

ESS Validate API - Developer Guide

Authentication.....	2
Authentication method and flow.....	2
OAuth2 parameters and Client App.....	2
How to generate the authorization token.....	2
Example in Curl.....	2
Example using Postman.....	3
ESS Validate API.....	4
Description.....	4
Calling the API.....	4
Via Swagger UI.....	4
url.....	4
Testing using Swagger UI.....	4
Authentication on Swagger.....	4
Swagger UI Endpoints.....	6
Single Validation.....	6
Multi-validation.....	8
Programmatic Access.....	10
OpenAPI definition.....	10
Setting up you preferred Report Language.....	10
Validation rules set format.....	10
File format in the request.....	10
Example of Requests using Postman.....	11
You can import the following Collection and check few examples of Postman API calls. You can use it to test and validate what are the parameters and how they should be passed before doing it from your program.....	11
Import the library.....	11
API call parameters in Postman.....	11
Validation Report format.....	16
Support/Help.....	18

Authentication

Authentication method and flow

- The ESS Validate API uses OAuth2 to secure the endpoints
- To access any endpoint, you will need to use a JWT access token
- This authentication token will be created using the OAuth2 protocol, following the Client Credentials Flow (see [example here](#) - *note*: this link is just to understand the flow logic you are not required to implement any AWS setup from this link)

OAuth2 parameters and Client App

- We are using Cognito as our Authorization server (the /token endpoint)
- We have already created an OAuth2 Client Application for you ready to use.
- You can use those credentials to call the Authorization Server and get the token

ClientID	7d30bi87iptegbrf2bp37p42gg
ClientSecret	880tema3rvh3h63j4nquvgoh0lgts11n09bq8597fgrkvvd62su
scope	eat/read
Grant type	client_credentials
Token endpoint uri	https://dev-eat.auth.eu-central-1.amazoncognito.com/oauth2/token

How to generate the authorization token

You can use your favorite language or tool to generate the token by calling the Token endpoint.

Example in Curl

Run the command then copy the value of the access-token in the response body:

```
curl --request POST \  
  --url 'https://dev-eat.auth.eu-central-1.amazoncognito.com/oauth2/token' \  
  --header 'content-type: application/x-www-form-urlencoded' \  
  --data grant_type=client_credentials \  
  --data client_id=7d30bi87iptegbrf2bp37p42gg \  
  --data client_secret=880tema3rvh3h63j4nquvgoh0lgts11n09bq8597fgrkvvd62su \  
  --data scope=eat/read
```

You can use Postman with the following parameters, then then copy the value of the access-token in the response body:

Once you have the token, you can use it to call any of the API endpoints either using [Swagger UI](#), or [Programmatically](#).

Note: when using the API programmatically, you will use your favorite language or framework to generate the token automatically within your program.

ESS Validate API

Description

- Validation of invoice XML documents against a set of business rules
- Takes invoice XML file as input
- Generates a report in json format as output (see [Report format](#)).

