API Developer Guide

GOVERNID API DOCUMENT

Technical Documentation

Prepared By : GovernID Development Team
Created : 12.06.2018
Updated : 25.02.202
Version : 2.2

If any information in this document is used, the terms specified in the "Terms of Use" section of the document are deemed to have been accepted. If these conditions are not accepted, the document should not be used.

Content

Abstract

GovernID provides personal data management tools to both institutions and end users as stipulated by the Personal Data Protection Law No.6698 (KVKK) or Europe GDPR Regulations:

For institutions:

- Ensuring that Personal Data is received in accordance with legal regulations.
- Providing consent management while receiving Personal Data.
- Tracking the flow of Personal Data in the organization.
- Tracking Personal Data retentions.
- Tracking and processing withdrawn consents.

For individuals:

- Keep track of data given by individual. (Which personal data is given to which institution?)
- Monitor whether expired data has been deleted.
- To be able to withdraw a consent (Consent Management).
- To monitor whether the consents withdrawn are processed or not.

GovernID has put into service an API service that works with the **HTTP1.1** protocol in order for institutions to fulfil their legal responsibilities stated above. **API** service uses **JSON** data type for mutual data transfer.

With the API service, organizations can perform the following operations:

1. Sending data to the API and receiving the transaction code
2. Listing personal data coming to the Data Entry Point (open and masked, based on access level)
3. Transaction codes of personal data arriving at the Data Entry Point and expired
   a. Listing (with internal / external flows, if any)
   b. Sending deletion / anonymization confirmation to the server
4. If Consent Management connected to the Institution Data Flow Lines is used, the flow codes whose consent is withdrawn
   a. Listing
   b. Confirmation notification to the server

API Activation

API Activation is a default built in service of GovernID for companies that has valid “GovernID Enterprise Licence” or “GovernID Enterprise Plus Licence”

In order to use the API service, this service must be activated by the authority of the institution. The activation process is free.

The authority of the institution opens the API Access page from the "Institution Settings" menu in the application, clicks the (+) button at the bottom right of the page that opens, and creates the API access key. During creation, it sets the active / passive state of personal data access. If this is activated, the data will be listed as explicit, if it is inactive the data will be listed as **** masked. Also, just below it selects which forms the generated key will be authorized to operate on. An API key can operate in multiple forms. The last section defines the IP addresses of the servers that are requested to access the API. If it is inside your institution, it is recommended that your institution have a fixed IP address accessing the internet, and this IP address should be defined for API access. In this way, a 2nd security layer is created in addition to the access key to be used in API access.

In the IP address definition, an IPv4 address must be entered in the form X.X.X.X. If there is more than one exit IP address in the same network, they can be defined one after the other, or CIDR notation expressing the relevant subnet can be used. For example, if you want to give API access to the entire 1.2.3.1 - 1.2.3.253 IP block of your organization, you can define it as 1.2.3.0/24.

Although not recommended, if you do not have a fixed IP address, you can enable access from any IP address by entering a 0.0.0.0/0 entry in the IP restriction field.

API access will not be made except for the specified IP addresses.

After the IP addresses are defined, the system automatically generates an API access key and identifies it to your organization.

The institution can create as many API keys as it wishes and set them to be accessed with different authorizations and from different IP addresses.

The institution will be able to perform the above-mentioned operations programmatically by using the API key provided.

Note: API keys are in UUID v4 format.

PROCESSES AND TRANSACTIONS

The type of requests to be sent to the API (REQUEST_TYPE) must be **POST**.

Requests must be specified as **Content-Type: application / json** in the header package.

For authentication; Again, the **API key** must be sent with the **Governid-Apikey** parameter in the header package.

In all API requests, a response will be returned as {success: true,…} or if you sent an incorrect request {success: false, reason: "error reason description"}.

Sample API Request (PHP):

```
<?php
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,"https://api.governid.com/api_endpoint_url");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
         'Content-Type: application/json',
         'Governid-Apikey: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'
));
...
...
```

Note: The curl_init and curl_setopt functions shown in the example above are functions of the PHP_CURL extension. The PHP_CURL package must be installed to use them.

