Smilart Web API Guide

Table of Contents

Overview . . . 1
Common protocol description . . . 1
  WebSockets heartbeat . . . 1
Root Endpoint . . . 2
Resources Summary . . . 3
  Person Collection . . . 3
  Person . . . 3
  Person Photo Collection . . . 3
  Person Photo . . . 3
  Camera Collection . . . 5
  Camera Info . . . 5
  Camera MJPEG . . . 5
  VCA Service . . . 5
  Photo Booth Service . . . 5
  Verification Service . . . 5
  IPA Service . . . 6
  Adaptive Verification Service . . . 6
  License Management Service . . . 6
Resources Reference . . . 7
  Person Collection . . . 7
  Person . . . 10
  Person Photo Collection . . . 14
  Person Photo . . . 15
  Camera Collection . . . 22
  Camera Info . . . 24
  Camera MJPEG . . . 25
  VCA Service . . . 27
  Photo Booth Service . . . 34
  Verification Service . . . 44
  IPA Service . . . 52
  Adaptive Verification Service . . . 58
  License Management Service . . . 65

Overview

Smilart Web API is a set of several API services which provide necessary functions to use Smilart recognition algorithms when you create your own Web application.

List of services included into Smilart API:

1. Person Service — person management service implements basic operations on persons, such as adding, retrieving, removing and updating person in Platform database.
2. Camera Service — camera management service implements basic operations on cameras in Platform.
3. Video Content Analytics Service (VCA Service) — provides access to some events generated by Platform services, such as frame streams from cameras, face detection results and identification result.
4. Photo Booth Service — selects best frames from the camera stream, in order to put them into Platform database to achieve best identification results.
5. Verification Service — implements verification case.
6. Instant Photo Analytics Service — the service for instant photo analysis.
7. Adaptive Verification Service — provides opportunities to manage Adaptive Verification (AV) service.
8. License Management Service — provides opportunities to manage License (LM) service.

Platform receives frames from connected cameras. Each camera can be either physical device or software emulation, and must implement one of the supported streaming protocols.

All services share common base of persons that used for face recognition.

Common protocol description

Smilart Web API uses HTTP and WebSockets as a transport layer for communication and JSON as a payload.

WebSockets heartbeat

After the handshake, the server will periodically send **ping** messages to the client.
When the **ping** is received, the recipient must send back a **pong** as soon as possible.
The server use this mechanism to make sure that the client is still alive. It is RECOMMENDED that client also send **pings** for checking a server.

A **ping** or **pong** is just a regular frame, but it is a control frame.
Pings have an opcode of `0x9`, and **pongs** have an opcode of `0xA`.
When you get a **ping**, send back a **pong** with the exact same Payload Data as the **ping** (for **pings** and **pongs**, the max payload length is 125).
You might also get a **pong** without ever sending a **ping**; ignore this if it happens.

Most WebSockets libraries already have PING/PONG mechanism.

Root Endpoint

By default root endpoint for Smilart Web API is located at `/api`. The default port is 9999, The default URL for Person Service, Camera Service, IPA Service: `http://<yourInstance>:9999/api` and `ws://<yourInstance>:9999/api` for other services.
Any Smilart Web API Resource endpoints are relative to these URLs, unless otherwise noted.

Resources Summary

Person Collection

For Person Collection Resource details, see the resource representation.

Method	HTTP Request	Description
ListPersons	`GET /persons`	Lists all persons in base.
DeletePersons	`DELETE /persons`	Deletes all persons from base.

Person

For Person Resource details, see the resource representation.

Method	HTTP Request	Description
AddPerson	`POST /persons`	Adds new person into base with auto-generated identifier.
AddPersonWithId	`PUT /persons/{personId}`	Adds new person into base with identifier specified by client.
GetPerson	`GET /persons/{personId}`	Retrieves the person from base.
DeletePerson	`DELETE /persons/{personId}`	Removes the person from base.

Person Photo Collection

For Person Photo Collection Resource details, see the resource representation.

Method	HTTP Request	Description
ListPersonPhotos	`GET /persons/{personId}/photos`	Lists all photos for the person.
DeletePersonPhotos	`DELETE /persons/{personId}/photos`	Removes all photos associated with the person.

Person Photo

For Person Photo Resource details, see the resource representation.

Method	HTTP Request	Description
AddPersonPhoto	`POST /persons/{personId}/photos`	Adds photo to the person.
AddPersonPhotoWithId	`PUT /persons/{personId}/photos/{photoId}`	Adds photo with specified identifier to the person.
GetPersonPhoto	`GET /persons/{personId}/photos/{photoId}`	Receives photo from the person.

GetPersonPhotoJpeg	`GET /persons/{personId}/photos/{photoId}/jpeg`	Receives binary JPEG photo from the person.
DeletePersonPhoto	`DELETE /persons/{personId}/photos/{photoId}`	Removes photo from the person.

Camera Collection

For Cameras Resource details, see the resource representation.

Method	HTTP Request	Description
ListCameras	`GET /cameras`	Lists all cameras.

Camera Info

For Camera Info Resource details, see the resource representation.

Method	HTTP Request	Description
GetCameraInfo	`GET /camera/{cameraPid}`	Receives the camera information from the system.

Camera MJPEG

For Camera MJPEG Resource details, see the resource representation.

Method	HTTP Request	Description
GetCameraMjpegStream	`GET /camera/{cameraPid}/mjpeg`	Starts subscription on the camera MJPEG stream.

VCA Service

For VCA Service description, see the details.

Method	HTTP Request	Description
SubscribeToVCAEvents	`GET /cameras/{cameraPid}/vca`	Subscribes to the events of VCA Service.

Photo Booth Service

For Photo Booth Service description, see the details.

Method	HTTP Request	Description
StartPhotoSelection	`GET /cameras/{cameraPid}/photobooth`	Starts frames selection from the camera.

Verification Service

For Verification Service description, see the details.

Method	HTTP Request	Description

Verify	`GET /cameras/{cameraPid}/verify`	Starts verification process from the camera.

IPA Service

For IPA Service description, see the details.

Method	HTTP Request	Description
PhotoAnalysis	`POST /photo_analysis`	Correlates image with base.

Adaptive Verification Service

For Adaptive Verification Service description, see the details.

Method	HTTP Request	Description
GetConfig	`GET /av/config`	Gets the service configuration.
SetConfig	`POST /av/config`	Sets the service configuration.
RemoveAllPhotos	`DELETE /av/photos`	Removes all sampled photos.
RemovePhotosByPerson	`DELETE /av/photos/persons/{personId}`	Removes all sampled photos of the person.
RemovePhotosByCamera	`DELETE /av/photos/cameras/{cameraPid}`	Removes all sampled photos from the camera for every person.

License Management Service

For License Management Service description, see the details.

Method	HTTP Request	Description
SetLicense	`PUT /lm/license`	Installs a new license file.
GetFingerprint	`GET /lm/fingerprint`	Gets a server fingerprint.
GetLicense	`GET /lm/license`	Gets the installed product license.

Resources Reference

Person Collection

Overview

Represents collection of persons in base.

Resource Representation

Content-Type: application/vnd.com.smilart.helios.persons+json

JSON array of string identifiers of persons in base.

Methods

ListPersons

Lists all persons in base.

Request

```
GET /persons
```

Optional Query Parameters

Parameter Name	Value Type	Default Value	Description
`filter_by_person_id_substring`	string	not provided	Option for filtering the list of persons by substring in the person identifier. **Case insensetive**.
`limit`	integer	not provided	Option to limit the number of identifiers.
`offset`	integer	not provided	Option to offset over the specified number of identifiers.

