CONTROL LIST CHECK API

Developers Guide

Content description

This is a reference manual and configuration guide for the NeoCheck Control List Check API product. It shows how to interact with the JSON Web API from an external client to check personal data against black and white lists.

Madrid, April 1st 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 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	01/04/2019	Initial Version

Index

1. NeoCheck API Endpoint

NeoCheck provides an API for Control List checks. This API allows users to check personal data against white lists, such as politicians, and/or black lists, such as financial and crime lists.

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

2. Authentication

The ControlListCheck 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 ControlListCheck 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 Control List Check 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 Control List Check 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 Control List Check 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. Control List Check

3.1. Check

HTTPS POST method to check personal data against Control Lists configured for logged user.

Endpoint	HTTP Method
`/v1/ControlList/check`	POST

Request Parameters:

Name	Type	Description
`authorization`	query parameter	`Bearer access_token` authorization header with a valid access token
`controlListFilter`	Body: JSON object	JSON object, with properties: - **name**: person's first name - **surname**: person's last name/s - **controlListType**: (optional) to check only in white lists, black lists, or both. Default value is Both - **exactMatch**: (optional) to perform the check and return results that matches exactly or approximately the words entered in the filter (equals vs contains). Default value is false

Response:

If request is correct, you will receive a Success response along with the Control List matches, or a bad request if no results were found

HTTP Code	Body	Description
`200`	JSON `list<ControlListHit>` object	List of ControlListHit object, containing: **- controlListType**: if hit is from a white list or a black list **- controlListName:** name of the control list **- documentNumber:** the person's passport or idcard number **- personalNumber:** the person's personal number **- issuingCountry:** the person's document issuing country **- dateOfBirth:** the person's date of birth **- name:** name of the person **- surname:** surname/s of the person **- gender:** gender of the person **- comments:** some non-structured information related to the person **- alias:** list of aliases known for the person **- pictureUrl:** a link to the person's face

		- descriptionUrl: a link to the person's online information
400		No results were found for the filter provided
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error

3.2. Multiple Check

HTTPS POST method to check multiple personal data against Control Lists configured for logged user.

Endpoint	HTTP Method
/v1/ControlList/multicheck	POST

Request Parameters:

Name	Type	Description
authorization	query parameter	`Bearer access_token` authorization header with a valid access token
controlListFilters	Body: JSON object	JSON object, composed of a list of person's filters, each one with properties: - **name**: person's first name - **surname**: person's last name/s - **controlListType**: to check only in white lists, black lists, or both. Default value is Both - **exactMatch**: (optional) to perform the check and return results that matches exactly or approximately the words entered in the filter (equals vs contains). Default value is false

Response:

If request is correct, you will receive a Success response along with a list of Control List matches, or a bad request if no results were found

HTTP Code	Body	Description
200	JSON `list<ControlListResult>` object	List of ControlListResult objects, containing: **- controlListFilter**: the original filter sent **- controlListHits:** a list of controlListHit objects, with the same structure as mentioned in chapter 3.1
400		No results were found for the filter provided
401		Unauthorized. Invalid token
500	JSON `ErrorMessage` object	Internal Error