FACECHECK API

Developers Guide

Content description

This is a reference manual and configuration guide for the NeoCheck Face Verification API product. It shows how to interact with the JSON Web API from an external client to verify faces against a previously stored record or list of records in a fast and easy way.

Madrid, March 21th 2019

Disclaimer

The information contained within this document is and shall remain the property of NeoCheck. It must not be produced in whole or in part, or given or communicated to any third party without the prior consent of NeoCheck.

Whilst careful attention have been paid to ensure the appropriate referencing of information and sources, it is possible that a document of this nature might contain some errors. In such cases, upon notification NeoCheck will immediately analyze and make any required corrections or amendments.

Release Notes

Version	Date	Description
1.0	01/10/2018	Initial Version
1.1	21/10/2018	1:1 between 2 images added
1.2	21/03/2019	1:N identification. Face List management

Index

1. NeoCheck API Endpoint

NeoCheck provides an API for Face verifications. This API allows users to:

- Manage face records, storing face images and personal information.
- Manage lists of face records.
- Retrieve face QA information of a portrait image.
- Compare two portrait images (1:1).
- Verify a face against a stored face record (1:1).
- Identify if a face is present in a configured list of face records (1:N).

Endpoint	Environment
`https://neocheck.net/api`	PRO

Features:

Face Detection

Face detection is a process when a subject can be detected by evaluating an image where the subject is present. The face detection process retrieves the location of the face with a high degree of confidence, even in cases when the face is rotated or the subject has a beard or is wearing glasses.

Face Matching

Face matching is a process when a subject can be identified by evaluating his biometric features and comparing them with another image to verify the individual is the person they claim to be. This process is called one-to-one matching because both images are from the same individual. Face matching is a fast way to compare images from the same subject, or a subject with several other subjects, which is called one-to-many matching.

Face QA

Face QA is the process to asses if a given subject image is compliant with the ICAO requirements checks. The different checks performed are:

- Good Vertical Face Position: Face has a good vertical face position
- Horizontally Center Face: Face is horizontally centered
- Good Exposure: Exposure of facial image is appropriate
- Eyes Not Red: Eyes are not red (no red eyes)
- Resolution: Image resolution is appropriate.
- Only One Face Visible: Only one face visible
- Is Frontal: Photo is frontal. Face image is frontally captured.
- Overall Photo Quality: This indicates the final score of the taken image if it is green the photo can be used for the face verification process meeting the min. quality requirements.

2. Authentication

The FaceCheck API access is protected by using basic authentication and OAuth 2.0 authorization tokens.

2.1. Obtain JWT Authentication Token

HTTPS POST method to authenticate. In order to request access to FaceCheck API, you must call the authentication method with your API `client_id` and `client_secret`, along with a valid `username` and `password`. Once validated, you will receive an authorization token, which must be used to call face API methods.

Endpoint	HTTP Method
connect/token	POST

Request Parameters:

Name	Type	Description
content_type	string	`application/x-www-form-urlencoded` indicates that parameters are passing as key/value pairs in the body of the HTTPS message
grant_type	string	`password` indicates that it's an authentication request including basic authentication
client_id	string	your Client Id
client_secret	string	Your client secret
username	string	valid username already created in the system for the Client Id
password	string	valid password for username provided

Response:

If request is correct, you will receive a Success response with the access token generated. This token is only valid for a short period, exposed in the `expires_in` property. You will also receive a `refresh_token`, which allows you to refresh your access token without having to send your credentials again. The API call to refresh token is explained in the next point.

HTTP Code	Body	Description
200	JSON `Authorization` object	Valid OAuth token for request authorization
400	JSON `ErrorMessage` object	Invalid ClientId
400	JSON `ErrorMessage` object	Unsupported grant type
400	JSON `ErrorMessage` object	Invalid credentials
500	JSON `ErrorMessage` object	Internal Error

Example of Successful Authorization Response:

```
ResultCode: HTTP/1.1 200 OK
Content-Type: application/json
Body:
{
  "token_type": "Bearer",
  "access_token": "yourAccessToken",
  "expires_in": 3600,
  "id_token": "youIdToken",
  "refresh_token": "yourRefreshToken"
}
```

Example of Invalid Client Response:

```
ResultCode: HTTP/1.1 400 BadRequest
Content-Type: application/json
Body:
{
  "error": "Invalid ClientId",
}
```

2.2. Token Refresh

If you already have authenticated, you can use the `refresh_token` property provided to renew your `access_token` authorization token, which must be used to call face API methods.

