OMNI Platform 1.9.3

API Reference

Contents

1 Object Model

Sample is a JSON-object that represents and stores the result of image processing (a face image, face attributes and/or a corresponding biometric template) used for face recognition.

Profile is an object that contains metadata about a person, person's photo (avatar), activity and the main sample - a biometric template used for detection. The main sample is used by an agent to check for a profile in the database, and by a server - to determine a profile which the new activities should be attached to. In case the sample from the new activity has better quality than the current main sample, the main sample can be replaced by the system.

Profile Group is an object used for profile grouping and receiving notifications. It contains a number of profile IDs, profile group metadata and attached triggers. If the detected activity belongs to a profile from this profile group, the system enables this attached trigger with the condition to create and send a notification.

Activity is an object created when a person is detected by an agent. When a person appears in the frame, the agent takes a photo, checks it for quality, identifies the person and collects activity data. After that, the agent checks the activity sample and the main samples for the similarity of the biometric template.

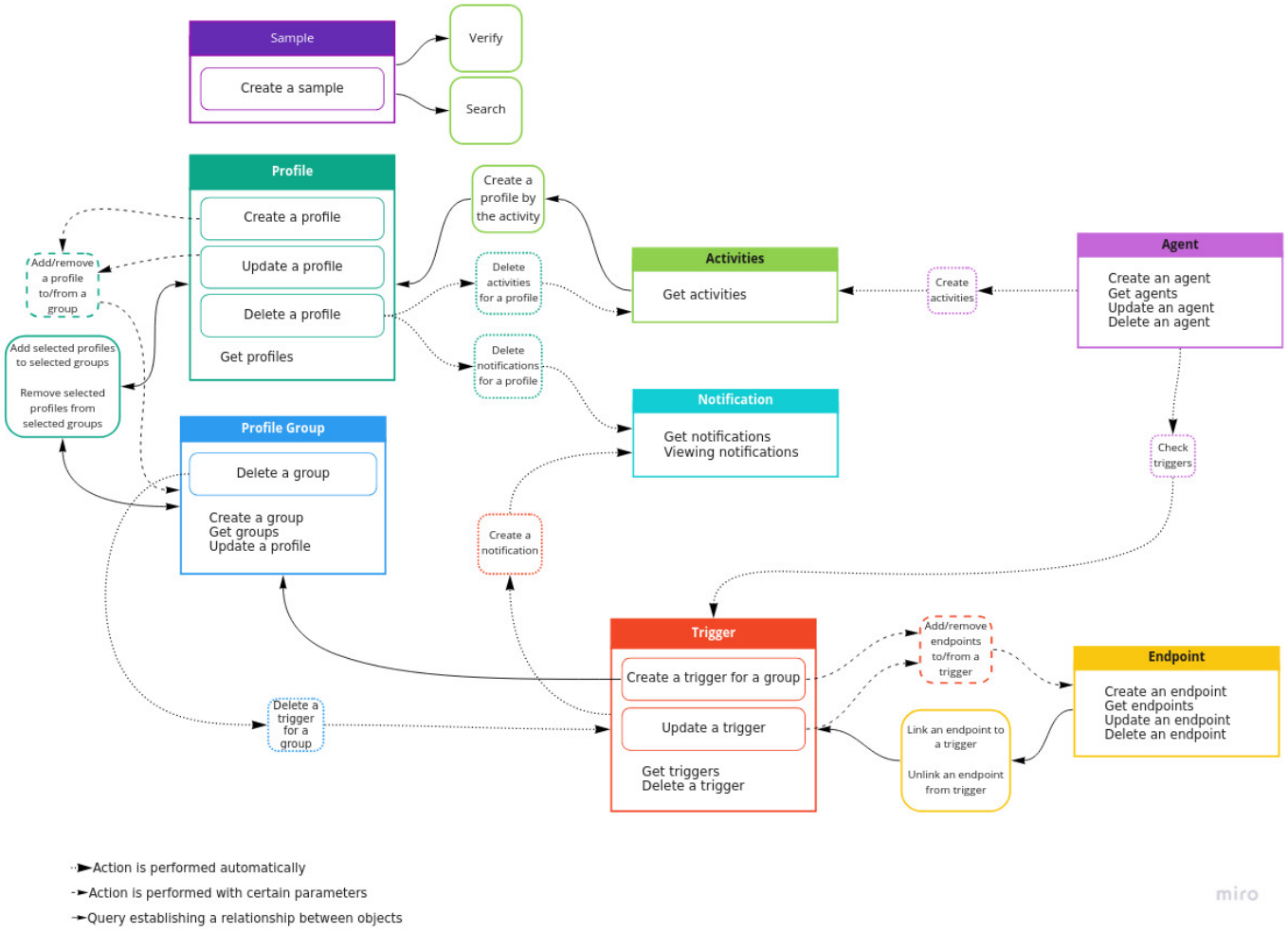
Notification is automatically generated when the system receives activity that satisfies the trigger condition. For a certain profile group this condition is as follows: the biometric data of received activity match the biometric data of a profile from this profile group.

Endpoint is an object that contains data about the point where notifications are sent to. To send notifications, you need to attach an endpoint to the trigger.

Trigger is the object needed to create and send a notification to an endpoint if the activity data satisfies the trigger condition. Trigger contains a trigger condition, a set of endpoints, and ID of the profile group it is attached to.

Agent is an object that stores information about edge device processing instances.

The object diagram that shows relation between the objects is given below:



2 Integration API

All available face recognition operations with OMNI Platform objects (such as Profiles, Profile Groups, Samples, etc.) can be performed using Integration API. Access to API is enabled via GraphQL - an open-source API data query and manipulation language and a runtime for running existing data queries. With GraphQL you can send requests to OMNI Platform and receive the requested data that can be integrated into your client application. For more information about GraphQL see the following sources:

- Introduction to GraphQL
- How to GraphQL
- Guides and Best Practices

To get started with API, sign in to OMNI Platform account. At the home page of the web interface click on the Platform *API* button at Resources panel to move to GraphQL interactive console.

To use Integration API in your application, add *Token* to HTTP request header and send request to the `https://cloud.3divi.ai/api/v2/`. To get the token, follow the instructions below:

1. Open GraphQL interactive console by clicking *Platform API* button at *Resources* widget of web interface.
2. Copy the request below to the console and click the Execute *Query* button.

```
{
  me {
    workspaces {
      accesses {
        id
      }
    }
  }
}
```

3.As a result, GraphQL returns the response with a token (id):

```
{
   "data":{
      "me":{
         "workspaces":[
            {
               "accesses ":[
                  {
                     "id":"b3a4f990-d2d0-4989-8684-41cb88f3d0f9"
                  }
               ]
            }
```

```
          ]
       }
    }
}
```

The user registered with PLATFORM_DEFAULT_EMAIL can get API access token via a command:

```
$ ./setup/get-token.sh
```

For more details on how to get and use the token, follow the Administrator Guide.
Except for GraphQL you can send the requests using cURL. cURL template for sending API requests is given below:

```
curl --location --request POST "<url>" --header "token: <your access token>" --header "Content-Type: application/json" --data-raw "{\"query\":\"<GraphQL query or mutation>\",\"variables\":{<GraphQL variables>}}"
```

where `<url>` and `<your access token>` are fields for the URL of your server and access token, respectively. Note that your cURL requests will not be sent without an access token.

Example cURL request: This request is used to get a list of first 5 profiles from the database.

```
curl --location --request POST "<url>" --header "token: b3a4f990-d2d0-4989-8684-41cb88f3d0f9" --header "Content-Type: application/json" --data-raw "{\"query\":\"query {profiles(limit: 5, offset: 0) {totalCount, collectionItems {id, info}}}\",\"variables\":{}}"
```

API returns the following result:

```
{
  "data": {
    "profiles": {
      "totalCount": 5,
      "collectionItems": [
        {
          "id": "195ed5fe-580e-476f-8496-befdb0003d85",
          "info": {
            "age": 46,
            "gender": "MALE",
            "avatar_id": "9945328f-ebd8-4683-bde5-25284686f439",
            "main_sample_id": "83800c22-5ed3-4c9f-91bf-ce79c0de747a"
          }
        },
        {
          "id": "8d46bf22-5d24-4e4f-bfd3-e215cbde53f0",
          "info": {
            "age": 24,
            "gender": "FEMALE",
            "avatar_id": "1e7c8293-75e8-485a-8861-415eec854011",
            "main_sample_id": "6cbd4c9a-7cd0-4c43-9561-387be9dff6f3"
          }
        },
        {
```

```
          "id": "944ab599-bd6d-47bf-b8e3-ed201a4e6530",
          "info": {
            "age": 31,
            "gender": "MALE",
            "avatar_id": "9f36bee0-efdf-42fc-a059-c60d5c7e6da9",
            "main_sample_id": "db9998a4-4331-41b5-9abd-30499b881be8"
          }
        },
        {
          "id": "aa40fb31-96d3-421b-96ae-05566fc57bee",
          "info": {
            "age": 35,
            "gender": "MALE",
            "avatar_id": "ae61ec9f-2953-4f24-86d8-7718df6f9ff5",
            "main_sample_id": "33a72136-175b-4361-8444-e1e0c4ce04d4"
          }
        },
        {
          "id": "c5e1b7d9-b446-4739-a3d0-a8cf8d717c70",
          "info": {
            "age": 21,
            "gender": "MALE",
            "avatar_id": "9938407f-d101-4005-8d30-4b1ce8319bc7",
            "main_sample_id": "f2b3d23a-6f73-4d9b-97aa-13edaf5e82f7"
          }
        }
      ]
    }
  }
}
```


The requests and responses for API testing in GraphQL are given below.

2.1 Face Recognition

2.1.1 General Incorrect Input Errors

Errors Occurred When Uploading an Image Via API:

Invalid base64 string:

```
{
  "message": "image file is truncated (21 bytes not processed)"
}
```

Image not transferred in base64:

```
{
  "message": "Expected value of type 'CustomBinaryType', found \"wrong_data\";
}
```

No image transmitted:

```
{
  "message": "Sample Data is not valid",
  "code": "0xc69c44d4"
}
```

Image uploaded in an incorrect format:

```
{
  "message": "Image decode failed"
}
```

No faces detected in the image:

```
{
  "message": "No faces found",
  "code": "0x95bg42fd"
}
```

Too big image size:

```
Bad Request (400)
```

Errors Occurred in Filtering, Pagination and Sorting of Objects:

Transmitted filters are not in dictionary format:

```
{
  "message": "'str' object has no attribute 'keys'"
}
```

Invalid field used for filtering:

```
{
  "message": "Cannot resolve keyword '' into field. Choices are: creation_date, id,
info, last_modified, link_to_label, person, person_id, profile_groups, samples,
workspace, workspace_id"
}
```

Using an invalid field to filter sub-objects or a function over a field:

```
{
  "message": "Unsupported lookup '1' for UUIDField or join on the field not
permitted."
}
```

Negative value specified in the pagination fields:

```
{
  "message": "Negative indexing is not supported."
}
```

Invalid field used for ordering:

```
{
  "message": "Cannot resolve keyword '' into field. Choices are: creation_date, id,
info, last_modified, link_to_label, person, person_id, profile_groups, samples,
workspace, workspace_id"
}
```

Remaining General Errors:

Object ID given in a non- UUID format:

```
{
  "message": "“%(value)s” is not a valid UUID."
}
```

Transmitted JSON did not pass validation by JSON schema:

```
{
  "message": "Invalid JSON request"
}
```

2.1.2 Face Detection

This query allows you to detect multiple faces in the image and get information about these faces (such as gender, age, emotions, liveness, mask presence, etc.). You can use this query if you're interested in detection result only, without saving it in your local database.

```
detect(
image: CustomBinaryType!
pupils: [EyesInput!] = null): JSON!
```

image: CustomBinaryType! : Base64 encoded image

pupils: [EyesInput!] : To increase face detection accuracy, you can specify X and Y coordinates of eye pupils.

