DOCUMENT VERIFICATION API

Developers Guide

Content description

This is a reference manual and configuration guide for the NeoCheck Document Verification API product. It shows how to interact with the JSON Web API from an external client to verify all kind of ID and travel security documents in a fast and easy way.

Madrid, November 07th 2018

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 has 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	05/02/2017	Initial Version
1.1	12/02/2017	Parameters added
1.2	01/03/2017	Callback added
2.0	12/03/2018	Updated to API V1
2.1	19/04/2018	Include images in results
2.2	07/11/2018	Create Video Identification and optional verification configuration

Index

Release Notes 2
1. NeoCheck API Endpoint 4
2. Authentication 5
2.1. Initial authentication 5
2.2. Token Refresh 6
2.3. Abusive usage countermeasures 7
3. Users 8
3.1. Retrieve User information 8
4. Verifications 10
4.1. Create a Verification 10
4.2. Verification Status 11
4.3. Verification Result 12
4.4. PDF Report 14
5. Video Identifications 15
5.1. Request a Video Identification 15

1. NeoCheck API Endpoint

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

- Automatically recognize and check ID documents from images.
- Detect ID document falsification.
- Detect specimen documents.
- Detect expired documents.
- Perform biometric verifications (Face Matching) between the holder's document photo and an existing photo.

Additionally, the API also includes a way to perform an unattended Video Identification process, by using a web plugin to capture client face portrait and ID document images, along with liveness detection checks.

The video identification plugin can be easily integrated in a web browser by using an iframe object, and is highly configured, including custom plugin design, web URL redirections and results notification via POST API callback. Images captured can be used to perform ID document check if needed.

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

2. Authentication

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

2.1. Initial authentication

HTTPS POST method to authenticate. In order to request access to NeoCheck 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 verification methods and to request verification results.

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 verification 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 verification methods and to request verification results.

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 verification 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 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. Users

3.1. Retrieve User information

HTTPS GET method to retrieve logged user information.

Endpoint	HTTP Method
v1/users/userdata	GET

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token

Response:

If request is correct, you will receive a Success response along with the Verification Status,

HTTP Code	Body	Description
200	JSON `UserData` object	UserData object with logged user information
401		Unauthorized. Invalid token

User Data is composed of:

- **Personal data**: First name, last name, email, phone number, etc.
- **Account credits:** number of credits remaining, if not unlimited credits configured
- **Verification level**: type of checks performed, and number of credits per verification
  - **L1**: MRZ checks
  - **L2**: MRZ, Visual OCR, Barcode and Visual checks
  - **L3**: MRZ, Visual OCR, Barcode, Visual and Chip authentication checks
- **Verification priority:** verification expiration, meaning the maximum time to complete the verification (in seconds), and number of credits per verification
  - **None**: 1 day
  - **Low**: 2 hours
  - **Medium**: 15 min
  - **High**: 1 min
- **Face Identification**: if the user can send live face image to be matched against document portrait
- **Notification Email**: email to receive verification results report (PDF)
- **Notification Callback Url**: if configured, a HTTP POST will be sent to the callback URL when the verification finishes, containing the verification results in the body request, formatted as a JSON object. See chapter 4.3 for details about the verification results object

Example of Successful Verification Result Response:

```
ResultCode: HTTP/1.1 200 OK
Content-Type: application/json
Body:
{
  "credits":504,
  "email":"myregisteremail@mymail.com",
  "firstName":"John",
  "lastName":"Snow",
  "phoneNumber":null,
  "userName":"Scarecrow",
  "created":"2017-03-04T20:42:02.95+00:00",
  "lastLoginDate":null,
  "unlimitedCredits":false,
  "verificationLevel":
  {
    "name":"L2",
    "credits":2
  },
  "verificationPriority":
  {
    "name":"Low",
    "duration":7200,
    "credits":2
  },
  "canUseExpert":true,
  "faceIdentification":false,
  "language":"es",
  "notificationEmail":"notifemail@mymail.com",
  "notificationCallbackUrl":null,
  "disabled":false
}
```

4. Verifications

4.1. Create a Verification

HTTPS POST method to create a new verification. You will receive a NeoCheck Verification Identifier, which you should store and use to check verification status and request verification results when they are ready.

