SAFR® from realnetworks

SAFR® REST API Documentation

Documentation Version = 3.030

Publish Date = October 6, 2021

Contents

1 REST API Overview

1.1 OpenAPI Documentation

OpenAPI documentation is available for SAFR services in the SAFR Cloud via HTTPS.

1.1.1 SAFR Cloud

Access to OpenAPI documentation in the SAFR Cloud can be found at the following locations:

OpenAPI Doc	URL
SAFR Computer Vision API (COVI)	`https://covi.real.com/docs/index.html`
SAFR Computer Vision Events Server API (CVEV)	`https://cv-event.real.com/docs/index.html`
SAFR VIRGA Server API (VIRGA)	`https://virga.real.com/docs/index.html`
SAFR Object Server API (CVOS)	`https://cvos.real.com/docs/index.html`

1.1.2 Local Access

If you're using an on-premise license, API documentation can be accessed locally at the following locations. (Substitute the server's IP address or DNS name as needed.)

OpenAPI Doc	URL
SAFR Computer Vision API (COVI)	`https://<ipaddress or localhost>:8080/docs/index.html` **or** `http://<ipaddress or localhost>:8081/docs/index.html`
SAFR Computer Vision Events Server API (CVEV)	`https://<ipaddress or localhost>:8082/cv-event/docs/index.html` **or** `http://<ipaddress or localhost>:8083/cv-event/docs/index.html`
SAFR VIRGA Server API (VIRGA)	`https://<ipaddress or localhost>:8084/virga/docs/index.html` **or** `http://<ipaddress or localhost>:8085/virga/docs/index.html`
SAFR Object Server API (CVOS)	`https://<ipaddress or localhost>:8086/cvos/docs/index.html` **or** `http://<ipaddress or localhost>:8087/cvos/docs/index.html`

1.2 Video Demos

To assist in using SAFR OpenAPI documentation, view the following video demos:

- Post an image for recognition
- Get a recognition event
- Get face and scene thumbnail images for a recognition event

1.3 System Logs and Privacy

To protect privacy, SAFR limits retention of system logs associated with events to a time frame configured using an admin system API. When used in conjunction with `eventArchiveTimeLimit` in the Admin Tenant API, no trace of individual whereabouts is kept beyond the configured retention time.

Recognition logs are reduced in their default logging level not to include any personally identifiable information (PII).

2 Computer Vision (COVI) REST API

The SAFR Computer Vision Service (COVI) provides the ability to:

- Import a face from an image as a new identity
- Retrieve stored identities
- Delete stored identities
- Retrieve images of stored identities
- Match images against stored identities

The following elements are used with the API requests:

Element	Description	Notes
-H “X-RPCAUTHORIZATION: userid:pwd”	Header. For authentication purposes, all examples use user identifier userid and user password pwd	Substitute userid and pwd with credentials issued for your account.
-H “X-RPC-DIRECTORY: main”	Header. Identifies the directory used.	The directory used in the examples is main; substitute with proper directory name.
localhost	Location of the API endpoint.	Substitute with a proper IP address or domain name based on the location of the service.

3 Import a Face from an Image as a New Identity

Use the following call to import a face from an image as a new identity.

For Local Host:

```
curl -v -X POST -H "Content-Type:application/octet-stream" -H
   "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" -H
   "XRPC-PERSON-NAME:First Last" -H "X-RPC-EXTERNAL-ID: 0000001"
   "http://localhost:8080/people?update=false" --data-binary
   @IMG_0000001.jpg
```

For SAFR Cloud:

```
curl -v -X POST -H "Content-Type:application/octet-stream" -H
   "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" -H
   "XRPC-PERSON-NAME:First Last" -H "X-RPC-EXTERNAL-ID: 0000001"
   "https://covi.real.com/people?update=false" --data-binary
   @IMG_0000001.jpg
```

This call first attempts to match a face already present in the `main` directory of the account and only imports it as a new identity if the face does not match an already existing one, and only if the new face meets default pose, sharpness, and contrast quality.

