Inhaltsverzeichnis

Server access

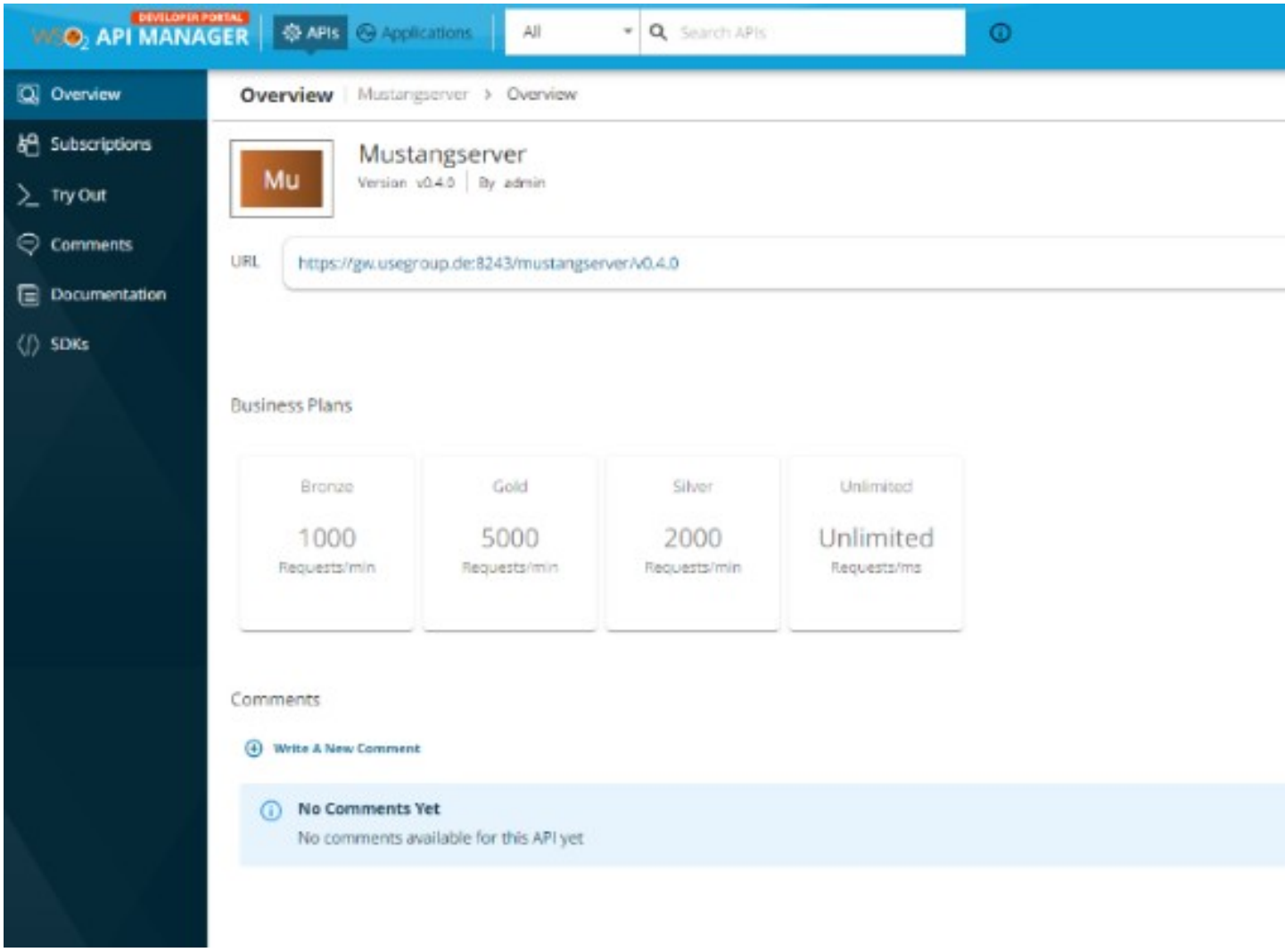
User access to the web interface is possible via https://api.usegroup.de/. Terms of service are listed in the chapter Terms of service on page 26.

To register please access https://api.usegroup.de:9443/devportal/services/configs
and click "Create Account" link on the bottom left. Select a username, "Proceed to self register", enter the rest of the data and have your email verified.
Afterwards you can login on https://api.usegroup.de/devportal/ .

You will need it your username in order to log in and in case you want to reset your password, both are *not possible using your email address*. In case you forgot your username feel free to inquire at info@mustangproject.org using the email address you are requesting the username for. This username does not have anything to do with optional username parameters mentioned below.

After being logged in you need to register your interest, i.e. „subscribe“ to the APIs you require.

If you select the desired Mustangserver version and click on the blue "try out" button (not the link in the navigation) on the next page you should be able to click a "get test key" button.



e.g. when you open the "ping" operation and click "try it out" and "execute" you should get a "pong" response.

You can always change your password on https://api.usegroup.de/devportal/settings/change-password/

There is a optional „username“ field for all operations: Please ignore it. It serves as a placeholder where the API management transmits your username, if it were set by any application the value would be overwritten anyway.

Authentication

OAuth2

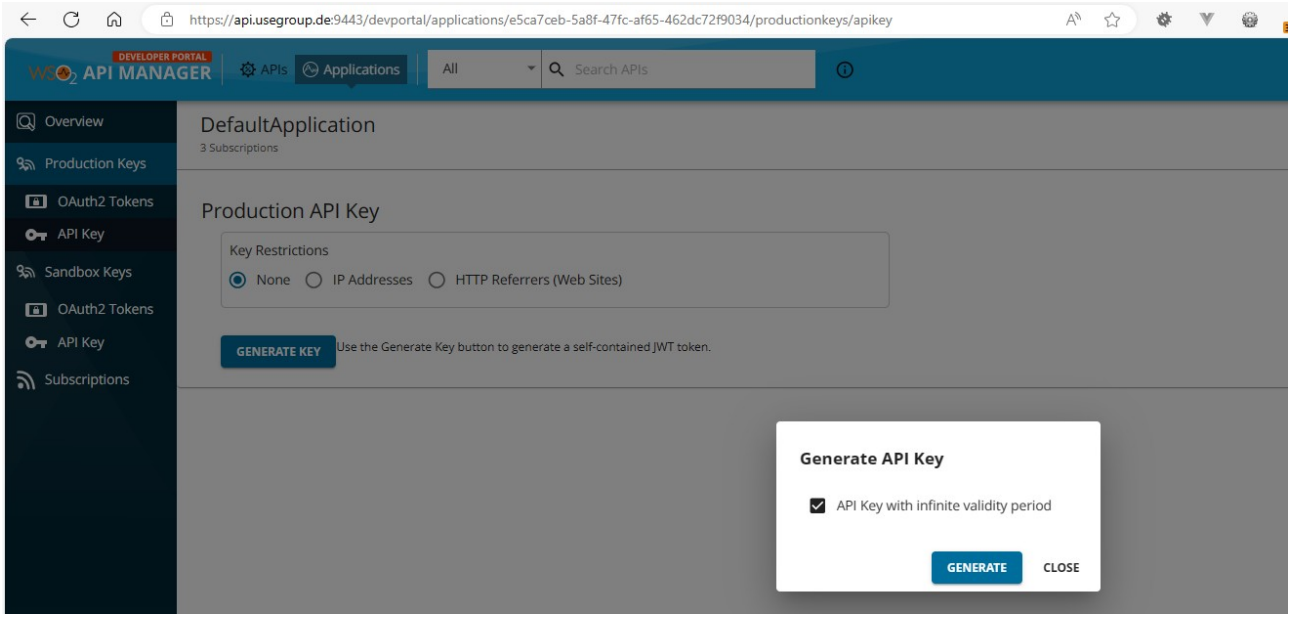
First of all you will have to subscribe to the API. You can manage your subscriptions in the left navigation of the devportal but it's often easiest to subscribe via overview|try out, which also allows you to try it.

