processed using FaceStream:

- AVI,
- MP4,
- MOV,
- MKV,
- FLV.

4.4.1.2 Encodings

Basic video compression standards that FaceStream works with:

- MPEG4,
- MS MPEG4,
- MS MPEG4v2,
- MJPEG,
- H.264,
- H.265.

4.4.1.3 Protocols

Basic protocols used by FaceStream for data receiving:

- HTTP,
- RTP,
- RTSP,
- TCP,
- HLS,
- UDP.

4.4.2 Memory consumption when running FaceStream

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

1. Each stream increases memory consumption. The amount of the consumed memory depends on the settings set for FaceStream:
 - the number of Ffmpeg threads in the `“ffmpeg_threads_number”` parameter (stream 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).
2. If the number of threads specified in the `“ffmpeg_threads_number”` parameter is greater than `“1”` (stream 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”` (stream management configuration). In this case, memory growth is not noticed.
3. 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.
4. Memory consumption increases when settings in the Debug section are enabled (FaceStream and Trackengine configurations).

4.4.3 Monitoring

Monitoring is implemented as sending data to the “[InfluxDB OSS 2](#)”. Monitoring is enabled in LUNA PLATFORM services by default, but can be disabled.

Monitoring is performed only for LUNA PLATFORM services. For FaceStream monitoring is not used.

There are two types of events that are monitored: *request* (all requests) and *error* (failed requests only).

Every event is a point in the time series. The point is represented using the following data:

- series name (*requests* or *errors*)
- timestamp of the request start
- tags
- fields

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.

See the LUNA PLATFORM administrator manual for more information.

4.4.4 InfluxDB OSS 2

For InfluxDB OSS 2 usage, you should:

- Install the DB. See the “InfluxDB OSS 2 container launch” in the installation manual.
- Register in the DB. InfluxDB has a user interface where you can register. You should visit `<server_ip>:<influx_port>`.
- Configure the display of monitoring information in the GUI. It is not described in this documentation.

4.4.4.1 InfluxDB configuration

The settings for InfluxDB are described below.

Table 4: InfluxDB settings

Setting name	Type	Description
send_data_for_monitoring	integer	Enables monitoring for the service.

Setting name	Type	Description
use_ssl	integer	Enables HTTPS protocol usage for connection to InfluxDB (0 – do not use, 1 – use).
flushing_period	integer	The frequency of sending monitoring data to InfluxDB.
port	integer	InfluxDB port.
host	integer	InfluxDB host.
organization	String	The organization name specified during registration.
token	String	Token received after registration.
bucket	String	Bucket name.

5 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
event_handler > frame_store	Enables you to set the URL of the Image Store service to send the source frame

5.1 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
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 bestshots 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
event_handler > origin	full network path to LP 5 - “http://<luna_api_adress>:5000/”. The example specifies the port “5000” for the API service, which is used by default
event_handler > api_version	The version of the API of the LP API service - 6

Parameter	Description
event_handler > bestshot_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.

5.2 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 > request_type	Request type for sending images to LP - jpeg
sending > aggregate_attr_requests	Enables aggregation of the bestshots to get a single descriptor in LUNA PLATFORM - true or false
sending > send_detection_path	Enables, along with the bestshots, to send a certain number of detections with the coordinates of the human body - x, y, width and height. The number of detections is determined by the parameters below - “true” или “false”

Parameter	Description
<code>sending > detection_path_length</code>	This parameter sets the maximum number of detections for the “send_detection_path” parameter. - “true” или “false”
<code>sending > minimal_body_track_length_to_send</code>	This parameter sets the number of detections for the “send_detection_path” parameter, less than the value of which they will not be sent. - “true” или “false”

Table 12: Streams management parameters

Parameter	Description
<code>data > type</code>	Type of signal source (tcp, udp, videofile, images) - string
<code>account_id</code>	“Luna_account_id”, to which the request is related - string in UUID format
<code>event_handler > origin</code>	full network path to LP 5 - “http://<luna_api_adress>:5000/”. The example specifies the port “5000” for the API service, which is used by default
<code>event_handler > api_version</code>	The version of the API of the LP API service - 6
<code>event_handler > bestshot_handler > handler_id</code>	LP handler that enables you to flexibly configure the processing of bodies - string in UUID format
<code>event_handler > detection_handler > handler_id</code>	dynamic LP handler that enables you to attach body coordinates to an event - string in UUID format

For detailed information about the handlers, see the documentation [APIReferenceManual.html](#) included in the LUNA PLATFORM 5 distribution package.

6 FaceStream configuration

Settings configuration is performed by one of the following ways:

- by editing parameters in the “FACE_STREAM_CONFIG” configuration in the Configurator service (see [“Use FaceStream with Configurator”](#));
- 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.

6.1 Logging section

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

6.1.1 Severity parameter

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.

```
"severity":  
{  
  "value": 1,  
  "description": "Logging severity levels ... "  
}
```

6.1.2 Tags parameter

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.

Table 13: Tags description

Tag	Description
streams	Information about LUNA Streams operation
common	General information
ffmpeg	Information about FFMPEG library operation
gstreamer	Information about GStream library operation

Tag	Description
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
image	Information about frames processing
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

```

"tags" : {
  "value" : ["common", "ffmpeg", "gstreamer", "bestshot", "primary-track",
    "image", "http_api", "client", "json", "streams"],
  "description" : "Logging specificity tags, full set: [streams, common,
    ffmpeg, gstreamer, liveness, primary-track, bestshot, angles, ags,
    mouth-occlusion, statistics, image, http_api, client, json, debug]"
},

```

6.1.3 Mode parameter

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.

```
"mode":  
{  
  "value": "l2b",  
  "description": " The mode of logging ... "  
}
```

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.

6.2 Sending section

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

6.2.1 Request_type section

Request_type is 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 or VisionLabs FaceStreamManager for further image processing.

```
"request_type":  
{  
  "value": "jpeg",  
  "description": " Type of request to server with portrait ..."  
},
```

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

Table 14: Request types

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

6.2.2 Portrait_type parameter

This parameter is used only for working with faces.

Portrait_type 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.

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:

- a 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.

```
"portrait_type":  
{  
  "value": "warp",  
  "description": "Image format type..."  
}
```

6.2.3 Send_source_frame parameter

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

When sending image to LUNA PLATFORM you should specify the URL of LUNA Image Store service in the “frame_store” parameter.

```
"send_source_frame" :  
{  
  "value": false,  
  "description": "Send source frame for portrait from stream (false by  
    default)."  
}
```

6.2.4 Send_detection_path parameter

This parameter is used only for working with bodies.

The “send_detection_path” parameter enables you, along with the bestshots, to send a certain number of detections with the coordinates of the human body - x, y, width and height (see the “save event” request in the OpenAPI LUNA PLATFORM document). The maximum number of sent detections is regulated by the “detection_path_length” parameter, the minimum - by the “minimal_body_track_length_to_send” parameter.

The parameter should be used in conjunction with the [detection handler > handler_id](#) parameter. When this parameter is enabled, in addition to generating the general event, one more event will be created, associated with the general one by “track_id”. This event will contain only the coordinates of the human body.

```
"send_detection_path" : {  
  "value" : false,  
  "description" : "Send detection path for Luna api version 6 or higher ('  
    false' by default)."  
}
```

6.2.5 Detection_path_length parameter

This parameter is used only for working with bodies.

This parameter sets the maximum number of detections for the “send_detection_path” parameter. Values from 1 to 100 inclusive are available.

```
"detection_path_length" : {  
  "value" : 100,  
  "description" : "Maximum length of detection path allowed for sending to  
    Luna ('100' by default)."  
}
```

6.2.6 Minimal_body_track_length_to_send parameter

This parameter is used only for working with bodies.

This parameter sets the number of detections for the send_detection_path parameter, less than the value of which they will not be sent. Values from 1 to 100 inclusive are available.


```
"minimal_body_track_length_to_send" : {  
  "value": 3,  
  "description" : "Minimal body track length to send"  
}
```

6.2.7 Async_requests parameter

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.

```
"async_requests" : {  
  "value" : true,  
  "description" : "Asynchronous requests to Luna server (true by default)."  
},
```

6.2.8 Aggregate_attr_requests parameter

The “aggregate_attr_requests” parameter enables the bestshots aggregation to receive a single descriptor in LUNA PLATFORM.

Aggregation is performed if there is more than one bestshot sent. The number of frames to send is set by the “number_of_bestshots_to_send” parameter.

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

```
"aggregate_attr_requests" :  
{  
  "value" : true,  
  "description" : "Set aggregate attributes in request to luna api 6 if  
    there are more than one bestshot (true by default)."  
},
```

6.2.9 Jpeg_quality_level parameter

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.

```
"jpeg_quality_level" : {  
  "value" : "best",  
  "description" : "Level of jpeg quality for source frames ['best', 'good',  
    , 'normal', 'average', 'bad'] ('best' by default)."  
}
```

6.3 Lunastreams section

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.