Calling the API

Via Swagger UI

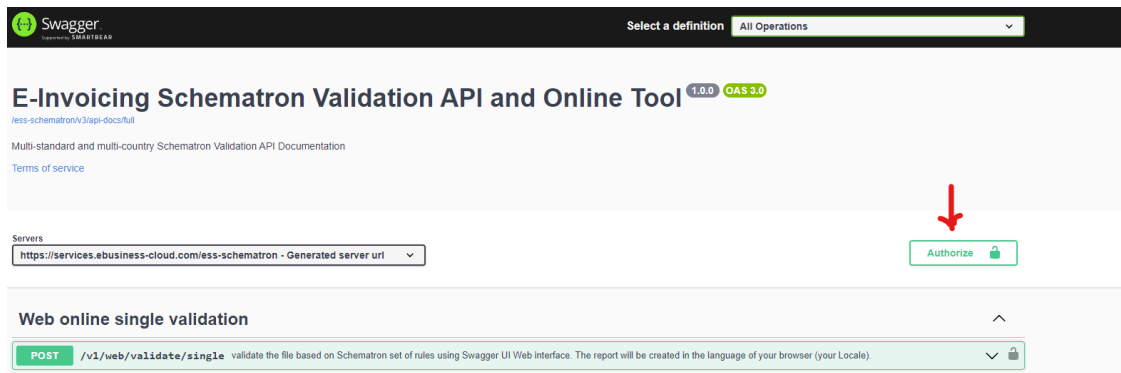
url

<https://services.ebusiness-cloud.com/ess-schematron/swagger-ui/index.html>

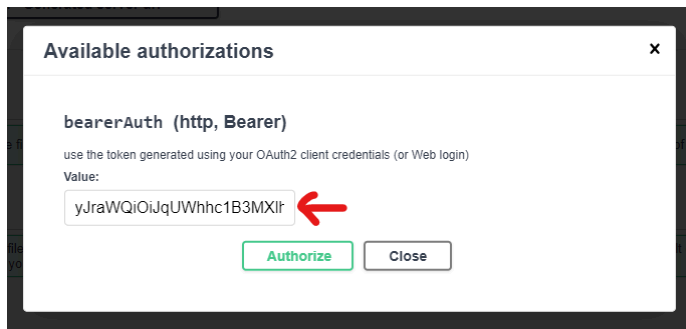
Testing using Swagger UI

Authentication on Swagger

- You have to generate your token separately (example: using Postman), once generated copy the value
- Go to swagger UI screen and click on “Authorize” button

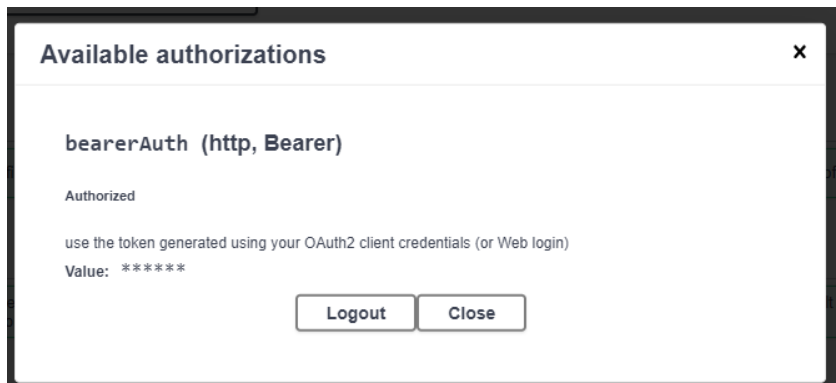


- Paste your token value, and click “Authorize”



The dialog box is titled "Available authorizations" with a close button (x) in the top right corner. It contains the text "bearerAuth (http, Bearer)" and a subtext "use the token generated using your OAuth2 client credentials (or Web login)". Below this, it says "Value:" followed by a text input field containing the token "yJraWQiOiJqUWwhc1B3MXlr". A red arrow points to the input field. At the bottom, there are two buttons: "Authorize" (highlighted in green) and "Close".

- The token will be saved and you can start using the endpoints



The dialog box is titled "Available authorizations" with a close button (x) in the top right corner. It contains the text "bearerAuth (http, Bearer)" and a subtext "use the token generated using your OAuth2 client credentials (or Web login)". Below this, it says "Authorized" and "Value: *****". At the bottom, there are two buttons: "Logout" and "Close".

- If when using the endpoints you get a 401 response code. This means your token is not valid or has expired. You have to repeat the steps above

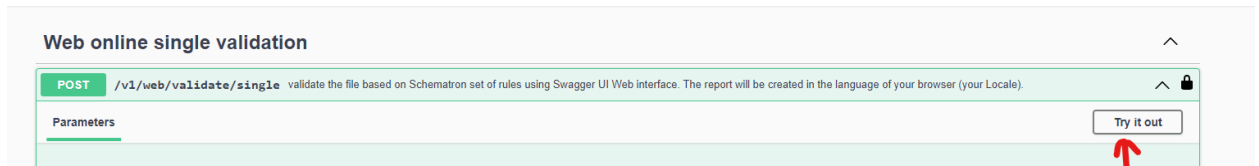
Swagger UI Endpoints

API endpoints can be tested online directly using Swagger UI.