EyesInput!

- **leftPupil: PointInputType!**
  - x: Float!
  - y: Float!
- **rightPupil: PointInputType!**
  - x: Float!
  - y: Float!

JSON! : API returns a sample in JSON format that contains the following parameters:

- **image :** Base64 encoded image
- **id :** The ordinal number of a face in the image
- **class :** Object class name, e.g., face
- **template :** A unique set of biometric features extracted from a face image. Templates are used to compare two face images and to determine a degree of similarity.
- **bbox :** A rectangle that surrounds a face and specifies face position and size in the original image.
- **crop image :** To determine face features, the system crops the uploaded image to extract face images. These extracted face images are called crop images.
- **keypoints :** A collection of 21-point face landmarks pointing to the positions of face components.
- **rotation angles :** Yaw (rotation along Z axis), pitch (rotation along Y axis), and roll (rotation along X axis) rotation angles. OMNI Platform algorithm allows to detect faces in the following range of angles: yaw [-60; 60], pitch [-60; 60], roll [-30; 30].
- **age : Age in years**
- **gender: Possible gender of a person in the image**
- **emotions :** Facial emotions described in a form of confidence
- **liveness :** Integrated liveness algorithm allows to detect if a person in the image is real or fake.
- **mask presence :** The system can detect if a person in the image is wearing a mask.

Example Request:

```
{
  detect(image: "your image in Base64 format")
}
```

Example Response:


```
{
  "data": {
    "detect": {
      "$image": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAUDBAQEAwUE***",
      "objects@common_capturer_uld_fda": [
        {
          "id": 1,
          "class": "face",
          "templates": {
            "$template10v100": "T5JnWgcfMPHQANMdP8NA8FwPQJLc5TDfcJFVFS/tYgEw9yN***"
          },
          "bbox": [
            0,
            0.21308482869466144,
            1,
            0.9869169769287109
          ],
          "$cropImage": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgH***",
          "keypoints": {
            "left_eye_brow_left": {
              "x": 0.1554061550564236,
              "y": 0.3466766866048177
            },
            "left_eye_brow_up": {
              "x": 0.25802788628472223,
              "y": 0.3236322784423828
            },
            "left_eye_brow_right": {
              "x": 0.3854702419704861,
              "y": 0.3430420430501302
            },
            "right_eye_brow_left": {
              "x": 0.6136109754774306,
              "y": 0.345932362874349
            },
            "right_eye_brow_up": {
              "x": 0.7370150417751736,
              "y": 0.3286321258544922
            },
            "right_eye_brow_right": {
              "x": 0.8359405517578125,
              "y": 0.35814239501953127
            },
            "left_eye_left": {
              "x": 0.21608330620659721,
              "y": 0.4157813008626302
            },
            "left_pupil": {
              "x": 0.30088453504774304,
              "y": 0.4109149678548177
            },
            "left_eye_right": {
              "x": 0.3838650173611111,
              "y": 0.4230572001139323
            },
            "right_eye_left": {
              "x": 0.6079323323567708,
              "y": 0.4287389628092448
            },
```

```
            "right_pupil": {
              "x": 0.6891607666015624,
              "y": 0.4170384724934896
            },
            "right_eye_right": {
              "x": 0.7724110243055555,
              "y": 0.4233682759602865
            },
            "left_ear_bottom": {
              "x": 0.1043272230360243,
              "y": 0.5988115437825521
            },
            "nose_left": {
              "x": 0.3953646511501736,
              "y": 0.5805702209472656
            },
            "nose": {
              "x": 0.4884576416015625,
              "y": 0.5780504862467448
            },
            "nose_right": {
              "x": 0.5848392740885416,
              "y": 0.5861359659830729
            },
            "right_ear_bottom": {
              "x": 0.8565327962239583,
              "y": 0.607865956624349
            },
            "mouth_left": {
              "x": 0.33830030653211807,
              "y": 0.6955980936686198
            },
            "mouth": {
              "x": 0.4904075113932292,
              "y": 0.7090469868977864
            },
            "mouth_right": {
              "x": 0.6333636474609375,
              "y": 0.6958428955078125
            },
            "chin": {
              "x": 0.4895579020182292,
              "y": 0.8763695271809896
            }
          },
          "age": 23,
          "emotions": {
            "neutral": 0.9834117889404297,
            "angry": 0.013813868165016174,
            "happy": 0.002322095213457942,
            "surprised": 0.00045228638919070363
          },
          "gender": "FEMALE",
          "liveness": {
            "value": "FAKE",
            "confidence": 0.801980197429657
          },
          "angles": {
            "yaw": -0.8611235618591309,
            "pitch": -15.951218605041504,
            "roll": 1.7437981367111206
```

```
          },
          "mask": {
            "value": false,
            "confidence": 1
          }
        }
      ]
    }
  }
}
```

2.1.3 Create a Sample

This mutation allows you to create samples with face attributes (age, gender, emotions, liveness, mask etc.).

```
createSample(
anonymousMode: Boolean = false
image: CustomBinaryType = null
pupils: [EyesInput!] = null
sampleData: JSON = null): [SampleOutput!]!
```

anonymousMode: Boolean = false : If you have to work with anonymous data, you can set *anonymousMode* to true (by default, it's set to false). In this case, the image is not saved at OMNI Platform Server.

image: CustomBinaryType : Base64 encoded image

pupils: [EyesInput!] : To increase face detection accuracy, you can specify X and Y coordinates of eye pupils.

EyesInput!

- **leftPupil: PointInputType!**
  - x: Float!
  - y: Float!
- **rightPupil: PointInputType!**
  - x: Float!
  - y: Float!

sampleData: JSON : Sample data is the face detection result, not saved at the database.

SampleOutput! : API returns JSON file with the following fields:

- **id :** Sample ID.
- **creationDate :** Sample creation date in ISO 8601 format with time zone
- **lastModified :** Last sample modification date in ISO 8601 format with time zone
- **data :** Image and/or template and/or detection result in sample format

The returned sample is automatically saved at OMNI Platform Server (if *anonymousMode* is set to false) and can be used to verify a face to a face or seach a person in the database.

Incorrect Input Errors:

No data has been entered to create a sample:

```
{
  "message": "One of the parameters sampleData or sourceImage is required",
  "code": "0xnf5825dh"
}
```

Invalid transmitted sample data:

```
{
  "message": "argument should be a bytes-like object or ASCII string, not 'NoneType'"
}
```

Transmitted wrong pupils coordinates:

```
{
  "message": "0x8905a659: Assertion '( transform_m(0, 0) * transform_m(0,
0) + transform_m(0, 1) * transform_m(0, 1) + transform_m(1, 0) * transform_m(1,
0) + transform_m(1, 1) * transform_m(1, 1) ) > 1e-5' failed, error code:
0x8905a659. wrap code: 0x7df96daf."
}
```

Example Request:

```
mutation {
  createSample(image/sampleData: "your image in Base64 format or sample data") {
    id
    creationDate
    lastModified
    data
  }
}
```

Example Response:

```
{
  "data": {
    "createSample": [
      {
        "id": "3f7352c9-94be-4449-aaa0-b93a3812e1c6",
        "creationDate": "2022-04-28T06:23:45.902315+00:00",
        "lastModified": "2022-04-28T06:23:46.450993+00:00",
        "data": {
          "$image": {
            "id": "3403c21e-44e1-4ea4-9e57-2d2c0933861e"
          },
          "objects@common_capturer_uld_fda": [
            {
              "id": 1,
              "age": 23,
```

```
            "bbox": [
              0,
              0.21308482869466144,
              1,
              0.9869169769287109
            ],
            "mask": {
              "value": false,
              "confidence": 1
            },
            "class": "face",
            "angles": {
              "yaw": -0.8611235618591309,
              "roll": 1.7437981367111206,
              "pitch": -15.951218605041504
            },
            "gender": "FEMALE",
            "emotions": {
              "angry": 0.013813868165016174,
              "happy": 0.002322095213457942,
              "neutral": 0.9834117889404297,
              "surprised": 0.00045228638919070363
            },
            "liveness": {
              "value": "FAKE",
              "confidence": 0.801980197429657
            },
            "keypoints": {
              "chin": {
                "x": 0.4895579020182292,
                "y": 0.8763695271809896
              },
              "nose": {
                "x": 0.4884576416015625,
                "y": 0.5780504862467448
              },
              "mouth": {
                "x": 0.4904075113932292,
                "y": 0.7090469868977864
              },
              "nose_left": {
                "x": 0.3953646511501736,
                "y": 0.5805702209472656
              },
              "left_pupil": {
                "x": 0.30088453504774304,
                "y": 0.4109149678548177
              },
              "mouth_left": {
                "x": 0.33830030653211807,
                "y": 0.6955980936686198
              },
              "nose_right": {
                "x": 0.5848392740885416,
                "y": 0.5861359659830729
              },
              "mouth_right": {
                "x": 0.6333636474609375,
                "y": 0.6958428955078125
              },
              "right_pupil": {
```

```
            "x": 0.6891607666015624,
            "y": 0.4170384724934896
          },
          "left_eye_left": {
            "x": 0.21608330620659721,
            "y": 0.4157813008626302
          },
          "left_eye_right": {
            "x": 0.3838650173611111,
            "y": 0.4230572001139323
          },
          "right_eye_left": {
            "x": 0.6079323323567708,
            "y": 0.4287389628092448
          },
          "left_ear_bottom": {
            "x": 0.1043272230360243,
            "y": 0.5988115437825521
          },
          "right_eye_right": {
            "x": 0.7724110243055555,
            "y": 0.4233682759602865
          },
          "left_eye_brow_up": {
            "x": 0.25802788628472223,
            "y": 0.3236322784423828
          },
          "right_ear_bottom": {
            "x": 0.8565327962239583,
            "y": 0.607865956624349
          },
          "right_eye_brow_up": {
            "x": 0.7370150417751736,
            "y": 0.3286321258544922
          },
          "left_eye_brow_left": {
            "x": 0.1554061550564236,
            "y": 0.3466766866048177
          },
          "left_eye_brow_right": {
            "x": 0.3854702419704861,
            "y": 0.3430420430501302
          },
          "right_eye_brow_left": {
            "x": 0.6136109754774306,
            "y": 0.345932362874349
          },
          "right_eye_brow_right": {
            "x": 0.8359405517578125,
            "y": 0.35814239501953127
          }
        },
        "templates": {
          "$template10v100": {
            "id": "708a5da8-104e-4b83-9f26-ec0329e08b89"
          }
        },
        "$cropImage": {
          "id": "3cf0792f-1dde-4e8b-841a-c18330080d21"
        },
        "quality": -540.9627075195312
```

```
            }
          ]
        }
      }
    ]
  }
}
```