If a match is found, information about matched identity is returned (`personId`). If a new identity is formed, returned information includes the `newId` property set to `true`:

Note: Always use `https` when making requests over the internet.

Complete request with overrides in place:

```
curl -v -X POST -H "Content-Type:application/octet-stream" -H
   "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" -H
   "X-RPC-PERSON-NAME:0000001" -H "X-RPC-EXTERNAL-ID: 0000001" -H
   "X-RPC-FACES-GROUPINGTHRESHOLD: 0"
   "http://localhost:8080/people?min-cpq=0&min-fsq=0&min-fcq=0&update=false"
   --data-binary @IMG_0000001.jpg
```

3.1 Response

Element	Description	Notes
accountUpdated	Indicates whether COVI has made changes to the account regarding person, face, or metadata about faces	Boolean. If value is `true`, changes have been made to the account.
detectionTime	Amount of time it takes to detect a face	Integer. Value in milliseconds.
identifiedFaces	Array of `identifiedFaces` data	
personid	ID indicates a recognized person	String.
name	Name associated with the recognized person	String.
newId	Identifies whether this is a new identity	Boolean. `true` signifies a new identity.

3.1.1 Sample Response

```
{
  "accountUpdated": true,
  "detectionTime": 324,
  "identifiedFaces": [
                {
              "personId": "866e75a6-e22a-4077-97bb- e5dbfe1c513e",
              "name": "First Last",
              "newId": true,
              ...
                }
        ]
}
```

3.2 Headers

Header Name	Description	Notes
X-RPC-DIRECTORY: main	Indicates directory of the identities used. In this example, directory is `main`. All data in different directories is separate. Use multiple directories when creating multiple identity sets for completely different uses.	(Required) Maximum number of directories supported is 10 per account.
X-RPC-AUTHORIZATION:{userid:pwd}	Provides credentials required for API access and identifies account used. Data in different accounts is entirely separate, even if the directory names in which data resides is the same.	(Required) Substitute the key-value `userid:pwd` with your credentials required for API access to identify the account being used. Data in different accounts is entirely separate, even if the directory names in which data resides is the same.
X-RPC-PERSON-NAME:{First Last}	Person name associated with the inserted identity.	(Required)
X-RPC-EXTERNAL-ID: {0000001}	External ID associated with the inserted identity. In this case, the test ID of the person is being used. This external ID is included in recognition responses, to allow the identity to be correlated to information maintained externally to the SAFR system.	(Required) Identity metadata can be retrieved from the SAFR system by `externalId` as well as internal `personId` that are returned in insertion call response.

Header Name	Description	Notes
X-RPC-FACES-GROUPING-THRESHOLD: 0 Example: “identifiedFaces”: [{ “personId”: “866e75a6-e22a-4077-97bb-e5dbfe1c513e”, “personExternalId”: “0000001”, “name”: “0000001”, “newId”: true, ... }]	Use to force insertion of the face as new identity even when a match can be normally found. Ensures that a newly detected face is inserted as a new identity, rather than being considered the same as an identity already present. Every insertion call is preceded with the check of the identity being inserted, to see if it is already present in the indicated directory. If present, the insertion is not made; the other already present identity is returned in response.	The threshold of `0` makes almost certain the identity being inserted is considered different from any other identity. In case of an identical image already in the directory, the face in the image would not be inserted — even at threshold of `0` — but be considered already matching existing identity. Attribute `newId=true` in the response indicates if this call inserted the new face and generated a new identifier.

3.3 Query Parameters

Element	Description	Notes
min-cpq	Refers to Center Pose Quality.	Set to 0 to ensure that image insertion is not rejected due to poor facial image quality. The API normally rejects the insertion of facial images that don’t have sufficient quality, according to configurable criteria to be used as solid reference. Storing low-quality references degrades the accuracy of the system and should be avoided if possible.
min-fsq	Refers to Face Sharpness Quality.	See `min-cpq` notes.
min-fcq	Refers to Face Contrast Quality.	See `min-cpq` notes.
update	Disallows or allows information from a new face to update image and metadata for a face that is a match to an identity already inserted in the indicated directory.	Set to `false` to disallow the update from the new face. Set to `true` if the last face posted is to be used to update the existing matching reference (assuming it meets the insertion quality criteria).