6.3.1 Origin parameter

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

```
"origin": {  
  "value": "http://127.0.0.1:5160",  
  "description": "LunaStreams url address."  
}
```

6.3.2 Api_version parameter

The parameter specifies the API version of the LUNA Streams service. At the moment, the API version “1” is supported.

```
"api_version": {  
  "value": 1,  
  "description": "Api version."  
}
```

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

6.3.3 Max_number_streams parameter

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

```
"max_number_streams": {  
  "value": 50,  
  "description": "Upper bound on the number of streams FS processes. Value  
    must be greater than 0."  
}
```

6.3.4 Request_stream_period parameter

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.

```
"request_stream_period": {  
  "value": 1.0,  
  "description": "Time period for requesting new streams from LUNA Streams  
    . Available range [0.1, 3600]. Default value 1 second."
```

6.3.5 Send_feedback_period parameter

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.

```
"send_feedback_period": {  
  "value": 5.0,  
  "description": "Time period for sending report of streams. Available  
    range [1.0, 3600]. Default value 5 seconds. Must not be larger than  
    STREAM_STATUS_OBSOLETING_PERIOD in LUNA Streams."  
}
```

6.3.6 Max_feedback_delay parameter

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”] (#send_feedback_period) and should not exceed the value of the parameter [“STREAM_STATUS_OBSOLETING_PERIOD”](#), set in the LUNA Streams service settings.

```
"max_feedback_delay": {  
  "value" : 10.0,
```

```
"description": Max feedback sending delay after which processing streams  
stops. Available range [1.0, 3600]. Default value 10 seconds. Must  
not be less than send_feedback_period and larger than  
STREAM_STATUS_OBSOLETING_PERIOD in LUNA Streams."
```

```
}
```

6.4 Performance section

6.4.1 Stream_images_buffer_max_size parameter

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.

```
"stream_images_buffer_max_size" : {  
  "value" : 40,  
  "description" : "Max images buffer size for a single stream. Higher value  
    provides better performance, but increases memory consumption. When  
    set to 0 buffer is not used. (40 by default)"  
}
```

6.4.2 Enable_gpu_processing parameter

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”).

```
"enable_gpu_processing" : {  
  "value" : false,  
  "description" : "When 'true' the processing is performed using  
    GPU instead of CPU. GPU could provide better performance, but  
    increases memory consumption. ('false' by default)"  
},
```

6.4.3 Convert_full_frame parameter

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.

```
"convert_full_frame" : {  
    "value" : true,  
    "description" : "Enables converting full raw frame from decoder  
        to rgb for processing. If value is 'true', then better  
        quality is achieved. 'false' value provides better performance  
        . ('true' by default)"  
}
```

6.5 Debug section

This section is used to configure and debug the application. Settings of this section are not recommended for use in industrial environment, since they consume significant resources and negatively affect performance.

6.5.1 Save_debug_info parameter

Save_debug_info parameter makes it possible to save information about the detector operation and recognition results. If the value is “true”, then the information is saved and used for debugging purposes to analyze the quality of the system.

```
"save_debug_info" :  
{  
  "value": false,  
  "description": "Save information for quality analysis ..."  
}
```

6.5.2 Save_only_jpegs_with_honest_detections parameter

Parameter enables saving only the frames with detected faces. Parameter is used for debugging purposes in frame-by-frame analysis.

This setting can significantly save hard disk space if faces rarely appear in the frame.

```
"save_only_jpegs_with_honest_detections" :  
{  
  "value": false,  
  "description": "Filter for save_jpegs flag to save only jpegs with  
    honest detections ('false' by default)."  
},
```

6.5.3 Save_jpegs parameter

Save-jpeg flag is used to save frames received for processing in the application. The parameter is used for debugging purposes for repeated frame-by-frame analysis.

Saved frames from the original stream may require considerable space on the hard disk.

```
"save_jpegs" :  
{
```



```
"value": false,  
"description": "Save jpegs for research visualization ..."  
}
```

7 Streams management configuration

The application supports simultaneous work with several stream sources.

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.

Several types of sources are supported:

- tcp, udp - real-time video signal sources. These can be both USB cameras and IP cameras (via RTSP protocol);
- videofile - video files;
- images – a set of frames in the form of separate image files.

Note that the 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 configuration”](#).

7.1 Account_id parameter

The parameter specifies the mandatory “account_id” field, which is passed to LUNA PLATFORM 5 service API in the request header.

Account ID is set in the UUID4 format. You can find the requirements for the Account ID in the LUNA PLATFORM 5 documentation.

The parameter is used to bind the received data to a specific user.

```
"account_id" : {  
  "value" : "aaba1111-2111-4111-a7a7-5caf86621b5a",  
  "description" : "Luna_account_id header value for Luna api version 6 or  
    higher (' ' by default)."  
}
```

7.2 Name parameter

Source identification. It is used to identify the source of the sent frames.

```
"name": "stream_0",
```

When sending to LUNA PLATFORM 5, the “name” value and the track ID are used to generate the external_id field.

This field has a size limit of 36 characters. Due to this, you must the size of the “name” parameter **to twenty characters**.

If characters are passed in the string (for example, “+”), then additional encoding of the string will be required. By default, the string is not encoded when passed to the LUNA PLATFORM, so characters that require encoding are lost.

7.3 Description parameter

User description of the stream.

```
"description": "Stream on Arbat street",
```

7.4 Data section

The general parameters required to configure stream are listed below.

7.4.1 Type parameter

The protocol of stream transmission. The application can use one of two network protocols to receive video data: **TCP** or **UDP**, and also process a set of frames as separate image files (**images**) or process a video file (**videofile**).

```
"type": "tcp",
```

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.

7.4.2 Reference parameter

Full path to the source or a USB-device number (for “tcp”/“udp” type).

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

Full path to the video file (for “videofile” type).

```
"reference": https://127.0.0.1:0000/super_server/
```

Full path to the directory with the images.

```
"reference": "/example1/path/to/images/"
```

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

7.4.3 Roi parameter

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.

Region of interest on the original frame is set in pixels as array **[x, y width, height]**, where (x, y) are the coordinates of the upper-left point of the region of interest. Coordinate system is set in the same way as it is shown on the picture below.

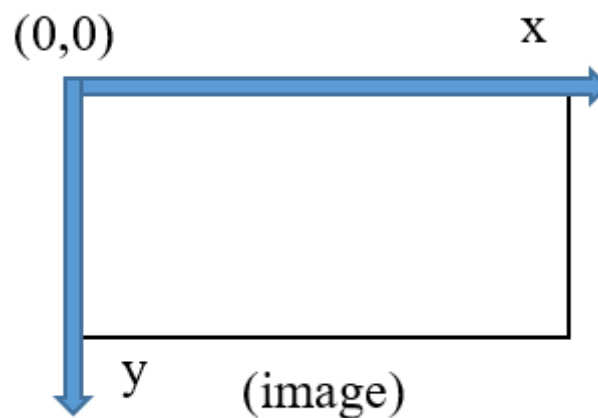


Figure 9: ROI coordinate system

When the values of width and height are set to “0”, the entire frame will be the region of interest.

```
"roi": [0, 0, 0, 0],
```

7.4.4 Droiparameter

This parameter is used only for working with faces.

The parameter specifies the region of interest within the ROI zone. Face detection is performed in ROI, but the best shot is selected only in the DROI area. Face detection must be completely within the DROI zone for the frame to be considered as a best shot.

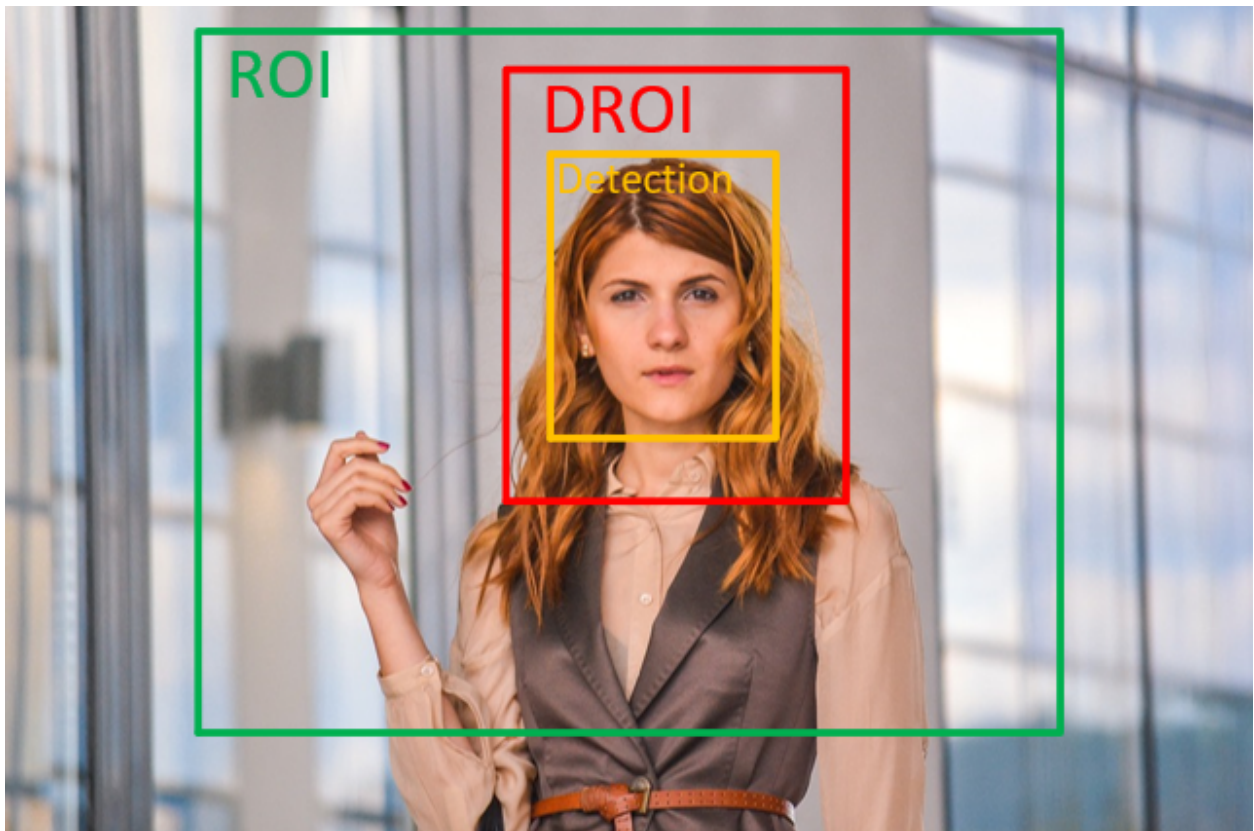


Figure 10: DROI

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.