Request Body

Do not supply a request body with this method.

Response

Persons listed

Status code: `200 OK`
Headers: `X-Smilart-TotalPersons`
Body: representation of the resource.

Headers

Property Name	Value Type	Description
`X-Smilart-TotalPersons`	integer	Number of all or filtered persons in the base.

DeletePersons

Deletes all persons in base.

Request

```
DELETE /persons
```

Request Body

Do not supply a request body with this method.

Response

Persons deleted

Status code: `204 No Content`

Person

Overview

Represents person. Person has a (possibly empty) collection of processed photos.

Resource Representation

Content-Type: application/vnd.com.smilart.helios.person+json

```
{
  "id":string,
  "creationTime":integer,
  "modificationTime":integer
}
```

Properties

Property Name	Value Type	Description
`id`	string	Identifier of the person.
`creationTime`	integer	Person creation time.
`modificationTime`	integer	Last person modification time in ms. It is updated on every modification of any person field, include initial person creation.

Methods

AddPerson

Adds new person into base with auto-generated identifier.

Request

```
POST /persons
```

Request Body

Do not supply a request body with this method.

Response

Person added

Status code: `201 Created`
Body: representation of the resource.

AddPersonWithId

Adds new person into base with specified identifier.

Request

```
PUT /persons/{personId}
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Client-generated identifier of new person. Non empty string. Max lenght is 50. ASCII symbols with codes [32, 126]. Should be unique for the person.

Request Body

Do not supply a request body with this method.

Response

Person added

Status code: `201 Created`
Body: representation of the resource.

Person with specified identifier already exists

Status code: `409 Conflict`

PersonId validation error

Status code: `400 Bad Request`

GetPerson

Gets person from base.

Request

```
GET /persons/{personId}
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.

Request Body

Do not supply a request body with this method.

Response

Person provided

Status code: `200 OK`
Body: representation of the resource.

Person not found

Status code: `404 Not Found`

DeletePerson

Deletes person from base.

Request

```
DELETE /persons/{personId}
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.

Request Body

Do not supply a request body with this method.

Response

Person deleted

Status code: `204 No Content`

Person not found

Status code: `404 Not Found`

Person Photo Collection

Overview

Represents collection of photos for the person in base.

Resource Representation

Content-Type: application/vnd.com.smilart.helios.photos+json

JSON array of string identifiers of photos for the person in base.

Methods

ListPersonPhotos

Lists photos of the person.

Request

```
GET /persons/{personId}/photos
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.

Request Body

Do not supply a request body with this method.

Response

Photos listed

Status code: `200 OK`
Body: representation of the resource.

Person not found

Status code: `404 Not Found`

DeletePersonPhotos

Deletes all photos of the person.

Request

```
DELETE /persons/{personId}/photos
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.

Request Body

Do not supply a request body with this method.

Response

Photos deleted

Status code: `204 No Content`

Person not found

Status code: `404 Not Found`

Person Photo

Overview

Represents photo of the person in base.

Resource Representations

JPEG photo of the person as multipart/form-data

Content-Type: multipart/form-data; boundary={your boundary}

```
{your boundary}\r\n
Content-Type: image/jpeg\r\n\r\n
{binary jpeg}
{your boundary}\r\n
```

Meta information about person photo

Content-Type: application/vnd.com.smilart.helios.photo+json

```
{
  "id":string,
  "creationTime":integer,
  "autoSampled":boolean
}
```

Properties

Property Name	Value Type	Description
`id`	string	Identifier of the person’s photo.
`creationTime`	integer	Photo creation time in ms.
`autoSampled`	boolean	True if this photo was sampled by Adaptive Verification service during self-learning process, other false.

Methods

AddPersonPhoto

Adds binary representation of a photo to the person.
Only JPEG images are supported.

Request

```
POST /persons/{personId}/photos
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.

Request Body

Content-Type

`multipart/form-data; boundary={your boundary}`

Payload

Representation of the resource

Response

Photo added

Status code: `201 Created`
Content-Type: `application/vnd.com.smilart.helios.photo+json`
Body: representation of the resource.

Face not found in the photo

Status code: `422 Unprocessable Entity`
Body: No face.

Payload validation error

Status code: `400 Bad Request` Body: reason.

Person not found

Status code: `404 Not Found`

AddPersonPhotoWithId

Adds binary representation of a photo to the person with specified identifier.
Only JPEG images are supported.

Request

```
PUT /persons/{personId}/photos/{photoId}
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.
`photoId`	string	Client-generated identifier of new photo. Non empty string. Max lenght is 50. ASCII symbols with codes [32, 126]. Should be unique for the photo for each person.

Request Body

Content-Type

`multipart/form-data; boundary={your boundary}`

Payload

Representation of the resource

Response

Photo added

Status code: `201 Created`
Content-Type: `application/vnd.com.smilart.helios.photo+json`
Body: representation of the resource.

Photo with specified identifier already exists

Status code: `409 Conflict`

Face not found in the photo

Status code: `422 Unprocessable Entity`
Body: No face.

Payload validation error

Status code: `400 Bad Request` Body: reason.

Person not found

Status code: `404 Not Found`

GetPersonPhoto

Receives information about photo with specified identifier.

Request

```
GET /persons/{personId}/photos/{photoId}
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.
`photoId`	string	Identifier of the photo.

Request Body

Do not supply a request body with this method.

Response

Photo received

Status code: `200 OK`
Content-Type: `application/vnd.com.smilart.helios.photo+json`
Body: representation of the resource.

Photo or person not found

Status code: `404 Not Found`

GetPersonPhotoJpeg

Receives binary JPEG photo from the person.

Request

```
GET /persons/{personId}/photos/{photoId}/jpeg
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.
`photoId`	string	Identifier of the photo.

Request Body

Do not supply a request body with this method.

Response

Photo received

Status code: `200 OK`
Content-Type: `image/jpeg`
Body: binary representation of the photo for the person.

Photo or person not found

Status code: `404 Not Found`

DeletePersonPhoto

Deletes photo of the person with specified identifier.

Request

```
DELETE /persons/{personId}/photos/{photoId}
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.
`photoId`	string	Identifier of the photo.

Request Body

Do not supply a request body with this method.

Response

Photo deleted

Status code: `204 No Content`

Photo or person not found

Status code: `404 Not Found`

Camera Collection

Overview

Represents collection of cameras in system.

Resource Representation

Cameras

Content-Type: application/vnd.com.smilart.helios.cameras+json

JSON array with string identifiers of cameras in system.

CamerasInfo

Content-Type: application/vnd.com.smilart.helios.cameras_info+json

JSON array with camera representation:

```
[
  {
    "id":string,
    "active":boolean,
    "running":boolean,
  },
  ...
]
```

Methods

ListCameras

Lists all cameras in system.

Request

```
GET /cameras
```

Request Headers

Header Name	Value Type	Description

`Accept`	string	Media type(s) that is/are acceptable for the response. Acceptable types are: • **application/vnd.com.smilart.helios.cameras+json** • **application/vnd.com.smilart.helios.cameras_info+json**. If not provided **application/vnd.com.smilart.helios.cameras+json** is used.

Request Body

Do not supply a request body with this method.

Response

Cameras listed

Status code: `200 OK`
Body: representation of the resource.