*For **Linux distributions using APT**: apt-get install php 8.0-curl (according to your php version, write the appropriate version code instead of 8.0)*
***For FreeBSD 10+:** pkg add php80-curl (according to the php version, write the appropriate version code instead of 8.0)*
***FreeBSD <10 için**: cd / usr / ports / lang / phpXX-extensions / then make config & make install clean (XX: version)*

Sample API Request (JavaScript/NodeJS):

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/api_endpoint_url');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
...
...
```

Some basic search / filtering / paging / sorting operations can also be performed in the listing results. These should be POST as the body package, not in the headerDetailed usage of these will be explained in the following sections. If no criteria are specified, the default result set will be listed in reverse order according to the posting date, that is, the **last 100 records**.

Data Package (JSON)

Data packets to be sent to the PI must be JSON encoded and encoded as strings. Commonly used programming languages (ASP.NET, PHP, Python, JavaScript, Java, VB, C # etc.), includes the functions of unpacking / packaging (encode / decode) JSON data type.

Note: You can check the validity of the JSON data types you send at https://jsonlint.com/

Sample API Request (PHP):

```
<?php
$data = array(
        "param1"=>"value1",
        "param2"=>"value2"
);
$data_encoded = json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,"https://api.governid.com/api_endpoint_url");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_HTTPHEADER, array(
        'Content-Type: application/json',
        'Governid-Apikey: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'
));
curl_setopt($ch, CURLOPT_POSTFIELDS, $data_encoded);
curl_exec($ch);
...
```

Sample API Request (Javascript/NodeJS):

```
var data = {
        param1: 'value1',
        param2: 'value2'
};
Var data_encoded = JSON.stringify(data);
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/api_endpoint_url');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send(data_encoded);
...
```

JSON encode / decode functions in different programming languages:

.NET: https://docs.microsoft.com/en-us/dotnet/api/system.web.helpers.json.encode?view=aspnet-webpages-3.2

PYTHON: https://pynative.com/python-json-encode-unicode-and-non-ascii-characters-as-is/

PERL: https://metacpan.org/pod/JSON

JAVA: https://www.javatpoint.com/java-json-example

Character encoding

The character encoding of all data sent to and received from the API is in UTF-8 format.

Time Zone

The time zone of all data sent to and received from the API is set to UTC. In the future, you will be able to choose a predefined time zone in your API access settings.

Time Display Format

Since HTTP protocol works with STRING type data in server-client data transfer, time values should also be sent in STRING format. The JSON communication standard in our API infrastructure automatically shows timestamp values in ISO8601 format. The advantage is that all commonly used software languages recognize this format, the string form can be sorted even without format conversion. See, https://en.wikipedia.org/wiki/ISO_8601

Sample display: 2018-06-12T16:52:35Z

The software for your own time zone Europe / Istanbul, display type in German (DD.MM.YYYY) If you set, for which Turkey is UTC + 3, the compiler will show in the history of this process 06.12.2018 19:52. If you set the time zone as America / New York in your software, it will be automatically displayed on the client side as 06/12/2018 11:52 in winter and 06/12/2018 12:52 in summer. (New York winter time UTC-5 summer time UTC-4)

Phone Numbers

In case personal data are listed openly, the phone numbers will be displayed in E.164 format and as + CCXXXXXX… without spaces. (CC country phone code) See https://www.itu.int/rec/T-REC-E.164-201011-I/en

Personal Data Fields (user_data in API results)

These fields indicate a set of personal data types that are fixed in the application. New user_data types are constantly being added.

_FULLNAME: Name Surname

_EMAIL: E-mail

_EMAIL_VERIFIED: true/false BOOLEAN value

_TEL: Phone number (display format +CCXXXXXXX, CC: Country Code)

_TEL_VERIFIED: true/false BOOLEAN

_TCKN: TC National ID Number

_PHOTO: base64 encoded string (you can directly use as seen here, …)

_COMPANY_TITLE: Company Title (if corporate profile is selected while receiving data)

_WORK_TITLE: Job Title (if corporate profile is selected when receiving data)

For Visitor Entrance Gates, the API also lists the following fields in the personal data field

_VISITING_TO: Person visited

_VISITING_REASON: Reason of Visit

1. Sending data to the API and receiving the transaction code

API Endpoint: https://api.governid.com/v2/submit/{form_uuid}

Sample API Request (JavaScript/NodeJS):