Region of interest on the original frame is set in pixels as array **[x, y width, height]**, where (x, y) are the coordinates of the upper-left point of the ROI.

Coordinate system is set in the same way as it is shown in the picture below.

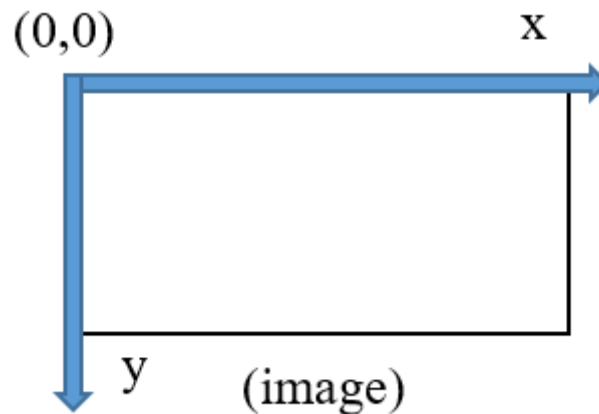


Figure 11: DROI coordinate system

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 frame.

```
"droi": [0, 0, 0, 0],
```

7.4.5 Rotation parameter

The rotation angle of the image source. It is used when the incoming stream is rotated, for example, if the camera is installed on the ceiling.

```
"rotation": 0,
```

7.4.6 Preferred_program_stream_frame_width parameter

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

The default value is 800.

```
"preferred_program_stream_frame_width": 800
```

7.4.7 Mask parameter

This parameter is used only for the **images** type and is a mandatory parameter.

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"
```


7.5 Event_handler section

This section defines the parameters of the handler created in the LUNA PLATFORM, with which video streams will be processed. Different handlers should be used for the face and bodies. Handler should be created in LP 5 in advance.

For more information about handlers, see the LUNA PLATFORM administrator manual.

7.5.1 Origin parameter

A full network path for sending images from this source.

```
"origin": "http://<luna_address>:<port>/"
```

<luna_address> - the address of the server with LUNA PLATFORM API service,

<port> - the port used by the LUNA PLATFORM API service. The default port is 5000.

7.5.2 Api_version parameter

The API version for generating events in the LUNA PLATFORM. Currently, version 6 of the API is supported.

7.5.3 Handler_id parameter of bestshot_handler section

The parameter enables you to use the external “handler_id” LUNA PLATFORM to process face or body samples according to the specified rules. When using this handler, LUNA PLATFORM generates an event that contains all the information received from FaceStream and processes it in accordance with the processing rules.

```
"handler_id": "aaba1111-2111-4111-a7a7-5caf86621b5a",
```

Handler should be created in LP 5 in advance.

7.5.4 Handler_id parameter of detection_handler section

The parameter enables you to use the external dynamic “handler_id” LUNA PLATFORM for working **with bodies**. This parameter is used in conjunction with the [send_detection_path](#) parameter and their interaction enables you to send detections along with the bestshots. These detections will contain exclusively the coordinates x, y, width and height of the human body (see the save event request in the OpenAPI LUNA PLATFORM document).

Handler should be created in LP 5 in advance.

```
"handler_id": "aaba1111-2111-4111-a7a7-5caf86621b5a",
```

This parameter is optional and is required only if it is necessary to detect the coordinates of the human body.

For more information about dynamic handlers, see the LUNA PLATFORM administrator manual.

7.5.5 Frame_store parameter

The parameter specifies the LUNA Image Store service URL for the source frame sending.

The [send_source_frame](#) option should be enabled for sending source frames.

The source frame is sent to the LUNA Handlers service for processing, as well as to the LUNA Image Store for storage.

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

127.0.0.1 - IP address of the Image Store service.

5020 - the default Image Store service port.

<frames> - the name of the LUNA Image Store bucket where the source image 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
```

7.6 Policies section

7.6.1 Sending section

Sending – object defines a period within a track during which the analysis for best shot is conducted. In addition, it allows to define parameters connected to forming bestshot collection.

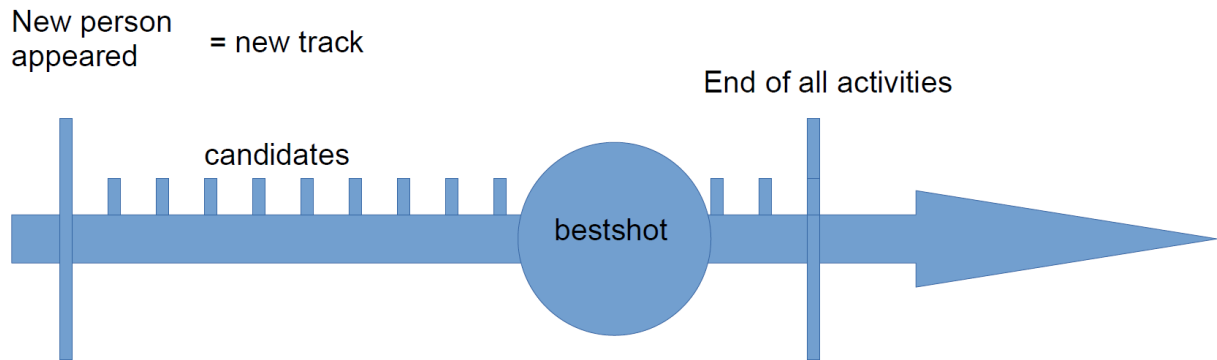


Figure 12: Best bestshot

```
"sending": {  
  "time_period_of_searching": -1,  
  "silent_period" : 0,  
  "type" : "sec",  
  "number_of_bestshots_to_send": 1  
},
```

7.6.1.1 Time_period_of_searching parameter

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.

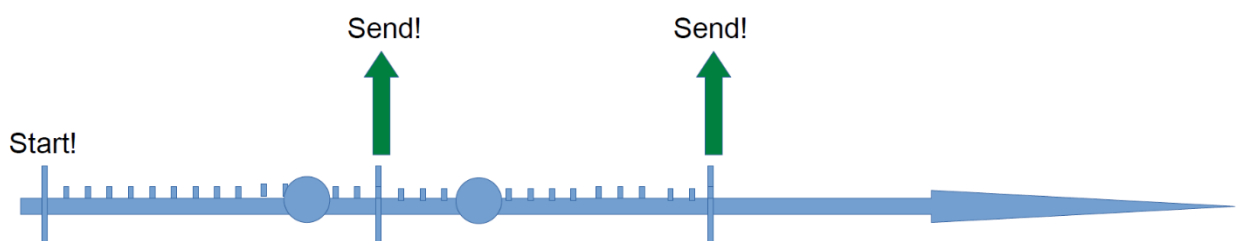


Figure 13: Sending period

If the value equals “-1”, 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.

```
"time_period_of_searching": -1,
```

7.6.1.2 Silent_period parameter

Silent_period – interval between period. Once the analysis period is over, the system holds this silent_period before starting next period of frame analysis.

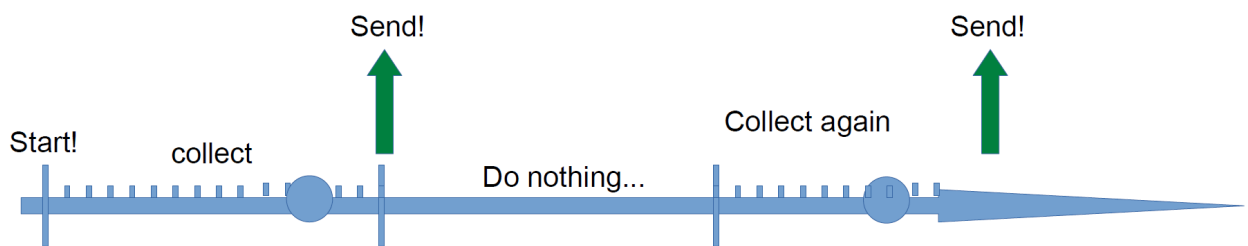


Figure 14: Silent period

If the value equals “-1”, system holds silent period indefinitely.

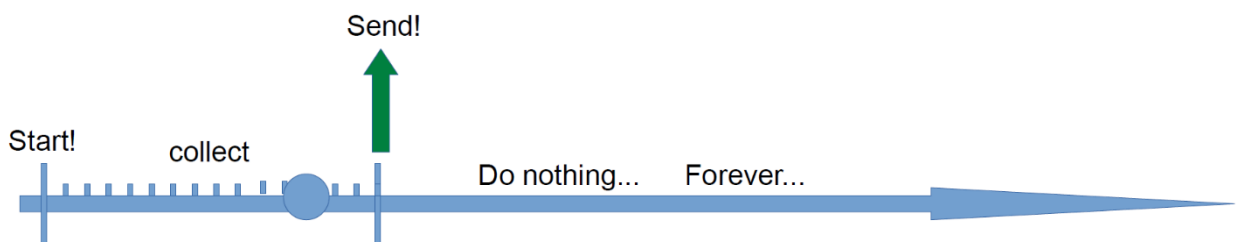


Figure 15: Endless waiting period

```
"silent_period" : 0,
```

7.6.1.3 Type parameter

Type – sets the measurement metric for analysis periods and intervals between those (“frames” or “sec”).

```
"type" : "sec",
```

7.6.1.4 [Number_of_bestshots_to_send](#) parameter

Number_of_bestshots_to_send – number of frames that the user sets to receive from the track or certain periods of this track. This parameter enables collection of best shots from a track or a certain period of a track set in parameter *time_period_of_searching-bd*. The collection of frames is then sent to the server.