Endpoint	HTTP Method
connect/token	POST

Request Parameters:

Name	Type	Description
content_type	string	`application/x-www-form-urlencoded` indicates that parameters are passing as key/value pairs in the body of the HTTPS message
grant_type	string	`refresh_token` indicates that it's an authentication request including basic authentication
refresh_token	string	the refresh token
client_id	string	your Client Id
client_secret	string	Your client secret

Response:

If request is correct, you will receive a Success response (StatusCode 200), with a new access token, and the same refreh_token. If request is not correct, you will receive a Bad Request response (StatusCode 400), with error description

HTTP Code	Body	Description
200	JSON `Authorization` object	Valid OAuth token for request authorization
400	JSON `ErrorMessage` object	Invalid ClientId
400	JSON `ErrorMessage` object	Unsupported grant type
400	JSON `ErrorMessage` object	Invalid ticket (`refresh_token` not valid)
500	JSON `ErrorMessage` object	Internal Error

2.3. Abusive usage countermeasures

In order to prevent a bad or abusive usage and assure the best performance, there is a maximum number of requests per hour that a user can perform. This maximum number is set to 3600 calls per hour (an average of 1 call per second). After this limit is reached, the Face API will respond with a BadRequest (StatusCode 400) and an error message stating that the maximum number of calls per hour has been reached.

3. Face Records

3.1. Create Face Record

HTTPS POST method to create a new Face Record.

Endpoint	HTTP Method
`v1/face`	POST

Request Parameters:

Name	Type	Description
`authorization`	query parameter	`Bearer access_token` authorization header with a valid access token
`addFaceRecordRequest`	Body: JSON object	JSON object, with properties: - **faceImageBase64**: face image in Base64 string format - **name**: person's name - **surname1**: person's surname1 - **surname2**: (optional) person's surname2 - **externalIdentifier**: (optional) custom identifier - **additionalDataFileBase64**: (optional) file with custom additional data, in Base64 string format - **additionalDataFileName**: (required if additionalDataFileBase64 has value) file name (with extension) of the custom additional data file - **faceListId**: (optional) the Face list in which the face record will be included - **enabled**: (optional) Boolean property to set the record as enabled or disabled. Default value is true

Response:

If request is correct, you will receive a Success response along with the Face Record Identifier (GUID)

HTTP Code	Body	Description
`200`	JSON `addFaceRecordResponse` object	AddFaceRecordResponse object with property **identifier** (GUID)
`400`	JSON `addFaceRecordResponse` object	AddFaceRecordResponse object with a list of **errors** (with properties **code** and **description**)
`401`		Unauthorized. Invalid token
`500`	JSON `ErrorMessage` object	Internal Error

3.2. Update Face Record

HTTPS PUT method to update a Face Record.

Endpoint	HTTP Method
v1/face	PUT

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
updateFaceRecordRequest	Body: JSON object	JSON object, with properties: - **faceImageBase64**: (optional) face image in Base64 string format. Not needed if does not change - **name**: person's name - **surname1**: person's surname1 - **surname2**: (optional) person's surname2 - **externalIdentifier**: (optional) custom identifier - **additionalDataFileBase64**: (optional) file with custom additional data, in Base64 string format. Not needed if does not change - **additionalDataFileName**: (required if already has value or additionalDataFileBase64 has changed) file name (with extension) of the custom additional data file - **faceListId**: (optional) the Face list in which the face record will be included - **enabled**: (optional) Boolean property to set the record as enabled or disabled. Default value is true

Response:

If request is correct, you will receive a Success response along with the Face Record Identifier (GUID)

HTTP Code	Body	Description
200	JSON `updateFaceRecordResponse` object	UpdateFaceRecordResponse object with property **result** = true
400	JSON `updateFaceRecordResponse` object	UpdateFaceRecordResponse object with property **result** = false and a list of **errors** (with properties **code** and **description**)
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

3.3. Delete Face Record

HTTPS DELETE method to delete a Face Record. It can be called using NeoCheck identifier or external identifier

Endpoint	HTTP Method
v1/face/{identifier}	DELETE

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifier	Query parameter: Face Record identifier (GUID)	NeoCheck GUID identifier, retrieved on record creation

Endpoint	HTTP Method
v1/face/externalIdentifier/{identifier}	DELETE

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifier	Query parameter: Face Record external identifier (string)	External identifier, set on Face Record creation

Response:

If request is correct, you will receive a boolean response with the result of the delete operation

HTTP Code	Body	Description
200	boolean	Boolean result of the delete operation
404		If Face Record was not found
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

3.4. Get Face Record

HTTPS GET method to retrieve a Face Record. It can be called using NeoCheck identifier or external identifier