Not Acceptable

Status code: `406 Not Acceptable`

Camera Info

Overview

Represents camera info.

Resource Representation

Content-Type: application/vnd.com.smilart.helios.camera_info+json

```
{
  "id":string,
  "active":boolean,
  "running":boolean,
}
```

Properties

Property Name	Value Type	Description
`id`	string	Identifier of the camera.
`active`	boolean	Whether the camera was activated (started) in the system, typically intentionally by the system administrator.
`running`	boolean	True if camera frames are available for processing, otherwise false.

Methods

GetCameraInfo

Gets the camera information from the system.

Request

```
GET /cameras/{cameraPid}
```

Path Parameters

Parameter Name	Value Type	Description
`cameraPid`	string	Identifier of the camera.

Request Body

Do not supply a request body with this method.

Response

Camera Info provided

Status code: `200 OK`
Body: representation of the resource.

Camera not found

Status code: `404 Not Found`

Camera MJPEG

Overview

Represents MJPEG Stream from cameras.

Resource Representation

M-JPEG over HTTP type.

Methods

GetCameraMjpegStream

Gets a stream from the camera.

Request

```
GET /cameras/{cameraPid}/mjpeg
```

Path Parameters

Parameter Name	Value Type	Description
`cameraPid`	string	Identifier of the camera.

Optional Query Parameters

Parameter Name	Value Type	Description
`resolution`	enumeration	Resolutions of the frames. Acceptable types are: **small**, **medium**, **large**, **original**. If not provided **original** is used.
`max_fps`	integer	Rate limit for incoming messages per seconds. Positive integer. If not provided produces all incoming frames.

Request Body

Do not supply a request body with this method.

Response

Camera MJPEG Stream provided

Status code: `200 OK`

Headers: multipart/x-mixed-replace; boundary=<*some boundary*>
Body: representation of the resource.

Incorrect request parameters

Status code: `400 Bad Request`

Camera not found

Status code: `404 Not Found`

VCA Service

Overview

VCA Service provides access to events generated by Platform in the process of handling frames.
Subscription to video analytics events is implemented via WebSockets.
To subscribe client should open WebSocket with header `Sec-WebSocket-Protocol: vca`.
Open WebSocket will receive messages with the result of the analysis of the frame from the selected camera.
The message is an aggregated event (frame + detection + correlation + identification) in JSON.

There is an aggregation timeout for event and if the system does not collect all the information during this time some information in the aggregated event may be missing.

A subscription for more then one type of event will add a delay (about the aggregation timeout value) to the sending.

In the subscription request you can specify what type of information with analysis of the frame you want to receive and how often you want to receive messages.
If an error occurs during validation of specified options in the beginning or during the process, WebSocket would be closed by the server with the corresponding code.

The server will only close connection if the client terminated subscription by closing WebSocket.

Correlations are primarily **debugging information** that can be completely correct interpreted only by the vendor's specialists and reflects the features of the **currently used** face recognition algorithm that **may change** in the future. Therefore, you **SHOULD NOT** make any conclusions based on the received coefficients, except for getting the top of the most similar persons in the database according to the **current** face recognition algorithm.

Request

```
GET /cameras/{cameraPid}/vca
```

Path Parameters

Parameter Name	Value Type	Description
`cameraPid`	string	Identifier of the camera.

Optional Query Parameters

Parameter Name	Value Type	Default Value	Description
`subscribe`	string	`"frame+detect+identification"`	Defines what types of events should be sent. Should not be empty. Represents composition of types joined by "+". Acceptable types are: • `"frame"`: subscribe to frames. • `"detect"`: subscribe to detects. • `"correlation"`: subscribe to correlations. • `"identification"`: subscribe to identifications.
`required`	string	not provided	Defines what types of events are required and should always be presented in aggregated message. These set of types should be subset of `subscribe` parameters set. Represents composition of types joined by "+". Acceptable types are: • `"frame"`: subscription to frames. • `"detect"`: subscription to detects. • `"correlation"`: subscription to correlations. • `"identification"`: subscription to identifications.
`frame_size`	string	`"medium"`	Size of frames from camera. Picture from the camera will be scaled to fit built-in dimensions. It converts so that the proportions do not change and do not become larger. Acceptable values are: • `"small"`: 320x240 pixels. • `"medium"`: 800x600 pixels. • `"large"`: 1400x1050 pixels. • `"original"`: original frame size.
`max_mps`	integer	not provided	Rate limit for incoming messages per seconds. Positive integer. If not provided produces all incoming events. If set over 100 reduced to 100.

`detect_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send detected faces. • `"jpeg"`: send detected faces as jpeg images.
`correlation_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send correlated faces. • `"jpeg"`: send correlated faces as jpeg images.
`identification_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send identified faces. • `"jpeg"`: send identified faces as jpeg images.
`correlation_max_persons`	integer	10	Maximum number of matches in correlation event. Positive integer.
`identification_throttle`	integer	not provided	Minimum time interval in seconds between identification messages of the same person. Positive integer.

Communication Protocol

Server Send Message Representation

Below is a scheme of an aggregated event in JSON.
Depending on the settings and system performance, some objects may be missing.

```
{
  "camera": string,
  "sequence": integer,
  "timestamp": integer,
  "frame": {
    "meta": {
      "width": integer,
      "height": integer,
    },
    "image": {
      "contentType": "image/jpeg",
      "data": string
    }
  },
  "detects": {
    "<detect id>": {
      "id": <detect id>,
      "top": integer,
      "left": integer,
      "right": integer,
      "bottom": integer,
      "leftEyeTop": integer,
      "leftEyeLeft": integer,
      "rightEyeTop": integer,
      "rightEyeLeft": integer,
      "detectFace": {
        "contentType": "image/jpeg",
        "data": string
      }
    }
  },
  "correlations": {
    "<detect id>": {
      "id":<detect id>,
      "matches": [
        {
          "correlation": float,
          "personId": string,
          "photoId": string,
          "databaseFace": {
            "contentType": "image/jpeg",
            "data": string
          }
        }
      ]
    }
  },
  "identifications": {
    <detect id>: {
      "id": <detect id>,
      "correlation": float,
      "threshold": float,
      "personId": string,
      "photoId": string,
      "databaseFace": {
        "contentType": "image/jpeg",
        "data": string
      }
    }
  }
}
```

Properties

Property Name	Value Type	Description
`camera`	string	Pid of camera.
`sequence`	integer	Sequential number of the message.
`timestamp`	integer	Server-side timestamp (with ms resolution) of the message.
`frame`	nested object	Frame info if requested by client.
`frame.meta`	nested object	Frame meta.
`frame.meta.width`	integer	Width of the image of frame.
`frame.meta.height`	integer	Height of the image of frame.
`frame.image`	nested object	Image of the frame.
`frame.image.contentType`	string	Content type of payload. Only `"image/jpeg"` supported.
`frame.image.data`	string	Base64 encoded binary content of the image.
`detects`	nested object	Information about detects. Present if requested by client.
`detects.<detect id>`	nested object	Information about detect with identifier `<detect id>`.
`detects.<detect id>.id`	string	Detect id = `<detect id>`.
`detects.<detect id>.top`	integer	Coordinate of top horizontal line of face rectangle.
`detects.<detect id>.bottom`	integer	Coordinate of bottom horizontal line of face rectangle.
`detects.<detect id>.left`	integer	Coordinate of left vertical line of face rectangle.