FaceStreamManager can aggregate several warped images from a single track into a LUNA PLATFORM.

```
"number_of_bestshots_to_send": 1
```

Increasing parameter's value increases the probability of correct object recognition but affects the network load.

7.6.1.5 [Send_only_full_set](#) parameter

This parameter enables to send data only if the required amount of data is available, namely, "number_of_bestshots_to_send" and "minimal_body_track_length_to_send".

```
"send_only_full_set" : true
```

7.6.1.6 [Delete_track_after_sending](#) parameter

This parameter enables to delete track after sending.

```
"delete_track_after_sending" : false
```

7.6.2 Primary_track_policy section

This group of parameters is used only for working with faces.

This group of parameters 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 parameters group **is not used** for the “images” type.

7.6.2.1 Use_primary_track_policy parameter

Use_primary_track_policy parameter is used in cases of Access Control Systems (turnstiles/gates at the office/bank entrances) for easier control and face recognition implementation in a secured area.

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)

```
"use_primary_track_policy": false,
```

7.6.2.2 Best_shot_min_size parameter

The parameter is used when “use-primary-track-policy” parameter is enabled.

Best_shot_min_size parameter sets the minimal size of detection at which the analysis of frames and bestshot definition begins.

```
"best_shot_min_size": 70,
```

7.6.2.3 Best_shot_proper_size parameter

The parameter is used when “use-primary-track-policy” parameter is enabled.

Best_shot_proper_size – parameter sets the size of detection for Primary Track policy. When a detection reaches the defined value, track immediately sends all its best shots to the server.

```
"best_shot_proper_size": 140
```

7.6.3 Liveness section

This group of parameters 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.

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

The parameters group is not used for the “images” type.

7.6.3.1 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.

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.

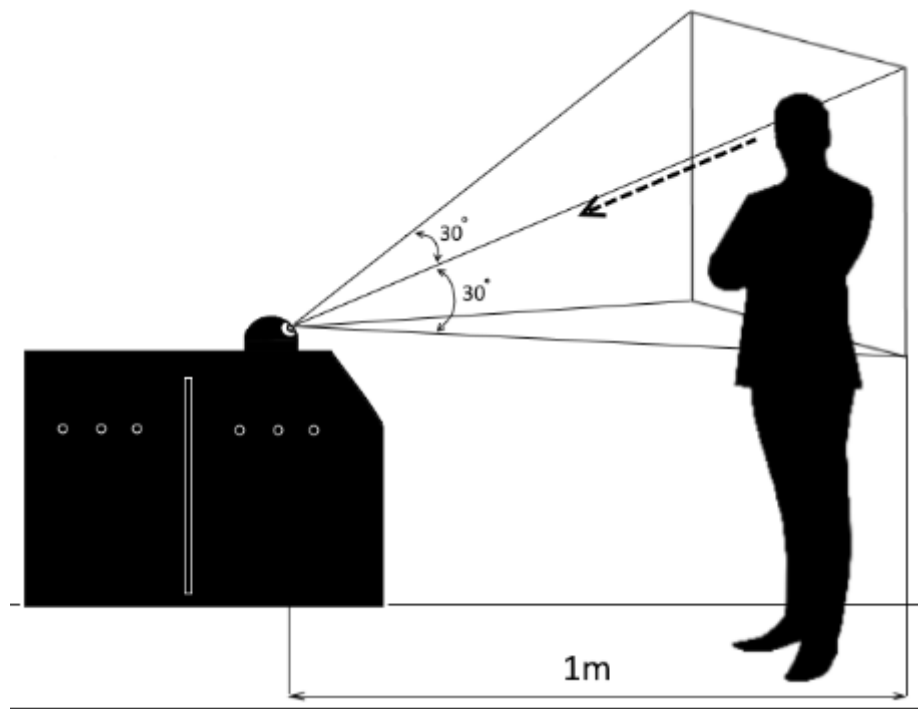


Figure 16: Proper camera placement for Liveness

FS starts collecting frames and selecting the bestshot 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 bestshot 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.

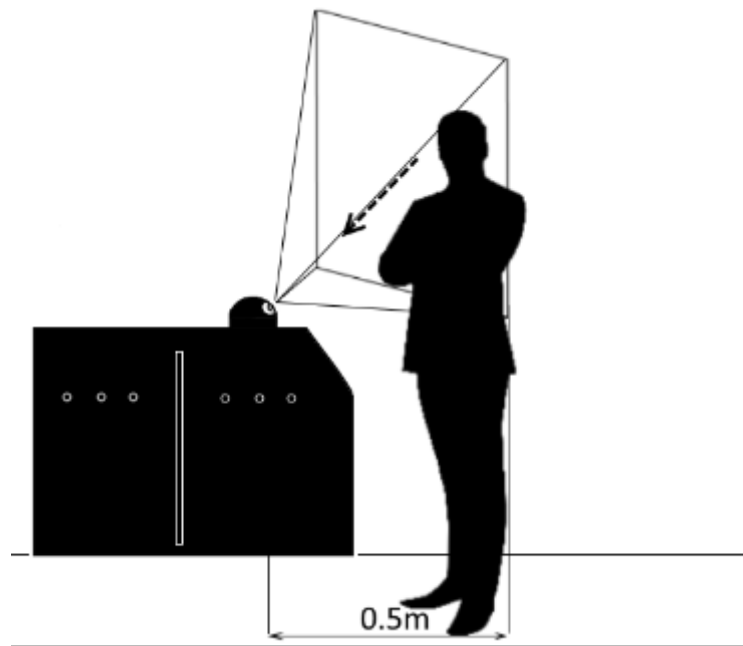


Figure 17: Inappropriate camera placement for Liveness

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.

Recommendations for configuring FS

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

```
"use_shoulders_liveness_filtration": true,  
"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": [0.05, 0.45, 0.5],  
"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.

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.

7.6.3.2 Use_shoulders_liveness_filtration parameter

The parameter enables checking the presence of a real person in the frame based on the head and shoulder areas.

```
"use_shoulders_liveness_filtration": false,
```

7.6.3.3 Use_mask_liveness_filtration parameter

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).

```
"use_mask_liveness_filtration": false,
```

7.6.3.4 Use_flying_faces_liveness_filtration parameter

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

```
"use_flying_faces_liveness_filtration": false,
```

7.6.3.5 Liveness_mode parameter

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.

```
"liveness_mode": 0,
```

7.6.3.6 Number_of_liveness_checks parameter

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

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

```
"number_of_liveness_checks": 10,
```

7.6.3.7 Liveness_threshold parameter

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”.

```
"liveness_threshold": 0.8,
```

7.6.3.8 Liveness_weights parameter

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

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

- Use_shoulders_liveness_filtration,
- Use_mask_liveness_filtration,
- Use_flying_faces_liveness_filtration.

In the example present (which is the system default) below 0,05 determines that 5% of liveness estimation will be based on shoulders_liveness, 0,45 - 45% on mask_liveness, and 0,5 - 50% on flying_faces_liveness.

The ratio is always calculated based on liveness_weights values, even if they don't add up to one, or not all liveness types are active.

```
"livenesses_weights": [0.05, 0.45, 0.5]
```

7.6.3.9 Mask_backgrounds_count parameter

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

Do not change this parameter.

```
"mask_backgrounds_count": 300
```

7.6.4 Filtering section

The section describes the filter object parameters and modes of sending the resulting portraits.

```
"filtering": {  
  "min_score": 0.5187,  
  "detection_yaw_threshold": 40,  
  "detection_pitch_threshold": 40,  
  "detection_roll_threshold": 30,  
  "yaw_number": 1,  
  "yaw_collection_mode": false,  
  "mouth_occlusion_threshold" : 0.0  
},
```

7.6.4.1 Min_score parameter

Min_score, also known as Approximate Garbage Score (AGS) for faces or Detector score for bodies – score that defines detection quality, threshold for filtering detections sent to the server. All detections with score higher than the value of this parameter can be sent to the server as HTTP-requests, otherwise detections are considered as not appropriate for further analysis.

If a new detection has a higher threshold than those in the existing collection, it will replace the detection with lowest threshold.

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

```
"min_score" : 0.5187,
```

7.6.4.2 Detection_yaw_threshold parameter

This parameter is used only for working with faces.

Detection_yaw_threshold 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.

```
"detection_yaw_threshold" : 40,
```

7.6.4.3 Detection_pitch_threshold parameter

This parameter is used only for working with faces.

Detection_pitch_threshold 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.

```
"detection_pitch_threshold" : 40,
```

7.6.4.4 *Detection_roll_threshold* parameter

This parameter is used only for working with faces.

Detection_roll_threshold 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.