Endpoint	HTTP Method
v1/face/{identifier}	GET

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifier	Query parameter: Face Record identifier (GUID)	NeoCheck GUID identifier, retrieved on record creation

Endpoint	HTTP Method
v1/face/externalIdentifier/{identifier}	GET

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifier	Query parameter: Face Record external identifier (string)	External identifier, set on Face Record creation

Response:

If request is correct, you will receive a Success response along with the Face Record JSON object

HTTP Code	Body	Description
200	JSON `FaceRecordResponse` object	JSON object, with properties: - **identifier**: Person Record Identifier (GUID) - **faceImageBase64**: face image in Base64 string format - **name**: person's name - **surname1**: person's surname1 - **surname2**: person's surname2 - **externalIdentifier**: custom identifier - **additionalDataFileBase64**: file with custom additional data, in Base64 string format - **additionalDataFileName**: file name with extension of the custom additional data file - **enabled**: Boolean property to set the record as enabled or disabled. Default value is true - **faceListId:** face list in which the face record is included - **createdUtc**: date of Face Record creation
404		If Face Record was not found
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

4. Face Lists

4.1. Create Face List

HTTPS POST method to create a new Face List.

Endpoint	HTTP Method
v1/face/list	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
addFaceListRequest	Body: JSON object	JSON object, with properties: - **name**: face list name - **description**: face list description

Response:

If request is correct, you will receive a Success response along with the Face Record Identifier (GUID)

HTTP Code	Body	Description
200	JSON `addFaceListResponse` object	AddFaceListResponse object with property **identifier** (GUID)
400	JSON `addFaceListResponse` object	AddFaceListResponse object with a list of **errors** (with properties **code** and **description**)
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

4.2. Update Face List

HTTPS PUT method to update a Face List.

Endpoint	HTTP Method
v1/face/list	PUT

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
updateFaceRecordRequest	Body: JSON object	JSON object, with properties: - **name**: face list name - **description**: face list description

Response:

If request is correct, you will receive a Success response along with the Face Record Identifier (GUID)

HTTP Code	Body	Description
200	JSON `updateFaceListResponse` object	UpdateFaceR`List`Response object with property **result** = true
400	JSON `updateFaceListResponse` object	UpdateFace`List`Response object with property **result** = false and a list of **errors** (with properties **code** and **description**)
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

4.3. Delete Face List

HTTPS DELETE method to delete a Face List. It must be called using NeoCheck Face List identifier

Endpoint	HTTP Method
v1/face/list/{identifier}	DELETE

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifier	Query parameter: Face List identifier (GUID)	NeoCheck GUID identifier, retrieved on list creation

Response:

If request is correct, you will receive a boolean response with the result of the delete operation

HTTP Code	Body	Description
200	boolean	Boolean result of the delete operation
404		If Face List was not found
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

4.4. Get Face List

HTTPS GET method to retrieve a Face List. It must be called using NeoCheck Face List identifier

Endpoint	HTTP Method
v1/face/list/{identifier}	GET

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifier	Query parameter: Face List identifier (GUID)	NeoCheck GUID identifier, retrieved on face list creation

Response:

If request is correct, you will receive a Success response along with the Face List JSON object

HTTP Code	Body	Description
200	JSON `FaceListResponse` object	JSON object, with properties: - **identifier**: face list Identifier (GUID) - **name**: face list name - **description**: face list description - **createdUtc**: date of face list creation
404		If Face List was not found
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

4.5. Add Face Record to Face List

HTTPS POST method to add an existing Face Record to a Face List, to be able to perform 1:N identifications

Endpoint	HTTP Method
v1/face/list/AddFaceRecord	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
AddFaceRecordToListRequest	Body: JSON object	JSON object, with properties: - **faceRecordId**: (optional)Face Record identifier (GUID). Required if faceRecordExternalIdentifier is not set - **faceRecordExternalIdentifier**: (optional) Face Record custom identifier. Required if faceRecordId is not set - **faceListId**: (optional) face list identifier (GUID). Required if faceListName is not set - **faceListName**: (optional) face list name. Required if faceListId is not set

Response:

If request is correct, you will receive a boolean response value with the result of the operation

HTTP Code	Body	Description
200	boolean	Boolean result of the operation
404		If Face Record or Face List not found
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

5. Face Recognition

5.1. Identify using an existing Face Record (1:1)

HTTPS POST method to identify a face image against a stored Face Record. You can select the Face Record by sending the NeoCheck Identifier (GUID) or the custom ExternalIdentifier.