`detects.<detect id>.right`	integer	Coordinate of right vertical line of face rectangle.
`detects.<detect id>.leftEyeTop`	integer	Top coordinate of face left eye. **May be missing**.
`detects.<detect id>.leftEyeLeft`	integer	Left coordinate of face left eye. **May be missing**.
`detects.<detect id>.rightEyeTop`	integer	Top coordinate of face right eye. **May be missing**.
`detects.<detect id>.rightEyeLeft`	integer	Left coordinate of face right eye. **May be missing**.
`detects.<detect id>.detectFace`	nested object	Face from frame. Present if requested by client.
`detects.<detect id>.detectFace.contentType`	string	Content type of payload. Only `"image/jpeg"` supported.
`detects.<detect id>.detectFace.data`	string	Base64 encoded binary content of the image.
`correlations`	nested object	Information about correlations. Present if requested by client.
`correlations.<detect id>`	nested object	Information about correlation for detect with identifier `<detect id>`.
`correlations.<detect id>.id`	string	Detect identifier = `<detect id>`.
`correlations.<detect id>.matches`	array	List of matches for correlation.
`correlations.<detect id>.matches[].correlation`	float	Correlation coefficient between detected face and photo from base.
`correlations.<detect id>.matches[].personId`	string	Person id correlated with.
`correlations.<detect id>.matches[].photoId`	string	Photo id from base correlated with.
`correlations.<detect id>.matches[].databaseFace`	nested object	Face image from base for correlated photo. Present if requested by client.
`correlations.<detect id>.matches[].databaseFace.contentType`	string	Content type of payload. Only `"image/jpeg"` supported.
`correlations.<detect id>.matches[].databaseFace.data`	string	Base64 encoded binary content of the image.
`identifications`	nested object	Information about identifications. Present if requested by client.
`identifications.<detect id>`	nested object	Information about identification for detect with identifier `<detect id>`.
`identifications.<detect id>.id`	string	Detect id = `<detect id>`.
`identifications.<detect id>.correlation`	float	Correlation coefficient between detected face and photo from base.
`identifications.<detect id>.threshold`	float	Current system threshold for correlation coefficient. Person is identified if the correlation is not less than the threshold.
`identifications.<detect id>.personId`	string	Person id identified with.

`identifications.<detect id>.photoId`	string	Photo id from base identified with.
`identifications.<detect id>.databaseFace`	nested object	Face image from base for identified photo. Present if requested by client.
`identifications.<detect id>.databaseFace.contentType`	string	Content type of payload. Only `"image/jpeg"` supported.
`identifications.<detect id>.databaseFace.data`	string	Base64 encoded binary content of the image.

Errors Handling

The table below shows error codes and descriptions that are returned if the emergency shutdown of the WebSocket on server side occurs.
Note that closing message is limited by 125 characters.

Close Event content

Code	Full-text Reason Description	Description	Proposed client's actions
1001	`"Going Away"`	Indicates that an endpoint is "going away", such as a server going down for some internal reason.	Contact tech support.
1011	`"Internal Server Error"`	Indicates that a server is terminating the connection because it encountered an unexpected condition that prevented it from fulfilling the request.	Contact to tech support.
4000	`"Unknown query entry: {key}={value}"`	Optional query parameter name `{key}` and its value `{value}` was not understood by the server.	Check the parameter's compliance with the service protocol.
4001	`"Unknown camera: {cameraPid}"`	Process couldn't be started due to absence of camera with identifier `{cameraPid}`.	Check availability of the specified camera.

Photo Booth Service

Overview

Service provides the way to get optimal set of faces for person who stands in front of the camera to enroll the person using these photos.
To start process client should open WebSocket with header `Sec-WebSocket-Protocol: photobooth`.

There are several steps to get the set of faces:

- The person to enroll should stands in front of the camera.
- The process of building the set of faces starts via WebSocket request.
- The person by moving his or her head to different poses provides to the service different images of his or her face.
- The service accepts the images depending on its sampler scheme.
- When all necessary images have been collected or operation timeout occurs, the service stops the sampling process and creates the optimal set of faces, which will be returned from the service.

Sampling process is finished when area of interest is full.
The area of interest consists of one or several groups, each of which indicates a certain head pose or face position and is called named position or named group. The area of interest is considered full, when each named group included in it is full.
A group of head pose is considered full when it collects enough detects where face relates to the group. Implementation of the service reserves the right to define strategy of group of head pose progress estimation.

Client will receive JSON messages about the start, progress, detects and result of the process for the selected camera through the opened WebSocket.

> There is a timeout to receive and convert an image for detect message and if the system does not succeed in the specified time, the message will not be sent.

In the subscription request, you can specify what information you want to receive and how often to receive messages.
There is a timeout for sampling process (20 seconds by default) after which partial completed result will be send.

> Clients can interfere with each other if they try to run the process from the same camera at the same time.

If an error occurs during validation of options, start or during process, the WebSocket is closed by the server with the corresponding code.

The subscription is terminated when the client closes the WebSocket (the server does not close the WebSocket unless an error occurs even after the sampling process finished).

Request

```
GET /cameras/{cameraPid}/photobooth
```

Path Parameters

Parameter Name	Value Type	Description
`cameraPid`	string	Identifier of the camera.

Optional Query Parameters

Parameter Name	Value Type	Default Value	Description
`subscribe`	string	`"detect"`	Defines what types of events should be sent besides sampling result. Represents composition of types joined by "+". Acceptable types are: • `"detect"`: subscribe to detects events. • `"progress"`: subscribe to progress events. To receive messages only about sampling result use empty string as a parameter value.
`detect_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send detected faces. • `"jpeg"`: send detected faces as jpeg images.

Communication Protocol

Detected Face Named Positions

Named position indicates a certain head pose or face position.

- `"leftwardTurn"`: person looks to the left.
- `"leftwardUpwardTurn"`: person looks to the left and up.
- `"leftwardDownwardTurn"`: person looks to the left and down.
- `"forwardTurn"`: person looks straight forward.
- `"forwardUpwardTurn"`: person looks up.
- `"forwardDownwardTurn"`: person looks down.
- `"rightwardTurn"`: person looks to the right.
- `"rightwardUpwardTurn"`: person looks to the right and up.
- `"rightwardDownwardTurn"`: person looks to the right and down.
- `"outsideLeftwardTurn"`: person looks to the left too far from forward position.
- `"outsideRightwardTurn"`: person looks to the right too far from forward position.
- `"outsideUpwardTurn"`: person looks up too far from forward position.
- `"outsideDownwardTurn"`: person looks down too far from forward position.

Areas Of Interest

Set of face positions taken into account during the sampling process:

- `"cross"`: 5 positions (`"forwardTurn"`, `"leftwardTurn"`, `"rightwardTurn"`, `"forwardUpwardTurn"`, `"forwardDownwardTurn"`).
- `"horizontal"`: 3 positions (`"forwardTurn"`, `"leftwardTurn"`, `"rightwardTurn"`).
- `"forwardTurn"`: only forward position (`"forwardTurn"`).
- `"allInnerPoses"`: all 9 positions.