Single Validation

To validate XML file against 1 single set of business Rules

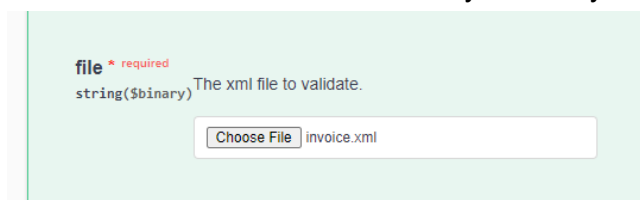
- Description: [Web online single validation](#)
- Click on the “Try it out” button to test online. This will enable the fields.



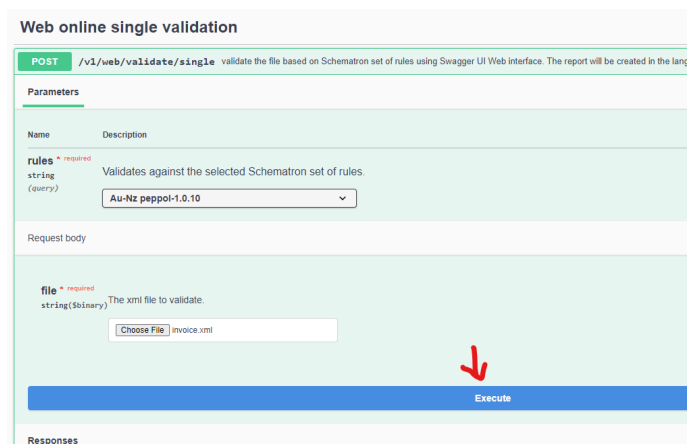
- Select the set of Rules you want to apply. Example “AU-Nz peppol-1.0.10”



- Select the XML file to validate from your file system



- Click on “execute” to run the Validation



- The response code should be 200, and Response body will contain the Validation report (see [report format below](#) for full details)

The screenshot displays a web application for online validation. At the top, there are 'Execute' and 'Clear' buttons. Below this, the 'Responses' section is active, showing the following details:

- Curl:** A terminal window showing a POST request to `https://services.ebusiness-cloud.com/ess-schematron/v1/web/validate/single?rules=Au-NrX20ub1-1.0.10` with headers for 'accept', 'authorization', and 'content-type', and a file named 'invoice.xml'.
- Request URL:** `https://services.ebusiness-cloud.com/ess-schematron/v1/web/validate/single?rules=Au-NrX20ub1-1.0.10`
- Server response:** A table with two columns: 'Code' and 'Details'. The 'Code' column shows '200', and the 'Details' column shows the 'Response body'.

The 'Response body' is a JSON object indicating a successful validation. A red bracket on the left side of the interface highlights the 'Server response' section. A 'Download' button is located at the bottom right of the response body area.

```
{
  "customer": "Online validation tool",
  "successful": true,
  "message": "Schematron validation on file 'invoice.xml' completed with status: SUCCESS. Total failed assertions count= 0. Total reports count= 0. ",
  "report": {
    "successful": true,
    "summary": "Schematron validation completed successfully",
    "filename": "invoice.xml",
    "reports": {
      "AUNZ_UBL_1_0_10": {
        "rules": "AUNZ_UBL_1_0_10",
        "successful": true,
        "summary": "Validation result for AUNZ_UBL_1_0_10: Successful. No assertion errors fired.Schematron Reports fired: no schematron reports fired. ",
        "firedAssertionErrors": [],
        "firedSuccessfulReports": [],
        "firedAssertionErrorsCount": 0,
        "firedSuccessfulReportsCount": 0,
        "firedAssertionErrorCodes": []
      }
    }
  },
  "firedAssertionErrorsCount": 0,
  "allAssertionErrorCodes": [],
  "firedSuccessfulReportsCount": 0
}
```

Multi-validation

To validate XML file against multiple sets of business Rules at the same time

- Description: [Web online multi-validation](#)
- Click on the “Try it out” button to test online. This will enable the fields.