2.1.4 Face Verification

Query `verify()` is used to compare two samples and verify whether two face images belong to the same person or whether one face image belongs to the person.

```
verify(
sourceImage: CustomBinaryType = null
sourceSampleData: JSON = null
sourceSampleId: ID = null
targetSampleId: ID!): MatchResult!
```

sourceImage: CustomBinaryType : Base64 encoded image

sourceSampleData: JSON : Face detection result, not saved at the database.

sourceSampleId: ID : Sample ID to be compared with target sample ID

targetSampleId: ID : Sample ID to be compared with a source image, source sample data or source sample ID

MatchResult : Verification result that contains the following parameters:

- **distance :** The parameter shows the distance between the compared template vectors. The shorter the distance, the higher the verification rate is.
- **faR :** False acceptance rate (FAR) shows the system resistance to false acceptance errors. Such an error occurs when the biometric system recognizes a new face as previously detected one. This rate is measured by the number of false-acceptance recognitions divided by the total number of recognition attempts.
- **frR :** False rejection rate (FRR). When a system fails to recognize previously detected face, false rejection occurs. The rate shows the percentage of recognition attempts with false rejection result.
- **score :** The parameter shows the verification rate from 0 (0%) to 1 (100%)

Incorrect Input Errors:

A comparison object is not passed or an ambiguous interpreted combination is passed:

```
{
  "message": "One of the parameters sourceSampleData or sourceSampleId or
sourceImage is required",
  "code": "0x963fb254"
}
```

No sample found by transmitted id:

```
{
  "message": "Sample matching query does not exist."
}
```

Transmitted source sample data is invalid:

```
{
  "message": "'objects@common_capturer_uld_fda'"
}
```

Example Request:

```
{
  verify(
    sourceSampleId: "fa76e8a4-3c90-4007-a72f-94d5fc655c36"
    targetSampleId: "a2d852e8-aa00-4403-bc5d-f8b94cc183ca"
  ) {
    distance
    faR
    frR
    score
  }
}
```

Example Response:

```
{
  "data": {
    "verify": {
      "distance": 0,
      "faR": 0,
      "frR": 1,
      "score": 1
    }
  }
}
```

2.1.5 Face Identification

This query allows you to search for a person in the database. Function search() is used to compare one sample with all other samples in the database.

```
search(
confidenceThreshold: Float = 0
maxNumOfCandidatesReturned: Int = 5
scope: ID = null
sourceImage: CustomBinaryType = null
sourceSampleData: JSON = null
sourceSampleIds: [ID!] = null): [SearchType!]!
```

confidenceThreshold: Float = 0 : To exclude matches with low confidence from the returned result, use the *confidenceThreshold* parameter (min: 0, max: 1; default: 0)

maxNumOfCandidatesReturned: Int = 5 : To set the max number of returned candidates, specify the value for the *maxNumOfCandidatesReturned* parameter (min: 1, max: 100). By default, 5 closest candidates are returned.

scope: ID : By default, a person is searched in an entire database. To get matches from a specific List, set the List id in the scope parameter.

sourceImage: CustomBinaryType : Base64 encoded image

sourceSampleData: JSON : Face detection result, not saved at the database.

sourceSampleIds: [ID!] : Sample IDs

SearchType! : The result is a list of candidates for each requested sample in descending order of confidence. Search result contains the following parameters:

- **template**
- **searchResult**
- **PersonSearchResult**
    - **sample:** SampleOutput! (id: ID! , creationDate: DateTime , lastModified: DateTime , data: JSON!)
    - **profile:** ProfileOutputData! (id: ID! , info: JSON! , lastModified: DateTime! , creationDate: DateTime! , personId: ID! , mainSample: SampleOutput)
    - **matchResult:** MatchResult! (distance: Float! , faR: Float! , frR: Float! , score: Float!)

Note: the source data is compared with *Profiles* from the server, not with the samples. So, before you start searching, make sure you have created at least 1 profile.

Incorrect Input Errors:

A comparison object is not passed or an ambiguous interpreted combination is passed:

```
{
  "message": "One of the parameters sourceSampleData or sourceSampleId or sourceImage
is required",
  "code": "0x963fb254"
}
```

The confidenceThreshold value is transmitted out of range:

```
{
  "message": "Confidence threshold must be between 0 and 1",
  "code": "0xf47f116a"
}
```

The maxNumOfCandidatesReturned value is transmitted out of range:

```
{
  "message": "Max num of candidates must be between 1 and 100",
  "code": "0xf8be6762"
}
```

Transmitted source sample data is invalid:

```
{
  "message": "'objects@common_capturer_uld_fda'"
}
```

Example Request:

```
{
  search(sourceSampleIds: "fa76e8a4-3c90-4007-a72f-94d5fc655c36") {
    template
    searchResult {
      matchResult {
        distance
        faR
        frR
        score
      }
      sample {
        id
      }
      profile {
        id
        info
      }
    }
  }
}
```

Example Response:

```
{
  "data": {
    "search": [
      {
        "template":
"T5JnWuAN4j5MHWRP7tDj7/AOcg9S8AWw4AAQZwJAHeMRw1MUGfkxFTTVUALbWmAHBHJTBKM9LfMgwCUvXlQg
AizEAzwP0BvkQDHQfykA4ZDvHrkDFhCvD3Eh73TSD+UQTMDSQBPj4DAtYBEi/wDQMH9A8PMObwQQAFHDAgMOA
QHjMB+vEHc2Q/ACH/EHAw+9MPUDA+ImodYPbeKv7Q9/Xw8wD39CPQpQd/wrfwPcZRDSIgkj0PEA8L0NZNoAId
ACER8Q/hAQPQAa8UDdovBN6eXBAHUwQOC7DjECPjBQ5FJBE+MNPeAB1OUPEW787jBCAsHQkBCSRXD0MHEP4E0
O49IgQ7MBGyBEDyMldtHx4RT0MH9MMReb0g==",
        "searchResult": [
          {
            "matchResult": {
```

```
              "distance": 0,
              "faR": 0,
              "frR": 1,
              "score": 1
            },
            "sample": {
              "id": "805be807-dd89-4265-9ee7-bbdc19473136"
            },
            "profile": {
              "id": "fd606132-9757-419b-97d6-d8cdd67ed476",
              "info": {
                "age": 25,
                "gender": "MALE",
                "main_sample_id": "805be807-dd89-4265-9ee7-bbdc19473136"
              }
            }
          },
          {
            "matchResult": {
              "distance": 9356,
              "faR": 0.31920063495635986,
              "frR": 0,
              "score": 0.000001847743988037109 4
            },
            "sample": {
              "id": "c18b3a5a-ccfb-4785-9248-b7ce90052754"
            },
            "profile": {
              "id": "218db30f-b6ff-4a46-a83d-a0701005a12b",
              "info": {
                "age": 23,
                "gender": "FEMALE",
                "main_sample_id": "c18b3a5a-ccfb-4785-9248-b7ce90052754"
              }
            }
          }
        ]
      }
    ]
  }
}
```


Note that if the database contains no profiles created before search start, the profile field in the returned result will be blank.

2.2 Profiles

2.2.1 Get Profiles

Query `profiles()` allows you to get a list of all created profiles.

```
profiles(
filter: JSONString = null
ids: [ID] = null
limit: Int = null
offset: Int = null
order: [String] = null): ProfilesCollection!
```

filter: JSON: You can filter a profile list by specifying one or several parameters:

- **creation_date:** Exact profile creation date in ISO 8601 format
- **id:** ID assigned to a profile from the database
- **info:** Object with the following parameters:
    - **age:** Profile age. For example, a request with a parameter **info_age:28** returns a list of profiles with age equal to 28.
    - **gender:** Profile gender. For example, a request with a parameter **info_gender: “MALE”** returns a list of male profiles.
    - **main_sample_id:** ID assigned to the best profile sample
    - **avatar_id:** ID of profile avatar
- **last_modified**: Exact date of the last profile modification in ISO 8601 format
- **link_to_label:**
- **person:**
- **person_id:** ID assigned to a profile from the database
- **profile_groups**: List of group IDs
- **samples:** Sample ID
- **workspace:** Workspace ID
- **workspace_id:** Workspace ID

ids: [ID]: To get a list with certain profiles, specify their IDs in the list.

limit: Int: The parameter allows to get the first n profiles from the list.

offset: Int: The parameter allows to exclude the first n profiles from the list.

order: [String]: You can sort the list by specifying the value for the following parameters: **creation_date, id, info, last_modified, link_to_label, person, person_id, profile_groups, samples, workspace,workspace_id.**

ProfilesCollection!: The query result is a profile list that contains the following parameters:

- **totalCount:** Number of returned profiles
- **collectionItems: [ProfileOutput!]**
  - **id:** Profile ID
  - **lastModified:** Last profile modification date in ISO 8601 format
  - **creationDate:** Profile creation date in ISO 8601 format
  - **info:** Profile information (approximate age, gender, avatar ID, the best sample ID)
  - **samples**: List of profile samples
  - **profileGroups:** List of profile groups
  - **mainSample:** The best profile sample
  - **avatar:** Profile avatar
  - **activities:** All profile activities
  - **activitiesCount:** Number of profile activities
  - **firstActivityDate:** The first profile activity date recorded by the system in IS O8601 format
  - **lastActivityDate:** The last profile activity date recorded by the system in ISO 8601 format

Example Request:

```
{
  profiles(ids: ["1cf13933-00be-4a6d-8dc3-3ff17f94aef8"]) {
    totalCount
    collectionItems {
      id
      avatar
      creationDate
      firstActivityDate
      info
      lastActivityDate
      lastModified
      mainSample {
        id
      }
      profileGroups {
        id
      }
      samples {
        id
      }
```

```
      activitiesCount
      activities {
        id
      }
    }
  }
}
```

