**How do I verify someone’s biometrics in BioPass ID ABIS?**

To perform the operation of verifying an individual’s biometrics, you need to use the **Verify** method. This operation conducts a 1:1 biometric comparison, using the biometric data of an individual with someone already registered in the database. Whether using facial or fingerprint biometrics, the response will be a message confirming whether the biometrics belong to the same person or not.

To use this operation, the client must make the call through the API. The complete documentation is available at <https://docs.biopassid.com/>.

**Input fields:**

* **CandidateId:** Unique identifier of the candidate as registered in the database.
* **Face:** Facial biometric information, including:
  + **ImageFileName:** Type of image to be provided (e.g, “WSQ”, “PNG”).
  + **ImageBase64:** Facial biometric image encoded as a base64 string.
  + **HorzResolution:** Horizontal image resolution in PPI.
  + **VertResolution:** Vertical image resolution in PPI.
* **Fingers:** List of objects containing each digital biometric to be used, as well as additional information:
  + **FingerName:** Name of the finger or, if no name, use the same value as “Position”.
  + **Position:** Positions of the finger from which the biometric was extracted, which can be: Unknown, RightThumb, RightIndex, RightMiddle, RightRing, RightLittle, LeftThumb, LeftIndex, LeftMiddle, LeftRing, LeftLittle.
  + **CollectedType:** Type of fingerprint capture, which can be: “Plain” for flat capture, “Rolled” for rolled capture, and “Latent” for latent capture.
  + **ImageFileName:** Type of image to be provided (e.g., “WSQ”, “PNG”).
  + **ImageBase64:** Fingerprint image encoded as a base64 string.
  + **HorzResolution:** Horizontal image resolution in PPI.
  + **VertResolution:** Vertical image resolution in PPI.
  + **Anomalydescription:** Additional information about fingerprint anomalies, if any, specifying the type and period (permanent, temporary) of the anomaly.

Example of how the request body should look:

{

"CandidateId": "example-candidate-id",

"Face": {

"ImageFileName": "string",

"ImageBase64": "string",

"HorzResolution": 0,

"VertResolution": 0

},

"Fingers": {

"Finger": [

{

"FingerName": "string",

"Position": "string",

"CollectedType": "string",

"ImageFileName": "string",

"ImageBase64": "string",

"HorzResolution": 0,

"VertResolution": 0,

"AnomalyDescription": {

"AnomalyType": "string",

"AnomalyTimeFrame": "string"

}

}

]

},

"DoMatchIfFound": true

}

The request header, in turn, is where the content type should be specified. Below is an example of how the header should be represented:

{

'Content-Type': 'application/json'

}

Finally, the request should include parameters specifying that it is a POST method, the URL ‘https://abis.api.biopassid.com/verify/post’, as well as the header and body described earlier. Below, we present a Python code example for making this request. Examples in other programming languages can be found in our documentation.

import http.client

import json

conn = http.client.HTTPSConnection("abis.api.biopassid.com")

payload = json.dumps({

"CandidateId": "example-candidate-id",

"Face": {

"ImageFileName": "string",

"ImageBase64": "string",

"HorzResolution": 0,

"VertResolution": 0

},

"Fingers": {

"Finger": [

{

"FingerName": "string",

"Position": "string",

"CollectedType": "string",

"ImageFileName": "string",

"ImageBase64": "string",

"HorzResolution": 0,

"VertResolution": 0,

"AnomalyDescription": {

"AnomalyType": "string",

"AnomalyTimeFrame": "string"

}

}

]

},

"DoMatchIfFound": True

})

headers = {

'Content-Type': 'application/json'

}

conn.request("POST", "/verify/post", payload, headers)

res = conn.getresponse()

data = res.read()

print(data.decode("utf-8"))

The response received in the case of a successful operation will include several response fields, as described below.

**Response fields:**

* **Success:** Boolean value indicating the success of the operation.
* **Message:** Additional message regarding the success of the operation.
* **Status:** Operation status (e.g., Processing, Done, Error).
* **MatchingSubject:** Object containing information about the individual with whom the biometric match occurred.
  + **ReferenceCandidateId:** Internal identifier of the candidate passed in the operation call.
  + **IdentifyCandidateId:** Internal identifier of the individual with whom the biometric match occurred.
  + **Score:** Numerical value representing the confidence level of the biometric match.
  + **ScoreFace:** Numerical value representing the confidence level of the facial biometric match.
  + **ScoreFinger:** Numerical value representing the confidence level of the fingerprint biometric match.
* **Faces:** Object containing information about the submitted facial biometrics and the match confidence score.
* **Fingers:** Object containing information about the submitted fingerprint biometrics and the match confidence score.

The response message for the provided example can be checked below:

{

"Success": true,

"Message": "string",

"Status": "string",

"MatchingSubject": {

"ReferenceCandidateId": "example-candidate-id",

"IdentifyCandidateId": "example-candidate-id",

"ScoreN": 0,

"Score": 0,

"ScoreFaceN": 0,

"ScoreFace": 0,

"ScoreFingerN": 0,

"ScoreFinger": 0,

"Threshold": 0,

"FacesMatched": [

{

"SondaTemplateFaceId": "example-candidate-id",

"ReferenceTemplateFaceId": "example-candidate-id",

"FaceSourceMatchedId": 0,

"MatchedIndex": 0,

"Score": 0,

"ScoreN": 0

}

],

"FingerMatched": [

{

"FingerSourcePosition": 0,

"FingerMatchedId": 0,

"SondaTemplateFingerId": "example-candidate-id",

"ReferenceTemplateFingerId": "example-candidate-id",

"FingerSourceMatchedId": 0,

"ScoreN": 0,

"Score": 0,

"CenterX": 0,

"CenterY": 0,

"BiometricType": 0,

"TranslationX": 0,

"TranslationY": 0

}

]

},

"Faces": {

"Faces": [

{

"Id": "string",

"CreateDateUtc": "2024-07-02T18:11:33.367Z",

"PathImageBlob": "string",

"ImageFileName": "string",

"ImageBase64": "string",

"ImagePngBase64": "string",

"ImageThumbnail120PngBase64": "string"

}

],

"FacesMatch": [

{

"InicialImageGuid": "example-candidate-id",

"ReceivedImageFileName": "string",

"Score": 0,

"Threshold": 0

}

]

},

"Fingers": {

"Fingers": [

{

"Id": "string",

"CreateDateUtc": "2024-07-02T18:11:33.367Z",

"PathImageBlob": "string",

"ImageFileName": "string",

"ImageBase64": "string",

"ImagePngBase64": "string",

"ImageThumbnail120PngBase64": "string",

"Nfiq1": 0,

"Nfiq2": 0,

"Nfiq21": 0,

"Position": 0,

"PositionName": "string",

"CollectedType": "string",

"AnomalyTimeFrame": "string",

"AnomalyType": 0,

"AnomalyTypeName": "string"

}

],

"FingersMatch": [

{

"InicialImageGuid": "example-candidate-id",

"ReceivedImageFileName": "string",

"Score": 0,

"Threshold": 0,

"StatusName": "string",

"ErrorMessage": "string"

}

]

}

}

If an error is returned, verify that the request body and content are correctly filled out. If you receive a message with the code 200 (OK), it means your request was successfully requested and executed, and you can proceed with subsequent requests.