4 Retrieve Stored Identities

Use the following API requests to retrieve stored identities:

Sample API Request	Description
curl -v -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" "http://localhost:8080/coviws/people"	Retrieve all inserted identities from a directory containing less than 10,000 faces.
curl -v -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" "http://localhost:8080/coviws/rootpeople?start-index=0"	Retrieve inserted identities from the directory in a highly efficient paged manner, supporting directories of any size. Note that the query related to rootpeople is, by default, capped at 100. Thus, if a directory has more than 100 the count query param should be used. Setting `count=0` means that no query limit should be applied.
curl -v -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" "http://localhost:8080/coviws/people/866e75a6-e22a-4077-97bb-e5dbfe1c513e"	Use an identity's `personId` to retrieve it from the Identity Database.
curl -v -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" "http://localhost:8080/coviws/people/external/0000001"	Use an identity's `externalId` (e.g. "0000001") to retrieve it from the Identity Database.

5 Delete Stored Identities

To delete an identity from the directory using its `personId`, use the following API request:

API Request	Description	Notes
curl -v -X DELETE -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userid:pwd" "http://localhost:8080/people/866e75a6-e22a-4077-97bb-e5dbfe1c513e?recursive=true"	Use an identity's `personId` to delete it from the Identity Database.	The `recursive=true` parameter ensures that all persons merged into the same identity are deleted. Each merged person may be represented with different face modalities of the person (e.g. with sunglasses, without sunglasses, and with makeup). If not specified, only a specific person (i.e. face modality) is deleted while retaining others (if they exist).

6 Retrieve Images of Stored Identities

6.1 Easy Method

The easiest way to retrieve face images of a stored identity is by doing the following:

- Use the object server `GET /person/<personId>/face` API and `personId` returned by `GET /people or GET /rootpeople` requests.

This method works for cloud and locally hosted systems for images with and without application-level encryption applied to them. It is not, however, as efficient as other methods. For more information, see the Efficient method section.

6.1.1 For Local Host

```
curl -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION:userid:pwd"
"http://localhost:8086/person/<personId>/face"'
```

6.1.2 For SAFR Cloud

```
curl -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION:userid:pwd"
"https://cvos.real.com/person/<personId>/face"
```

Example:

```
curl -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION:userid:pwd"
"https://cvos.real.com/person/60046fb9-5b5d-4d2b-a3aa-6345e43da53d/face"
```

6.2 Efficient Method

For a highly efficient and the most direct method of retrieval of images stored for identities, use the identity image URL. The stored identity image URL is returned by `GET /people` and `GET /rootpeople` requests in the `imageURI` property. The image referenced is used to represent the identity.

Note: Use `unmergedImageURIis` if the image that is different from the one referenced by `imageURI` exists for the identity and can be used to retrieve the image of the specific face modality (for example, a facial image with sunglasses) associated with a matching `personId`.

For a locally hosted system

For a locally hosted system, the provided URI will be of the following form: `cvos://obj/<image-guid>`

Retrieve the image by issuing the following `http://` request to the Object Server (CVOS) API endpoint:

```
curl -X GET -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION:
   userid:pwd"
"http://localhost:8086/obj/<imageguid>"
```

For a cloud hosted system

- For images not requiring application-level decryption (applicable to some early and demo accounts), the provided UI will be of the following form: `https://<path>`
- For application-level encrypted images, the provided URI will be of the following form: `ehttps://<path>`

  Images referenced via the ehttps:// scheme will have to be decrypted after being downloaded before they can be viewed. The encrypted image data can be downloaded by using https:// protocol scheme with the same path: `https://<path>`

The decryption is described in the next section.