Example Response:


```
{
  "data": {
    "profiles": {
      "totalCount": 1,
      "collectionItems": [
        {
          "id": "1cf13933-00be-4a6d-8dc3-3ff17f94aef8",
          "avatar": "dc3e1949-b4e8-4feb-a51c-67740e323e8b",
          "creationDate": "2022-07-07T08:59:56.227343+00:00",
          "firstActivityDate": "2022-07-07T12:36:51.644000+05:00",
          "info": {
            "age": 36,
            "gender": "MALE",
            "avatar_id": "dc3e1949-b4e8-4feb-a51c-67740e323e8b",
            "main_sample_id": "3071e862-7116-4ed3-94f1-309b0b5b912f"
          },
          "lastActivityDate": "2022-07-07T12:37:07.425000+05:00",
          "lastModified": "2022-07-07T09:00:15.028365+00:00",
          "mainSample": {
            "id": "3071e862-7116-4ed3-94f1-309b0b5b912f"
          },
          "profileGroups": [
            {
              "id": "97c1a9fa-bfde-460a-9bda-f610060423ca"
            }
          ],
          "samples": [
            {
              "id": "3071e862-7116-4ed3-94f1-309b0b5b912f"
            }
          ],
          "activitiesCount": 5,
          "activities": [
            {
              "id": "78f7c795-0f28-42ce-a9d2-818095dd5f84"
            },
            {
              "id": "a4ddc400-8fb8-41ef-aacb-9fc195868368"
            },
            {
              "id": "ea4ba238-4c64-4e07-bed6-d8f29f426e7e"
            },
            {
              "id": "29c8b4f5-11d0-4523-a17f-748ba25d4b97"
            },
            {
              "id": "4a1cad4a-ed81-409e-8a16-38abe9145a77"
            }
          ]
        }
```

```
      ]
    }
  }
}
```

2.2.2 Create a Profile

Mutation `createProfile()` is used to create a new profile. The created profile is automatically saved at OMNI Platform Server.

```
profiles(
filter: JSON = null
ids: [ID] = null
limit: Int = null
offset: Int = null
order: [String] = null): ProfilesCollection!
```

image: CustomBinaryType: To add an avatar to a new profile, submit a base64 encoded image with the size up to 8MB.

profileData: ProfileInput: To add the supplemental information to be saved at the new profile, submit the following parameters:

- **profileGroupIds: [ID]:** List of IDs assigned to groups which a new profile is linked to
- **info: JSON:** Additional information about profile (age: Int, gender: "MALE" | "FEMALE")

ProfileCreateOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean:** Flag that mutation is successfully completed
- **profile: ProfileOutput!:** New profile object
- **isCreated: Boolean!:** The parameter determines whether a new profile is created or a photo is attached to the existing profile.

Incorrect input errors:

Image is transmitted in poor quality:

```
{
"message": "Low quality photo",
"code": "0x86bd49dh"
}
```

Profile groups not found by passed ids:

```
{
"message": "One or several profiles_groups does not exist",
"code": "0x573bkd35"
}
```

Example Request:

```
mutation {
  createProfile {
    isCreated
    ok
    profile {
      id
    }
  }
}
```

Example Response:

```
{
  "data": {
    "createProfile": {
      "isCreated": true,
      "ok": true,
      "profile": {
        "id": "d5ca09fb-199a-471f-99ca-b9b4954fb45f"
      }
    }
  }
}
```

2.2.3 Update a Profile

Mutation `updateProfile()` is used to update profile information.

```
updateProfile(profileData: ProfileInput!
profileId: ID): ProfileUpdateOutput!
```

profileData: ProfileInput!: Profile information to be updated (profileGroupIds: [ID], info: JSON)

profileId: ID: Profile ID

ProfileUpdateOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean:** Flag that mutation is successfully completed
- **profile: ProfileOutput!:** Updated profile object

Incorrect input errors:

Profile not found by passed id:

```
{
  "message": "Profile matching query does not exist."
}
```

Profile groups not found by passed ids

```
{
  "message": "One or several profiles_groups does not exist",
  "code": "0x573bkd35"
}
```

Example Request:

```
mutation {
  updateProfile(
    profileData: {info: {age: 20}}
    profileId: "d5ca09fb-199a-471f-99ca-b9b4954fb45f"
  ) {
    ok
    profile {
      id
    }
  }
}
```

Example Response:

```
{
  "data": {
    "updateProfile": {
      "ok": true,
      "profile": {
        "id": "d5ca09fb-199a-471f-99ca-b9b4954fb45f"
      }
    }
  }
}
```

2.2.4 Delete a Profile

Mutation `deleteProfiles()` is used to delete profiles.

```
deleteProfiles(profileIds: [ID!]!): MutationResult!
```

profileIds: [ID!]: List of IDs assigned to profiles to be deleted

MutationResult!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean:** Flag that mutation is successfully completed

Incorrect input errors:

List of profile ids passed for deletion is empty:

```
{
  "message": "Empty profiles ids list",
  "code": "0xd2ae0ef8"
}
```

Example Request:

```
mutation {
  deleteProfiles(profileIds: "d5ca09fb-199a-471f-99ca-b9b4954fb45f") {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "deleteProfiles": {
      "ok": true
    }
  }
}
```

2.3 Profile Groups

2.3.1 Get Profile Groups

Query `profileGroups()` allows to get a list of all groups stored in the database.

```
profileGroups(
filter: JSON = null
ids: [ID] = null
limit: Int = null
offset: Int = null
order: [String] = null): ProfileGroupsCollection!
```

filter: JSON: You can filter a group list by specifying one or several parameters:

- **area_type:**
- **attention_areas:**
- **camera_location:**
- **cameras:**
- **creation_date:** Exact group creation date in ISO 8601 format
- **id:** ID assigned to a group from the database
- **info:** JSON completely similar to info parameter of a group object
- **last_modified**: Exact date of the last group modification in ISO 8601 format
- **link_to_profile:**
- **profiles:** List of profile IDs
- **title:** Group name
- **type:**
- **workspace:** Workspace ID
- **workspace_id:** Workspace ID

ids: [ID]: To get a list of certain groups, specify their IDs in the list.

limit: Int: The parameter allows to get the first n profiles from the list.

offset: Int: The parameter allows to remove the first n profiles from the list.

order: [String]:

ProfileGroupsCollection!: The mutation result is a group list that contains the following parameters:

- **totalCount:** Number of returned groups
- **collectionItems: [ProfileGroupOutput!]!:**
  - **id:** Group ID
  - **title:** Group name
  - **info:** Additional group information in JSON format
  - **lastModified:** Last group modification date in ISO 8601 format
  - **creationDate:** Group creation date in ISO 8601 format
  - **profileIds:** List of IDs assigned to profiles which are linked to a group

Example Request:

```
{
  profileGroups(ids: ["97c1a9fa-bfde-460a-9bda-f610060423ca"]) {
    totalCount
    collectionItems {
      id
      creationDate
      info
      lastModified
      profileIds
      title
    }
  }
}
```

Example Response:

```
{
  "data": {
    "profileGroups": {
      "totalCount": 1,
      "collectionItems": [
        {
          "id": "97c1a9fa-bfde-460a-9bda-f610060423ca",
          "creationDate": "2022-07-07T07:21:06.428216+00:00",
          "info": {
            "color": "red.600"
          },
          "lastModified": "2022-07-07T07:21:06.428205+00:00",
          "profileIds": [
            "e4975119-cac7-4ced-ae3f-779bb1093c89",
            "1cf13933-00be-4a6d-8dc3-3ff17f94aef8",
            "b4a16647-1336-4ed8-a440-e4e8896e1de3"
          ],
          "title": "My persons"
        }
      ]
    }
  }
}
```

2.3.2 Create a Profile Group

Mutation `createProfileGroup()` is used to create a new group. The created group is automatically saved at OMNI Platform Server.

```
createProfileGroup(profileGroupData: ProfileGroupInput!): ProfileGroupModifyOutput!
```

profileGroupData: ProfileGroupInput!: JSON with the information required for creating a new group:

- **title: String!:** New group name
- **info: JSON = null:** Additional group information

ProfileGroupInput!: The mutation result is JSON with the following parameters:

- **ok: Boolean:** Flag that mutation is successfully completed
- **profileGroup: ProfileGroupOutput:** New group object

Example Request:

```
mutation {
  createProfileGroup(profileGroupData: {title: "My new group"}) {
    ok
    profileGroup {
      creationDate
      id
      info
      lastModified
      profileIds
      title
    }
  }
}
```

Example Response:

```
{
  "data": {
    "createProfileGroup": {
      "ok": true,
      "profileGroup": {
        "creationDate": "2022-07-08T07:44:10.766783+00:00",
        "id": "d77c80eb-fffb-4812-a63c-cd8007270ef3",
        "info": {},
        "lastModified": "2022-07-08T07:44:10.766767+00:00",
        "profileIds": [],
        "title": "My new group"
      }
    }
  }
}
```

2.3.3 Delete a Profile Group

Mutation `deleteProfileGroup()` allows to delete a group list.

```
deleteProfileGroup(groupIds: [ID!]!): MutationResult!
```

groupIds: [ID!]!: List of IDs assigned to groups to be deleted

MutationResult!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean:** Flag that mutation is successfully completed

Example Request:

```
mutation {
  deleteProfileGroup(groupIds: ["d77c80eb-fffb-4812-a63c-cd8007270ef3"]) {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "deleteProfileGroup": {
      "ok": true
    }
  }
}
```

2.3.4 Update a Profile Group

Mutation `updateProfileGroupInfo()` is used to update group information.

```
updateProfileGroupInfo(
profileGroupData: ProfileGroupModifyInput!
profileGroupId: ID!): ProfileGroupModifyOutput!
```

profileGroupData: ProfileGroupModifyInput!: JSON with the information to be updated:

- **title:** Group name
- **info:** Additional information about group

profileGroupId: ID!: Group ID

ProfileGroupModifyOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean:** Flag that mutation is successfully completed
- **profileGroup: ProfileGroupOutput:** Updated group object

Incorrect input errors:

Profile group not found by transmitted id:

```
{
  "message": "Label matching query does not exist."
}
```

Example Request:

```
mutation {
  updateProfileGroupInfo(
    profileGroupData: {title: "New group's name"}
    profileGroupId: "800f0b65-7dbe-42b2-8f66-89af95a734e7"
  ) {
    ok
    profileGroup {
      id
      title
    }
  }
}
```

Example Response:

```
{
  "data": {
    "updateProfileGroupInfo": {
      "ok": true,
      "profileGroup": {
        "id": "800f0b65-7dbe-42b2-8f66-89af95a734e7",
        "title": "New group's name"
      }
    }
  }
}
```

2.3.5 Add Profiles to Profile Groups

Mutation `addProfilesToGroups()` allows to add the selected profiles to one or several shared groups.

```
addProfilesToGroups(
groupIds: [ID!]!
profilesIds: [ID!]!): ProfilesUpdateOutput!
```

groupIds: [ID!]!: List of group IDs

profilesIds: [ID!]!: List of profile IDs

ProfilesUpdateOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean**: Flag that mutation is successfully completed
- **profiles: [ProfileOutput!]!:** List of updated profiles

Incorrect input errors:

Profile not found by passed id:

```
{
  "message": "Profile matching query does not exist."
}
```

Profile groups not found by passed ids:

```
{
  "message": "One or several profiles_groups does not exist",
  "code": "0x573bkd35"
}
```

Example Request:

```
mutation {
  addProfilesToGroups(
    groupIds: ["5b53aa69-d515-4fd3-81bf-c3d8696c3ab8",
"800f0b65-7dbe-42b2-8f66-89af95a734e7"]
    profilesIds: ["1cf13933-00be-4a6d-8dc3-3ff17f94aef8",
"292a2f47-8bfd-4782-8fdf-c8b8189e300e", "3f4c1579-4e5e-4ef4-b9e1-a19808c4d931"]
  ) {
    ok
    profiles {
      id
      profileGroups {
        id
      }
    }
  }
}
```