Then you will have to enable client credentials in the applications tab, https://api.usegroup.de:9443/devportal/applications Default Application, Oauth2 tokens. The procedure is described more detailled in the PHP Client chapter „Allowing Client Credentials“ on page 10 but is generic to all examples and does not apply for PHP only.

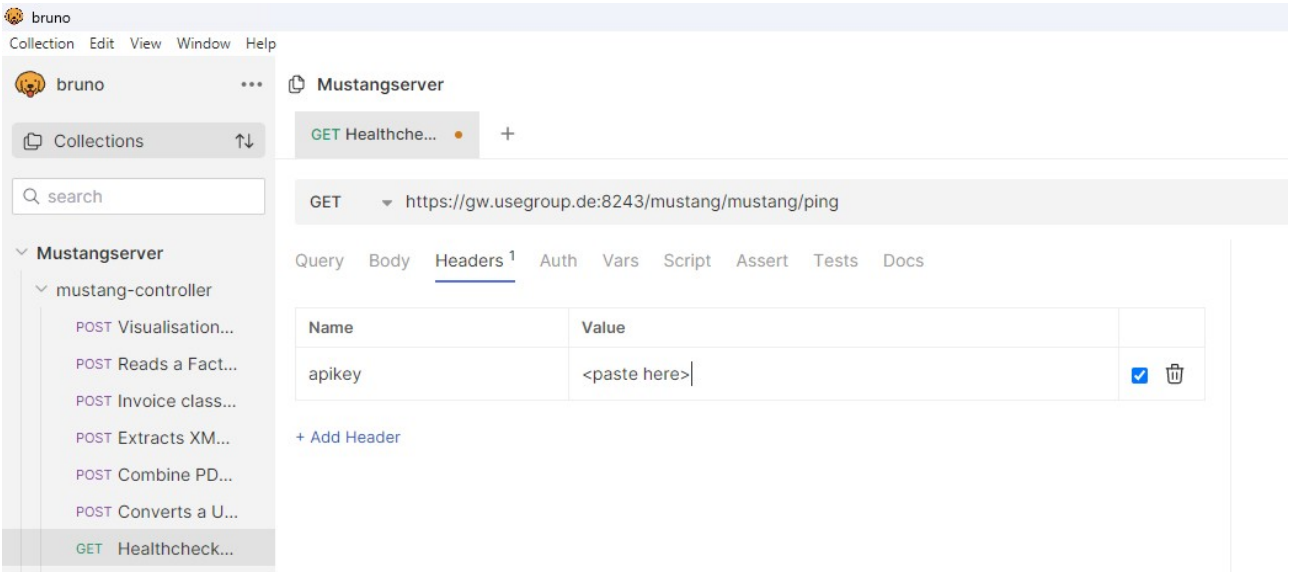
Please note that most clients will use the Mustangserver version which had been *selected* in the backend when downloading the OpenAPI definition. Feel free to replace /mustang/<version>/mustang by as described in Endpoint for most-recent API versions on page 4.

Api Key

In the applications tab, https://api.usegroup.de:9443/devportal/applications select Default Application, Oauth2 tokens, Production Keys, API Key. Select according restrictions if required, click Generate Key, select lifetime and click generate. Copy&safely store the generated key.



An API key can be used e.g. within Bruno (pp 20ff)



Within PHP you also might pass just as additional header attribute making your source code look like

```
 <?php

require_once(__DIR__ . '/vendor/autoload.php');

$apikey="<your key>";
$config = Swagger\Client\Configuration::getDefaultConfiguration();
$gc=new GuzzleHttp\Client(['headers' => ['apikey' => $apikey]]);

$apiInstance = new Swagger\Client\Api\MustangControllerApi(
    $gc,
    $config
);

try {
    $result = $apiInstance->ping();
    print_r($result);
} catch (Exception $e) {
    http_response_code(500);
    echo 'Exception when calling ErrorControllerApi->handle: ', $e->getMessage(), PHP_EOL;
}
```

Endpoint for most-recent API versions

If you leave out the version number in the request to the gateway you will always be using the latest recommended (usually the latest) version. In that case e.g. your „ping“ endpoint changes from https://gw.usegroup.de:8243/mustang/1.0.0/mustang/ping to https://gw.usegroup.de:8243/mustang/mustang/ping .

New versions may be retired as soon as six month after release of the successor. It is possible to always use the latest (more precisely: recommended) version by removing the version from the endpoint. This is an example of the validate endpoint:

```
-         "https://gw.usegroup.de:8243/mustang/v0.5.0/mustang/validate",
+         "https://gw.usegroup.de:8243/mustang/mustang/validate",
```

Errors and Exceptions

In case of an exception Mustangserver will return a http status code of 400 and the message of the Exception will be returned in the „message“ field of the according JSON.

```
{
  "requestUrl": "http://127.0.0.1:8000/mustang/combineXML",
  "httpCode": 400,
  "errorCode": "MSE1000:Unbekannte Fehler während der Request Ausführung!",
  "message": "File is not a valid PDF/A-1 input file"
}
```

Mediation, Transformation and Orchestration

The http://api.usegroup.de/ uses WSO² as API management which in turn uses Apache Synapse (https://synapse.apache.org/) for mediation/transformation/orchestration. This means that mediation and orchestration can be developed e.g. in WSO²‘s Integration Studio (https://wso2.com/integration/integration-studio/) and uploaded as XML file. Apart from acting as a load balancer and central authentication this allow to

- override certain states in the process, e.g. implement a timeout after a certain number of seconds
- invoke a chain of operations in only one virtual endpoint, e.g. conversion from plain PDF, parallely converting invoice data to XML, merging PDF/A and XML and validation thereof and/or
- map any custom specific input- or output parameter to the values used by Mustangserver internally

Operations/Endpoints

Mustangserver’s available operations are

Ping

Just a test, always just returns „pong“. The only operation acessible via HTTP GET, the rest is POST.

Validate

validate: Validate a Factur-X/ZUGFeRD or XRechnung CII or Order-X-CIO File using Mustang’s validator. Requires a file and returns a XML report. The format to be validated against will be read from it’s guideline ID.

Phive

Validate a CII or UBL file using https://github.com/phax/phive . Requires a file and returns JSON.

