**How to register a person in BioPass ID ABIS?**

To perform the operation of registering an individual, you need to use the **Enroll** method. This operation is one of the main functionalities of BioPass ID ABIS and aims to record a person's biometric information in the system, allowing for their recognition in future consultations.

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

**Input fields:**

* **CustomId:** Unique identifier of the individual in the database.
* **EnrollWithDeduplication:** It aims to indicate the need to perform deduplication of the record.
* **BiographicData**: Biographic information of the user, such as name, CPF, date of birth, and signature.
* **Face:** Object containing information about the facial biometrics for the registration.
* **Fingers:** Object containing information about the fingerprint biometrics for each finger to be registered.

An example of how the request body would look like:

{

"Candidate": {

"GalleryNames": [

"string"

],

"CustomId": "string",

"EnrollWithDeduplication": true,

"BiographicData": {

"Name": "string",

"Id": "string",

"Renach": [

{

"Id": "string"

}

],

"BirthDate": "2024-07-01",

"MotherName": "string",

"FatherName": "string",

"Gender": "string",

"Signature": {

"ImageFileName": "string",

"ImageBase64": "string"

},

},

"Face": {

"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"

}

}

]

}

},

"PriorityOrder": 0,

"DelayOrder": 0

}

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

{

'Content-Type': 'application/json'

}

Lastly, in the request, the parameters specifying that it is a POST method, the URL ‘https://abis.api.biopassid.com/enroll/create’, along with the previously described header and body, must be included. Below, we present an example of how the code for this request would look like, written in Python. 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({

"Candidate": {

"GalleryNames": [

"string"

],

"CustomId": "string",

"EnrollWithDeduplication": True,

"BiographicData": {

"Name": "string",

"Id": "string",

"Renach": [

{

"Id": "string"

}

],

"BirthDate": "2024-07-01",

"MotherName": "string",

"FatherName": "string",

"Gender": "string",

"Signature": {

"ImageFileName": "string",

"ImageBase64": "string"

},

},

"Face": {

"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"

}

}

]

}

},

"PriorityOrder": 0,

"DelayOrder": 0

})

headers = {

'Content-Type': 'application/json'

}

conn.request("POST", "/enroll/create", payload, headers)

res = conn.getresponse()

data = res.read()

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

The response received in case of a successful operation will include several response fields, as we can see below.

**Response fields:**

* **Success:** Boolean value indicating whether the operation was successful.
* **Message:** Additional message regarding the success of the operation.
* **TransactionId:** Internal ID of the performed operation.
* **Candidate:** Information about the newly registered candidate. These include:
* **Person:** Object created in the database with candidate information, including their identifier, facial and digital biometrics (if provided), among others.
* **BiographicData:** Biographical data of the individual passed in the operation call.
* **Faces:** Information about the individual's facial biometrics.
* **Fingers:** Information about the individual's digital biometrics.
* **CreateDateUtc:** Date of the individual's registration creation in the database.
* **MatchingCandidates:** Users registered in the database who have a biometric match with the newly registered candidate.
* **CustomId:** Unique identifier of the user in the database.
* **CandidateStatus:** Candidate status.

If an error is returned, please verify if the JSON text is correctly formatted. If you receive a message with code 200 (OK), it means your request was successfully submitted and processed, and you can proceed with further requests. The response message for the provided example can be checked below:

{

"Success": true,

"Message": "string",

"TransactionId": "example-candidate-id",

"Candidate": {

"Id": "string",

"Galleries": [

{

"Id": "example-candidate-id",

"Name": "string",

"Description": "string",

"CreateDateUtc": "2024-07-01T18:09:17.293Z"

}

],

"Person": {

"Id": "string",

"BiographicData": {

"Id": "string",

"Name": "string",

"Signature": {

"Id": "string",

"ImageFileName": "string",

"PathImageBlob": "string",

"ImageBase64": "string",

"ImagePngBase64": "string",

"ImageThumbnail120PngBase64": "string",

"CreateDateUtc": "2024-07-01T18:09:17.293Z"

},

"Renachs": [

{

"Id": "example-candidate-id",

}

],

"BirthDate": "2024-07-01T18:09:17.293Z",

"MotherName": "string",

"FatherName": "string",

"Gender": "string",

"AbisImportedDateUtc": "2024-07-01T18:09:17.293Z"

},

"Faces": {

"Faces": [

{

"Id": "string",

"CreateDateUtc": "2024-07-01T18:09:17.293Z",

"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": "2024-07-01T18:09:17.293Z",

"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"

}

]

},

"CreateDateUtc": "2024-07-01T18:09:17.293Z",

"MatchingPersons": [

{

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

"Score": 0,

"Threshold": 0,

"FacesMatched": [

{

"MatchedIndex": 0,

"Score": 0,

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

"ReferenceTemplateFaceId": "example-candidate-id"

}

],

"FingerMatched": [

{

"FingerMatchedId": 0,

"Score": 0,

"FingerSourceMatchedId": 0,

"FingerSourcePosition": 0,

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

"ReferenceTemplateFingerId": "example-candidate-id"

}

]

}

]

},

"BiographicData": {

"Id": "string",

"Name": "string",

"Signature": {

"Id": "string",

"ImageFileName": "string",

"PathImageBlob": "string",

"ImageBase64": "string",

"ImagePngBase64": "string",

"ImageThumbnail120PngBase64": "string",

"CreateDateUtc": "2024-07-01T18:09:17.293Z"

},

"Renachs": [

{

"Id": "example-candidate-id",

}

],

"BirthDate": "2024-07-01T18:09:17.293Z",

"MotherName": "string",

"FatherName": "string",

"Gender": "string",

"AbisImportedDateUtc": "2024-07-01T18:09:17.293Z"

},

"Faces": {

"Faces": [

{

"Id": "string",

"CreateDateUtc": "2024-07-01T18:09:17.293Z",

"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": "2024-07-01T18:09:17.293Z",

"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"

}

]

},

"CreateDateUtc": "2024-07-01T18:09:17.293Z",

"MatchingCandidates": [

{

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

"Score": 0,

"Threshold": 0,

"FacesMatched": [

{

"MatchedIndex": 0,

"Score": 0,

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

"ReferenceTemplateFaceId": "example-candidate-id"

}

],

"FingerMatched": [

{

"FingerMatchedId": 0,

"Score": 0,

"FingerSourceMatchedId": 0,

"FingerSourcePosition": 0,

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

"ReferenceTemplateFingerId": "example-candidate-id"

}

]

}

],

"CustomId": "string",

"CandidateStatus": "string",

"HitStatus": "string"

}

}