```
var postData = {
        _FULLNAME: 'Ahmet YILMAZ',
        _EMAIL: 'xxxxxxxxx@xxxxxx.xxx',
        _TEL: '+90XXXXXXXXXX',
        ...
        ...
}
var postDataEncoded = JSON.stringify(postData);
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/submit/{form_uuid}');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send(postDataEncoded);
...
...
```

Sample API response:

```
'{"success":true,"transids":["ab12xy34","cd56tz78"]}'
```

In the server response, a single transaction code is normally required, but in some cases, it may be that the e-mail address belongs to another user and the phone belongs to another user. In this case, KVKK transaction record is made for two different persons on the GovernID side. Expected to be here; The other end user who has not done the transaction is using the application efficiently and objecting on the grounds that he did not do the operation himself. If there is a request in this way, you should delete the records related to the transaction code of that person from your database.

2. Sending the transaction confirmation of the form which is filled by using QR Code

API Endpoint: https://api.governid.com/v2/verify/{transid}

In forms filled with QR Code, a secret governid_transid name is automatically added to the form on your page by the system and a transaction code (transid) value is assigned to it. When the form is submitted (submitted), you need to receive this valuable information in your backend code and send the information that the process is completed to the GovernID API server.

Sample API Request (JavaScript/NodeJS):

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/verify/{transid}');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send(postDataEncoded);
...
...
```

Sample API response:

```
'{"success":true}'
```

3. Listing personal data received by the Data Entry Point

API Endpoint (all authorized forms): https://api.governid.com/v2/entries
API Endpoint (for single form): https://api.governid.com/v2/entries/{form_uuid}
Sample API Request (JavaScript/NodeJS):

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/entries');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send();
```

As a result of this process, the last 100 records will return as JSON stringified object.

In the incoming data:

totalPages: Total number of data pages, if no page number is specified: as total number of data / 100
rows: Data is in array format, array elements (rows) are in object format.
- row.form_uuid: Descriptive code of the form from which data is taken (uuid v4)
- row.indate: Shows the transaction date in the UTC time zone in ISO8601 format
- row.transid: Contains the transaction code that you will associate in the database
- row.userdata: Lists the **data** collected at the data entry point in Object format

Not: Since different parameters can be specified at the data entry point as user_data, these fields are also shown in an object. For example, in entry A, the name, surname and phone number, in entry B, the name, surname, e-mail and phone may be taken..
Sample API response:


```
'{"success":true,"totalPages":24,"rows":[{"form_uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
"transid":"ab12xy34","indate":"2018-06-12T16:52:35Z","user_data":{"_FULLNAME":"Ahmet
YILMAZ","_EMAIL":"xxxxxxxxx@xxxxxx.xxx","_EMAIL_VERIFIED":true,"_TEL":"+90XXXXXXXXXX","_TEL_VERIFIED":false}},{"
form_uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "transid":"cd56tz78","indate":"2018-06-
13T17:52:35Z","user_data":{"_FULLNAME":"Mehmet
YILMAZ","_EMAIL":"xxxxxxxxx@xxxxxx.xxx","_EMAIL_VERIFIED":false,"_TEL":"+90XXXXXXXXXX","_TEL_VERIFIED":true,"TCK
N":"12345678901"}}]}'
```


Incoming data can be transformed into object structure with functions such as json decode, json parse. For example, we can try it in the Chrome Developer Tools console, if a data like the following is returned through the API.:

Filtering / Paging operations

These parameters should be assigned to an object and POST to the API as JSON encoded string. All parameters are optional and can be used individually.

Paging

Parameter: **paging**
Value: Any INTEGER value between 5-500.

Its default value is set to 100. It determines how many lines will be drawn on each page during data listing. For example, records between 1-100 are listed on page 1, records between 101-200 are listed on page 2. If you set the pagination as 50, records between 1-50 will be listed on page 1, records between 51-100 will be listed on page 2.

Page No

Parameter: **page**
Value: Up to the total number of pages 1 or greater INTEGER, default 1

Its default value is set to 1. Use the total number of pages in the value of totalPages in the result set that comes after the first request is made.

For total page only send request to API Endpoint: https://api.governid.com/v2/entries/total or https://api.governid.com/v2/entries/total/{form_uuid}. will be replied as : {totalPages: 24}.

Sorting

Parameter: **sortby**
Value: indate|transid|(Possible values in user_data like _FULLNAME, _EMAIL), DEFAULT: indate

Sorting Direction

Parameter: **sorttype**
Value: ASC|DESC, DEFAULT: DESC

Query

Parameter: **query**
Value: text of at least 2 characters long is searched in all data fields

Filtering by Date

Parameters : **date_after, date_before**

Value : should be valued as YYYY-MM-DD. Will be queried as >= ve <=. Can be used individually or both.

Sample API Request (JavaScript/NodeJS):