Example Response:


```
{
  "data": {
    "addProfilesToGroups": {
      "ok": true,
      "profiles": [
        {
          "id": "1cf13933-00be-4a6d-8dc3-3ff17f94aef8",
          "profileGroups": [
            {
              "id": "97c1a9fa-bfde-460a-9bda-f610060423ca"
            },
            {
              "id": "5b53aa69-d515-4fd3-81bf-c3d8696c3ab8"
            },
            {
              "id": "800f0b65-7dbe-42b2-8f66-89af95a734e7"
            }
          ]
        },
        {
          "id": "292a2f47-8bfd-4782-8fdf-c8b8189e300e",
          "profileGroups": [
            {
              "id": "5b53aa69-d515-4fd3-81bf-c3d8696c3ab8"
            },
            {
              "id": "800f0b65-7dbe-42b2-8f66-89af95a734e7"
            }
          ]
        },
        {
          "id": "3f4c1579-4e5e-4ef4-b9e1-a19808c4d931",
          "profileGroups": [
            {
              "id": "edbb2c49-bacc-4b43-aac8-e06fd3da35eb"
            },
            {
              "id": "5b53aa69-d515-4fd3-81bf-c3d8696c3ab8"
            },
            {
              "id": "800f0b65-7dbe-42b2-8f66-89af95a734e7"
            }
          ]
        }
      ]
    }
  }
}
```

2.3.6 Delete Profiles from Profile Groups

Mutation `removeProfilesFromGroups()` is used to remove the selected profiles from one or several groups.

```
removeProfilesFromGroups(
groupIds: [ID!]!
profilesIds: [ID!]!): ProfilesUpdateOutput!
```

groupIds: [ID!]!: List of group IDs

profilesIds: [ID!]!: List of profile IDs

ProfilesUpdateOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean:** Flag that mutation is successfully completed
- **profiles: [ProfileOutput!]!:** List of updated profiles

Incorrect input errors:

Profile not found by passed id:

```
{
  "message": "Profile matching query does not exist."
}
```

Profile groups not found by passed ids:

```
{
  "message": "One or several profiles_groups does not exist",
  "code": "0x573bkd35"
}
```

Example Request:

```
mutation {
  removeProfilesFromGroups(
    groupIds: ["5b53aa69-d515-4fd3-81bf-c3d8696c3ab8",
"800f0b65-7dbe-42b2-8f66-89af95a734e7"]
    profilesIds: ["1cf13933-00be-4a6d-8dc3-3ff17f94aef8",
"292a2f47-8bfd-4782-8fdf-c8b8189e300e", "3f4c1579-4e5e-4ef4-b9e1-a19808c4d931"]
  ) {
    ok
    profiles {
      id
      profileGroups {
        id
      }
    }
  }
}
```

Example Response:


```
{
  "data": {
    "removeProfilesFromGroups": {
      "ok": true,
      "profiles": [
        {
          "id": "1cf13933-00be-4a6d-8dc3-3ff17f94aef8",
          "profileGroups": [
            {
              "id": "97c1a9fa-bfde-460a-9bda-f610060423ca"
            }
          ]
        },
        {
          "id": "292a2f47-8bfd-4782-8fdf-c8b8189e300e",
          "profileGroups": []
        },
        {
          "id": "3f4c1579-4e5e-4ef4-b9e1-a19808c4d931",
          "profileGroups": [
            {
              "id": "edbb2c49-bacc-4b43-aac8-e06fd3da35eb"
            }
          ]
        }
      ]
    }
  }
}
```

2.4 Endpoints

2.4.1 Get Endpoints

Query `endpoints()` allows to get a list of endpoints where notifications can be sent to.

```
endpoints(
filter: JSON = null
ids: [ID] = null
limit: Int = null
offset: Int = null
order: [String] = null
withArchived: WithArchived = null): EndpointCollection!
```

filter: JSON: You can filter a list of endpoints by specifying one or several parameters:

- **creation_date:** Exact endpoint creation date in ISO 8601 format
- **id:** ID assigned to endpoint from the database
- **is_active:**
- **last_modified:** Exact date of the last endpoint modification in ISO 8601 format
- **meta:** JSON that contains the data for reaching an endpoint
- **triggers:** ID assigned to a trigger that activates notifications
- **type:** The parameter sorts the endpoints by type (WI = WebInterface, EM = Email, WH = Webhook)
- **workspace:** Workspace ID
- **workspace_id:** Workspace ID

ids: [ID]: To get a list with certain endpoints, specify their IDs in the list.

limit: Int: The parameter allows to get the first n profiles from the list.

offset: Int: The parameter allows to remove the first n profiles from the list.

order: [String]:

withArchived: WithArchived: To get all endpoints including the archived ones, specify all. To get only archived endpoints, specify archived.

EndpointCollection!: Query result is a list of endpoints that contains the following parameters:

- **totalCount:** Number of returned endpoints
- **profiles: [EndpointOutput!]!:**

- **id: ID!:** Endpoint ID
- **type: EndpointType!:** Endpoint type can have the following values: Email, Webhook or WebInterface
- **meta: JSONString!:** Endpoint meta-information for reaching this endpoint
- **lastModified: DateTime!**: Last endpoint modification date in ISO 8601 format
- **creationDate: DateTime!:** Endpoint creation date in ISO 8601 format
- **defaultAlias: DefaultAlias:**
  - OWNER_EMAIL
  - WEB_INTERFACE
- **archived: Boolean!:** Attribute that the endpoint is archived.

Example Request:

```
{
  endpoints(ids: ["6fd67f81-a195-442b-a991-1a0eca6bbb69"]) {
    totalCount
    collectionItems {
      archived
      creationDate
      id
      defaultAlias
      lastModified
      meta
      type
    }
  }
}
```

Example Response:

```
{
  "data": {
    "endpoints": {
      "totalCount": 1,
      "collectionItems": [
        {
          "archived": false,
          "creationDate": "2022-07-07T07:21:06.421413+00:00",
          "id": "6fd67f81-a195-442b-a991-1a0eca6bbb69",
          "defaultAlias": "OWNER_EMAIL",
          "lastModified": "2022-07-07T07:21:06.421425+00:00",
          "meta": "{\"target_email\": \"aa@aa.ru\", \"default_alias\":
\"owner_email\"}",
          "type": "Email"
        }
      ]
    }
  }
}
```

2.4.2 Create an Email Endpoint

Mutation `createEmailEndpoint()` allows to create a new endpoint of Email type.

```
createEmailEndpoint(endpointData: EmailEndpointInput!): EndpointManageOutput!
```

endpointData: EmailEndpointInput!: JSON with the information required for creating an endpoint:

- **targetEmail: String!:** Email where notifications will be sent to.

EndpointManageOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed
- **endpoint: EndpointOutput!:** New endpoint object

Example Request:

```
mutation {
  createEmailEndpoint(endpointData: {targetEmail: "test@test.com"}) {
    ok
    endpoint {
      id
      archived
      creationDate
      defaultAlias
      lastModified
      meta
      type
    }
  }
}
```

Example Response:

```
{
  "data": {
    "createEmailEndpoint": {
      "ok": true,
      "endpoint": {
        "id": "2eff291d-8c8c-4325-8953-5a70b31bc63e",
        "archived": false,
        "creationDate": "2022-07-10T12:27:22.568988+00:00",
        "defaultAlias": null,
        "lastModified": "2022-07-10T12:27:22.569003+00:00",
        "meta": "{\"target_email\": \"test@test.com\"}",
        "type": "EMAIL"
      }
    }
  }
}
```

2.4.3 Create a Webhook Endpoint

Mutation `createWebhookEndpoint()` allows to create a new endpoint of Webhook type.

```
createWebhookEndpoint(endpointData: WebhookEndpointInput!): EndpointManageOutput!
```

endpointData: WebhookEndpointInput!: JSON with the information required for creating an endpoint:

- **url: String!:** URL where requests will be sent to
- **requestMethod: String!:** Request method

EndpointManageOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed
- **endpoint: EndpointOutput!:** New endpoint object

Example Request:

```
mutation {
  createWebhookEndpoint(
    endpointData: {url: "https://endpoint_test.requestcatcher.com/test /",
requestMethod: "POST"}
  ) {
    ok
    endpoint {
      archived
      creationDate
      defaultAlias
      id
      lastModified
      meta
      type
    }
  }
}
```

Example Response:

```
{
  "data": {
    "createWebhookEndpoint": {
      "ok": true,
      "endpoint": {
        "archived": false,
        "creationDate": "2022-07-10T12:42:34.957171+00:00",
        "defaultAlias": null,
        "id": "afd7c121-3140-4a8c-b0ee-3717581eb76d",
        "lastModified": "2022-07-10T12:42:34.957188+00:00",
        "meta": "{\"url\": \"https://endpoint_test.requestcatcher.com/test /\",
\"method\": \"POST\"}",
        "type": "WEBHOOK"
      }
```

```
    }
  }
}
```

2.4.4 Update an Endpoint

Mutation `updateEndpoint()` allows to update the endpoint information.

```
updateEndpoint(endpointId: ID!
endpointInfo: JSON): EndpointManageOutput!
```

endpointId: ID!: Endpoint ID

endpointInfo: JSON: JSON-object with parameters to be updated. These parameters are similar to the parameters from meta: JSON

EndpointManageOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed
- **endpoint: EndpointOutput!:** Updated endpoint object

Incorrect input errors:

Endpoint not found by transmitted id:

```
{
  "message": "Endpoint matching query does not exist."
}
```

Example Request:

```
mutation {
  updateEndpoint(
    endpointId: "6fd67f81-a195-442b-a991-1a0eca6bbb69"
    endpointInfo: {target_email: "new-email@test.com"}
  ) {
    ok
    endpoint {
      id
      meta
    }
  }
}
```

Example Response:

```
{
  "data": {
    "updateEndpoint": {
```

```
      "ok": true,
      "endpoint": {
        "id": "6fd67f81-a195-442b-a991-1a0eca6bbb69",
        "meta": "{\"target_email\": \"new-email@test.com\", \"default_alias\":
\"owner_email\"}"
      }
    }
  }
}
```

2.4.5 Delete an Endpoint

Mutation `deleteEndpoint()` is used to archive a list of endpoints.

```
deleteEndpoint(endpointIds: [String!]!): MutationResult!
```

endpointIds: [String!]!: List of endpoint IDs

MutationResult!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed

Example Request:

```
mutation {
  deleteEndpoint(
    endpointIds: ["2eff291d-8c8c-4325-8953-5a70b31bc63e",
"29bcfb3a-3a26-4544-9def-d5d44bbd3be4"]
  ) {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "deleteEndpoint": {
      "ok": true
    }
  }
}
```

2.5 Triggers

2.5.1 Get Triggers

The query allows to get a list of triggers for creating notifications. Triggers are related to the endpoints where notifications will be sent to.

```
triggers(
filter: JSON = null
ids: [ID] = null
limit: Int = null
offset: Int = null
order: [String] = null
targetId: ID = null
withArchived: WithArchived = null): TriggerCollection!
```

filter: JSON: You can filter a list of triggers by specifying one or several endpoints:

- **creation_date:** Exact trigger creation date in ISO 8601 format
- **endpoints:** Linked endpoint ID
- **id:** ID assigned to a trigger from the database
- **is_active:**
- **last_modified:** Exact date of the last trigger modification in ISO 8601 format
- **meta:**
- **workspace:** Workspace ID
- **workspace_id:** Workspace ID

ids: [ID]: To get a list of certain triggers, specify their IDs in the list.

limit: Int: The parameter allows to get the first n triggers from the list.

offset: Int: The parameter allows to remove the first n triggers from the list.

order: [String]: You can sort a list by specifying the value for the following parameters: creation_date, endpoints, id, is_active, last_modified, meta, workspace, workspace_id

targetId: ID: ID assigned to a group which the trigger is linked to

withArchived: WithArchived: To get all triggers, including the archived ones, specify all. To get only archived triggers, specify archived.

