HyperVerge India KYC API Documentation

Overview

This documentation describes version1(v1) HyperVerge's proprietary API for ascertaining if an image of a user is not a photo of a photo(not live). The postman collection can be found at this link.

Schema

We recommend using HTTPS for all API access. All data is received as JSON, and all image uploads are to be performed as form-data (POST request).

Parameters

All optional and compulsory parameters are passed as part of the request body.

Header

Only 'appId', 'appKey' and 'content-type' are required fields in the header. Optional parameters are described later

Root Endpoint

The root production endpoint for this API is

```
https://ind-liveness.hyperverge.co/v1/photo
```

Authentication

Currently, a simple appId, appKey combination is passed in the request header. The appId and appKey are provided on request by the HyperVerge team. If you would like to try the API, please reach out to contact@hyperverge.co

```
curl -X POST https://ind-liveness.hyperverge.co/v1/photo/verifyLiveness \
  -H 'appid: xxx' \
  -H 'appkey: yyy' \
  -H 'content-type: multipart/form-data;' \
  -F 'image=@abc.png'
```

On failed attempt with invalid credentials or unauthorized access the following error message should be received :

```
{
  "status": "failure",
  "statusCode": "401",
  "error": "Missing/Invalid credentials"
}
```

Please do not expose the appid and appkey on browser/front-end applications. In case of a browser/front-end application, a temporary authorization token can be received for a small duration of time. For more details, please contact your Point of Contact at HyperVerge.

Media Types

Currently, `jpeg, png and tiff` images are supported by the the Liveness API.

1. `/verifyLiveness` on an image

   ```
   curl -X POST https://ind-liveness.hyperverge.co/v1/photo/verifyLiveness \
     -H 'appid: xxx' \
     -H 'appkey: yyyy' \
     -H 'content-type: multipart/form-data;\
     -F 'image=@image_path.png'
   ```

Input Image Constraints

Recommendations on input image to ensure highest accuracy:

- Please use a portrait image
- Preferred Image size: Height 1280 x Width 960
- Maximum Image size: Height 1920 x Width 1080
- Minimum Image size: Height 640 x Width 480
- Size (width) of the face in the image should be between 20% to 60% of the original image. Preferred 40%

If integrating into a native Android or iOS app, please use the HyperSnap SDK for Android and iOS. This SDK provides in-built liveness API with necessary optimizations and quality control.

API Call Structure

- **Endpoint**: /verifyLiveness
- **Method:**: `POST`
- **Header**
  - content-type : 'formdata'
  - appid
  - appkey
- **Request Body**
  1. image: content of the image;

Response Structure

- **Success Response:**
  - **Code:** 200
    Incase of a properly made request, the response would follow schema.

```
{
    "status" : "success",
    "statusCode" : "200",
    "result" : {
        "live": "<yes/no>",
        "liveness-score": "<score>",
        "to-be-reviewed": "<yes/no>"
    },
}
```

- **Error Response:**

There are multiple types of request errors and following error is returned in all of these cases:

1. Missing or invalid credentials

```
{
  "status": "failure",
  "statusCode": "400",
  "error": "Missing/Invalid credentials"
}
```

2. No Image input

```
{
  "status": "failure",
  "statusCode": "400",
  "error": "image is required"
}
```

3. Face is not detected in the image

```
{
    "status": "success",
    "statusCode": "200",
    "result": {
        "error": "No face detected"
    }
}
```

4. Face Size is small in proportion to the image

```
{
    "status": "failure",
    "statusCode": "422",
    "error": "Face size in proportion to image is too small"
}
```

5. Face Size is big in proportion to the image

```
{
    "status": "failure",
    "statusCode": "422",
    "error": "Face size in proportion to image is too big"
}
```

6. Eyes Closed when value passed for *allowEyesClosed* parameter is **no**

```
{
    "status": "failure",
    "statusCode": "423",
    "error": "Eyes closed"
}
```

7. Rate limit error: This error occurs when input rate is greater than the rate allocated to a credential.

```
{
    "statusCode": 429,
    "status": "failure",
    "message": "Requests rate limit exceeded"
}
```

- **Server Errors** We try our best to avoid these errors, but if by chance they do occur the response code will be 5xx.

Sample Code

```
curl -X POST https://ind-liveness.hyperverge.co/v1/photo/verifyLiveness \
      -H 'appid: xxx' \
      -H 'appkey: yyyy' \
      -H 'content-type: multipart/form-data;' \
      -F 'image=@image_path.png'
```

Optional parameters

Following are optional parameters that could be set as part of the request body.

parameter	Body or Header	value	default	description
referenceId	Header	String	null	`referenceId` is an identifier for the API call and is echoed back by the server in the response. This referenceId can be used to create logical grouping of single/multiple API calls based on business logic of the consumer of API
uuid	Header	String	null	If `uuid` is present in the headers of the API call, response will include an `X-Response-Signature` header which will have a checksum of the response body signed with `uuid` sent in the request and a private key(whose public key will be provided by HyperVerge). This is an advanced feature designed to prevent tampering of response by a man in the middle. For more details on implementation of this feature, please contact your Point of Contact from HyperVerge
allowEyesClosed	Body	"yes"/"no"	"yes"	This parameter adds an additional verification for eyes being closed. If passed value is *no* and eyes are closed in the passed image, then a `423` error will be sent by the server with appropriate error message
dataLogging	Body	"yes"/"no"	"no"	By default, the input images are not stored by HyperVerge systems, however, if the 'dataLogging' parameter is set to "yes", then the images will be stored and the requestId can be provided to HyperVerge to check the uploaded image incase of an inaccurate extraction. **This will also provide access to HyperVerge and your team for debugging various issues that might have occurred during AI processing**