The screenshot shows the top part of the 'Web online multi-validation' interface. It includes a header with the title and a 'Try it out' button in the top right corner. Below the header, there is a section for parameters with a table that has columns for 'Name' and 'Description'. The 'Try it out' button is highlighted with a red arrow.

- Select the sets of Rules you want to apply. Set to “true” the sets of rules you want to apply. Example “AU-Nz peppol-1.0.10” and “AU-Nz ubl-1.0.10”

The screenshot shows the 'Web online multi-validation' interface with the 'Parameters' section expanded. It displays a table with columns for 'Name' and 'Description'. The table lists several rules, each with a dropdown menu to select its status. The 'true' option is selected for 'aunz_peppol_1_0_10' and 'aunz_ubl_1_0_10', while 'false' is selected for 'aunz_peppol_sb_1_0_10' and 'france_EN16931_UBL_1_3_11'. The 'france_EN16931_CII_1_3_11' rule is also listed. The 'true' selections are highlighted with red boxes.

- Select the XML file to validate from your file system

The screenshot shows the 'Web online multi-validation' interface with the 'file' parameter section. It includes a label 'file * required' and a description 'The xml file to validate.' Below this, there is a 'Choose File' button and a text input field containing the filename 'invoice.xml'.

- Click on “execute” to run the Validation

ia_RO16931_UBL_1_0_8_En16931 * required Validates against Schematron Rules of Romania ro16931-ubl-1.0.8

ia_RO16931_UBL_1_0_8_CiusRo * required Validates against Schematron Rules of Romania ro16931-ubl-1.0.8

body

* required The xml file to validate.

Choose File | invoice.xml

Execute

- The response code should be 200, and Response body will contain the Validation report (see [report format below](#) for full details)

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'https://services.ebusiness-cloud.com/ess-schematron/v1/web/validate/multi?aunz_peppol_1_0_10=false&aunz_peppol_sb_1_0_10=false&aunz_ubl_1_0_10=true&france_EN16931_CTI_1_3_11=false&france_EN16931_UBL_1_3_11=false&romania_R016931_UBL_1_0_8_En16931_CiusRo=false'
```

Request URL

```
https://services.ebusiness-cloud.com/ess-schematron/v1/web/validate/multi?aunz_peppol_1_0_10=false&aunz_peppol_sb_1_0_10=false&aunz_ubl_1_0_10=true&france_EN16931_CTI_1_3_11=false&france_EN16931_UBL_1_3_11=false&romania_R016931_UBL_1_0_8_En16931_CiusRo=false
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "customer": "Online validation tool", "successful": true, "message": "Schematron validation on file 'invoice.xml' completed with status: SUCCESS. Total failed assertions count= 0. Total reports count= 0. ", "report": { "successful": true, "summary": "Schematron validation completed successfully", "filename": "invoice.xml", "reports": { "AUNZ_UBL_1_0_10": { "rules": "AUNZ_UBL_1_0_10", "successful": true, "summary": "Validation result for AUNZ_UBL_1_0_10: Successful. No assertion errors fired.Schematron Reports fired: no schematron reports fired. ", "firedAssertionErrors": [], "firedSuccessfulReports": [], "firedAssertionErrorsCount": 0, "firedSuccessfulReportsCount": 0, "firedAssertionErrorCodes": [] } }, "firedAssertionErrorsCount": 0, "allAssertionErrorCodes": [], "firedSuccessfulReportsCount": 0 } }</pre>

Note: Although the Endpoint [API programmatic validation](#) is available on Swagger UI, it is designed for programmatic access, therefore it is recommended to call this endpoint directly from your program instead of using the UI (see [Programmatic access](#) below)

Programmatic Access

OpenAPI definition

The definition can be found here

<https://services.ebusiness-cloud.com/ess-schematron/v3/api-docs/full>

You can use it to automatically generate the code of your Rest Client using your favorite tool (depends on what language you use - search on Google, plenty of tools available for all languages).

Note: Although the Endpoint paths “/v1/web/validate/single” and “” are available on the OpenAPI doc, it is preferable to not use them for programmatic access as they were designed for the Swagger UI use (see [Swagger UI endpoints](#)).