Server Send Message Representation

All messages from the service can be divided into 3 types:

- initial message
- info message
- final message

Initial message

```
{
  "camera": string,
  "sequence": integer,
  "timestamp": integer,
  "startOptions": {
    "timeLimitSeconds": integer,
    "scheme": {
      "grid3x3": {
        "areaOfInterest": string
      }
    }
  }
}
```

Info messages

```
{
  "camera": string,
  "sequence": integer,
  "timestamp": integer,
  "detects": {
    "<detect id>": {
      "id": <detect id>,
      "top": integer,
      "left": integer,
      "right": integer,
      "bottom": integer,
      "leftEyeTop": integer,
      "leftEyeLeft": integer,
      "rightEyeTop": integer,
      "rightEyeLeft": integer,
      "headPose": string,
      "detectFace": {
        "contentType": "image/jpeg",
        "data": string
      }
    }
  }
}
```

```
{
  "camera": string,
  "sequence": integer,
  "timestamp": integer,
  "progress": {
    "progressPercentage": integer,
    "perPoseStatistics":
    [{
      "namedPosition": string,
      "collectedPercentage": integer
    }]
  }
}
```

Final message

```
{
  "camera": string,
  "sequence": integer,
  "timestamp": integer,
  "done": {
    "progressPercentage": integer,
    "photos": [{
      "contentType": string,
      "data": string
    }],
    "description": string
  }
}
```

Properties of messages

Properties

Property Name	Value Type	Description
`camera`	string	Pid of camera.
`sequence`	integer	Sequential number of the message.
`timestamp`	integer	Server-side timestamp (with ms resolution) of the message.
`startOptions`	nested object	Initial information about sampling process. Sent only once in the beginning.
`startOptions.timeLimitSeconds`	integer	Duration of sampling process in seconds.
`startOptions.scheme`	nested object	Sampling scheme.
`startOptions.scheme.grid3x3`	nested object	Type of scheme. Only `"grid3x3"` supported.
`startOptions.scheme.grid3x3.areaOfInterest`	string	Type of area where a face should be detected. One of area types.
`detects`	nested object	Information about detects. Present if requested by client and the system succeed to aggregate detect information.
`detects.<detect id>`	nested object	Information about detect with identifier `<detect id>`.
`detects.<detect id>.id`	string	Detect id = `<detect id>`.
`detects.<detect id>.top`	integer	Coordinate of top horizontal line of face rectangle.
`detects.<detect id>.bottom`	integer	Coordinate of bottom horizontal line of face rectangle.
`detects.<detect id>.left`	integer	Coordinate of left vertical line of face rectangle.
`detects.<detect id>.right`	integer	Coordinate of right vertical line of face rectangle.
`detects.<detect id>.leftEyeTop`	integer	Top coordinate of face left eye. **May be missing**.

`detects.<detect id>.leftEyeLeft`	integer	Left coordinate of face left eye. **May be missing**.
`detects.<detect id>.rightEyeTop`	integer	Top coordinate of face right eye. **May be missing**.
`detects.<detect id>.rightEyeLeft`	integer	Left coordinate of face right eye. **May be missing**.
`detects.<detect id>.headPose`	string	Detected head pose. One of named positions.
`detects.<detect id>.detectFace`	nested object	Face from frame. Present if requested by client.
`detects.<detect id>.detectFace.contentType`	string	Content type of payload. Only `"image/jpeg"` supported.
`detects.<detect id>.detectFace.data`	string	Base64 encoded binary content of the image.
`progress`	nested object	Information about progress. Present if requested by client.
`progress.progressPercentage`	integer	Information about progress. It grows from 0 to 100 and corresponds to the proportion of frames with a face received from the camera and taken for sampling, from the number of frames with a face required for sampling process.
`progress.perPoseStatistics`	array	Progress of filling the pose with frames.
`progress.perPoseStatistics[].namedPosition`	string	Named head pose. One of named positions.
`progress.perPoseStatistics[].collectedPercentage`	integer	Information about progress for the head pose. It grows from 0 to 100 and corresponds to the proportion of frames with a face received from the camera and taken for sampling, from the number of frames with a face required for sampling process in the current head pose.
`done`	nested object	Information about sampling process results. Send only once on finish.
`done.progressPercentage`	integer	Information about progress. It grows from 0 to 100 and corresponds to the proportion of frames with a face received from the camera and taken for sampling, from the number of frames with a face required for sampling process.
`done.photos`	array	Optimal list of faces selected during training.
`done.photos[].contentType`	string	Content type of payload. Only `"image/jpeg"` provided.
`done.photos[].data`	string	Base64 encoded binary content of the image.

`done.description`	string	Describes result status. One of: • `"complete"`: process fully completed. • `"timeLimitExceeded"`: process partially completed and stopped due to exceeding time limit. • `"terminated"`: process partially completed and stopped due to termination request.

Errors Handling

The table below shows error codes and descriptions that are returned while emergency shutdown of the WebSockets by server side.
Note that closing message is limited by 125 characters.

Close Event content

Code	Full-text Reason Description	Description	Proposed client’s actions
1001	`"Going Away"`	Indicates that an endpoint is "going away", such as a server going down for some internal reason.	Contact to tech support.
1011	`"Internal Server Error"`	Indicates that a server is terminating the connection because it encountered an unexpected condition that prevented it from fulfilling the request.	Contact to tech support.
4000	`"Unknown query entry: {key}={value}"`	Optional query parameter name `{key}` and its value `{value}` was not understood by the server.	Check the parameter’s compliance with the service protocol.
4001	`"Unknown camera: {cameraPid}"`	Process couldn’t be started due to absence of camera with identifier `{cameraPid}`.	Check availability of the specified camera.
4002	`"Process already started from: {cameraPid}"`	Process couldn’t be started due to sampling process already started for camera with identifier `{cameraPid}`.	Terminate last sampling process or wait for some time.

Verification Service

Overview

Service implements verification scenario: is the person who stands in front of the camera a person from base? Subscription to verification events is carried out via WebSockets.
To subscribe client should open WebSocket with header `Sec-WebSocket-Protocol: verification`.
The opened WebSocket will receive messages about the result of the analysis of the frame from the selected camera.
The message is an aggregated event (frame + detection + correlation + successfull verification) in JSON.

There is an aggregation timeout for event and if the system does not collect an information during the specified time, then some information in the aggregated event may be missing.

A subscription for more then only successfull verification event will add a delay (about the aggregation timeout value) to the sending.

In the subscription request, you can specify what information you want to receive, how often to receive messages.

Clients can interfere with each other if they try to run the process from the same camera at the same time.

If an error occurs during validation of options, start or during process, the WebSocket is closed by the server with the corresponding code.

The subscription is terminated when the client closes the WebSocket (the server does not close the WebSocket on successful workflow even after the verification finished).

When starting verification, the client does not specify in advance the maximum verification time. The verification process continues for as long as the client keeps the connection open (except as described above when the process is stopped by the server for other reasons).

Correlations are primarily **debugging information** that can be completely correct interpreted only by the vendor’s specialists and reflects the features of the **currently used** face recognition algorithm that **may change** in the future. Therefore, you **SHOULD NOT** make any conclusions based on the received coefficients, except for getting the top of the most similar photos in the database according to the **current** face recognition algorithm.

Request

```
GET /cameras/{cameraPid}/verify
```

Path Parameters

Parameter Name	Value Type	Description
`cameraPid`	string	Identifier of the camera.

Required Query Parameters

Parameter Name	Value Type	Description
`person_id`	string	Identifier of the person to match.