6.3 Decrypt Application-Level Encrypted Images

Images referenced via the `https://` scheme (applicable to cloud accounts only when efficient image retrieval method is used) must be decrypted after being downloaded before they can be viewed. The encrypted image data can be downloaded by using the `https://` protocol scheme with the same path provided in the `https:// URL`: `https://<path>`

To decrypt the downloaded image:

1. Retrieve the account-specific decryption key using the following API request:

   ```
   curl -X GET -H "X-RPC-DIRECTORY: main" -H
      "X-RPC-AUTHORIZATION:userid:pwd""https://covi.real.com/obj/imagekey"
   ```

   The response contains a JSON formatted base64 form of the key: `{ "key": "<base64_encoded_key>"}`

   Example:

   ```
   { "key": "yJgLFSH/Ypsb1wycx2TzZnvTwCob4KZHrcIYgqZxrz0=" }
   ```

2. To decrypt the image, extract the 16-byte IV prefix from the encrypted image data and the rest of the data decrypted using the IV and the decryption key.

   - `IV = first_16_bytes_of_encrypted_image_data`
   - `Key = base64Decode(base64_encoded_key)`
   - `EncryptedImageBody = encrypted_image_data_starting_with_byte_17`
   - `DecryptedImage = getCipher(AES/CBC/PKCSPadding).decode(IV, Key, EncryptedImageBody)`

   Example:

   ```
   openssl dec -aes-256-cbc -in EncryptedImageBody.enc -out ImageBody.jpg -K
      $Key -iv $IV
   ```

7 Match Images Against Stored Identities

Use the following parameters in your POST request to perform recognition of an image against the identities inserted in a directory, and request a specific number of matches along with match probability.

7.1 Method, Headers, and URL

`POST -H "Content-Type:application/octet-stream" -H "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userId:pwd" "http://localhost:8080/people`

Element	Description	Notes
X-RPC-AUTHORIZATION	Header information for authentication purposes.	(Required) Substitute the `userid` (user identifier) and `pwd` (password) with the credentials issued for your account. Example: `"X-RPC-AUTHORIZATION: exampleInc:123456"`
X-RPC-DIRECTORY: main	The header identifies the directory used.	If necessary, substitute `main` with the appropriate directory name.
localhost	Location of the API endpoint.	Substitute with a proper IP address or domain name based on the service location.
–data-binary @{image file name}	Identifies the graphic file used to perform recognition against the identities in a directory.	(Required) Where {image file name} is the graphic file name to be used.
insert=false&update=false	Ensures nothing in the directory is inserted or updated.	(Required)
similar_limit={integer}	Requests the top number of matches with probabilities of match.	(Optional) Valid values are integers.

Use `similar_limitquery` parameter in `POST /people` request.

7.2 cURL Examples

- Perform recognition in an image against the identities inserted into a directory:

```
curl -v -X POST -H "Content-Type:application/octet-stream" -H
   "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userId:pwd"
"http://localhost:8080/people?insert=false&update=false" --data-binary
   @IMG_0001001.jpg
```

- Request the top five (5) probability matches:

```
curl -v -X POST -H "Content-Type:application/octet-stream" -H
   "X-RPC-DIRECTORY: main" -H "X-RPC-AUTHORIZATION: userId:pwd"
"http://localhost:8080/people?insert=false&update=false&similar_limit=5"
   --data-binary @IMG_0001001.jpg
```

Note: The previous request returns the top five (5) matches for each face found in the image. The order in which found faces/identities are returned is arbitrary; they're not sorted by `similarityScore`. For each found

face, the top five (5) matching identities are provided in order of most similar first (highest `similarityScore` first).

The probability of a match is returned in the `similarityScore` attribute of records returned in the similar array. The score of 1.0 (100%) is tied to the set `X-RPC-FACES-GROUPING-THRESHOLD`; faces matching exactly at the threshold have `similarityScore` of 1.0. Those matching to a greater extent score > 1.0, while those matching to a lesser extent score < 1.0 .