TriggerCollection!: The query result is a list of triggers that contains the following parameters:

- **totalCount: Int:** Number of returned triggers

- **collectionItems: [TriggerType!]!:**
  - **id: ID!:** Trigger ID
  - **creationDate: DateTime!:** Trigger creation date in ISO 8601 format
  - **lastModified: DateTime!:** Last trigger modification date in ISO 8601 format
  - **meta: Meta!:** Trigger meta-information
  - **endpoints: [EndpointOutput!]!:** List of endpoints linked to a trigger
  - **archived: Boolean!:** Attribute that a trigger is archived.

Example Request:

```
{
  triggers(ids: ["3e57a95e-028c-4d27-9419-a9c4bfd6f3fb"]) {
    totalCount
    collectionItems {
      id
      creationDate
      lastModified
      meta {
        notificationParams
        conditionLanguage {
          condition
          variables {
            name
            target {
              type
              uuid
            }
            type
          }
        }
      }
      endpoints {
        id
      }
      archived
    }
  }
}
```

Example Response:

```
{
  "data": {
    "triggers": {
      "totalCount": 1,
      "collectionItems": [
        {
          "id": "3e57a95e-028c-4d27-9419-a9c4bfd6f3fb",
          "creationDate": "2022-07-07T08:38:22.982190+00:00",
          "lastModified": "2022-07-08T05:03:48.961423+00:00",
          "meta": {
            "notificationParams": "{\"lifetime\": 5}",
            "conditionLanguage": {
              "condition": null,
              "variables": [
```

```
                {
                  "name": "0_v",
                  "target": [
                    {
                      "type": "Label",
                      "uuid": "edbb2c49-bacc-4b43-aac8-e06fd3da35eb"
                    }
                  ],
                  "type": "presence"
                }
              ]
            }
          },
          "endpoints": [
            {
              "id": "db38ec53-c09d-45f2-bfed-5877c9bd9016"
            }
          ],
          "archived": false
        }
      ]
    }
  }
}
```


2.5.2 Create a Trigger for a Profile Group

Mutation `createProfileGroupTrigger` allows to create a new trigger for a profile group. Everytime the camera detects a person from this group, the system creates a notification.

```
createProfileGroupTrigger(
endpointAliases: [DefaultAlias!] = null
endpointIds: [ID!] = null
endpointUrl: String = null
profileGroupId: ID!): TriggerManageOutput!
```

endpointAliases: [DefaultAlias!]: You can submit OWNER_EMAIL or WEB_INTERFACE. In this case one of endpoints with the same value for parameter defaultAlias will be linked to a trigger.

endpointIds: [ID!]: You can submit a list of IDs assigned to endpoints which are linked to a trigger.

endpointUrl: String: You can submit URL to your webhook. This will create a new endpoint for sending notifications to your URL address.

profileGroupId: ID!: Group ID necessary for creating a trigger.

TriggerManageOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed
- **trigger: TriggerType!:** New trigger object

Incorrect input errors:

Profile group not found by transmitted id:

```
{
  "message": "Label matching query does not exist."
}
```

Endpoint not found by transmitted id:

```
{
  "message": "Endpoint does not exist."
}
```

Example Request:

```
mutation {
  createProfileGroupTrigger(
    profileGroupId: "800f0b65-7dbe-42b2-8f66-89af95a734e7"
    endpointIds: ["afd7c121-3140-4a8c-b0ee-3717581eb76d"]
  ) {
    trigger {
      endpoints {
        id
      }
      id
    }
  }
}
```

Example Response:

```
{
  "data": {
    "createProfileGroupTrigger": {
      "trigger": {
        "endpoints": [
          {
            "id": "afd7c121-3140-4a8c-b0ee-3717581eb76d"
          }
        ],
        "id": "9675aede-3ee8-495b-ac8b-e284a07db99e"
      }
    }
  }
}
```

2.5.3 Update a Trigger

Mutation `updateTrigger` allows to update endpoints linked to a trigger. List of linked endpoints is replaced by the new one.

```
updateTrigger(
endpointAliases: [DefaultAlias!] = null
endpointIds: [ID!] = null
triggerId: ID!): TriggerManageOutput!
```

endpointAliases: [DefaultAlias!]: You can submit OWNER_EMAIL or WEB_INTERFACE. In this case one of endpoints with the same value in parameter defaultAlias will be linked to a trigger.

endpointIds: [ID!]: You can submit a list of IDs assigned to endpoints which will be linked to a trigger.

triggerId: ID!: Trigger ID

TriggerManageOutput!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed
- **trigger: TriggerType!:** Updated trigger object

Incorrect input errors:

Trigger not found by passed id:

```
{
  "message": "Trigger matching query does not exist."
}
```

Endpoint not found by transmitted id:

```
{
  "message": "Endpoint does not exist."
}
```

Example Request:

```
mutation {
  updateTrigger(
    triggerId: "64e0a7ec-0df4-4734-a460-601fa1b65a1f"
    endpointIds: ["6e5a6d3b-8247-488e-95d7-57a306b294ed"]
    endpointAliases: OWNER_EMAIL
  ) {
    ok
    trigger {
```

```
      id
      endpoints {
        id
        defaultAlias
      }
    }
  }
}
```

Example Response:

```
{
  "data": {
    "updateTrigger": {
      "ok": true,
      "trigger": {
        "id": "64e0a7ec-0df4-4734-a460-601fa1b65a1f",
        "endpoints": [
          {
            "id": "6e5a6d3b-8247-488e-95d7-57a306b294ed",
            "defaultAlias": null
          },
          {
            "id": "6fd67f81-a195-442b-a991-1a0eca6bbb69",
            "defaultAlias": "OWNER_EMAIL"
          }
        ]
      }
    }
  }
}
```

2.5.4 Link an Endpoint to a Trigger

Mutation `linkEndpoint` allows to link endpoint to trigger. The endpoint is to be added to a list of endpoints linked to the trigger.

```
linkEndpoint(
endpointAliases: [DefaultAlias!] = null
endpointIds: ID! = null
triggerId: ID!): MutationResult!
```

endpointAliases: [DefaultAlias!]: You can submit OWNER_EMAIL or WEB_INTERFACE. In this case one of endpoints with the same value for parameter defaultAlias will be linked to a trigger.

endpointIds: ID!: You can submit ID assigned to an endpoint that will be linked to a trigger.

triggerId: ID!: Trigger ID

MutationResult!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed

Incorrect input errors:

Endpoint id for binding to a trigger has not been passed:

```
{
  "message": "Unknown exception code for 'bad_input_data' type",
  "code": "No id or alias provided"
}
```

Endpoint not found by transmitted id:

```
{
  "message": "Endpoint matching query does not exist."
}
```

Trigger not found by passed id:

```
{
  "message": "Trigger matching query does not exist."
}
```

Example Request:

```
mutation {
  linkEndpoint(
    triggerId: "64e0a7ec-0df4-4734-a460-601fa1b65a1f"
    endpointAlias: WEB_INTERFACE
  ) {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "linkEndpoint": {
      "ok": true
    }
  }
}
```

2.5.5 Unlink an Endpoint from a Trigger

Mutation `unlinkEndpoint` is used to unlink endpoint from trigger. The endpoint is removed from a list of endpoints linked to the trigger.

```
unlinkEndpoint(
endpointAliases: [DefaultAlias!] = null
endpointIds: ID! = null
```

```
triggerId: ID!): MutationResult!
```

endpointAliases: [DefaultAlias!]: You can submit OWNER_EMAIL or WEB_INTERFACE. In this case one of endpoints with the same value for parameter defaultAlias is removed from a list of endpoints linked to this trigger.

endpointIds: ID!: You can submit ID of an endpoint to be removed from a list of endpoints linked to this trigger.

triggerId: ID!: Trigger ID

MutationResult!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed.

Incorrect input errors:

Endpoint id to unlink from the trigger has not been passed:

```
{
  "message": "Unknown exception code for 'bad_input_data' type",
  "code": "No id or alias provided"
}
```

Endpoint not found by transmitted id:

```
{
  "message": "Endpoint matching query does not exist."
}
```

Trigger not found by passed id:

```
{
  "message": "Trigger matching query does not exist."
}
```

Example Request:

```
mutation {
  unlinkEndpoint(
    triggerId: "64e0a7ec-0df4-4734-a460-601fa1b65a1f"
    endpointAlias: WEB_INTERFACE
  ) {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "unlinkEndpoint": {
      "ok": true
    }
  }
}
```

2.5.6 Delete a Trigger

Mutation `deleteTrigger` allows to delete a trigger.

```
deleteTrigger(triggerId: ID!): MutationResult!
```

triggerId: ID!: Trigger ID

MutationResult!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!**: Attribute that mutation is successfully completed.

Incorrect input errors:

Trigger not found by passed id:

```
{
  "message": "Trigger matching query does not exist."
}
```

Example Request:

```
mutation {
  deleteTrigger(triggerId: "64e0a7ec-0df4-4734-a460-601fa1b65a1f") {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "deleteTrigger": {
      "ok": true
    }
  }
}
```

2.6 Notifications

2.6.1 Get Notifications

The query is used to get a list of notifications sent from the system.

```
notifications(
filters: NotificationFilter
order: NotificationOrdering
pagination: OffsetPaginationInput): NotificationOutputCountList!
```

filters: NotificationFilter: To get only required notifications, you can adjust filtration

- **id:** Filtration by notification IDs:
    - **exact: ID:** Get an exact notification object using ID notification value.
    - **iExact: ID:** Get an exact notification object using ID notification value. Not case-sensitive.
    - **contains: String:** Get notification objects using value matching with notification ID.
    - **iContains: String:** Get notification objects using value matching with notification ID. Not case-sensitive.
    - **inList: [ID!]:** Get exact notification objects using a list of notification IDs.
    - **gt: ID:** Get notification objects which IDs are greater(>) than submitted IDs. String matching is used.
    - **gte: ID:** Get notification objects which IDs are greater than or equal to (>=) submitted ID. String matching is used.
    - **lt: ID:** Get notification objects which IDs are less (<) than submitted ID. String matching is used.
    - **lte: ID:** Get notification objects which IDs are less than or equal to(<=) submitted ID. String matching is used.
    - **startsWith: String:** Get notification objects which IDs start with submitted value.
    - **iStartsWith: String:** Get notification objects which IDs start with submitted value. Not case-sensitive.
    - **endsWith: String:** Get notification objects which IDs end with submitted value.
    - **iEndsWith: String:** Get notification objects which IDs end with submitted value. Not case-sensitive.
    - **range: [ID!]:** Get notification objects which IDs are within the range of submitted IDs. String matching is used.