Available format/standard/version-combinations (VES IDs) are

de.xrechnung:cii:1.2.0
de.xrechnung:cii:1.2.1
de.xrechnung:cii:1.2.2
de.xrechnung:cii:2.0.0
de.xrechnung:cii:2.0.1
de.xrechnung:cii:2.1.1
de.xrechnung:cii:2.2.0
de.xrechnung:cii:2.3.1
de.xrechnung:cii:3.0.1
de.xrechnung:ubl-creditnote:1.2.0
de.xrechnung:ubl-creditnote:1.2.1
de.xrechnung:ubl-creditnote:1.2.2
de.xrechnung:ubl-creditnote:2.0.0
de.xrechnung:ubl-creditnote:2.0.1
de.xrechnung:ubl-creditnote:2.1.1
de.xrechnung:ubl-creditnote:2.2.0
de.xrechnung:ubl-creditnote:2.3.1
de.xrechnung:ubl-creditnote:3.0.1
de.xrechnung:ubl-invoice:1.2.0
de.xrechnung:ubl-invoice:1.2.1
de.xrechnung:ubl-invoice:1.2.2
de.xrechnung:ubl-invoice:2.0.0
de.xrechnung:ubl-invoice:2.0.1
de.xrechnung:ubl-invoice:2.1.1
de.xrechnung:ubl-invoice:2.2.0
de.xrechnung:ubl-invoice:2.3.1
de.xrechnung:ubl-invoice:3.0.1
eu.cen.en16931:cii:1.0.0
eu.cen.en16931:cii:1.1.0
eu.cen.en16931:cii:1.2.0
eu.cen.en16931:cii:1.2.1
eu.cen.en16931:cii:1.2.3
eu.cen.en16931:cii:1.3.0
eu.cen.en16931:cii:1.3.1
eu.cen.en16931:cii:1.3.10
eu.cen.en16931:cii:1.3.11
eu.cen.en16931:cii:1.3.2
eu.cen.en16931:cii:1.3.3
eu.cen.en16931:cii:1.3.4
eu.cen.en16931:cii:1.3.5
eu.cen.en16931:cii:1.3.6
eu.cen.en16931:cii:1.3.6a
eu.cen.en16931:cii:1.3.7
eu.cen.en16931:cii:1.3.8
eu.cen.en16931:cii:1.3.9
eu.cen.en16931:ubl-creditnote:1.0.0
eu.cen.en16931:ubl-creditnote:1.1.0
eu.cen.en16931:ubl-creditnote:1.2.0
eu.cen.en16931:ubl-creditnote:1.2.1
eu.cen.en16931:ubl-creditnote:1.2.3
eu.cen.en16931:ubl-creditnote:1.3.0
eu.cen.en16931:ubl-creditnote:1.3.1
eu.cen.en16931:ubl-creditnote:1.3.10
eu.cen.en16931:ubl-creditnote:1.3.11
eu.cen.en16931:ubl-creditnote:1.3.2
eu.cen.en16931:ubl-creditnote:1.3.3
eu.cen.en16931:ubl-creditnote:1.3.4
eu.cen.en16931:ubl-creditnote:1.3.5
eu.cen.en16931:ubl-creditnote:1.3.6
eu.cen.en16931:ubl-creditnote:1.3.6a
eu.cen.en16931:ubl-creditnote:1.3.7
eu.cen.en16931:ubl-creditnote:1.3.8
eu.cen.en16931:ubl-creditnote:1.3.9
eu.cen.en16931:ubl:1.0.0
eu.cen.en16931:ubl:1.1.0
eu.cen.en16931:ubl:1.2.0
eu.cen.en16931:ubl:1.2.1
eu.cen.en16931:ubl:1.2.3
eu.cen.en16931:ubl:1.3.0
eu.cen.en16931:ubl:1.3.1
eu.cen.en16931:ubl:1.3.10
eu.cen.en16931:ubl:1.3.11
eu.cen.en16931:ubl:1.3.2
eu.cen.en16931:ubl:1.3.3
eu.cen.en16931:ubl:1.3.4
eu.cen.en16931:ubl:1.3.5
eu.cen.en16931:ubl:1.3.6
eu.cen.en16931:ubl:1.3.6a
eu.cen.en16931:ubl:1.3.7
eu.cen.en16931:ubl:1.3.8
eu.cen.en16931:ubl:1.3.9
eu.peppol.bis3.aunz.ubl:creditnote-self-billing:1.0.10
eu.peppol.bis3.aunz.ubl:creditnote-self-billing:1.0.9
eu.peppol.bis3.aunz.ubl:creditnote:1.0.10
eu.peppol.bis3.aunz.ubl:creditnote:1.0.9

eu.peppol.bis3.aunz.ubl:invoice-self-billing:1.0.10
eu.peppol.bis3.aunz.ubl:invoice-self-billing:1.0.9
eu.peppol.bis3.aunz.ubl:invoice:1.0.10
eu.peppol.bis3.aunz.ubl:invoice:1.0.9
eu.peppol.bis3.sg.ubl:creditnote:1.0.3
eu.peppol.bis3.sg.ubl:creditnote:2023.7.0
eu.peppol.bis3.sg.ubl:invoice:1.0.3
eu.peppol.bis3.sg.ubl:invoice:2023.7.0
eu.peppol.bis3:catalogue-response:2023.11.0
eu.peppol.bis3:catalogue-response:2023.5.0
eu.peppol.bis3:catalogue:2023.11.0
eu.peppol.bis3:catalogue:2023.5.0
eu.peppol.bis3:creditnote:2023.11.0
eu.peppol.bis3:creditnote:2023.5.0
eu.peppol.bis3:despatch-advice:2023.11.0
eu.peppol.bis3:despatch-advice:2023.5.0
eu.peppol.bis3:invoice-message-response:2023.11.0
eu.peppol.bis3:invoice-message-response:2023.5.0
eu.peppol.bis3:invoice:2023.11.0
eu.peppol.bis3:invoice:2023.5.0
eu.peppol.bis3:mlr:2023.11.0
eu.peppol.bis3:mlr:2023.5.0
eu.peppol.bis3:order-agreement:2023.11.0
eu.peppol.bis3:order-agreement:2023.5.0
eu.peppol.bis3:order-cancellation:2023.11.0
eu.peppol.bis3:order-cancellation:2023.5.0
eu.peppol.bis3:order-change:2023.11.0
eu.peppol.bis3:order-change:2023.5.0
eu.peppol.bis3:order-response-advanced:2023.11.0
eu.peppol.bis3:order-response-advanced:2023.5.0
eu.peppol.bis3:order-response:2023.11.0
eu.peppol.bis3:order-response:2023.5.0
eu.peppol.bis3:order:2023.11.0
eu.peppol.bis3:order:2023.5.0
eu.peppol.bis3:punch-out:2023.11.0
eu.peppol.bis3:punch-out:2023.5.0
eu.peppol.directory:businesscard:1.0.0
eu.peppol.directory:businesscard:2.0.0
eu.peppol.directory:businesscard:3.0.0
eu.peppol.reporting:eusr:1.0.0
eu.peppol.reporting:eusr:1.0.0RC2
eu.peppol.reporting:eusr:1.0.1
eu.peppol.reporting:eusr:1.1.0
eu.peppol.reporting:eusr:1.1.1
eu.peppol.reporting:eusr:1.1.2
eu.peppol.reporting:eusr:1.1.3
eu.peppol.reporting:eusr:1.1.4
eu.peppol.reporting:tsr:1.0.0
eu.peppol.reporting:tsr:1.0.1
eu.peppol.reporting:tsr:1.0.2
eu.peppol.reporting:tsr:1.0.3
eu.peppol.reporting:tsr:1.0.4
org.peppol.jp.pint:credit-note:0.1.2
org.peppol.jp.pint:invoice:0.1.2
org.peppol.pint:credit-note:1.0.0
org.peppol.pint:credit-note:1.0.1
org.peppol.pint:invoice:1.0.0
org.peppol.pint:invoice:1.0.1

pdf

(in the Mustangserver-docs API) Create a PDF/A file from any input PDF. Requires a PDF file (plain PDF, PDF A/1, PDF/A-3 or PDF/X) and a integer PDFAVersion. This operation will remove all non-PDF/A features as well as any embedded files, including potentially available Factur-X/ZUGFeRD files, and embed only available fonts. PDFAVersion should be 1, 2 or 3 for PDF/A-1, PDF/A-2 or PDF/A-3 respectively.

parse

Read a Factur-X/ZUGFeRD/XRechnung and create a JSON representation. Requires a Factur-X or Order-X file.