Endpoint	HTTP Method
v1/verifications	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
createVerificationRequest	Body: JSON object	JSON object, with properties: - **frontImage**: front image as Base64 string - **backImage** (optional): back image as Base64 string - **faceImage** (optional): face image as Base64 string - **verificationConfiguration**: (optional) custom configuration, with settings: - **storeInNeoCheck**: Boolean value to configure whether Verification will be stored in NeoCheck or not - **useNeoCheckReview**: Boolean value to configure whether verification can be reviewed using NeoCheck platform or not - **notificationApiUrl**: string value to set the URL callback in which verification results will be notified

Response:

If request is correct, you will receive a Success response along with the NeoCheck Verification Identifier, which can be used to look for verification status and request verification result.

HTTP Code	Body	Description
200	VerificationId (Guid)	Verification identifier to check for status and result
401		Unauthorized. Invalid token
400	JSON `ErrorMessage` object	Invalid content. If no files or files without correct format
400	JSON `ErrorMessage` object	Insufficient credits. If your credits are not enough to create a new verification
500	JSON `ErrorMessage` object	Internal Error

4.2. Verification Status

HTTPS GET method to check a verification status

Endpoint	HTTP Method
v1/verifications/{verificationIdentifier}/status	GET

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
verificationIdentifier	query parameter (Guid)	NeoCheck Verification Identifier retrieved when verification was created

Response:

If request is correct, you will receive a Success response along with the Verification Status,

HTTP Code	Body	Description
200	Verification Status	**- WaitingIdentification**: verification not processed yet **- Error**: error processing verification **- WaitingChecking**: waiting for verification checking **- ManualIdentification**: verification under reviewer identification **- WaitingChecked**: completing verification **- Checked**: verification completed **- ManualChecking**: verification under reviewer checking **- ExpertManualIdentification**: verification under expert identification **- ExpertManualChecking**: verification under expert checking **- VideoIdentificationExternal**: custom video identification **- VerificationExternal**: custom verification
401		Unauthorized. Invalid token
400	Error Message	Transaction not found

4.3. Verification Result

HTTPS GET method to retrieve the verification results.

Endpoint	HTTP Method
v1/verifications/{verificationIdentifier}/results	GET

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
verificationIdentifier	query parameter (Guid)	NeoCheck Verification Identifier retrieved when verification was created
includeImages	query parameter (boolean)	Optional parameter to include document images in results as Base64 string format. Default value is false.

Response:

If request is correct, you will receive a Success response along with the Verification Status,

HTTP Code	Body	Description
200	JSON `VerificationResults` object	VerificationResults object with document data and verification checks performed
401		Unauthorized. Invalid token
400	Error Message	Result not ready. Please check that the verification is in Checked status

Verification Results are composed of:

- **Document data**:
  - **Holder's personal information**: name, surname, date of birth, nationality, etc.
  - Document type, country of issue, date of expiry, date of issue, etc.
- **Document verifications:** all checks performed, including:
  - Name: verification check name
  - Source: the verification check source: MRZ,OCR,BARCODE, CHIP
  - Category: the verification check category: Identification, Data Format, Data Integrity, Security Features, Biometrics, etc
  - Result: the verification check result: Ok, Failed, Warning, NotExecuted, Overriden
- **Verification summary**: total verifications, global confidence, document name, etc.
- **Document images**: if request parameter IncludeImages is set to true, the document images are included in Base64 string format

Optionally, you can configure a callback URL during the integration process. If so, a HTTP POST will be sent to the configured callback URL when the verification finishes, containing the verification results in the body request, formatted as a JSON object.

Example of Successful Verification Result Response:

```
ResultCode: HTTP/1.1 200 OK
Content-Type: application/json
Body:
{
  "documentData": [
    { "key": "IssuingState",
      "name": "Issuing State",
      "value": "USA" },
    { "key": "Surname",
      "name": "Surname",
      "value": "SMITH" },
     . . .
  ],
  "documentVerifications": [
    { "name": "DocumentExpired",
      "category": "DataFormat",
      "result": "OK" },
    { "name": "PatternImage",
      "category": "SecurityFeatures",
      "result": "OK" },
    { "name": "FaceMatching",
      "category": "Biometrics",
      "result": "OK" },
     . . .
  ],
  "totalChecks": 31,
  "successChecks": 31,
  "warningChecks": 0,
  "failedChecks": 0,
  "expertUsed": false,
  "identificationTime": "00:00:04.1647703",
  "checkTime": "00:00:01.4440000",
  "expertComments": null,
  "sourceType": "Camera",
  "status": "Checked",
  "userName": "userName",
  "spentCredits": 6,
  "startDate": "2017-03-09T18:50:50.4454074+01:00",
  "endDate": "2017-03-09T18:50:55.1864382+01:00",
  "startDateUtc": "2017-03-09T17:50:50.4454074",
  "endDateUtc": "2017-03-09T17:50:55.1864382",
  "securityLevel": "Medium",
  "securityLevelDescription": "[securityLevelDescription]",
  "globalResult": "OK",
  "globalResultDescription": "[globalResultDescription]",
  "language": "en"
}
```

4.4. PDF Report

In some cases, user may be interested to get the analysis results as a PDF file. It provides a very convenient way to store the analysis results. To do so, call the HTTPS GET method to retrieve the PDF report.