    - **isNull: Boolean:** Get notification objects which IDs are equal(true) or not equal(false) to null.
    - **regex: String:** Get notification objects which IDs meet a submitted regular expression.
    - **iRegex: String:** Get notification objects which IDs meet a submitted regular expression. Not case-sensitive.
- **creationDate: DatetimeFilterLookupCustom:** Filtration by notification creation date. The parameters are similar to the ones for filtration by ID.
- **lastModified: DatetimeFilterLookupCustom:** Filtration by date of the last notification modification. The parameters are similar to the ones for filtration by ID.
- **isViewed: Boolean:** Filtration of viewed (true) and not viewed (false) notifications.
- **isActive: Boolean**: Filtration of active (true) and not active (false) notifications.
- **endpointId: UUID:** Filtration of notifications by ID assigned to the endpoint they were sent to.
- **isSent: Boolean:** Filtration of notifications by in-system notifications (true) and notifications sent to external endpoints (false).
- **triggerId: ID:** Filtration of notifications by trigger ID.
- **profileId: ID:** Filtration of notifications by profile ID.
- **type: NotificationType:** Filtration of notifications by type.
- **profileGroupTitle: String:** Filtration of notifications by group name.

order: NotificationOrdering: You can adjust sorting of notification list:

- **id: Ordering:** Sorting by ID. String matching is used.(ASC: Ascending sorting, DESC: Descending sorting)
- **creationDate: Ordering:** Sorting by notification creation date.
- **lastModified: Ordering**: Sorting by date of the last notification modification.

pagination: OffsetPaginationInput:

- **limit: Int!:** This parameter allows to get the first n notifications from the list.
- **offset: Int!:** This parameter allows to remove the first n notifications from the list.

NotificationOutputCountList!: The query result is a list of notifications that contains the following parameters:

- **totalCount: Int!**: Total number of notifications.

- **collectionItems: [NotificationOutput!]!:**
  - **id: ID!:** Notification ID.
  - **isActive: Boolean!:** Attribute that notification is active.
  - **isViewed: Boolean!:** Attribute that notification is viewed.
  - **lastModified: DateTime:** Date of the last notification modification in ISO 8601 format.
  - **activityId: ID:** ID assigned to activity that causes creating of notification.
  - **avatarId: ID:** Profile avatar ID.
  - **cameraId: ID:** ID assigned to a camera that detected the activity.
  - **cameraTitle: String:** Camera name
  - **creationDate: DateTime!**: Notification creation date in ISO 8601 format.
  - **currentCount: Int:** Number of persons in the camera's field of view
  - **description: String:** Profile description
  - **endpointStatuses: [EndpointStatusOutput!]:** List of endpoints and their statuses
    - **endpoint: EndpointOutput!:** Endpoint object.
    - **status: String!:** Submission status.
  - **limit: Int:** Max number of persons.
  - **name: String:** Profile name.
  - **profileGroupColor: String:** Color of the group linked to a trigger.
  - **profileGroupId: ID:** ID assigned to the group linked to a trigger.
  - **profileGroupTitle: String:** Name of the group linked to a trigger.
  - **profileId: ID:** Profile ID.
  - **realtimeBodyPhotoId: String:** ID assigned to a photo of the person detected by camera (Missing, if the anonymous mode is enabled).
  - **realtimeFacePhotoId: String:** ID assigned to a face image of a person detected by camera (Missing, if the anonymous mode is enabled).
  - **triggerId: ID:** ID assigned to a trigger that caused the notification.
  - **type: String!:** Notification type.

Example Request:

```
{
  notifications(filters: {id: {exact: "ed1a55af-6daf-4368-ac68-0defdff5159c"}}) {
    totalCount
    collectionItems {
      profileGroupColor
```

```
      activityId
      avatarId
      cameraId
      cameraTitle
      creationDate
      currentCount
      description
      id
      endpointStatuses {
        status
        endpoint {
          id
        }
      }
      isActive
      isViewed
      lastModified
      limit
      name
      profileGroupId
      profileGroupTitle
      profileId
      realtimeBodyPhotoId
      realtimeFacePhotoId
      triggerId
      type
    }
  }
}
```

Example Response:

```
{
  "data": {
    "notifications": {
      "totalCount": 1,
      "collectionItems": [
        {
          "profileGroupColor": "red.600",
          "activityId": "f3bf0ac7-5849-45d7-85e0-12de925633da",
          "avatarId": "4312809b-99b2-47a7-beb9-b60b87c07594",
          "cameraId": "3c818dc4-352c-47a7-b32b-465fbd4a9665",
          "cameraTitle": "My <ShortProductName/> Agent",
          "creationDate": "2022-07-07T13:10:53.619145+00:00",
          "currentCount": null,
          "description": "gap\n",
          "id": "ed1a55af-6daf-4368-ac68-0defdff5159c",
          "endpointStatuses": [
            {
              "status": "success",
              "endpoint": {
                "id": "db38ec53-c09d-45f2-bfed-5877c9bd9016"
              }
            }
          ],
          "isActive": false,
          "isViewed": true,
          "lastModified": "2022-07-07T13:11:12.264337+00:00",
          "limit": null,
          "name": null,
```

```
          "profileGroupId": "97c1a9fa-bfde-460a-9bda-f610060423ca",
          "profileGroupTitle": "My persons",
          "profileId": "b4a16647-1336-4ed8-a440-e4e8896e1de3",
          "realtimeBodyPhotoId": "339a4020-a9af-49e8-83d4-3fc34ef794e5",
          "realtimeFacePhotoId": null,
          "triggerId": "66b3eb1a-de88-469e-83c9-697785a0a4c2",
          "type": "presence"
        }
      ]
    }
  }
}
```

2.6.2 View Notifications

Mutation `viewingNotifications` marks one or several notifications as viewed.

```
viewingNotifications(notificationIds: [String!]!): MutationResult!
```

notificationIds: [String!]!: List of IDs assigned to notifications which should be marked as viewed.

MutationResult: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed.

Example Request:

```
mutation {
  viewingNotifications(notificationIds: ["5a1a1bed-5bc0-4e8e-8154-e56c5ef5ea87"]) {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "viewingNotifications": {
      "ok": true
    }
  }
}
```

2.6.3 View All Notifications

Mutation `markAllNotificationsAsViewed` marks all notifications as viewed.

```
markAllNotificationsAsViewed: MutationResult!
```

MutationResult: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!**: Attribute that mutation is successfully completed.

Example Request:

```
mutation {
  markAllNotificationsAsViewed {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "markAllNotificationsAsViewed": {
      "ok": true
    }
  }
}
```

2.7 Activities

2.7.1 Get Activities

The query allows to get a list of activities detected by the camera.

```
notifications(
filters: ActivityFilter order: ActivityOrdering
pagination: OffsetPaginationInput): ActivityOutputCountList!
```

filters: ActivityFilter: To get only required activities, you can adjust filtration

- **id:** Filtration by activity ID:
    - **exact: ID:** Get an exact activity object using the value of activity ID.
    - **iExact: ID:** Get an exact activity object using the value of activity ID. Not case-sensitive.
    - **contains: String:** Get activity objects using the value matching with activity ID.
    - **iContains: String:** Get activity objects using the value matching with activity ID. Not case-sensitive.
    - **inList: [ID!]:** Get certain activity objects using a list of activity IDs.
    - **gt: ID:** Get activity objects which IDs are greater (>) than submitted ID. String matching is used.
    - **gte: ID:** Get activity objects which IDs are greater than or equal to (>=) to submitted ID. String matching is used.
    - **lt: ID:** Get activity objects which IDs are less (<) than submitted ID. String matching is used.
    - **lte: ID:** Get activity objects which IDs are less than or equal to (<=) to submitted ID. String matching is used.
    - **startsWith: String:** Get activity objects which IDs start with submitted value.
    - **iStartsWith: String:** Get activity objects which IDs start with submitted value. Not case-sensitive.
    - **endsWith: String:** Get activity objects which IDs end with submitted value.
    - **iEndsWith: String:** Get activity objects which IDs end with submitted value. Not case-sensitive.
    - **range: [ID!]:** Get activity objects which IDs are within the range of submitted IDs. String matching is used.

    - **isNull: Boolean:** Get activity objects which IDs are equal(true) or not equal to(false) null.
    - **regex: String:** Get activity objects which IDs meet a submitted regular expression.
    - **iRegex: String:** Get activity objects which IDs meet a submitted regular expression. Not case-sensitive.
- **creationDate: DatetimeFilterLookupCustom:** Filtration by activity creation date. The parameters are similar to the ones for filtration by ID.
- **lastModified: DatetimeFilterLookupCustom:** Filtration by date of the last activity modification. The parameters are similar to the ones for filtration by ID.
- **profileId: ID:** Activity filtration by profile ID.

order: ActivityOrdering: You can adjust sorting of activity list:

- **id: Ordering:** Sorting by ID. String matching is used.(ASC: Ascending sorting, DESC: Descending sorting)
- **creationDate: Ordering:** Sorting by activity creation date.
- **lastModified: Ordering:** Sorting by date of the last activity modification.

pagination: OffsetPaginationInput:

- **limit: Int!:** The parameter allows to get the first n activities from the list.
- **offset: Int!:** The parameter allows to remove the first n activities from the list.

ActivityOutputCountList!: The query result is an activity list that contains the following parameters:

- **totalCount: Int!:** Number of returned activities.
- **collectionItems: [ActivityOutput!]!:**
    - **id: ID!:** Activity ID.
    - **data: JSON:** Additional information about activity.
    - **lastModified: DateTime:** Date of the last activity modification in ISO 8601 format.
    - **creationDate: DateTime!:** Activity creation date in ISO 8601 format.
    - **bestShotId: ID:** ID of the best face image received from the camera that detected the activity (Missing, if the anonymous mode is enabled or the camera couldn't detect a face image).
    - **cameraId: ID!:** ID assigned to the camera that detected the activity.

- **locationId: String!:** ID assigned to location which the camera is linked to.
- **profileId: ID:** Profile ID.
- **status: ActivityType!:**

  - Progress

  - Finalized

  - Failed
- **timeStart: String! :** Activity starting time.
- **timeEnd: String! :** Activity ending time.