Invoice2xml

Convert a Factur-X/ZUGFeRD/XRechnung JSON representation to XML. Requires a input JSON string, a format (ZUGFeRD = zf, XRechnung = xr, Factur-X = fx or Order-X=ox), a version (usually 2 for ZUGFeRD and 1 for Factur-X) and a profile

("MINIMUM","BASICWL","BASIC","EN16931","EXTENDED" or "XRECHNUNG" for Factur-X, for ZUGFeRD 1 "BASIC","COMFORT" or "EXTENDED"). For XRechnung only "XRECHNUNG".

Extract

Extracts just the XML (not as JSON like parse) from a Factur-X/ZUGFeRD/Order-X file.

combineXML

Combines CII XML and a PDF/A document to a Factur-X/ZUGFeRD PDF/A-3 document. Requires a input PDF/A-1 or A-3 file, a format (ZUGFeRD = zf, Factur-X = fx or Order-X = ox), a version (usually 2 for ZUGFeRD and 1 for Factur-X) and a profile ("MINIMUM","BASICWL","BASIC","EN16931","EXTENDED" or "XRECHNUNG" for Factur-X, for ZUGFeRD 1 "BASIC","COMFORT" or "EXTENDED"). Order-X currently accepts only PDF/A-1 as input PDF.

cii2ubl

transforms XML from the UN/CEFACT Cross Industry Invoice (CII) XML format, the basis of factur-x/ZUGFeRD and the CII version of the XRechnung, to the Universal Business Language format, UBL. Requires a CII string.

xmltohtml

converts a UBL or CII XML file (parameter file) into a human readable HTML in the language specfied in language, which can be EN, DE or FR. The resulting file will require the additional files in the same directory:

- xrechnung-viewer.css and
- xrechnung-viewer.js

Example PHP Client

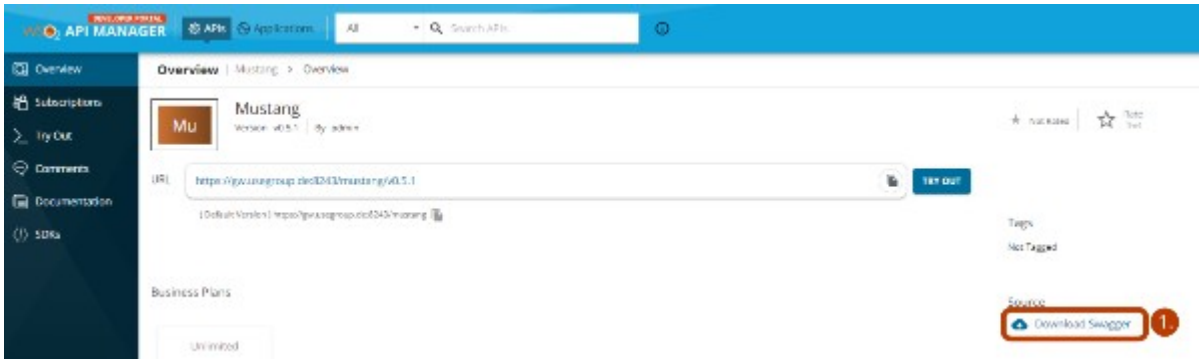
This example operates in a PHP context but https://editor.swagger.io/ also allows

C#, Dart, HTML, Go, Java, Javascript, Kotlin, Python, R, Ruby, Scala, Swift and Typescript.

Mustangserver Hello World

Preparations

Screenshot 1:

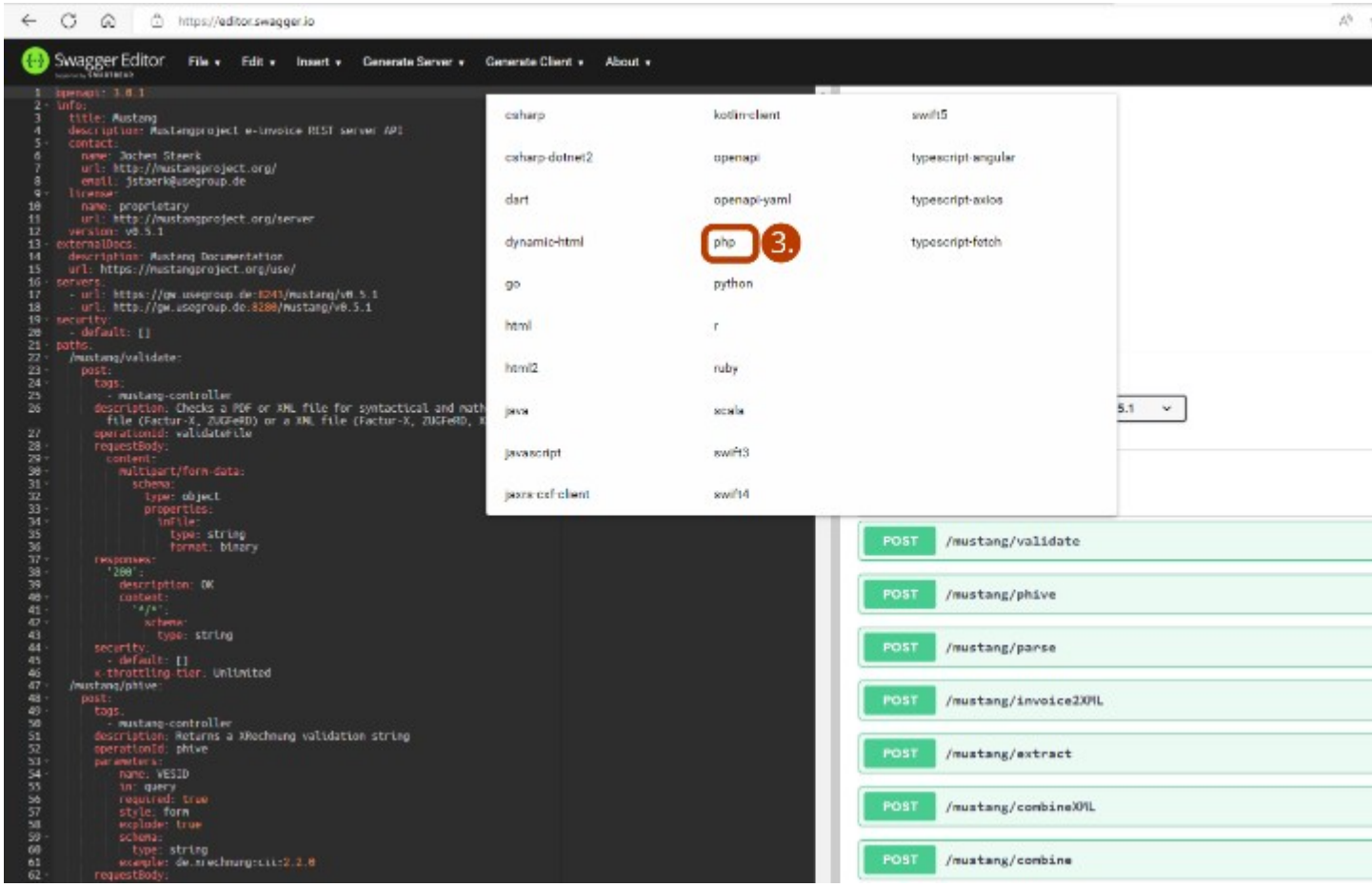


#1. Log in on https://api.usegroup.de/devportal/ , select the latest Mustangserver API and download the OpenAPI (=Swagger) definition of the API (->1.)
#2. Open the file in a text editor, select all and copy
#3. Go to editor.swagger.io, paste the definition and confirm conversion to yaml. Select Generate Client|PHP (3.). The API is public so usually there is no need to create code in a private matter. However, it is possible: Swagger editor is open source under the APL license (https://github.com/swagger-api/swagger-editor) and e.g. a Docker Image can bei obtained from https://registry.hub.docker.com/r/sebp/swagger-editor , i.e. using

```
sudo docker run --rm -p 8080:8080 sebp/swagger-editor
```