Setting up you preferred Report Language

To set English as the report language, you need to pass in the Header:

`Accept-Language=en`

Validation rules set format

When using the API programmatically, note that the sets of rules to validate against are passed as a List of String. Possible values are listed in the API doc.

```
ValidationRules v string
Enum:
  [ AUNZ_PEPPOL_1_0_10, AUNZ_PEPPOL_SB_1_0_10, AUNZ_UBL_1_0_10, FR_EN16931_CIT_1_3_11, FR_EN16931_UBL_1_3_11, RO_RO16931_UBL_1_0_8_EN16931, RO_RO16931_UBL_1_0_8_CTUS_RO ]
```

File format in the request

When using the API programmatically, note that the file content is passed as Base64 encoded content. A checksum is added as well to make sure the content is correct.

content field = the content of the file Base64 encoded

Checksum = the MD5 hash of the Base64 encoded content

Use your favorite language to encode base64 and MD5 your content.

See API doc for detailed DTO structure.

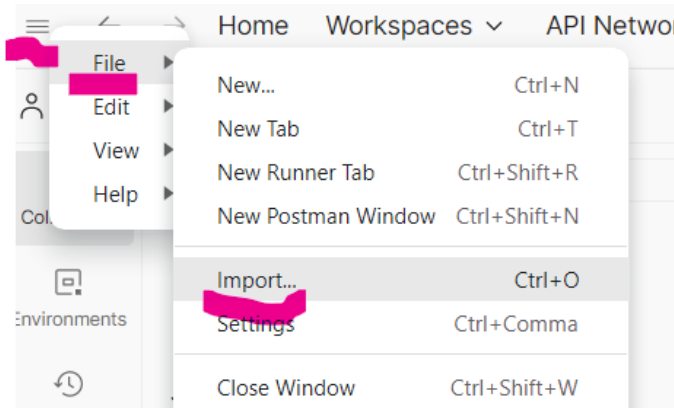
```
FileDto v {
  description: The input xml file Dto. Content needs to be Base 64 encoded and will be validated against checksum value. Checksum is the md5Hex of the encoded content. Used for API usage.
  filename      string
  content       string
  checksum      string
}
```

Example of Requests using Postman

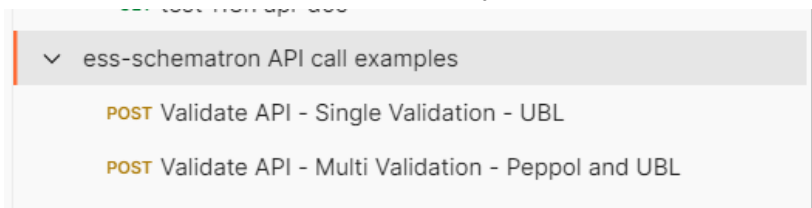
You can import the following Collection and check few examples of Postman API calls. You can use it to test and validate what are the parameters and how they should be passed before doing it from your program.

Import the library

1. Download this file on your machine [ess-schematron API call examples.postman_collection](#)
2. Open your postman
3. Click on File > import



4. Select the file you downloaded previously and import
5. You should have a new collection on your Workspace with the test APIs created for you

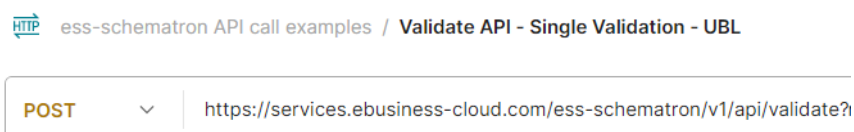


6. There are 2 sample methods. Click on the one you want to explore the details. You can choose to duplicate those methods and create as different tests as you want.
7. Check next section to know more about the parameters inside those examples