```
"detection_roll_threshold" : 30,
```

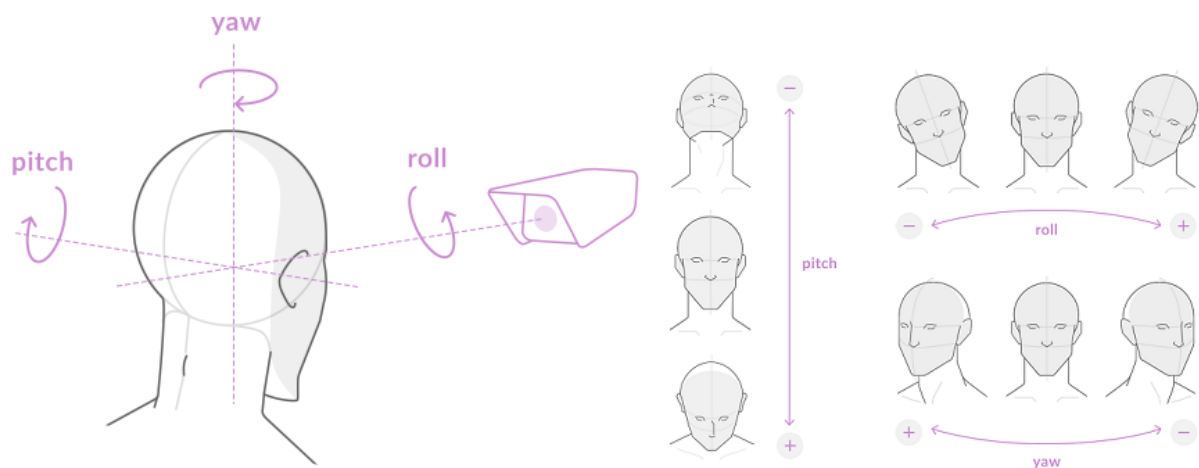


Figure 18: Head pose

7.6.4.5 *Yaw_number* parameter

This parameter is used only for working with faces.

Yaw_number parameter defines the number of frames for image filtration based on head tilt 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”.

```
"yaw_number": 1,
```

7.6.4.6 Yaw_collection_mode parameter

This parameter is used only for working with faces.

Yaw_collection_mode parameter sets the number of frames the system must collect to analyze head yaw angle. Best shot will be selected from those.

If *yaw_collection_mode* 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-mode parameter.

Parameter is disabled by default.

```
"yaw_collection_mode" : false,
```

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

7.6.4.7 Mouth_occlusion_threshold parameter

This parameter is used only for working with faces.

Mouth_occlusion_threshold 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 face 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 lower, the filtration is **disabled**.

```
"mouth_occlusion_threshold" : 0.0,
```

7.6.5 Frame_processing_mode parameter

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.

```
"frame_processing_mode": "auto",
```

7.6.6 Real_time_mode_fps parameter

This parameter is used for “videofiles” type only.

The video is processed with the specified number of FPS. The video can't be processed with FPS higher than the one specified in this parameter.

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”.

```
"real_time_mode_fps" : 0
```


7.6.7 Ffmpeg_threads_number

The parameter allows 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).

```
"ffmpeg_threads_number" : 0
```

7.6.8 Health_check section

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

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

7.6.8.1 Max_error_count parameter

The maximum number of errors that may occur during a video playback. For example, when it is unable to retrieve or decode a frame. Network problems or inaccessibility of a video can cause the errors.

```
"max_error_count": 10,
```

7.6.8.2 Period parameter

The parameter represents the period during which the number of errors is calculated. The value is set in seconds. An attempt to reconnect to the stream is performed when the number of errors, specified in the *max_error_count* parameter, is reached.

```
"period": 3600,
```

7.6.8.3 Retry_delay parameter

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

```
"retry_delay": 5
```

7.7 Location section

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

- “city”
- “area”
- “district”
- “street”
- “house_number”
- “geo_position” - latitude and longitude in degrees. Geo position is considered as properly specified if both longitude and latitude are set.

The `send_location_data` parameter enables the sending of location data of the video source.

```
"location": {  
  "send_location_data" : false,  
  "city": "Moscow",  
  "area": "CAO",  
  "district": "Arbat",  
  "street": "Arbat",  
  "house_number": "37",  
  "geo_position": {  
    "longitude": 36.616,  
    "latitude": 55.752  
  }  
}
```

This parameter is used to generate events in the LUNA PLATFORM (see the LUNA PLATFORM documentation).

7.8 Autorestart section

This group of parameters enables you to configure the automatic restart of the stream. Three parameters are available:

- `restart` - whether to use automatic restart of the stream
- `attempt_count` - number of attempts to automatically restart the thread (default 10)
- `delay` - stream automatic restart delay, in seconds (default 60 seconds)

```
"autorestart": {  
  "restart": 1,  
  "attempt_count": 7,  
  "delay": 600  
}
```

7.9 Status parameter

The status at the start of processing. Two states are available - “pending” and “pause”.

```
"status": "pending"
```

In addition to the two states at the start of processing, other states that occur during FaceStream operation are also available (see the [“Stream distribution in LUNA Streams”](#) section).

8 Trackengine configuration

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

Settings configuration is performed by one of the following ways:

- by editing parameters in the “TRACK_ENGINE_CONFIG” configuration in the Configurator service (see [“Use FaceStream with Configurator”](#));
- by editing the configuration file “trackengine.conf” in the mode of operation without the Configurator service (see [“Use FaceStream with configuration files”](#)).

8.1 Use-face-detector and use-body-detector

These parameters enable you to change the detection of faces to bodies and vice versa. Simultaneous detection of faces and bodies is not possible.

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.

8.2 Detector-step

The “detector-step” parameter in “trackengine.conf” 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" />
```

8.3 Detector-scaling

The “detector-scaling” option (Trackengine configuration) enables you to scale the frame before processing.

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

8.4 Scale-result-size

The appropriate frame size should be selected using the “scale-result-size” parameter (Trackengine configuration file). This parameter sets the maximum frame size after scaling the largest side of the frame. 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 1960X1080

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.

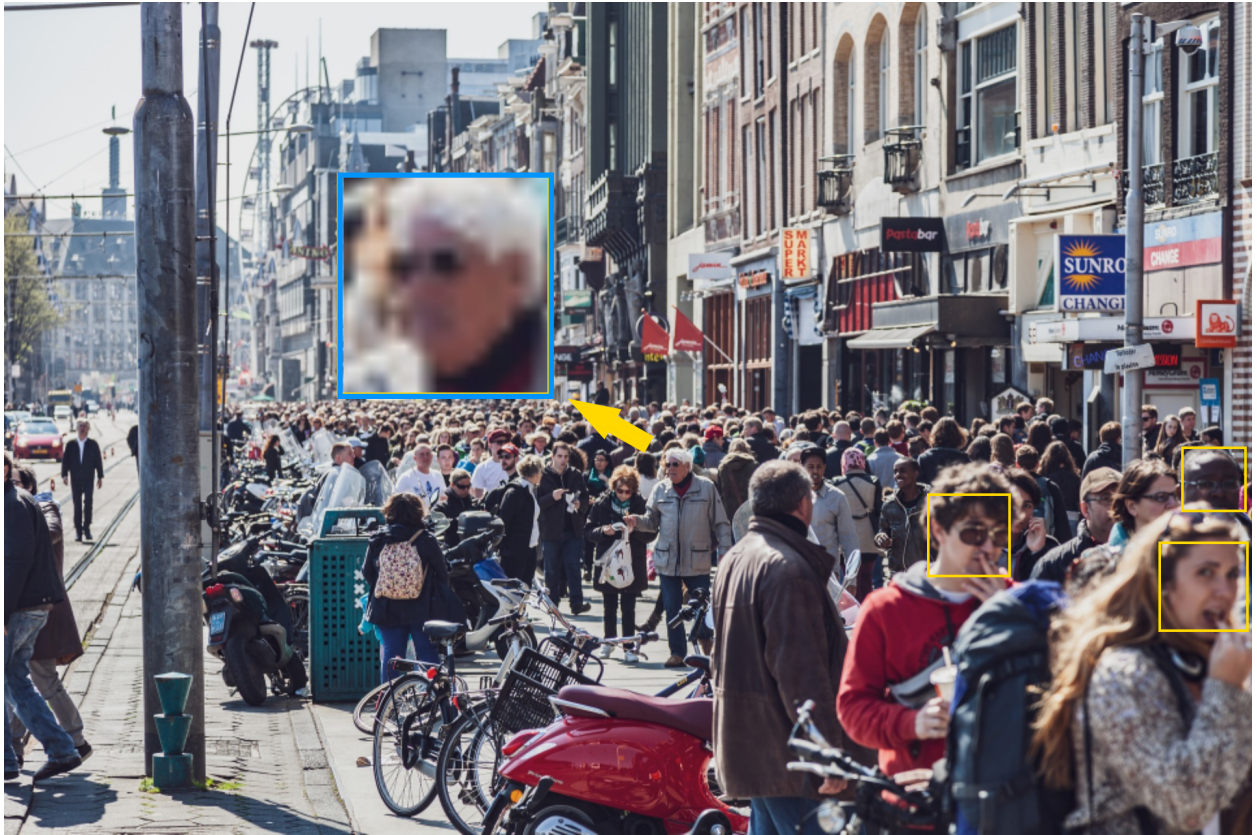


Figure 20: Detections in image 960X640

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" />
```

8.5 Frg-subtractor

When the “frg-subtractor” parameter (Trackengine configuration) 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" />
```

8.6 Frg-regions-alignment

The “frg-regions-alignment” parameter (trackengine.conf) 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" />
```

8.7 Frg-regions-square-alignment

When the “frg-regions-square-alignment” parameter (Trackengine configuration) 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" />
```

8.8 Batched-processing

The “batched-processing” 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.