Optional Query Parameters

Parameter Name	Value Type	Default Value	Description
`subscribe`	string	`"frame+detect"`	Defines what types of events should be sent besides successfull verification event. Represents composition of types joined by "+". Acceptable types are: • `"frame"`: subscription to frames. • `"detect"`: subscription to detects. • `"correlation"`: subscription to correlations. To receive messages only about successfull verification use empty string as a parameter value.
`frame_size`	string	`"medium"`	Size of frames from camera. Picture from the camera will be scaled to fit built-in dimensions. It converts so that the proportions do not change and do not become larger. Acceptable values are: • `"small"`: 320x240 pixels. • `"medium"`: 800x600 pixels. • `"large"`: 1400x1050 pixels. • `"original"`: original frame size.
`max_mps`	integer	not provided	Rate limit for incoming messages per seconds. Positive integer. If not provided produces all incoming events. If set over 100 reduced to 100.
`detect_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send detected faces. • `"jpeg"`: send detected faces as jpeg images.
`correlation_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send detected faces. • `"jpeg"`: send detected faces as jpeg images.

`threshold_name`	string	not provided	Name of the predefined verification threshold in the vendor implementation. Case insensetive. If not set — verification process will be started with threshold selected by implementation. List of available names should be provided by the vendor implementation.

Communication Protocol

Server Send Message Representation

Below is a scheme of an aggregated event in JSON.
Depending on the settings and system performance, some objects may be missing.

```
{
  "camera": string,
  "sequence": integer,
  "timestamp": integer,
  "frame": {
    "meta": {
      "width": integer,
      "height": integer
    },
    "image": {
      "contentType": "image/jpeg",
      "data": string
    }
  },
  "detects": {
    "<detect id>": {
      "id": <detect id>,
      "top": integer,
      "left": integer,
      "right": integer,
      "bottom": integer,
      "leftEyeTop": integer,
      "leftEyeLeft": integer,
      "rightEyeTop": integer,
      "rightEyeLeft": integer,
      "detectFace": {
        "contentType": "image/jpeg",
        "data": string
      }
    }
  },
  "correlations": {
    "<detect id>": {
      "id": <detect id>,
      "matches": [
        {
          "correlation": float,
          "personId": string,
          "photoId": string,
          "databaseFace": {
            "contentType": "image/jpeg",
            "data": string
          }
        }
      ]
    }
  },
  "terminated": {},
  "verified": {
    "id": <detect id>,
    "correlation": float,
    "threshold": float,
    "personId": float,
    "photoId": string
  }
}
```

Properties

Property Name	Value Type	Description
`camera`	string	Pid of camera.
`sequence`	integer	Sequential number of the message.
`timestamp`	integer	Server-side timestamp (with ms resolution) of the message.
`frame`	nested object	Frame info if requested by client.
`frame.meta`	nested object	Frame meta.
`frame.meta.width`	integer	Width of the image of frame.
`frame.meta.height`	integer	Height of the image of frame.
`frame.image`	nested object	Image of the frame.
`frame.image.contentType`	string	Content type of payload. Only `"image/jpeg"` provided.
`frame.image.data`	string	Base64 encoded binary content of the image.
`detects`	nested object	Information about detects. Presents if requested by client.
`detects.<detect id>`	nested object	Information about detect with identifier `<detect id>`.
`detects.<detect id>.id`	string	Detect id = `<detect id>`.
`detects.<detect id>.top`	integer	Coordinate of top horizontal line of face rectangle.
`detects.<detect id>.bottom`	integer	Coordinate of bottom horizontal line of face rectangle.
`detects.<detect id>.left`	integer	Coordinate of left vertical line of face rectangle.
`detects.<detect id>.right`	integer	Coordinate of right vertical line of face rectangle.
`detects.<detect id>.leftEyeTop`	integer	Top coordinate of face left eye. **May be missing**.
`detects.<detect id>.leftEyeLeft`	integer	Left coordinate of face left eye. **May be missing**.

`detects.<detect id>.rightEyeTop`	integer	Top coordinate of face right eye. **May be missing**.
`detects.<detect id>.rightEyeLeft`	integer	Left coordinate of face right eye. **May be missing**.
`detects.<detect id>.detectFace`	nested object	Face from frame. Presents if requested by client.
`detects.<detect id>.detectFace.contentType`	string	Content type of payload. Only `"image/jpeg"` provided.
`detects.<detect id>.detectFace.data`	string	Base64 encoded binary content of the image.
`correlations`	nested object	Information about correlations. Presents if requested by client.
`correlations.<detect id>`	nested object	Information about correlation for detect with identifier `<detect id>`.
`correlations.<detect id>.id`	string	Detect identifier = `<detect id>`.
`correlations.<detect id>.matches`	array	List of matches for correlation.
`correlations.<detect id>.matches[].correlation`	float	Correlation coefficient between detected face and photo from base.
`correlations.<detect id>.matches[].personId`	string	Person id correlated with.
`correlations.<detect id>.matches[].photoId`	string	Photo id from base correlated with.
`correlations.<detect id>.matches[].databaseFace`	nested object	Face image from base for correlated photo. Presents if requested by client.
`correlations.<detect id>.matches[].databaseFace.contentType`	string	Content type of payload. Only `"image/jpeg"` provided.
`correlations.<detect id>.matches[].databaseFace.data`	string	Base64 encoded binary content of the image.
`terminated`	nested object	Indicates that the process was terminated by another start request with the same camera pid. Send only once.
`verified`	nested object	Information about verification. Send only once on finish.
`verified.id`	string	Detect identifier = `<detect id>`.
`verified.correlation`	float	Correlation coefficient between detected face and photo from base.
`verified.threshold`	float	Current system threshold for correlation coefficient. Person is verified if the correlation is not less than the threshold.
`verified.personId`	string	Person id identified with.
`verified.photoId`	string	Photo id from base identified with.

Errors Handling

The table below shows error codes and descriptions that are returned if the emergency shutdown of the WebSocket on server side occurs.
Note that there is a limitation on the size of the message when closing WebSockets (125 characters).

Close Event content

Code	Full-text Reason Description	Description	Proposed client’s actions
1001	`"Going Away"`	Indicates that an endpoint is "going away", such as a server going down for some internal reason.	Contact to tech support.
1011	`"Internal Server Error"`	Indicates that a server is terminating the connection because it encountered an unexpected condition that prevented it from fulfilling the request.	Contact to tech support.
4000	`"Unknown query entry: {key}={value}"`	Optional query parameter name `{key}` and its value `{value}` was not understood by the server.	Check the parameter’s compliance with the service protocol.
4001	`"Unknown camera: {cameraPid}"`	Process couldn’t be started due to absence of camera with identifier `{cameraPid}`.	Check availability of the specified camera.
4002	`"Unknown person: {personId}"`	Process couldn’t be started due to absence of person with identifier `{personId}`.	Check existence of the specified person in base.
4003	`"Unknown threshold name: {threshold_name}"`	Process couldn’t be started due to absence of threshold named `{threshold_name}`.	Check the correctness of the threshold name.

IPA Service

Overview

Instant Photo Analytics (IPA) Service provides capability to analyze an image and generate analysis report.
Current supported content type of binary representation of a photo is `"image/jpeg"` only.

Methods

PhotoAnalysis

Correlates image with base and generate analysis report.

Request

```
POST /photo_analysis
```

Optional Query Parameters

Parameter Name	Value Type	Default Value	Description
`operation`	string	`"detect+identification"`	Defines what types of events should be sent. Should not be empty. Represents composition of types joined by "+". Acceptable types are: • `"detect"`: receive detect information. • `"correlation"`: receive correlation information. • `"identification"`: receive identification information.
`detect_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send detected faces. • `"jpeg"`: send detected faces as jpeg images.
`correlation_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send correlated faces. • `"jpeg"`: send correlated faces as jpeg images.