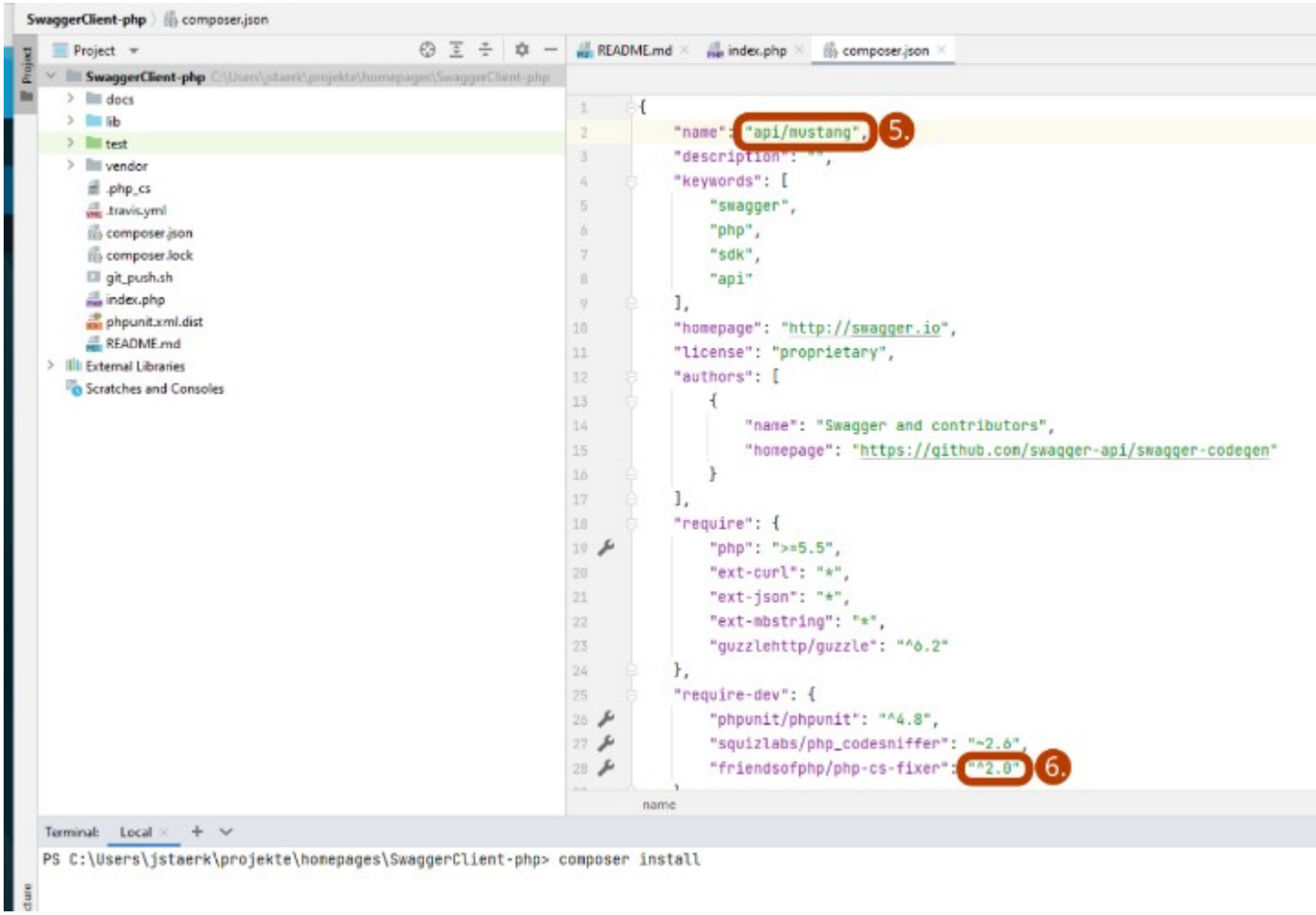
to run locally via port 8080.

Screenshot 2:



#4. Extract the downloaded file, edit composer.json.
#5. change the name of the project in the composer.json file to lowercaps/lowercaps (5.)

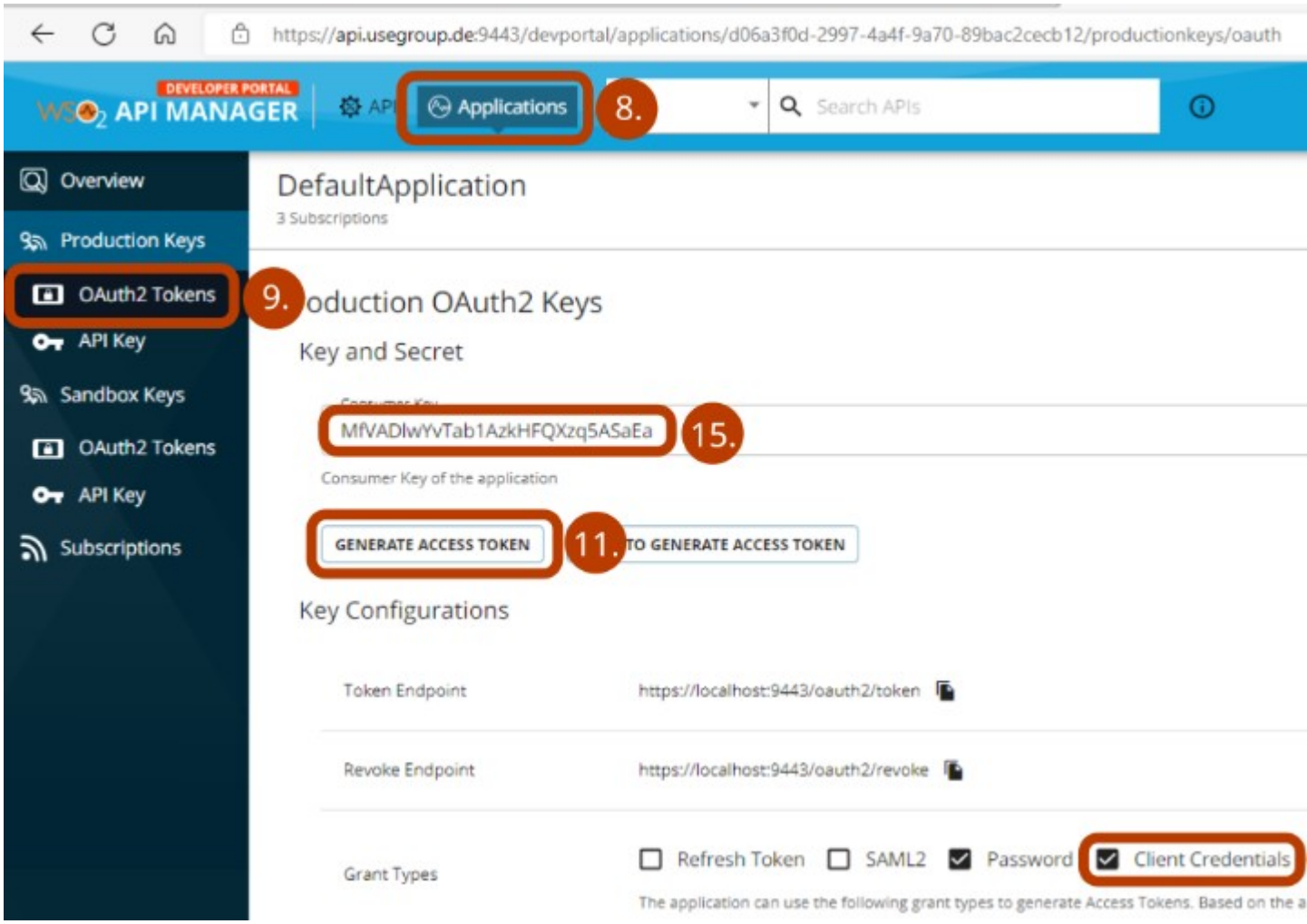
Screenshot 3:



#6. If you want to use PHP8+ upgrade the version number of php-cs-fixer to ^2.0. Then run "composer install" in that directory.
#7. Copy the example from the "Getting started" section of the readme.md to a new file, called index.php

Allowing Client Credentials

Screenshot 4:



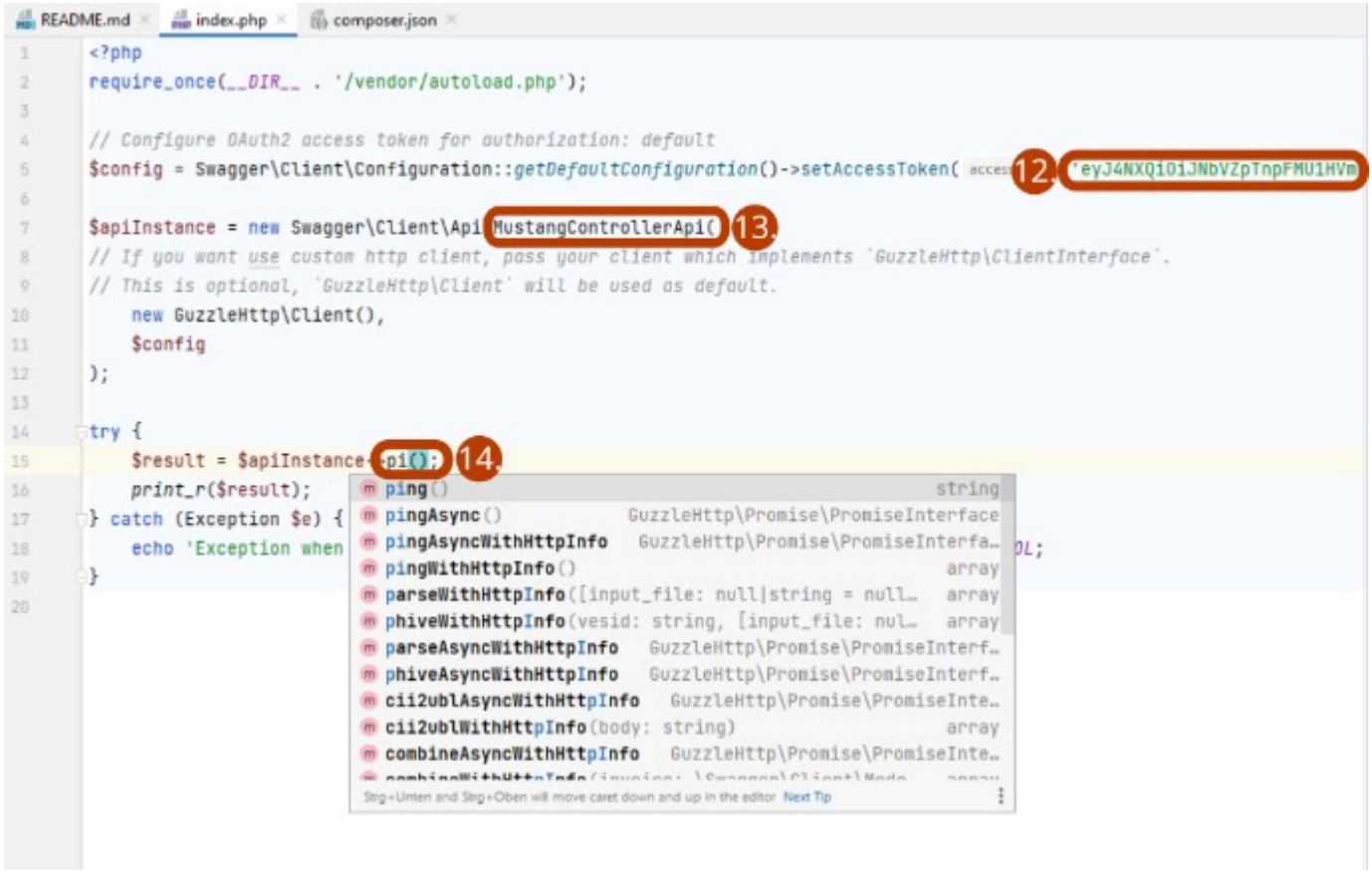
#8. Click on Applications (8.), Default Application,
#9. Production Keys/OAuth2 Token (9.).

#10. As preparation for authetication: Check Client Credentials and click the Update button on the bottom of the page.

Get access token

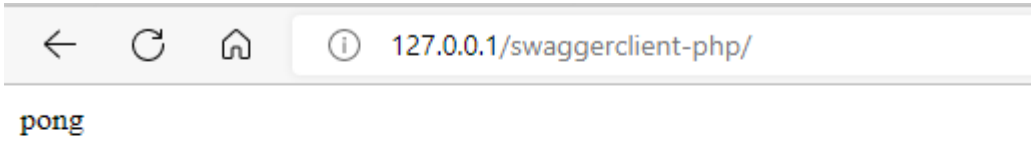
#11. For this part we will use a token which will expire shortly. Click Generate Access Token (11.), Generate and copy the resulting token. Paste it in

Screenshot 5:



#12. index.php (12.), in the same file

#13. change ErrorController to Mustangcontroller (13.) and
#14. handle() to ping() (14.). Please note that usual PHP editors will give you code completion.
Now you can open resulting index.php via your server and PHP processort in your browser, it should now look like Screenshot 6:



OAuth2 Authentication

Back to Screenshot 4:

In index.php paste the following code

$client = new GuzzleHttp\Client();

```
$res = $client->request('POST', 'https://gw.usegroup.de:9443/oauth2/token', [
   'auth' => ['<15.>', '<16.>'],
   'form_params' => [
      'grant_type' => 'client_credentials',
   ]
]);

$json = json_decode($res->getBody(), true);
```

Screenshot 7:

#15. copy Consumer Key (15., from screen 4) to the beginning of index.php (16.),

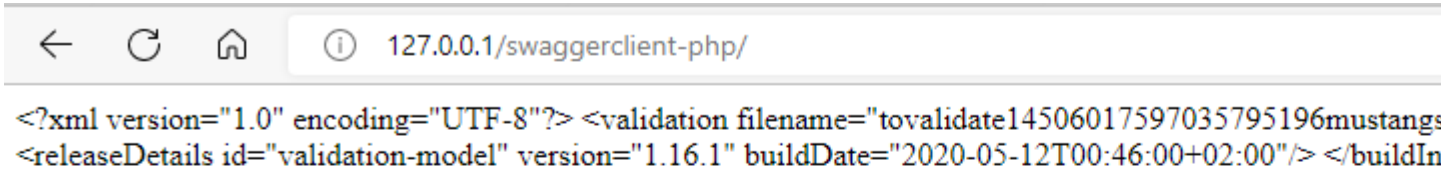
#17. reveal and copy Consumer Secret (18.).