API call parameters in Postman

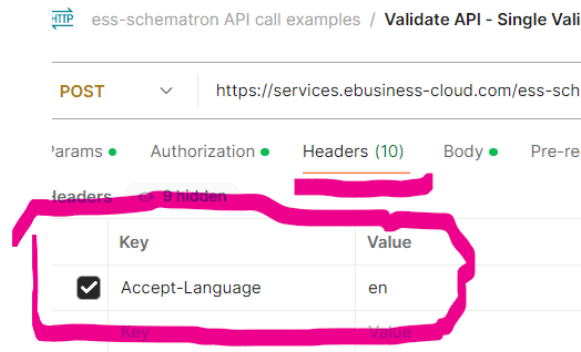
Let's take as an example the request "Validate API - Single Validation - UBL"

1. URL and http method: the first parameter set up is the url of the request, in our case it's always "<https://services.ebusiness-cloud.com/ess-schematron/v1/api/validate?>" and POST http method:



2. Request Headers (including Authentication)

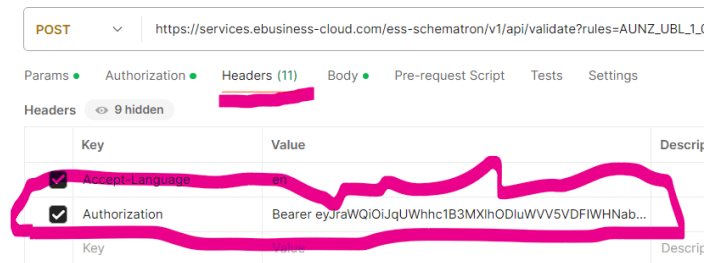
- a. Language: you need to set your preferred language if you want to override the default one:



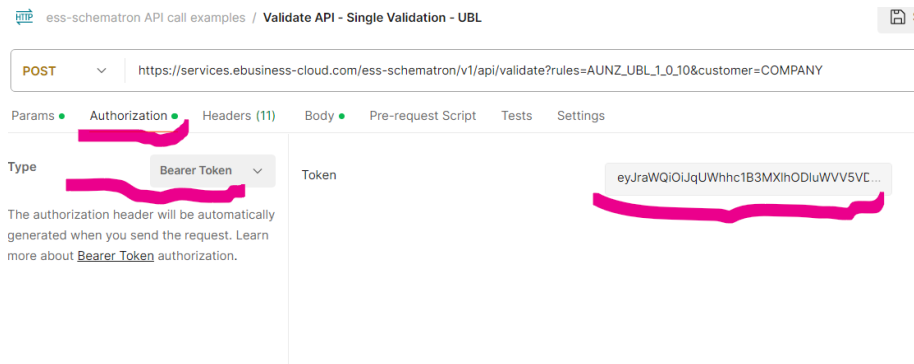
- b. Authentication:

You need first to Generate your token using your preferred library to do so ([see how to generate the token here](#)), then to setup the token in postman you have 2 options: (whether you use option 1 or 2, the Request should be *exactly the same* -in the example you imported, we are using Option ii)

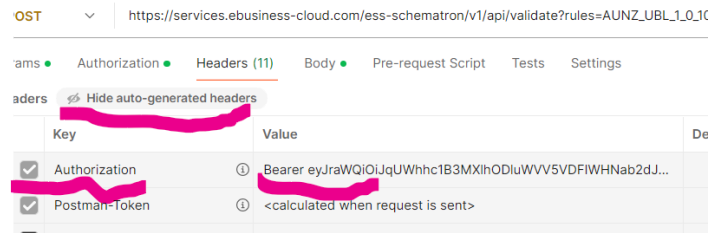
- i. You can set yourself the token in the header: in that case you add a new parameter to the Header called “Authorization”, and you add as value “Bearer replace-this-by-your-token-value”



- ii. You can let Postman generate this header for you as well. To do that, you Go to the tab Authorisation and select the “Bearer Token” type, then just insert the value of your token in that field

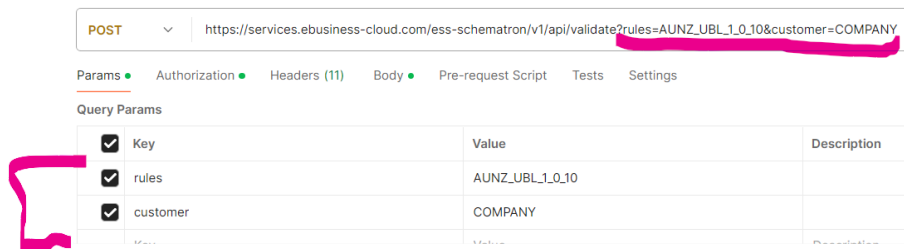


This will make postman generate automatically the same Header value (it's hidden field if you want to see it you must unhide them)



3. Request Parameters: This is where you pass the parameters as defined by the API. In our example we are passing mandatory param which is the “rules” list (only 1 validation “AUNZ_UBL1_0_10”), and we are passing an optional param customer=COMPANY.