`identification_face`	string	`"none"`	Type of images to send. Acceptable values are: • `"none"`: do not send identified faces. • `"jpeg"`: send identified faces as jpeg images.
`max_persons`	integer	10	Maximum number of matches in correlation event. Positive integer.
`max_faces`	integer	not provided	Maximum number of faces in detect event. Positive integer.

Request Body

Content-Type

`multipart/form-data; boundary={your boundary}`

Payload

Binary

Success Response

Status code

`200 OK`

Content-Type

`application/vnd.com.smilart.helios.ipa.result+json`

Body

```
{
  "faces":[{
    "detect":{
      "face":{
        "top": integer,
        "left": integer,
        "right": integer,
        "bottom": integer,
        "leftEyeTop": integer,
        "leftEyeLeft": integer,
        "rightEyeTop": integer,
        "rightEyeLeft": integer
      },
      "cutting": {
        "top": integer,
        "left": integer,
        "right": integer,
        "bottom": integer,
        "leftEyeTop": integer,
        "leftEyeLeft": integer,
        "rightEyeTop": integer,
        "rightEyeLeft": integer,
        "detectedFace": {
          "contentType": "image/jpeg",
          "data": string
        }
      }
    },
    "correlations":[{
      "correlation": float,
      "personId": string,
      "photoId": string,
      "databaseFace": {
        "contentType": "image/jpeg",
        "data": string
      }
    }],
    "identification":{
      "threshold": float,
      "correlation": float,
      "personId": string,
      "photoId": string,
      "databaseFace": {
        "contentType": "image/jpeg",
        "data": string
      }
    }
  }]
}
```

Properties

Property Name	Value Type	Description
`faces`	array	List of information on found faces sorted by face size in descending order.
`faces[].detect`	nested object	Information about face detect. Presents when query parameter `operation` contains `detect` and a face was found.
`faces[].detect.face`	nested object	Information about a face at the original image.
`faces[].detect.face.top`	integer	Coordinate of top horizontal line of face rectangle.
`faces[].detect.face.bottom`	integer	Coordinate of bottom horizontal line of face rectangle.
`faces[].detect.face.left`	integer	Coordinate of left vertical line of face rectangle.
`faces[].detect.face.right`	integer	Coordinate of right vertical line of face rectangle.
`faces[].detect.face.leftEyeTop`	integer	Top coordinate of face left eye. **May be missing**.
`faces[].detect.face.leftEyeLeft`	integer	Left coordinate of face left eye. **May be missing**.
`faces[].detect.face.rightEyeTop`	integer	Top coordinate of face right eye. **May be missing**.
`faces[].detect.face.rightEyeLeft`	integer	Left coordinate of face right eye. **May be missing**.
`faces[].detect.cutting`	nested object	Information about cut face. Presents when query parameter `detect_face` is `jpeg`.
`faces[].detect.cutting.top`	integer	Coordinate of top horizontal line of cut face rectangle.
`faces[].detect.cutting.bottom`	integer	Coordinate of bottom horizontal line of cut face rectangle.
`faces[].detect.cutting.left`	integer	Coordinate of left vertical line of cut face rectangle.
`faces[].detect.cutting.right`	integer	Coordinate of right vertical line of cut face rectangle.
`faces[].detect.cutting.leftEyeTop`	integer	Top coordinate of cut face left eye. **May be missing**.
`faces[].detect.cutting.leftEyeLeft`	integer	Left coordinate of cut face left eye. **May be missing**.
`faces[].detect.cutting.rightEyeTop`	integer	Top coordinate of cut face right eye. **May be missing**.
`faces[].detect.cutting.rightEyeLeft`	integer	Left coordinate of cut face right eye. **May be missing**.
`faces[].detect.cutting.detectedFace`	nested object	Information about cut face image.
`faces[].detect.cutting.detectedFace.contentType`	string	Content type of payload. Only `"image/jpeg"` provided.
`faces[].detect.cutting.detectedFace.data`	string	Base64 encoded binary content of the image.
`faces[].correlations`	array	Information about correlations. Presents when query parameter `operation` contains `correlation` and a face was found.

`faces[].correlations[].correlation`	float	Correlation coefficient between detected face and photo from base.
`faces[].correlations[].personId`	string	Person identifier correlated with.
`faces[].correlations[].photoId`	string	Photo identifier from base correlated with.
`faces[].correlations[].databaseFace`	nested object	Face image from base for correlated photo. Presents when query parameter `correlation_face` is `jpeg`
`faces[].correlations[].databaseFace.contentType`	string	Content type of payload. Only `"image/jpeg"` provided.
`faces[].correlations[].databaseFace.data`	string	Base64 encoded binary content of the image.
`faces[].identification`	nested object	Information about identification. Presents when query parameter `operation` contains `identification` and a face was found.
`faces[].identification.threshold`	float	Current system threshold for correlation coefficient. Person is identified if the correlation is not less than the threshold. Always presents.
`faces[].identification.correlation`	float	Correlation coefficient between detected face and photo from base. Presents when a person was identified.
`faces[].identification.personId`	string	Person identifier identified with. Presents when a person was identified.
`faces[].identification.photoId`	string	Photo identifier from base identified with. Presents when a person was identified.
`faces[].identification.databaseFace`	nested object	Face image from base for identified photo. Presents when a person was identified and query parameter `identified_face` is `jpeg`
`faces[].identification.databaseFace.contentType`	string	Content type of payload. Only `"image/jpeg"` provided.
`faces[].identification.databaseFace.data`	string	Base64 encoded binary content of the image.

Error Responses

Incorrect request parameters

Status code: `400 Bad Request`

Request timeout

Status code: `408 Request Timeout`

Image file is too large

Status code: `413 Payload Too Large`

Unsupported image type

Status code: `422 Unprocessable Entity`

Service is overloaded

Status code: `429 Too Many Requests`

Adaptive Verification Service

Overview

Adaptive Verification (AV) provides the way to improve the user experience in verification process by **populating person base** by sampled photos during successful verification from cameras and **adjustment of verification thresholds**.

Side effects on other services

Impact on Person Management service

Being activated, AV service can modify list of person photos (add and delete photos), but it **can delete only those photos which were added by this service (sampled photos). This service will not delete any person's photos, added by another service (e.g. Person Management service)**.

All sampled photos will be accessible in Person Management service with special flag indicating whether the photo was added (sampled) by AV service or not.

Being deactivated this service does not delete sampled photos. Client can get rid of sampled photos via explicit remove requests at any moment.

Impact on Verification service

Being activated AV service will change thresholds for verification requests and provide additional person photos for verification.

Impact on other services

Other services will not take into account sampled photos and will not change their behavior because of the activity of AV service.

Methods

GetConfig

Gets the service configuration.

Request

```
GET /av/config
```

Request Body

Do not supply a request body with this method.

Success Response

Status code

`200 OK`

Content-Type

`application/vnd.com.smilart.helios.av.config+json`

Body

```
{
  "active": boolean
}
```

Properties

Property Name	Value Type	Description
`active`	boolean	Service activity status.

Error Responses

Request timeout

Status code: `408 Request Timeout`

Service is overloaded

Status code: `429 Too Many Requests`

SetConfig

Sets the service configuration.