#19 replace the static access token which will become invalid by $json["access_token"]

#20 Create or download a invoice to be validated, e.g. https://www.mustangproject.org/files/MustangGnuaccountingBeispielRE-20201121_508.pdf and save it as factur-x.pdf

#21 Change the method to validateFile and

#22 html escape the validation result, so that the result in the browser looks like screenshot 8:

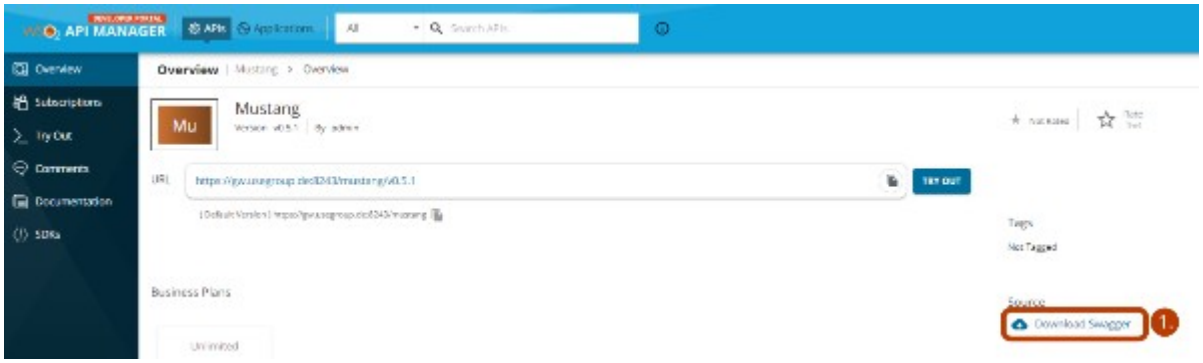


<?xml version="1.0" encoding="UTF-8"?> <validation filename="tovalidate14506017597035795196mustangs
<releaseDetails id="validation-model" version="1.16.1" buildDate="2020-05-12T00:46:00+02:00"/> </buildIn
validation profile" statement="PDF file is compliant with Validation Profile requirements." isCompliant="true"
finish="1665170599476">00:00:04.691</duration> </job> </jobs> <batchSummary totalJobs="1" failedToPar
<repairReports failedJobs="0">0</repairReports> <duration start="1665170587541" finish="1665170599531"
</pdf> <xml> <info> <version>2</version> <profile>urn:cen.eu:en16931:2017#conformant#urn:factur-x.eu:1
status="valid"/> </xml> <summary status="valid"/> </validation>

That's it. Instead of displaying the XML you can now parse it :-)

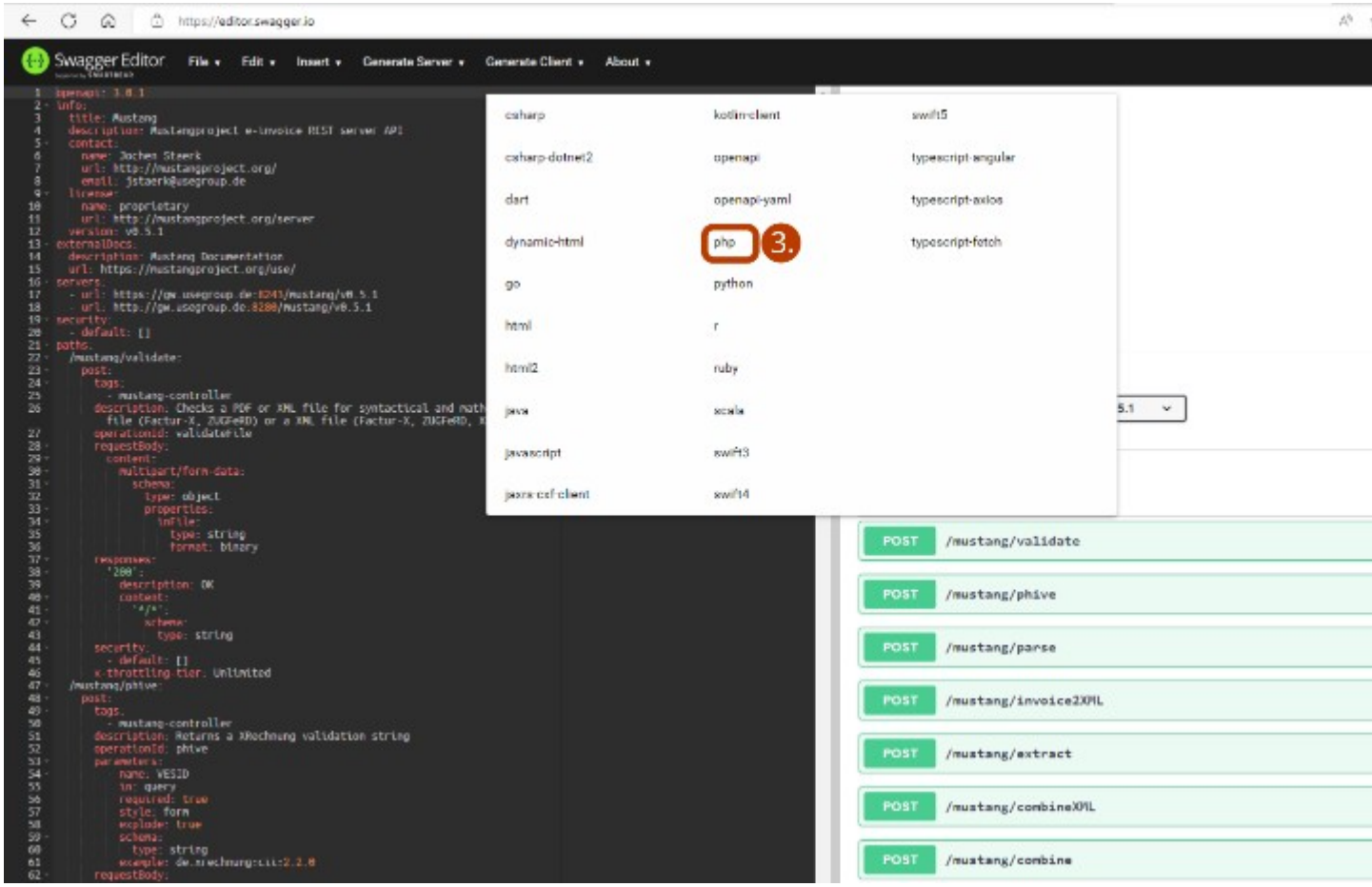
Feel free to also try the async functions.