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

8.9 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.

```
<!-- 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" />
```

8.10 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" />
```

9 LUNA Streams configuration

This section describes the parameters of the LUNA Streams service.

You can configure the service using the LUNA Configurator service.

For LUNA PLATFORM services settings, see the LUNA PLATFORM administrator manual.

9.1 LUNA_STREAMS_DB section

This section sets connection settings to the created LUNA Streams database.

9.1.1 Db_type parameter

The parameter sets the type of database used. Two options are available - “postgres” or “oracle”.

The default type is “postgres”.

```
"db_type": "postgres"
```

9.1.2 Db_user parameter

The parameter specifies the database username.

The default username is “luna”.

```
"db_user": "luna"
```

9.1.3 Db_password parameter

The parameter sets the database password.

The default password is “luna”.

```
"db_password": "luna"
```

9.1.4 Db_name parameter

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

The default name is “luna_streams”.

```
"db_name": "luna_streams"
```

9.1.5 Db_host parameter

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.

```
"db_host": "luna_streams"
```

9.1.6 Db_port parameter

The parameter sets LUNA Streams database listener port.

The default port is “5432” for “postgres” and “1521” for “oracle”.

```
"db_port": 5432
```

9.1.7 Connection_pool_size parameter

The parameter sets the database connection pool size.

The default value is “5”.

```
"db_settings": {  
    "connection_pool_size": 5  
}
```

9.2 LUNA_STREAMS_LOGGER section

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

9.2.1 Log_level parameter

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

```
"log_level": "INFO"
```

The default value is “INFO”.

9.2.2 Folder_with_logs parameter

The parameter sets the folder where the logs are stored (the relative path starts from the directory with the application).

The default value is “./”.

```
"folder_with_logs": "./"
```

9.2.3 Log_time parameter

The parameter sets the time format used in log entries: “LOCAL” or “UTC”.

```
"log_time": "LOCAL"
```

The default value is “LOCAL”.

9.2.4 Log_to_stdout parameter

The parameter enables you to send the log to standard output (stdout). Two options are available - “true” or “false”.

```
"log_to_stdout": true
```

The default value is “true”.

9.2.5 Log_to_file parameter

The parameter enables you to send the log to a file. Two options are available - “true” or “false”.

```
"log_to_file": true
```

The default value is “true”.

9.2.6 Multiline_stack_trace parameter

The parameter enables or disables multi-line traces in logs. Two options are available - “true” or “false”.

```
"multiline_stack_trace": true
```

The default value is “true”.

9.3 LUNA_LICENSES_ADDRESS section

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

9.3.1 Origin parameter

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.

```
"origin": "http://127.0.0.1:5120",
```

The default value is “http://127.0.0.1:5120”.

9.3.2 Api_version parameter

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

```
"api_version": 1
```

9.4 STREAM_WORKER_ASYNC_LOCK_TIMEOUT parameter

Parameter sets timeout of the LUNA Streams instance to lock a row in a database table in seconds in the range (0, 1]. 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.

The default value is “10”.

9.5 STREAM_STATUS_OBSOLETING_PERIOD parameter

Parameter sets stream status obsolescence period in seconds in range (0, 86400]. 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.

The default value is “0.1”.

9.6 LUNA_STREAMS_ACTIVE_PLUGINS parameter

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

The default value is “[]”.

9.7 STORAGE_TIME parameter

Parameter sets time format used in LUNA Streams database records: “LOCAL” or “UTC”.

The default value is “LOCAL”.

9.8 INFLUX_MONITORING section

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

9.8.1 Send_data_for_monitoring parameter

The parameter enables you to enable or disable sending monitoring data to InfluxDB. Two options are available - “0” or “1”.

```
"send_data_for_monitoring": 1
```

The default value is “1”.

9.8.2 Use_ssl parameter

The parameter enables you to use HTTPS to connect to InfluxDB. Two options are available - “0” or “1”.

```
"use_ssl": 0
```

The default value is “0”.

9.8.3 Flushing_period parameter

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

```
"flushing_period": 1
```

The default is 1 second.

9.8.4 Host parameter

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.

```
"host": "127.0.0.1"
```

9.8.5 Port parameter

Parameter sets InfluxDB 2.x port.

```
"port": 8086
```

The default value is “8086”.

9.8.6 Bucket parameter

Parameter sets InfluxDB 2.x bucket name.

```
"bucket": "luna_monitoring"
```

The default bucket name is “luna_monitoring”.

9.8.7 Organization parameter

Parameter sets InfluxDB 2.x workspace.

```
"organization": "luna"
```

The default value is “luna”.

9.8.8 Token parameter

Parameter sets InfluxDB 2.x authentication token.

10 Use FaceStream with LUNA Configurator

When FaceStream operates in the server mode, it can use LUNA Configurator service. The service enables you to store FaceStream settings and apply them to the FaceStream instances.

Note Streams management settings configured in the LUNA Streams service are not stored in the LUNA Configurator service.

The LUNA Configurator service also allows you to store the settings of LUNA Streams and all the LUNA PLATFORM services necessary to launch FaceStream. See the LUNA Streams settings in the “[LUNA Streams configuration](#)” section, and the LUNA PLATFORM settings in the LUNA PLATFORM 5 administrator manual. This section describes the use of the LUNA Configurator service only with FaceStream.

After FaceStream is launched it uses the parameters specified in the LUNA Configurator service and does not request them until restart.

To apply the FaceStream settings changed in the Configurator, you should restart FaceStream.

10.1 Features of working with Configurator

If the LUNA Configurator is specified in the startup line, but is not available, FaceStream will issue an error in the log. At the same time FaceStream settings will be taken from local configuration files, if they were uploaded to the container.

If there are also no local configuration files, then the default settings specified in the FaceStream code will be taken.

10.2 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

10.3 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.



The screenshot shows the LUNA Configurator interface. On the left, there are three sections: 'Name' with a text input containing 'FACE_STREAM_CONFIG', 'Description' with a text input containing 'Face stream configuration', and 'Id and Times' with an unchecked checkbox. In the center, a JSON configuration is displayed in a text area:

```
{
  "logging": {
    "severity": 1,
    "tags": [
      "ffmpeg",
      "gstreamer",
      "bestshot",
      "primary-track"
    ]
  },
}
```

. On the right, there are two buttons: 'Duplicate' and 'Save'. The 'Duplicate' button is highlighted with a blue border.

Figure 21: Duplicate record

- Set a tag and specify parameters values.

Create new setting

Limitation

FACE_STREAM_CONFIG

Description (str <= 128 chars)

Face stream configuration

Value (depends on schema)

```
{"logging":{"severity":1,"tags":["ffmpeg","gstreamer","bestshot","primary-track"],"mode":"I2b"},"sending":{"request-type":"jpeg","portrait-type":"warp","send-source-frame":false,"luna-api":3,"async-requests":true,"luna-account-id":""},"web_tasks":{"concurrent-max-count":3,"max-file-size":52428800},"performance":{"stream-images-buffer-max-size":10,"enable-gpu-processing":false,"convert-full-frame":true}}
```

Tags (str, separate by ',')

setting tags(separate by ',')

Cancel

Create

Figure 22: Change tag

Tags are not created for the default records.

11 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 \  
-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.5.1.2 \  
--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
```

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 \
```

11.1 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 \
--http-address http://0.0.0.0:34569
```

The following keys are available:

- `--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.
- `--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».

12 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.

12.1 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 number.
- [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.

12.2 Additional services API errors

This section describes the errors returned by the LUNA Streams and LUNA Licenses services, as well as general errors that may occur when interacting with these services. 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.

12.2.1 General errors

12.2.1.1 Code 0 returned

Error Message:

Success

Error source:

General errors

Error Description:

This code is returned when there are no errors and the request is successfully processed.

Note that the result may still contain images filtered according to the parameters defined in the request.

12.2.1.2 Code 1 returned

Error Message:

Internal server error. Unknown error

Error source:

General errors

Error Description:

An unexpected internal error occurred. The error is not properly processed in the code.

If any additional trace is provided and the error continues to appear, send the trace to VisionLabs technical support.

12.2.1.3 Code 3 returned

Error Message:

Invalid url {value}

Error source:

General errors

Error Description:

Invalid URL was specified in the request.

Check the URL in the request. It may contain spaces or erroneous symbols.

Check API specification for the examples of the URL.

12.2.1.4 Code 4 returned

Error Message:

Client payload error {value}

Error source:

General errors

Error Description:

Client payload error.

Possible causes: the response object was closed before the response received all data. A transfer encoding error occurred.

12.2.1.5 Code 5 returned

Error Message:

Server fingerprint mismatch {value}

Error source:

General errors

Error Description:

Server fingerprint error.

An attempt to connect to an invalid server was performed or an invalid key was used.

12.2.1.6 Code 6 returned

Error Message:

Socket read timeout. Request timeout on {value} method {value}

Error source:

General errors

Error Description:

Socket reading error. Request completion time limit exceeded.

The problem may occur due to the following reasons:

- Problems in the network. Check network.
- Internal server errors. Check service logs.

12.2.1.7 Code 7 returned

Error Message:

Socket connect timeout Request timeout on {value} method {value}

Error source:

General errors

Error Description:

Socket connection error. Request completion time limit exceeded.

The request may take long due to the following reasons:

- Problems in the network. Check network.
- Internal server errors. Check service logs.
- High service load.

12.2.1.8 Code 8 returned

Error Message:

Connect timeout on {value} method {value}

Error source:

General errors

Error Description:

Connection error. The connection time limit was exceeded.

The problem may occur due to the following reasons:

- Problems in the network. Check network.
- Service is unavailable. Check that the service is running and check its logs.