Endpoint	HTTP Method
v1/face/identifyRecord	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifyRecordRequest	Body: JSON object	IdentifyRecordRequest JSON object, with properties: - **faceImageBase64**: face image to identify, in Base64 string format - **faceRecordIdentifier** (optional): face record identifier (GUID) of the record used to perform the face identification - **faceRecordExternalIdentifier** (optional): face record external identifier (string) of the record used to perform the face identification - **doFaceQA:** (optional) Boolean value to indicate whether to extract face QA information from the face image to identify. Default value is false **NOTE**: either the **faceRecordIdentifier** or **faceRecordExternalIdentifier** field must be set to perform the identification

Response:

If request is correct, you will receive a Success response along with a response JSON object, containing the identification result, along with the person record information, if identified.

HTTP Code	Body	Description
200	JSON `IdentifyRecordResponse` object	JSON object, with properties: - **identifier**: Person Record Identifier (GUID) - **faceImageBase64**: face image in Base64 string format - **name**: person's name - **surname1**: person's surname1 - **surname2**: person's surname2

		- **externalIdentifier**: custom identifier - **confidence**: identification confidence value (0-100) - **result**: Result of the identification, with values: -ok(0) -notExecuted(1) -warning(2) -failed(3) - **resultDescription**: detailed result description text - **faceQA:** (optional) Face QA information from image sent to identification, including face rectangle coordinates, main face feature points and image features (blur, exposure, etc). Information included only if doFaceQA set to true when calling identification
401		Unauthorized. Invalid token
400	JSON `ErrorMessage` object	Invalid content
400	JSON `ErrorMessage` object	Insufficient credits. If your credits are not enough to perform a identification
500	JSON `ErrorMessage` object	Internal Error

5.2. Identify using a reference image (1:1)

HTTPS POST method to identify a face image against a reference image. You must send the reference image in Base64 String format.

Endpoint	HTTP Method
v1/face/identifyReferenceImage	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifyReferenceImageRequest	Body: JSON object	IdentifyReferenceImageRequest JSON object, with properties: - **faceImageBase64**: face image to identify, in Base64 string format - **referenceImageBase64**: reference image for identification, in Base64 string format - **doFaceQA:** (optional) Boolean value to indicate whether to extract face QA information from the face image to identify. Default value is false

Response:

If request is correct, you will receive a Success response along with a response JSON object, containing the identification result, along with the person record information, if identified.

HTTP Code	Body	Description
200	JSON `IdentifyReferenceImageResponse` object	JSON object, with properties: - **confidence**: identification confidence value (0-100) - **result**: Result of the identification, with values: -ok(0) -notExecuted(1) -warning(2) -failed(3) - **resultDescription**: detailed result description text - **faceQA:** (optional) Face QA information from image sent to identification, including face rectangle coordinates, main face feature points and image features (blur, exposure, etc). Information included only if doFaceQA set to true when calling identification

5.3. Identify using a Face List (1:N)

HTTPS POST method to identify a face image amongst faces included in a Face List. You can select the Face List using the Id (GUID) or the Face List name.

Endpoint	HTTP Method
v1/face/list/identify	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
identifyListRequest	Body: JSON object	IdentifyListRequest JSON object, with properties: - **faceImageBase64**: face image to identify, in Base64 string format - **faceListId** (optional): face list identifier (GUID) of the face list used to perform the face identification - **faceListName** (optional): face List name (string) of the face list used to perform the face identification - **doFaceQA:** Boolean value to indicate whether to extract face QA information from the face image to identify. Default value is false

		NOTE: either the **faceListId** or **faceListName** field must be set to perform the identification

Response:

If request is correct, you will receive a Success response along with a response JSON object, containing the identification result, along with the person record information, if identified.

HTTP Code	Body	Description
200	JSON `IdentifyRecordResponse` object	JSON object, with properties: - **identifier**: Person Record Identifier (GUID) - **faceImageBase64**: face image in Base64 string format - **name**: person's name - **surname1**: person's surname1 - **surname2**: person's surname2 - **externalIdentifier**: custom identifier - **confidence**: identification confidence value (0-100) - **result**: Result of the identification, with values: -ok(0) -notExecuted(1) -warning(2) -failed(3) - **resultDescription**: detailed result description text - **faceQA:** (optional) Face QA information from image sent to identification, including face rectangle coordinates, main face feature points and image features (blur, exposure, etc). Information included only if doFaceQA set to true when calling identification
401		Unauthorized. Invalid token
400	JSON `ErrorMessage` object	Invalid content
400	JSON `ErrorMessage` object	Insufficient credits. If your credits are not enough to perform a identification
500	JSON `ErrorMessage` object	Internal Error