Request

```
POST /av/config
```

Request Body

Content-Type

`application/vnd.com.smilart.helios.av.config+json`

Payload

JSON

```
{
  "active": boolean
}
```

Properties

Property Name	Required	Value Type	Description
`active`	true	boolean	Desired service activity status.True if service should sample person photos and adapt verification thresholds, otherwise false.

Success Response

Status code

`200 OK`

Content-Type

`application/vnd.com.smilart.helios.av.config+json`

Body

```
{
  "active": boolean
}
```

Properties

Property Name	Value Type	Description
`active`	boolean	Service activity status.

Error Responses

Incorrect request parameters

Status code: `400 Bad Request`

Request timeout

Status code: `408 Request Timeout`

Payload is too large

Status code: `413 Payload Too Large`

Unsupported payload type

Status code: `422 Unprocessable Entity`

Service is overloaded

Status code: `429 Too Many Requests`

RemoveAllPhotos

Removes all sampled photos.

Request

```
DELETE /av/photos
```

Request Body

Do not supply a request body with this method.

Success Response

Sampled photos deleted

Status code: `204 No Content`

Error Responses

Request timeout

Status code: `408 Request Timeout`

Service is overloaded

Status code: `429 Too Many Requests`

RemovePhotosByPerson

Removes all sampled photos of the person.

Request

```
DELETE /av/photos/persons/{personId}
```

Path Parameters

Parameter Name	Value Type	Description
`personId`	string	Identifier of the person.

Request Body

Do not supply a request body with this method.

Success Response

Sampled photos deleted

Status code: `204 No Content`

Error Responses

Request timeout

Status code: `408 Request Timeout`

Service is overloaded

Status code: `429 Too Many Requests`

RemovePhotosByCamera

Removes all sampled photos from the camera for every person.

Request

```
DELETE /av/photos/cameras/{cameraPid}
```

Path Parameters

Parameter Name	Value Type	Description
`cameraPid`	string	Identifier of the camera.

Request Body

Do not supply a request body with this method.

Success Response

Sampled photos deleted

Status code: `204 No Content`

Error Responses

Request timeout

Status code: `408 Request Timeout`

Service is overloaded

Status code: `429 Too Many Requests`

License Management Service

Overview

The service provides the following features to automate the process of license management of the system:

1. Get the server fingerprint, which must be transferred to the vendor to obtain a license.
2. Try to install an obtained license and store it on the server if the DRM (Digital Rights Management) subsystem allows the system to function under this license: has a valid signature, not expired, suitable for the current installed product, etc.
3. Request information about the stored license or get its raw version (for example, to backup it).

Stored license can be retrieved from the server, but may be neither readable no suitable by the implementation in case of some internal changes on the server (hardware changes, different product installation etc.).

A server can store no more than one license at a time. Successful license installation overwrites the stored version.

Resource Representation

License File

Content-Type: application/octet-stream

Binary representations of a license file.

License Information

Content-Type: application/vnd.com.smilart.helios.lm.license+json

JSON representations of a license:

```
{
  "activationDate":integer,
  "licensePeriodDays":integer,
  "licenseProduct":string,
  "serialNumber":integer,
  "checks":{
    "passed":boolean,
    "productCheck":{
      "passed":boolean,
      "serverProduct":string
    },
    "activityTimeIntervalCheck":{
      "passed":boolean,
      "serverDate":integer
    },
    "fingerprintCheck":{
      "passed":boolean
    }
  }
}
```

Properties

Property Name	Required	Value Type	Description
`activationDate`	`false`	integer	License activation date in ms. If absent, then the license is not limited in time.
`licensePeriodDays`	`false`	integer	The number of days that the license will work from the activation date (include activation date). If absent, then the license is timeless.
`licenseProduct`	`true`	string	Product name that has been installed.
`serialNumber`	`true`	integer	License Serial Number.
`checks`	`true`	nested object	License checks. The presence of some not passed checks denotes the system in a non operability state.
`checks.passed`	`true`	boolean	It is an aggregate value for all checks. True if all checks are passed.
`checks.productCheck`	`true`	nested object	Information about the correspondence of the product name from the license and the installed product name.
`checks.productCheck.passed`	`true`	boolean	True, if the product name from the license and the installed product name match. Otherwise false.
`checks.productCheck.serverProduct`	`true`	string	Name of the installed product.

`checks.activityTimeIntervalCheck`	`false`	nested object	Information about correspondence of the current server time and license activity time interval. If present, the license may be inactive at some moment according to licensing scheme.
`checks.activityTimeIntervalCheck.passed`	`true`	boolean	True, if at the time of the request the server time is in license activity period. Otherwise false.
`checks.activityTimeIntervalCheck.serverDate`	`true`	integer	Date on the server in ms.
`checks.fingerprintCheck`	`false`	nested object	Information about the correspondence of the server fingerprint to the server fingerprint in the license.
`checks.fingerprintCheck.passed`	`true`	boolean	True, if server fingerprints match. Otherwise false.

Methods

SetLicense

Installs a new license. By default, license will be accepted only if the **license allows the current installed product to operates at the moment**:

- License is in its activity time interval.
- License is suitable for this server.

These assertions could be bypassed with optional query request parameters.

Request

```
PUT /lm/license
```

Optional Query Parameters

Parameter Name	Value Type	Default Value	Description
`dry_run`	boolean	`false`	If true: license would NOT really installed on the server. Use it to check a license applicability to the server.
`allow_activation_in_future`	boolean	`false`	If true: bypasses a license activity time interval check if the license activity period has not begun and has not expired.

Request Body

Content-Type

`multipart/form-data; boundary={your boundary}`

Payload
  Binary

Success Response

Status code
  `200 OK`

Content-Type
  `application/vnd.com.smilart.helios.lm.license+json`

Body
  Representation of the resource

Error Responses

Bad Request
  Status code: `400 Bad Request`

Bad license
  Description: The license was not installed due to the fact that some not bypassed checks have not passed.
  Status code: `431 Bad License`
  Content-Type: `application/vnd.com.smilart.helios.lm.license+json`
  Body: Representation of the resource

Unknown license format
  Description: License was not installed due to the inability to correctly parse the license file.
  Status code: `432 Unknown License Format`

GetFingerprint

Gets a server fingerprint.

Request

```
GET /lm/fingerprint
```

Request Body

Do not supply a request body with this method.

Success Response

Status code
  `200 OK`

Content-Type
  `application/octet-stream`

Body

`Binary`

GetLicense

Downloads the installed license as JSON representation of the resource, or as an origin license file.

Request

```
GET /lm/license
```

Request Headers

Header Name	Value Type	Description
`Accept`	string	Media type(s) that is/are acceptable for the response. Acceptable types are: • **application/vnd.com.smilart.helios.lm.license+json**; • **application/octet-stream**. If not provided, **application/vnd.com.smilart.helios.lm.license+json** is used.

Request Body

Do not supply a request body with this method.

Success Response for application/vnd.com.smilart.helios.lm.license+json

Status code

`200 OK`

Content-Type

`application/vnd.com.smilart.helios.lm.license+json`

Body

Representation of the resource

Error Responses for application/vnd.com.smilart.helios.lm.license+json

License not found

Status code: `404 Not Found`

Unknown license format

Description: License cannot be presented due to the inability to correctly parse the license file.
Status code: `532 Unknown License Format`

Success Response for application/octet-stream

Status code

`200 OK`

Content-Type

`application/octet-stream`

Body

Representation of the resource

Error Responses for application/octet-stream

License Not Found

Status code: `404 Not Found`