12.2.1.9 Code 9 returned

Error Message:

Request timeout on {value} method {value}

Error source:

General errors

Error Description:

Connection error. Request completion time limit exceeded.

The problem may occur due to the following reasons:

- Problems in the network. Check network.
- Service is unavailable. Check that the service is running and check its logs.

12.2.1.10 Code 10 returned

Error Message:

Server disconnected {value}

Error source:

General errors

Error Description:

The connection with the server is lost.

The problem may occur due to the following reasons:

- Problems in the network. Check network.
- Server is unavailable. Check that the server is running.

12.2.1.11 Code 11 returned

Error Message:

Server connection error {value}

Error source:

General errors

Error Description:

Server connection error.

The connection with the server is lost.

The problem may occur due to the following reasons:

- Problems in the network. Check network.
- Server is unavailable. Check that the server is running.

12.2.1.12 Code 12 returned

Error Message:

Client proxy connection error {value}

Error source:

General errors

Error Description:

Client proxy connection error.

The problem may occur due to the following reasons:

- Problems in the network. Check network.
- Proxy connection is unavailable.

12.2.1.13 Code 13 returned

Error Message:

Client connector SSL error {value}

Error source:

General errors

Error Description:

Client connection via SSL error. Check SSL certificate.

12.2.1.14 Code 14 returned

Error Message:

Client connector certificate error {value}

Error source:

General errors

Error Description:

Client SSL error.

Possible causes are: invalid SSL certificate, outdated client SSL version. Check SSL certificate.

12.2.1.15 Code 15 returned

Error Message:

Client SSL error {value}

Error source:

General errors

Error Description:

Client SSL error.

Possible causes are: invalid SSL certificate, outdated client SSL version. Check SSL certificate.

12.2.1.16 Code 16 returned

Error Message:

Client connector error {value}

Error source:

General errors

Error Description:

Client connection error.

12.2.1.17 Code 17 returned

Error Message:

Client OS error {value}

Error source:

General errors

Error Description:

The error in the client operating system occurred.

Check the client OS.

12.2.1.18 Code 18 returned

Error Message:

Client connection error {value}

Error source:

General errors

Error Description:

Client connection error.

12.2.1.19 Code 19 returned

Error Message:

Client HTTP proxy error {value}

Error source:

General errors

Error Description:

The HTTP proxy server error has occurred on the client-side.

12.2.1.20 Code 21 returned

Error Message:

Content-Type error {value}

Error source:

General errors

Error Description:

Request content-type error. Invalid value in “Content-Type” field in the request header parameters. It does not match with the data provided in the request.

Check the “HEADER PARAMETERS” section in the API documentation for your request.

12.2.1.21 Code 22 returned

Error Message:

Client response error {value}

Error source:

General errors

Error Description:

The client response error occurred.

12.2.1.22 Code 23 returned

Error Message:

HTTP client error {value}

Error source:

General errors

Error Description:

Client HTTP error.

12.2.1.23 Code 24 returned

Error Message:

HTTP client error {value}

Error source:

General errors

Error Description:

Client HTTP error. The response was received in incorrect JSON format.

[12.2.1.24 Code 11055 returned](#)**Error Message:**

Forbidden. License problem: "{value}"

Error source:

General errors

Error Description:

License error. The license has expired or is unavailable.

[12.2.2 CORE errors, returned in API responses](#)[12.2.2.1 Code 100 returned](#)**Error Message:**

Bad/incomplete configuration. {value}

Error source:

CORE errors, returned in API responses

Error Description:

Cannot read the configuration file.

The error may occur due to:

- Invalid configuration file name or format.
- Invalid parameter name or parameter format.
- Required parameter is missed in the configurations.

Check the configuration file of the service.

- If you are using configuration files, check configuration files in the service directory. The `/srv/luna_{service_name}/configs/config.conf` path is used inside Docker container.
- If you are using the Configurator service, check the corresponding settings in it.

[12.2.2.2 Code 101 returned](#)**Error Message:**

Bad/incomplete query {value}

Error source:

CORE errors, returned in API responses

Error Description:

Invalid request parameters are set or mandatory parameters are absent.

One or several query parameters are misspelled or missed. Invalid values may be set to the query parameters.

12.2.2.3 Code 102 returned

Error Message:

Bad/incomplete body {value}

Error source:

CORE errors, returned in API responses

Error Description:

Invalid request body.

There is an error in the request body or it does not match the required schema.

See “REQUEST BODY SCHEMA” in the API documentation and compare it with the specified one.

12.2.2.4 Code 105 returned

Error Message:

Out of memory {value}

Error source:

CORE errors, returned in API responses

Error Description:

Not enough memory for the request processing.

Check the server memory usage. It may be required to increase the server memory.

12.2.2.5 Code 106 returned

Error Message:

Invalid http request {value}

Error source:

CORE errors, returned in API responses

Error Description:

Invalid HTTP request.

Make sure the request URL and method (POST, GET, etc.) are correct.

12.2.2.6 Code 200 returned

Error Message:

Generic error {value}

Error source:

CORE errors, returned in API responses

Error Description:

Generic error: invalid parameter, logical errors.

The problem may appear due to various reasons. Check that the request to the system is correct in the API documentation.

12.2.3 Database Errors

12.2.3.1 Code 10015 returned

Error Message:

SQL error. SQL request execution failed

Error source:

Database Errors

Error Description:

SQL request failed. A database error has occurred.

The database is not available for an unknown reason, or the required tables are missing. You should request the status of the database, check the availability of the database over the network, check the existence of the required tables in the database.

12.2.3.2 Code 10016 returned

Error Message:

Database error. Could not connect to database

Error source:

Database Errors

Error Description:

The connection to the database failed.

Check that the database is available. There may be errors during database launch or it is not available in the current network (due to server restrictions or network problems). Check the database settings in services parameters. They may be incorrect.

12.2.3.3 Code 10017 returned

Error Message:

Database error Database connection timeout error

Error source:

Database Errors

Error Description:

Database connection timeout error.

Check that the database is available. There may be errors during database launch or it is not available in the current network (due to server restrictions or network problems). Check the database settings in services parameters. They may be incorrect.

12.2.3.4 Code 10018 returned

Error Message:

Database error {value}

Error source:

Database Errors

Error Description:

An error occurred in the database. Check the provided description of the error.

12.2.4 REST API common errors

12.2.4.1 Code 12001 returned

Error Message:

Bad/incomplete input data. Object in query is not UUID4 format: xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx

Error source:

REST API common errors

Error Description:

The provided ID for the object is not in UUID4 format. Check that the provided IDs are correct.

12.2.4.2 Code 12002 returned

Error Message:

Bad/incomplete input data. Request does not contain json

Error source:

REST API common errors

Error Description:

The request does not contain JSON.

Check the JSON syntax in the request body:

- There are extra symbols or spaces.
- The JSON structure is wrong.
- Important syntax elements were missed

12.2.4.3 Code 12003 returned

Error Message:

Bad/incomplete input data. Field. {value} not found in json

Error source:

REST API common errors

Error Description:

The specified field was not found in the JSON body. Check that all required fields were added to the request.

12.2.4.4 Code 12005 returned

Error Message:

Bad/incomplete input data. Request contain empty json

Error source:

REST API common errors

Error Description:

The request contains an empty JSON body. A body is required to execute the request.

12.2.4.5 Code 12012 returned

Error Message:

Bad/incomplete input data. Bad query parameters {value}

Error source:

REST API common errors

Error Description:

The specified query parameters are set incorrectly.

[12.2.4.6 Code 12013 returned](#)**Error Message:**

Resource not found. Page not found

Error source:

REST API common errors

Error Description:

The HTTP resource was not found.

The URL specified in the request was not found.

For example, the request contains the path “/6/sample” instead of “/6/samples”. Compare the specified path with the path in the documentation.

[12.2.4.7 Code 12014 returned](#)**Error Message:**

Bad/incomplete input data. Required parameters {value} not found

Error source:

REST API common errors

Error Description:

The required query parameters were not found in the request. These parameters are listed in the error.

[12.2.4.8 Code 12016 returned](#)**Error Message:**

Bad/incomplete input data. No one parameters {value} not found

Error source:

REST API common errors

Error Description:

Listed query parameters were not found in the request.

12.2.4.9 Code 12017 returned

Error Message:

Bad/incomplete input data. Bad content type

Error source:

REST API common errors

Error Description:

Invalid “Content-Type” header value transmitted in the request.

No header was specified, an invalid header value was specified, or there is a mistake in the header.

12.2.4.10 Code 12018 returned

Error Message:

Bad/incomplete input data. Unsupported param “{value}”

Error source:

REST API common errors

Error Description:

The query parameter is unsupported.

12.2.4.11 Code 12021 returned

Error Message:

Method not allowed. Method not allowed

Error source:

REST API common errors

Error Description:

The HTTP resource does not support this method. Check the HTTP method specified to send the request.

12.2.4.12 Code 12022 returned

Error Message:

Bad/incomplete input data. Failed to validate input json. Path: “{value}” message: “{value}”

Error source:

REST API common errors

Error Description:

There is an error in JSON syntax at the specified path.

The request body contains invalid fields. Check the request for compliance with the template specified the API documentation for the request.

12.2.4.13 Code 12023 returned

Error Message:

Bad/incomplete input data. Content type is unacceptable allowed types: "{value}"

Error source:

REST API common errors

Error Description:

The request contains invalid “content-type”. Most likely, there is a typo in the request content-type.

See the API documentation for the request. Check the available content types.