```
var postData = {
          paging: 50,
          page: 2,
          sortby: '_FULLNAME',
          sorttype: 'ASC',
          query: '%test%',
          date_after: '2019-06-01'
}
var postDataEncoded = JSON.stringify(postData);
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/entries');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send(postDataEncoded);
...
```

4. Transaction codes of personal data arriving at the Data Entry Point and expiring

a. Listing (with data flows)

API Endpoint: https://api.governid.com/v2/expired or

API Endpoint: https://api.governid.com/v2/expired/{form_uuid}

Sample API Request (Javascript/NodeJS):

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/expired');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send();
...
...
```

As a result of this process, the last 100 records **waiting for you to confirm** will return as JSON stringified objects.

Filtering

1. All filtering / paging parameters can be used except the query specified in the step.

In the incoming data:

totalPages: The total number of pages of data, if the page number is not specified, the total number of data / 100
rows: Data are in array format, array elements (rows) are in object format.
row.indate: Displays the transaction date in the UTC time zone in ISO8601 format
row.transid: Contains the transaction code that you will associate in the database
row.userdata: Lists the **types of data** collected at the data entry point in Object format.

pipes: (optional) If you have defined your Data Flow Lines, the data's internal / external flow paths, flow lines are listed as arrays in order of flow.

Sample API response:


```
' {"success":true,"totalPages":11,"rows":[{"transid":"ab12xy34","indate":"2018-06-
12T16:52:35Z","user_data":["_FULLNAME","_EMAIL","_PHONE"],"pipes":["pipe1","pipe3","pipe5"]},{"transid":"cd56tz78","indate":"201
8-06-13T17:52:35Z","user_data":["_FULLNAME","_EMAIL","_PHONE","_TCKN"],"pipes":["pipe1","pipe2","pipe6"]}]}'
```


The information on which personal data fields have been extracted depending on the transaction codes in the incoming data set is displayed in user_data.

example:

It is seen that the name, surname, e-mail and phone are received from the user during the transaction with ab12xy34 code. As a legal obligation, you are required to delete or anonymize the name, surname, e-mail, phone numbers associated with this code from all of your (printed or digital) data recording fields.

Anonymization; It is making the data unrelated to an identified or identifiable natural person even if they are matched with other data. ... The data controller is obliged to take all necessary technical and administrative measures regarding the anonymization of personal data.

Depending on the other code named cd56tz78, it is seen that extra TCKN data is also taken. In this case, you need to delete or anonymize the data that includes the Turkish ID number in the records associated with this code.

b. Sending deletion / anonymization confirmation to the server

API Endpoint: https://api.governid.com/v2/expired_feedback

Confirmation submissions can be made individually or for batch codes. You must post the array transaction codes to the API server as JSON stringified.

Sample API Request (JavaScript/NodeJS):

```
var postData = [
         'ab12xy34',
         'cd56tz78'
];
var postDataEncoded = JSON.stringify(postData);
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/expired_feedback);
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send(postDataEncoded);
...
...
```

As a result of this process, it is reported that the transaction codes in the posted array are deleted or anonymized.

Whether the transaction is successful or not, a result as success: true is returned by the server

Sample API response:

```
' {"success":true}'
```

5. If Consent Management connected to Institution Data Flow Lines is used, the flow codes whose consent is withdrawn

a. Listing

API Endpoint: https://api.governid.com/v2/revoked

Note: Data Flow Lines (PIPE) and Data Crossing / Registration Points (NODE) must be defined in order to list transaction codes whose consent has been withdrawn.

Sample API Request (JavaScript/NodeJS):

```
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/revoked');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send();
...
...
```

As a result of this process, the last 100 records **waiting for you to confirm** will return as JSON stringified objects.