The lower bound is 0, and the upper bound is `2 / (2 - sqrt(X-RPC-FACES-GROUPINGTHRESHOLD))`. By default the `X-RPC-FACES-GROUPING-THRESHOLD` setting is 0.54; the upper bound is 1.581. The identity returned outside of the similar array is the best match with `similarityScore` >= 1.0, if any.


```
{
    "accountUpdated": false,
    "detectionTime": 616,
    "identifiedFaces": [
    {
        "attributes": {
            "centerPoseQuality": 0.58317417,
            "confidence": 0.99978894,
            "contrastQuality": 0.61625,
            "detectionVersion": 2,
            "dimension": {
                "height": 200,
                "width": 164
            },
            "landmarks": {
                "left-eye-center": {
                    "x": 0.31614387,
                    "y": 0.38013837
                },
                "left-mouth-corner": {
                    "x": 0.34892383,
                    "y": 0.7837046
                },
                "nose-tip": {
                    "x": 0.55630475,
                    "y": 0.6482113
                },
                "right-eye-center": {
                    "x": 0.7781885,
                    "y": 0.37358207
                },
                "right-mouth-corner": {
                    "x": 0.74510413,
                    "y": 0.7801639
                }
            },
            "provider": "RCV",
            "sharpnessQuality": 0.6402198
        },
    "lastOccurrenceDate": 1521757646850,
    "mediaId": "a058d139-4429- 4e3f-8603- 38290ac d3326",
    "occurrence": 2,
    "offsetX": 0.23326,
    "offsetY": 0.36203,
```

```
    "personId": "60046fb9-5b5d-4d2b-a3aa-6345e43da53d",
    "relativeHeight": 0.10982,
    "relativeWidth": 0.06004,
    "rootPersonAddDate": 1521747721069,
    "similar": [
        {
            "ignore": false,
            "lastOccurrenceDate": 1521747721069,
            "occurrence": 1,
            "personId": "60046fb9-5b5d-4d2b-a3aa-6345e43da53d",
            "similarityScore": 1.3789116
        },
        {
            "ignore": false,
            "lastOccurrenceDate": 1521747721069,
            "occurrence": 1,
            "personId": "328a6c14-c4b6-483b-8163- ac7390078a3f",
            "similarityScore": 0.6425859
        },
        {
            "ignore": false,
            "lastOccurrenceDate": 1521747721069,
            "occurrence": 1,
            "personId": "48328c18-044d-41d8-93c6- 73b7e6b68297",
            "similarityScore": 0.5377595
        },
        {
            "ignore": false,
            "lastOccurrenceDate": 1521747721069,
            "occurrence": 2,
            "personId": "83b86722-99d2-4327-8ded- a21be3d2382c",
            "similarityScore": 0.42808503
        },
        {
            "ignore": false,
            "lastOccurrenceDate": 1521723923106,
            "occurrence": 1,
            "personId": "ae4c5292-1e3a-4495-ba9a-6bd3d6c371e2",
            "similarityScore": 0.4004748
            }
    ],
    "tags": []
        }
    ]
}
```


7.3 API Response

The response is described in the following table:

Element	Description	Notes

identifiedFaces	The esponse returns one entry in `identifiedFaces` array for each detected face.	The order in which found faces (identities) are returned is arbitrary (i.e. they're not sorted by `similarityScore`)
		
		Example:
		`"identifiedFaces": [ { ... }, { ... }, { ... }, ... ]`
		Only positively identified faces (i.e. recognized with a `similarityScore` >= 1.0) are given a `personId`:
		`"identifiedFaces": [ { "personId": "866e75a6-e22a-4077-97bb-e5dbfe1c513e", "personExternalId": " 0000001", "name": "0000001", ... }, { ... }, { ... }, ... ]`

X-RPC-FACES-GROUPING-THRESHOLD	Positive identification is governed by `X-RPC-FACES-GROUPING-THRESHOLD`. To override the default setting, the `X-RPC-FACES-GROUPING-THRESHOLD` header can be passed in each request.	