12.2.4.14 Code 12024 returned

Error Message:

Bad/incomplete input data. Unsupported media type

Error source:

REST API common errors

Error Description:

The data type transmitted in the request is not supported.

See the API documentation for the request. Check the content type available for the request.

12.2.4.15 Code 12025 returned

Error Message:

Bad/incomplete input data. Specified content type does not match data type

Error source:

REST API common errors

Error Description:

The content-type specified in the “Content-Type” header does not match the data type.

See the API documentation for the request. Check the available content types.

12.2.4.16 Code 12027 returned

Error Message:

Bad/incomplete input data. Failed to validate input json. Message: {value}

Error source:

REST API common errors

Error Description:

The request body parameters are incorrect. Check the JSON body for compliance with the request requirements. Details are listed in the “Message:” field.

Check the “REQUEST BODY SCHEMA” of the request. See the examples provided for the request in the documentation.

12.2.4.17 Code 12028 returned

Error Message:

Bad/incomplete input data. Failed to parse Flatbuf

Error source:

REST API common errors

Error Description:

Internal error. Failed to deserialize Flatbuf.

12.2.4.18 Code 12029 returned

Error Message:

Functionality is not implemented. Required server functionality is not implemented

Error source:

REST API common errors

Error Description:

The error is returned for the server functionality that is not implemented.

12.2.4.19 Code 12030 returned

Error Message:

Bad/incomplete input data. Request does not contain data

Error source:

REST API common errors

Error Description:

The request contains no data. The request body is empty.

Check the “REQUEST BODY SCHEMA” of the request and other request parameters. See the examples provided for the request in the documentation.

[12.2.4.20 Code 12031 returned](#)**Error Message:**

Bad/incomplete input data. {value}

Error source:

REST API common errors

Error Description:

The input data is incomplete or wrong.

The detailed description of the error is given in the field “message”.

[12.2.4.21 Code 12032 returned](#)**Error Message:**

Internal server error. Document file not found

Error source:

REST API common errors

Error Description:

The document with the specified name was not found. The error is returned when requesting service documentation.

Check that the request is correct and the document name is set properly.

[12.2.4.22 Code 12033 returned](#)**Error Message:**

Forbidden. Access to this resource on the server is denied

Error source:

REST API common errors

Error Description:

The access to the resource on the server is denied.

Check the resource availability. The access to the service may be restricted in the server settings.

12.2.4.23 Code 12034 returned

Error Message:

Bad/incomplete input data. Descriptor has incorrect length {value}

Error source:

REST API common errors

Error Description:

The descriptor has an incorrect length.

Make sure that you are using a descriptor of the appropriate format. Descriptors received from different sources may have different lengths.

12.2.4.24 Code 12035 returned

Error Message:

Bad/incomplete input data. Failed to parse xpk file

Error source:

REST API common errors

Error Description:

Failed to parse the descriptor XPK file. Check that the file is correct.

12.2.4.25 Code 12036 returned

Error Message:

Bad/incomplete input data. Descriptor version {value} is not registered in the system

Error source:

REST API common errors

Error Description:

The specified descriptor version is not registered in the system.

A wrong/nonexistent descriptor version is set, or an incompatible service version is used that cannot process this descriptor version.

Check that you use a proper descriptor version.

12.2.4.26 Code 12037 returned

Error Message:

Bad/incomplete input data. XPK file does not contain descriptor

Error source:

REST API common errors

Error Description:

The input data XPK does not contain a descriptor. Check that the file is correct.

12.2.4.27 Code 12038 returned

Error Message:

Bad/incomplete input data. SDK descriptor is not valid

Error source:

REST API common errors

Error Description:

SDK descriptor is not valid. Check the provided descriptor data.

Make sure that you are using a descriptor of the appropriate format. Descriptors received from different sources may have different lengths.

12.2.4.28 Code 12039 returned

Error Message:

Bad/incomplete input data. Unknown multipart name "{value}"

Error source:

REST API common errors

Error Description:

There is an unknown multipart name in the request. Check that the name in the "message" field is correct.

12.2.4.29 Code 12040 returned

Error Message:

Bad/incomplete input data. Duplicate multipart name "{value}"

Error source:

REST API common errors

Error Description:

There is a duplicated multipart name in the request. Check the name in the “message” field. Duplaccate names are not available.

[12.2.4.30 Code 12041 returned](#)**Error Message:**

Bad/incomplete input data. Multipart with name “{value}” has bad Content-Type

Error source:

REST API common errors

Error Description:

The input data with the specified name has bad Content-Type in the multipart request. Check the specified content type.

[12.2.4.31 Code 12042 returned](#)**Error Message:**

Gone. Access to this resource on the server is no longer available

Error source:

REST API common errors

Error Description:

The access to the resource on the server is not available. Check that the corresponding service is working, has correct version and is available. Check that the resource is correct.

[12.2.4.32 Code 12043 returned](#)**Error Message:**

Unknown error. Service “{value}” unknown error method: “{value}” url: “{value}”

Error source:

REST API common errors

Error Description:

Unknown service error has occurred. Check the method and URL provided in the error message.

12.2.4.33 Code 12044 returned

Error Message:

Bad/incomplete input data. Failed to decompress input body using gzip encoder

Error source:

REST API common errors

Error Description:

Failed to decompress input body using GZIP encoder.

12.2.4.34 Code 12045 returned

Error Message:

Bad/incomplete input data. Failed to decompress input body using deflate encoder

Error source:

REST API common errors

Error Description:

Failed to decompress input body using deflate encoder.

12.2.5 LUNA Licenses service errors

12.2.5.1 Code 33001 returned

Error Message:

License problem Failed to check HASP License feature {value}

Error source:

LUNA Licenses service errors

Error Description:

A license problem occurred. The request for the feature availability failed. Check your license.

Possible reasons: the license was not applied, you use a wrong license, connection to the license server could not be established.

12.2.5.2 Code 33002 returned

Error Message:

License problem Failed to get value of HASP License feature {value}

Error source:

LUNA Licenses service errors

Error Description:

A license problem occurred. The request for the feature value failed. Check your license. Possible reasons: the license was not applied, you use a wrong license, connection to the license server could not be established.

[12.2.5.3 Code 33003 returned](#)

Error Message:

License problem No value found for required HASP License feature {value}

Error source:

LUNA Licenses service errors

Error Description:

A license problem occurred. The license does not include required feature.

Possible reasons:

- The required feature was not added to the license.
- A wrong license is used. A wrong license file was applied.
- A new license was not applied.
- Connection to the license server could not be established.

[12.2.5.4 Code 33004 returned](#)

Error Message:

License problem Failed to consume {value}

Error source:

LUNA Licenses service errors

Error Description:

The specified licensed LP feature cannot be used because the limit on the number of transactions was exceeded. Check your license.

[12.2.5.5 Code 33005 returned](#)

Error Message:

License problem Failed to consume {value}: license expired

Error source:

LUNA Licenses service errors

Error Description:

The specified licensed LP feature cannot be used because the license has expired.

Check your license.

12.2.6 LUNA Streams service errors

12.2.6.1 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.

12.2.6.2 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 (see the section “[Statuses transition table](#)”). The error description shows the expected status.

12.2.6.3 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).

12.2.6.4 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.

12.2.6.5 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.

13 Cameras Compatibility

Compatibility of the specified IP cameras with FaceStream is shown in the table below.

Table 16: cameras compatibility

Camera model	Testing	FaceStream	Results
		Version	
Hikvision ds-2cd2822f	Internal	3.2.2	No problems detected
Hikvision DS-2CD7126G0-IZS	Partners	3.2.2	No problems detected
Dahua IPC-HDBW8630E-Z	Partners	3.2.2	No problems detected
WiseNet XNV 8040 WiseNet XNV 8030	Partners	3.2.2	No problems detected
AXIS Q3515-LV	Partners	3.2.2	No problems detected
Hikvision DS-2CD5126	Partners	3.2.2	No problems detected
Vivotek cc8160	Partners	3.2.2	No problems detected
Vivotek cc837	Partners	3.2.2	No problems detected
ACTi E38	Partners	3.2.2	No problems detected
ACTi A92	Partners	3.2.2	No problems detected
ACTi E928	Partners	3.2.2	No problems detected
Vivotek FD9365-EHTV	Partners	3.2.2	No problems detected
Vivotek IB9367-EHT	Partners	3.2.2	No problems detected
Bosch NDI-4502-A	Partners	3.2.2	No problems detected

Camera model	Testing	FaceStream	
		Version	Results
Bosch NBN-50022-V3 (Use fisheye Lens 4mp 2.7-12mm Lens(DH-PLZ1040-D FOC 201706050010))	Partners	3.2.2	No problems detected
Bosch NUC-21012-F2	Partners	3.2.2	No problems detected
Dahua ipc-hdbw2220rp-vfs	Partners	3.2.2	No problems detected
Dahua ipc-hfw2221r-vfs-ire6	Partners	3.2.2	No problems detected
Samsung XNV-8040RP	Partners	3.2.2	No problems detected
Samsung SNV-SNV-6013P	Partners	3.2.2	No problems detected
Hikvision DS-2CD2125FWD-IS	Partners	3.2.2	No problems detected
Hikvision DS-2CD2025FWD-I	Partners	3.2.2	No problems detected
Hikvision DS-2CD4525FWD-IZH	Partners	3.2.2	No problems detected
Mobotix MX-S16B (MX-O-SMA-S-6D041)	Partners	3.2.3	No problems detected
Panasonic WV-S6131/WV-S6130	Partners	3.2.3	No problems detected
Panasonic WV-SC588A	Partners	3.2.3	No problems detected