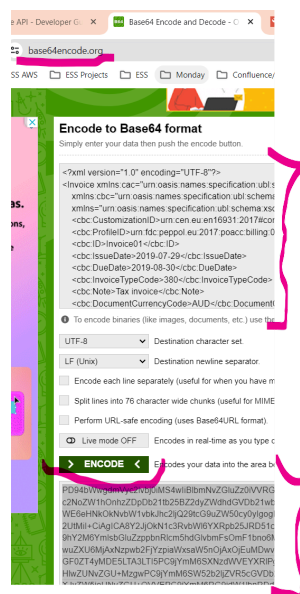
ess-schematron API call examples / Validate API - Single Validation - UBL



The second example shows how to do Multi validation. Refer to the API doc to know full list of possible rules to pass.

4. Request Body: That’s where we pass the fileDto object as json. The body contains:
 - a. “filename”: you can pick any name you want (it’s more for your convenience and tracking client side)
 - b. content: you file content base64 encoded. This will be generated by your program. You can do it manually using any online tool by copying your XML content there and generating the base64 content. Example of free tool:

<https://www.base64encode.org/>

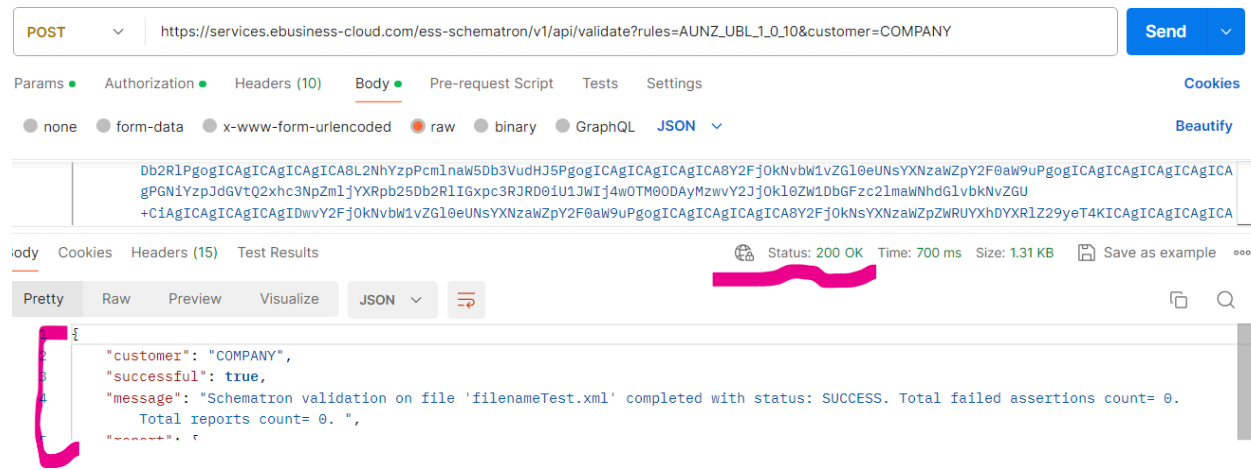


- c. `"checksum"`: you need to calculate the md5 checksum of your encoded content. You can use an online tool as well to generate the checksum manually. Example <https://onlinehashtools.com/calculate-md5-hash>

- d. Full body will look like this once all data in

5. Making the request:

- a. Just click on the “Send” Button
- b. You should get status 200 OK and the Validation Report



c. If you get the errors:

- i. 401 -> it means your token not set correctly
- ii. 403 -> it means your user not allowed to access this API (should never happen)
- iii. 400 -> it means your request parameters or body not set correctly
- iv. 500 -> it means the rules or file format, content or encoding are not correct

The details about the report are in the following section.