Example Request:

```
{
  activities(filters: {id: {exact: "78f7c795-0f28-42ce-a9d2-818095dd5f84"}}) {
    totalCount
    collectionItems {
      id
      data
      cameraId
      bestShotId
      locationId
      profileId
      timeStart
      creationDate
      lastModified
    }
  }
}
```

Example Response:

```
{
  "data": {
    "activities": {
      "totalCount": 1,
      "collectionItems": [
        {
          "id": "78f7c795-0f28-42ce-a9d2-818095dd5f84",
          "data": {
            "processes": [
              {
                "id": "78f7c795-0f28-42ce-a9d2-818095dd5f84",
                "type": "track",
                "object": {
                  "id": "c4398bcc-6fee-495a-be86-a59918a875e5",
                  "class": "human"
                },
                "time_interval": [
                  "2022-07-07T12:36:51.644000+05:00",
                  "2022-07-07T12:36:51.725000+05:00"
                ]
              },
              {
                "id": "ecb65ed1-e3c5-484f-b1e3-7f4bdf869757",
                "type": "track",
```

```
                "object": {
                  "id": "c4398bcc-6fee-495a-be86-a59918a875e5",
                  "age": 36,
                  "class": "face",
                  "gender": "MALE",
                  "quality": -1114.097412109375,
                  "embeddings": {
                    "$binary_image": {
                      "id": "7fd8abfb-258b-4d6a-bdd7-eec1e30fd0b4"
                    },
                    "$template10v100": {
                      "id": "35b60b5f-d6da-4998-ac4c-5725d3e64520"
                    }
                  }
                },
                "parent": "78f7c795-0f28-42ce-a9d2-818095dd5f84",
                "sample_id": "3071e862-7116-4ed3-94f1-309b0b5b912f",
                "$best_shot": {
                  "id": "ab8d76c8-0a8c-4575-8d81-e9e469f8ef64"
                },
                "time_interval": [
                  "2022-07-07T12:36:51.644000+05:00",
                  "2022-07-07T12:36:51.725000+05:00"
                ]
              },
              {
                "id": "3e33004f-6531-40ca-90d5-8345f66002ac",
                "type": "attention",
                "parent": "ecb65ed1-e3c5-484f-b1e3-7f4bdf869757",
                "time_interval": [
                  "2022-07-07T12:36:51.806000+05:00",
                  "2022-07-07T12:36:52.413000+05:00"
                ]
              }
            ]
          },
          "cameraId": "3c818dc4-352c-47a7-b32b-465fbd4a9665",
          "bestShotId": "ab8d76c8-0a8c-4575-8d81-e9e469f8ef64",
          "locationId": "",
          "profileId": "1cf13933-00be-4a6d-8dc3-3ff17f94aef8",
          "timeStart": "2022-07-07T12:36:51.644000+05:00",
          "creationDate": "2022-07-07T07:36:52.848815+00:00",
          "lastModified": "2022-07-07T08:59:56.385040+00:00"
        }
      ]
    }
  }
}
```


2.7.2 Create a Profile by Activity

Mutation `createProfileByActivity` is used to create a profile using a detected activity.

```
createProfileByActivity(
activityId: ID!
profileData: ProfileInput = null): ProfileCreateOutput!
```

activityId: ID!: ID assigned to the activity upon which a profile should be created.

profileData: ProfileInput: You can specify the additional information to be saved at profile:

- **profileGroupIds: [ID]:** List of IDs assigned to the groups which the new profile is linked to.
- **info: JSON:** Additional information about profile (age: Int, gender: "MALE" | "FEMALE")

ProfileCreateOutput: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed.
- **profile: ProfileOutput!:** New profile object.
- **isCreated: Boolean!:** Determines if a new profile is created or a photo was linked to the existing profile.

Incorrect input errors:

Error when trying to create a profile from an anonymous activity:

```
{
  "message": "Activity is anonymous",
  "code": "0x358vri3s"
}
```

Activity not found by passed id:

```
{
  "message": "Activity matching query does not exist."
}
```

Profile groups not found by passed ids:

```
{
  "message": "One or several profiles_groups does not exist",
  "code": "0x573bkd35"
}
```

Example Request:

```
mutation {
  createProfileByActivity(activityId: "78f7c795-0f28-42ce-a9d2-818095dd5f84") {
    ok
    isCreated
    profile {
      id
    }
  }
}
```

Example Response:

```
{
```

```
  "data": {
    "createProfileByActivity": {
      "ok": true,
      "isCreated": false,
      "profile": {
        "id": "1cf13933-00be-4a6d-8dc3-3ff17f94aef8"
      }
    }
  }
}
```

2.8 Agents

2.8.1 Get Agents

Query `agents` allows to get a list of agents.

```
agents(
filter: JSON = null
ids: [ID] = null
limit: Int = null
offset: Int = null
order: [String] = null
withArchived: WithArchived = null): AgentsCollection!
```

filter: JSON: You can filter a list of agents by specifying one or several parameters:

- **activations:**
- **cameras:** ID assigned to a camera stored in the database
- **creation_date:** Exact agent creation date in ISO 8601 format
- **id:** ID assigned to an agent stored in the database.
- **info__title:** Agent name.
- **is_active:** Agent activity attribute.
- **last_modified:** Exact date of the last agent modification in ISO 8601 format
- **workspace:** Workspace ID
- **workspace_id**: Workspace ID

ids: [ID]: To get a list of certain agents, specify their IDs in the list.

limit: Int: The parameter allows to get the first n agents from the list.

offset: Int: The parameter allows to remove the first n agents from the list.

order: [String]: You can sort the list by specifying one of the values for the following parameters: activations, cameras, creation_date, id, info, is_active, last_modified, workspace, workspace_id.

withArchived: WithArchived: To get all agents, including the archived ones, specify all. To get only archived agents, specify archived.

AgentsCollection!: The query result is a list of agents that contains the following parameters:

- **totalCount: Int:** Number of returned agents

- **collectionItems: [AgentsCollection!]!:**
  - **id:** ID!: Agent ID
  - **creationDate: DateTime!:** Agent creation date in ISO 8601 format.
  - **lastModified: DateTime!:** Last agent modification date in ISO 8601 format.
  - **token: String!:** Agent's token
  - **camerasIds: [ID!]:** List of IDs assigned to cameras connected to an agent.
  - **title: String:** Agent name
  - **agentStatus: String!:** Agent status
  - **agentLastActiveTime: String:** Last activity date recorded by agent's camera.
  - **archived: Boolean!:** Agent archivation attribute.

Example Request:

```
{
  agents {
    totalCount
    collectionItems {
      token
      title
      lastModified
      id
      creationDate
      camerasIds
      archived
      agentStatus
      agentLastActiveTime
    }
  }
}
```

Example Response:


```
{
  "data": {
    "agents": {
      "totalCount": 1,
      "collectionItems": [
        {
          "token": "19df2b38-7d23-44a5-85e7-c5a29b304581",
          "title": "My Agent",
          "lastModified": "2022-07-12T08:50:45.296225+00:00",
          "id": "19df2b38-7d23-44a5-85e7-c5a29b304581",
          "creationDate": "2022-07-07T07:23:08.965949+00:00",
          "camerasIds": [
            "3c818dc4-352c-47a7-b32b-465fbd4a9665"
          ],
          "archived": false,
          "agentStatus": "active",
          "agentLastActiveTime": "2022-07-12T08:50:45.296115"
        }
      ]
    }
  }
}
```


2.8.2 Create an Agent

Mutation `createAgent` allows to create a new agent.

```
createAgent(agentData: AgentInput!): AgentCreateOutput!
```

agentData: AgentInput!: Information required for creating an agent:

- **title: String:** Agent name.
- **extra: JSON:** Additional information about agent.

AgentCreateOutput: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed.
- **agent: AgentOutput!:** New agent object.
- **channelCost: String:** Monthly service charge.
- **writeoffDate: String:** Date of the next service in ISO 8601 format.

Incorrect input errors:

Error when creating a new agent because the limit has been exceeded:

```
{
  "message": "Agent limit exceeded",
  "code": "0x6245cd00"
}
```

Incorrect extra transmitted:

```
{
  "message": "'int' object has no attribute 'get'"
}
```

Example Request:

```
mutation {
  createAgent(agentData: {title: "My new agent"}) {
    ok
    agent {
      id
    }
  }
}
```

Example Response:

```
{
  "data": {
    "createAgent": {
      "ok": true,
      "agent": {
        "id": "2b769e5a-a49a-46ac-ac38-ec924bd4ec83"
      }
    }
  }
}
```

2.8.3 Update an Agent

Mutation `updateAgent` is used to update information about agent.

```
updateAgent(
agentData: AgentUpdateInput!
agentId: ID!): AgentManageOutput!
```

agentData: AgentUpdateInput!: Information necessary for updating an agent:

- **title: String:** Agent name.

agentId: ID!: ID assigned to the agent to be updated.

AgentManageOutput: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed.
- **agent: AgentOutput!:** Updated agent object.

Incorrect input errors:

Agent not found by the transmitted id:

```
{
  "message": "Camera matching query does not exist."
}
```

Example Request:

```
mutation {
  updateAgent(
    agentId: "19df2b38-7d23-44a5-85e7-c5a29b304581"
    agentData: {title: "My old agent"}
  ) {
    ok
    agent {
      id
      title
    }
  }
}
```

Example Response:

```
{
  "data": {
    "updateAgent": {
      "ok": true,
      "agent": {
        "id": "19df2b38-7d23-44a5-85e7-c5a29b304581",
        "title": "My old agent"
      }
    }
  }
}
```

2.8.4 Delete an Agent

Mutation `deleteAgent` allows to delete one or several agents.

```
deleteAgent(agentId: ID = "" agentIds: [ID!] = null): MutationResult!
```

agentId: ID: ID assigned to an agent to be deleted.

agentIds: [ID!]: List of IDs assigned to agents to be deleted.

MutationResult!: The mutation result is JSON that contains the following parameters:

- **ok: Boolean!:** Attribute that mutation is successfully completed.

Incorrect input errors:

No id has been passed for agent deletion:

```
{
  "message": "One of the parameters agentId or agentIds is required",
  "code": "0xe509f74d"
}
```

Example Request:

```
mutation {
  deleteAgent(agentId: "19df2b38-7d23-44a5-85e7-c5a29b304581") {
    ok
  }
}
```

Example Response:

```
{
  "data": {
    "deleteAgent": {
      "ok": true
    }
  }
}
```

2.9 Others

2.9.1 Get User Information

Query `me` allows to get information about authorized user.

```
me: UserType!
```

UserType!: The query result is a link list:

- **username: String!:** User name.
- **email: String!:** User email.
- **firstName: String!:** User first name.
- **lastName: String!:** User last name.
- **workspaces: [WorkspaceType!]!:** Information about the user's workspaces.

Example Request:

```
{
  me {
    email
    firstName
    lastName
    username
    workspaces {
      id
    }
  }
}
```

Example Response:

```
{
  "data": {
    "me": {
      "email": "aa@aa.ru",
      "firstName": "",
      "lastName": "",
      "username": "aa@aa.ru",
      "workspaces": [
        {
          "id": "8d100a02-1b5b-4da1-b645-9fd01ef09c77"
        }
      ]
    }
  }
}
```