In the incoming data:

totalPages: The total number of pages of data, if the page number is not specified, the total number of data / 100
rows: Data are in array format, array elements (rows) are in object format.
row.indate: Displays the transaction date in the UTC time zone in ISO8601 format
row.transid: Contains the transaction code that you will associate in the database
row.userdata: Lists the **types of data** collected at the data entry point in Object format.

pipes: (optional) If you have defined the inflow / outflow paths of the data are listed in the sequential array in order of flow.

Sample API response:

```
' {"success":true,"totalPages":11,"rows":[{"transid":"ab12xy34","indate":"2018-06-
12T16:52:35Z","revoked_from":["pipe1"]},{"transid":"cd56tz78","indate":"2018-06-13T17:52:35Z"," revoked_from":["pipe3"]}]}'
```

The information on which consent was withdrawn depending on the transaction codes in the incoming data set is displayed in revoked_from. The pipe name specified here is the code given by you of the data stream line that is subject to the consent process you define.

For example:

It is seen that during the transaction coded ab12xy34, a consent defined as pipe1 was received from the user and this was wanted to be withdrawn by the user. As a **legal obligation**, the transfer of personal data from all your (printed or digital) data recording areas during this pipe must be stopped.

b. Confirmation notification to the server

API Endpoint: https://api.governid.com/v2/revoked_feedback

Confirmation submissions can be made individually or for batch codes. You must post the array transaction codes to the API server as JSON stringified.

Sample API Request (JavaScript/NodeJS):

```
var postData = [
        {
                transid: 'ab12xy34',
                revoked_from: 'pipe1'
        },
        {
                transid: 'cd56tz78',
                revoked_from: 'pipe3'
        }
];
var postDataEncoded = JSON.stringify(postData);
var xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.governid.com/v2/revoked_feedback');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.setRequestHeader('Governid-Apikey', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX');
xhr.send(postDataEncoded);
...
...
```

As a result of this process, the withdrawal of consents based on transaction codes in the posted array is notified to the server that has been approved.

Whether the transaction is successful or not, a result as success: true is returned by the server

Sample API response:

```
' {"success":true}'
```

TERMS OF USE

This document is a document created for institutions receiving service from GovernID ™. Although the information in the document is in use as of the date of publication, there may be changes as a result of continuous development work. For this reason, we recommend that you contact GovernID™ before you start working with our document.

A. Access and use of the document:

1. A. Access to and use of the document is subject to the following terms and conditions set by GovernID™.
2. All information contained in this document cannot be changed, reproduced, published, distributed, publicized, or translated into another language without the permission of GovernID ™.
3. The "Terms of Use" of the person using the document are deemed to have been accepted. GovernID™ reserves the right to change the "Terms of Use" at any time, by posting changes online. It is the responsibility of the document user to regularly follow the "Terms of Use" in order to be informed about the changes published online. The person who continues to use the document after the changes is deemed to have accepted the changes in the "Terms of Use".
4. GovernID™ works to be accessible 24 hours a day. However, it is not responsible for the inaccessibility of the Site for various reasons.
5. Although GovernID™ makes every effort to ensure that the information it provides to document users is correct, it is not responsible for incorrect or incomplete information if GovernID™ is not contacted about the up-to-dateness of the document.

B. Downloading and using written and / or visual information from this document:

1. Personal use;

1.1. The information in the document is limited to personal use and / or informational purposes. However, the acts of reproduction carried out within this scope cannot harm the legitimate interests of the right owner without a justified reason or cannot be against the normal use of the work and cannot be used for commercial purposes.

1.2. Written or visual material in the document cannot be changed in any way, and cannot be used by deleting the copyright statements.

1.3. The whole or part of the information in the document cannot be used by changing or in any other way in another document / website without permission.

1.4. The data in this document can be downloaded or printed for non-commercial, informational and personal use.

1.5. This document can be sent to third parties without any commercial purpose for their personal information, provided that the content is provided by GovernID™ and that these terms and conditions apply to them and that they must comply with the.

2. Non-personal use;

Reproduction of this document, such as written and / or visual information download and printing, or use on websites:

Permissions for the information to be used must be requested in writing from GovernID ™.

C. Disclaimer

GovernID™ does not guarantee that the document at hand is up to date. The statements in this document are not considered as commitments and are not binding. In the event that information is given about the work to be done by contacting GovernID and the update of the document is not approved, it does not give any warranty, either implicitly, explicitly or legally, but not limited to these.