7.4 Threshold Setting Guidelines

- 0.54 – Secure Access (default)
  - Very high confidence identification with very low false positives
- 0.67 – Traffic monitoring
- 0.84 – Celebrity matching
- 0.94 – Matching the list of interest in very small, grainy, blurry images or video.

7.5 Similar Identities

For the `similar_limit=5` parameter, each detected face contains the five most similar identities. Each similar identity has a `personId` and, if set, an `externalId`, as well as a `similarityScore`. The identities in the `similar` array are sorted by `similarityScore` in descending order with the highest `similarityScore` first:

```
"identifiedFaces": [
              {
                ...
                "similar" : [
                                    {
                        "personId": "866e75a6-e22a-4077-97bb-e5dbfe1c513e",
                        "name": " 0000001 ",
                        "externalId": " 0000001 ",
                        "similarityScore": 1.0804045,
                        ...
```

```
                                        },
                                        {
                        "personId": "92ca0413-4e67-4981-a649-cffa29c6d56c",
                        "name": " 0000023 ",
                        "externalId": " 0000023 ",
                        "similarityScore": 0.62635994,
                        ...
                                         },
                                         ...
              ]
            },
            {
              ...
              "similar" : [
                             ...
              ]
             },
             ...
]
```

7.6 Face Attributes

Each detected face has attributes detailing positioning, face landmarks, face image quality, CV algorithm provider (RCV), and face detection confidence. None of this is relevant for identification.

```
"identifiedFaces": [
    {
        ...
        "attributes": {
        provider": "RCV",
        ...
        }
    },
    {
        ...
        "attributes": {
        "provider": "RCV",
        ...
        }
    },
    {
        ...
        "attributes": {
        "provider": "RCV",
        ...
        }
    }
]
```

8 Computer Vision Events (CVEV) Server API

The Computer Vision Events Server API (CVEV) provides the ability to:

- Retrieve events stored in the directory
- Retrieve images associated with the events
- Listen for new events and retrieve them as they occur

The following elements are used with the API requests:

Element	Description	Notes
-H “X-RPC-AUTHORIZATION: userid:pwd”	Header. For authentication purposes, all examples use user identifier userid and user password pwd.	Substitute userid and pwd with credentials issued for your account.
-H “X-RPC-DIRECTORY: main”	Header. Identifies the directory used.	The directory used in the examples is `main`; substitute with the proper directory name.
localhost	Location of the API endpoint.	Substitute with a proper IP address or domain name based on the location of the service.

9 Retrieve Events Stored in the Directory

To retrieve all events recorded in a directory, use the following request:

For local host:

```
curl -X GET "http://localhost:8082/events?sinceTime=0" -H "X-RPC-AUTHORIZATION: userId:pwd"
-H "X-RPC-DIRECTORY: main"
```

For SAFR Cloud:

```
curl -X GET "https://cv-event.real.com/events?sinceTime=0" -H "X-RPC-AUTHORIZATION:
userId:pwd" -H "X-RPC-DIRECTORY: main"
```

Note: Always use `https` when making requests over the internet.

To retrieve events currently in progress (not yet ended):

```
curl -X GET "http://localhost:8082/events" -H "X-RPC-AUTHORIZATION: userId:pwd" -H
"X-RPC-DIRECTORY: main"
```

To retrieve any events that occurred in last 60 seconds:

```
curl -X GET "http://localhost:8082/events?sinceTime=<currentEpochTimeInMs-60000>" -H
"X-RPC-AUTHORIZATION: userId:pwd" -H "X-RPC-DIRECTORY: main"
```

Element	Description	Notes
http://{domain name}/events	URL for the endpoint	Substitute {domain name} with your local host name or SAFR Cloud domain
X-RPC-AUTHORIZATION: userId:pwd	See Event Service (CVEV) API for more information	
X-RPC-DIRECTORY: main	See Event Service (CVEV) API for more information	
sinceTime=<currentEpochTimeInMs-60000>	For use in determining the precise past time for retrieving events.	Replace `=<currentEpochTimeInMs-60000>` with the current epoch time in milliseconds subtracted by 60000. `sinceTime=0` returns all events recorded in a directory.