Validation Report format

The report is in json format. It can both be read directly, or parsed by a program.

The high level structure is the following:

- Overall Validation result (Successful or Not, and Summary of the result)
- List of individual Validations, one Section for each Set of Rules:
 - For Each Set of Rules: List of Failed Validations:
 - For Each Failed Validation:
 - id=Rule Code
 - text=Description of the Validation error
 - location=the XPath of the data (path in the XML)

Example of Failed report:

```
{
  "customer": "Online validation tool",
  "successful": false,
  "message": "Schematron validation on file 'invoice.xml' completed with status:
FAILED. Total failed assertions count= 2. Failed assertions codes: {
PEPPOL-EN16931-R007-AUNZ-SB, PEPPOL-EN16931-R004-AUNZ-SB }. Total reports count= 0. ",
  "report": {
    "successful": false,
    "summary": "Validation failed. Check individual validation reports for details",
    "filename": "invoice.xml",
    "reports": {
      "AUNZ_PEPPOL_SB_1_0_10": {
        "rules": "AUNZ_PEPPOL_SB_1_0_10",
        "successful": false,
        "summary": "Validation result for AUNZ_PEPPOL_SB_1_0_10: Failed. Failed
assertions count = 2. Assertion errors: { PEPPOL-EN16931-R007-AUNZ-SB,
PEPPOL-EN16931-R004-AUNZ-SB }. Schematron Reports fired: no schematron reports fired.
",
        "firedAssertionErrors": [
          {
            "id": "PEPPOL-EN16931-R007-AUNZ-SB",
            "text": "Business process MUST be in the format
'urn:fdc:peppol.eu:2017:poacc:selfbilling:NN:1.0' where NN indicates the process
number.",
            "location":
"/*:Invoice[namespace-uri()='urn:oasis:names:specification:ubl:schema:xsd:Invoice-2'] [
1]",
            "test": "$profile != 'Unknown'",
            "flag": "fatal"
          },
          {
            "id": "PEPPOL-EN16931-R004-AUNZ-SB",
            "text": "Specification identifier MUST have the value
'urn:cen.eu:en16931:2017#conformant#urn:fdc:peppol.eu:2017:poacc:selfbilling:internati
onal:aunz:3.0'.",
            "location":
"/*:Invoice[namespace-uri()='urn:oasis:names:specification:ubl:schema:xsd:Invoice-2'] [
1]",
```



```

        "test": "starts-with(normalize-space(cbc:CustomizationID/text()),
'urn:cen.eu:en16931:2017#conformant#urn:fdc:peppol.eu:2017:poacc:selfbilling:international:aunz:3.0')",
        "flag": "fatal"
    }
},
"firedSuccessfulReports": [],
"firedAssertionErrorsCount": 2,
"firedSuccessfulReportsCount": 0,
"firedAssertionErrorCodes": [
    "PEPPOL-EN16931-R007-AUNZ-SB",
    "PEPPOL-EN16931-R004-AUNZ-SB"
]
},
"AUNZ_UBL_1_0_10": {
    "rules": "AUNZ_UBL_1_0_10",
    "successful": true,
    "summary": "Validation result for AUNZ_UBL_1_0_10: Successful. No assertion
errors fired.Schematron Reports fired: no schematron reports fired. ",
    "firedAssertionErrors": [],
    "firedSuccessfulReports": [],
    "firedAssertionErrorsCount": 0,
    "firedSuccessfulReportsCount": 0,
    "firedAssertionErrorCodes": []
}
},
"firedAssertionErrorsCount": 2,
"allAssertionErrorCodes": [
    "PEPPOL-EN16931-R007-AUNZ-SB",
    "PEPPOL-EN16931-R004-AUNZ-SB"
],
"firedSuccessfulReportsCount": 0
}
}

```

Support/Help

For any question related to the API, contact yacine@ebsoftwareservices.com.au or comment on this document. Thanks