Endpoint	HTTP Method
v1/verifications/{verificationIdentifier}/report	GET

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
verificationIdentifier	query parameter	NeoCheck Verification Identifier retrieved when verification was created

Response:

If request is correct, you will receive a Success response along with the Verification Status,

HTTP Code	Body	Description
200	FileStreamResult	Byte array with the PDF verification report
401		Unauthorized. Invalid token
400	Error Message	Transaction not found or report not available yet

Optionally, you can configure a notification email address during the integration process. If so, an email with the PDF report attached is sent when the verification finishes.

5. Video Identifications

5.1. Request a Video Identification

HTTPS POST method to request a Video Identification. You will receive a Video Identification plugin URL, which can be integrated in any website by using an iframe. You can optionally set configuration for the plugin design customization and to configure results notification and storage options.

Endpoint	HTTP Method
v1/ VideoIdentification/requestVideoIdentification	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
videoIdentification Configuration	Body: JSON object	JSON object, with properties: **VideoIdentificationWebConfiguration**: (optional) Configuration for Video Identification plugin customization. Settings: + **RedirectUrlOk**: Redirect URL when video identification process was successfully sent for revision. + **RedirectUrlKo**: Redirect URL when video identification process failed to be sent for revision + **CompanyName**: Company name that will appear on video Identification plugin. Default value is "NeoCheck". + **BackgroundColor**: Plugin background color, in HEX value (e.g. #58585A). + **MainColor**: Plugin text color for title elements, in HEX value (e.g. #58585A). + **SecondaryColor**: Plugin text color for secondary text elements, in HEX value (e.g. #58585A). **VideoIdentificationApiConfiguration**: Configuration options for Video Identification workflow and results notification. Settings: + **IncludeBinaries**: Boolean value to configure whether video identification results will be notified including captured images and video. + **UseNeoCheckReview**: Boolean value to configure whether video identification will be reviewed using NeoCheck platform or not. + **NotificationApiUrl**: string value to set the URL callback in which video identification results will be notified.

Response:

If request is correct, you will receive a Success response along with the NeoCheck Verification Identifier, which can be used to look for verification status and request verification result.

HTTP Code	Body	Description
200	VideoIdentificationURL (string)	Video Identification plugin URL
401		Unauthorized. Invalid token
400	JSON ErrorMessage object	Invalid content.
400	JSON ErrorMessage object	Insufficient credits. If your credits are not enough to create a new video identification
500	JSON ErrorMessage object	Internal Error

Once Video Identification is finished, results are notified via POST URL Callback to the address configured as **NotificationApiUrl**. The results object includes:

- **Captured binaries:** If IncludeBinaries is configured, images and video captured are notified as Base64 string
- **Process information:** Information about each captured image timestamp in UTC, and liveness detection checks
- **VideoIdentificationResult:** the process result after review, only if video identification is configured as UseNeoCheckReview = true. Values are: NotSet(0), Accepted(1), Rejected(2)
- **Reviewer comments:** Reviewer comments, only if video identification is configured as UseNeoCheckReview = true
- **DocumentType:** Document type used during video identification. Values are: NotSet(0), Passport(1), IdCard(2)

Example of Video Identification Callback results notification:

```
Content-Type: application/json
Body:
{
  "TransactionIdentifier": "a36dc639-595c-431b-a7a9-8a9dc8702e3c",
  "FrontImage": "IMAGE_BASE64",
  "BackImage": "IMAGE_BASE64",
  "FaceImage": "IMAGE_BASE64",
  "Video": "VIDEO_BASE64",
  "StartTimeUtc": "2018-10-28T15:00:28.354Z",
  "EndTimeUtc": "2018-10-28T15:01:17.136Z",
  "FrontCaptureTimeUtc": "2018-10-28T15:01:03.881Z",
  "BackCaptureTimeUtc": "2018-10-28T15:01:07.561Z",
  "FaceCaptureTimeUtc": "2018-10-28T15:01:11.129Z",
  "QuoteText": "Well begun is half done",
  "GestureText": "Turn your head to your left for 2 seconds",
  "VideoIdentificationResult": 0,
  "ReviewerComments": null,
  "DocumentType": 1,
}
```