10 Retrieve Images Associated with Events

Images associated with events — if recorded — can be retrieved from the object server using the event ID.

To retrieve a face image that triggered the event, use the following call:

For Local Host:

```
curl -v -X GET "http://localhost:8086/obj/<base64{event_id}>/face" -H
   "X-RPC-AUTHORIZATION: userId:pwd"
-H "X-RPC-DIRECTORY: main" > face.jpg
```

For SAFR Cloud:

```
curl -v -X GET "https://cvos.real.com/obj/<base64{event_id}>/face"
-H "X-RPC-AUTHORIZATION: userId:pwd=" -H "X-RPC-DIRECTORY: main" >
   face.jpg
```

Note: Always use `https` when making requests over the internet.

In these calls, replace `<base64{event_id}>` with a base64 form of the Event ID GUID for which the image is required.

For example, an Event ID GUID value of `D4565B3D-D5C2-4F02-AD81-49DE55AAEFF1`, should be replaced with `RDQ1NjVCM0QtRDVDMi00RjAyLUFEODEtNDlERTU1QUFFRkYx`.

10.1 Example

```
echo -n D4565B3D-D5C2-4F02-AD81-49DE55AAEFF1 | base64
   RDQ1NjVCM0QtRDVDMi00RjAyLUFEODEtNDlERTU1QUFFRkYx
```

To retrieve a scene thumbnail associated with the event, if recorded, issue the following request:

```
curl -v -X GET "http://localhost:8086/obj/<base64{event_id}>/sceneThumb"
-H "X-RPC-AUTHORIZATION: userId:pwd" -H "X-RPC-DIRECTORY: main" >
   scene.jpg
```

11 Listen for New Events and Retrieve Them As They Occur

To listen for new events as well as modifications to prior events, issue the following event status request:

```
curl -X GET
   "http://localhost:8082/event/status?since=<currentEpochTimeInMs>"
-H "X-RPC-AUTHORIZATION: userId:pwd" -H "X-RPC-DIRECTORY: main"
```

Note: Replace `=<currentEpochTimeInMs>` with the current epoch time in milliseconds.

This request blocks until a new event is recorded or started, or until an existing event is updated. Previously recorded but still active events without an end date undergo updates (e.g. a person identity being assigned or an event simply being given an end time as the face disappears from view).

If the request times out or returns `HTTP 204`, submit the request again. Once the request returns `HTTP 200`, it includes the following information:

Element	Type	Notes
lastModDate	integer	Epoch time in milliseconds of last inserted or updated event.
serverDate	integer	Epoch time on server, in milliseconds.
since	integer	Ensures all events returned that have ended–`endDate`, prior to their parameter, are excluded from query results. Number based on epoch time in milliseconds.
sinceModDate	integer	Ensures all events returned have a `modDate` greater than their parameter. Number based on epoch time in milliseconds.

On a `HTTP 200` response, the record returns `lastModDate` and retrieves all newly added or updated events:

```
curl -X GET
   "http://localhost:8082/events?sinceModDate=<epochTimeInMsUsedInStatusCallAbove>"
-H "X-RPC-AUTHORIZATION: userId:pwd" -H "X-RPC-DIRECTORY: main"
```

Note: Replace `<epochTimeInMsUsedInStatusCallAbove>` with the value used in the `since` parameter of the `/event/status` request described previously.

Once the events are retrieved, listen for more events by issuing an `/event/status` request again, but instead use the value of the recorded `lastModDate` for the value of the `since` parameter. Continue to repeat by alternating retrieval and listening steps:

```
curl -X GET "http://localhost:8082/event/status?since=<lastModDate>"
-H "X-RPC-AUTHORIZATION: userId:pwd" -H "X-RPC-DIRECTORY: main"
```