Screenshot 1:



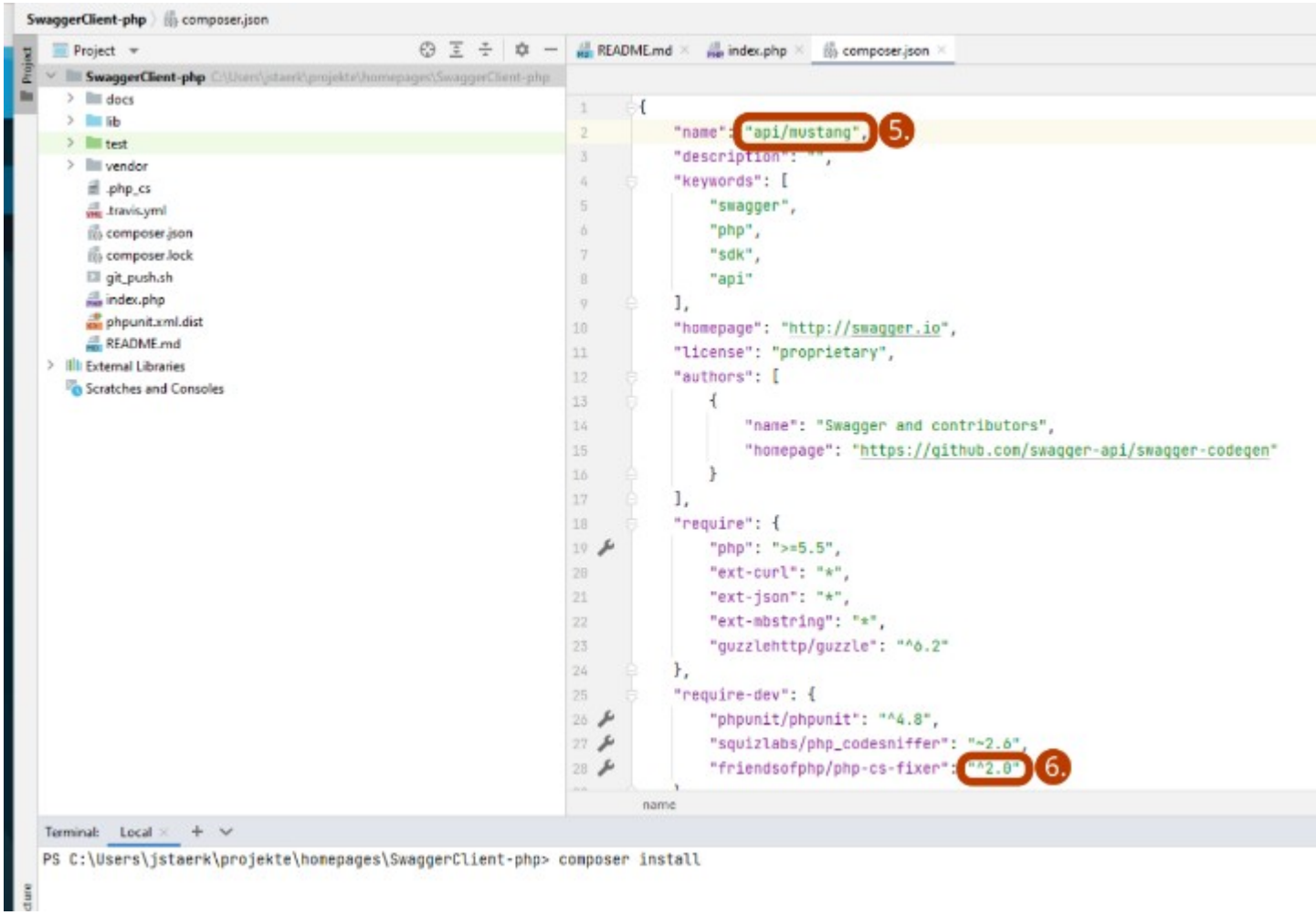
#1. Log in on https://api.usegroup.de/devportal/ , select the latest Mustangserver API and download the OpenAPI (=Swagger) definition of the API (->1.)
#2. Open the file in a text editor, select all and copy
#3. Go to editor.swagger.io, paste the definition and confirm conversion to yaml. Select Generate Client|PHP (3.)

Screenshot 2:



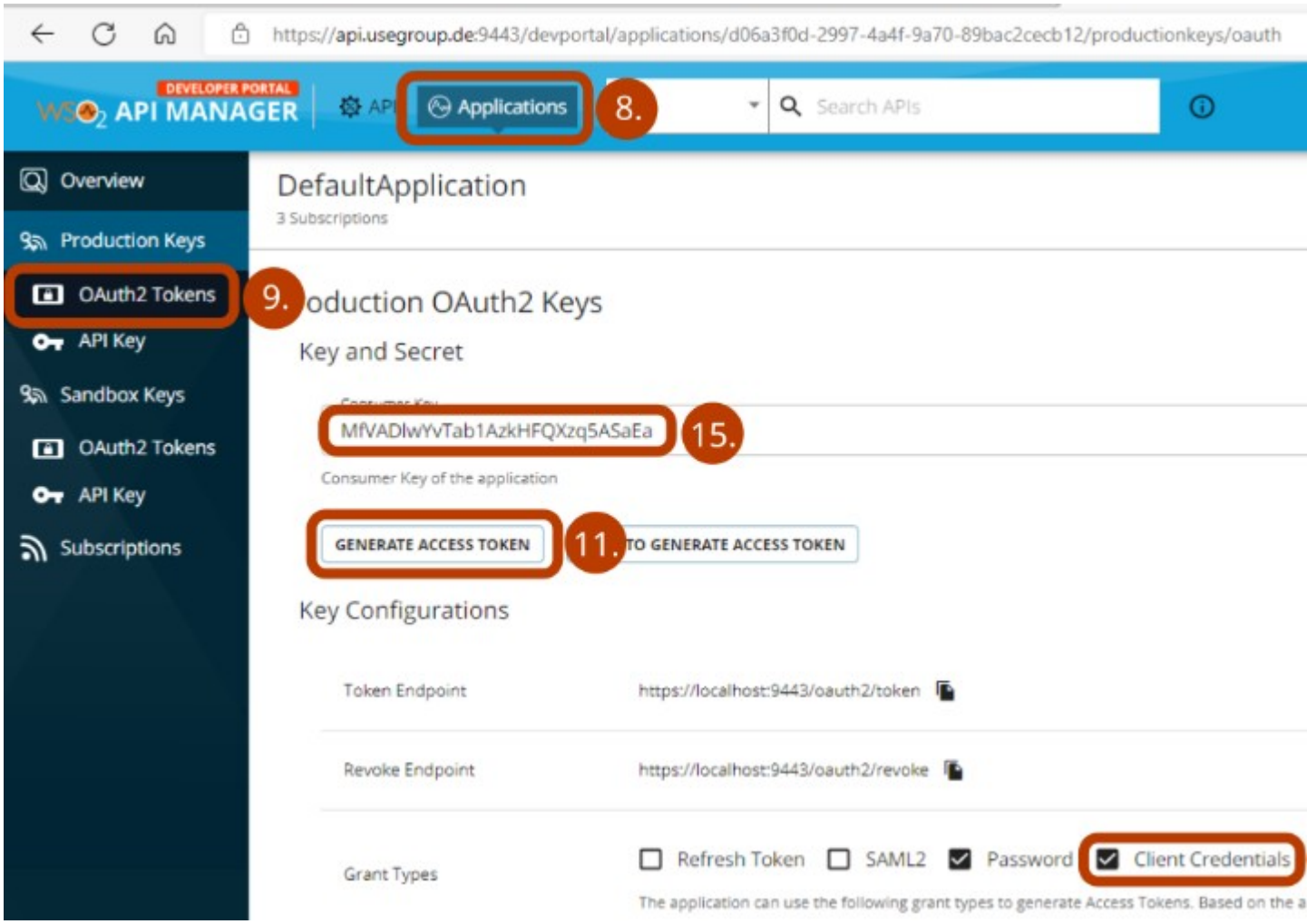
#4. Extract the downloaded file, edit composer.json.
#5. change the name of the project in the composer.json file to lowercaps/lowercaps (5.)

Screenshot 3:



#6. If you want to use PHP8+ upgrade the version number of php-cs-fixer to ^2.0. Then run "composer install" in that directory.
#7. Copy the example from the "Getting started" section of the readme.md to a new file, called index.php

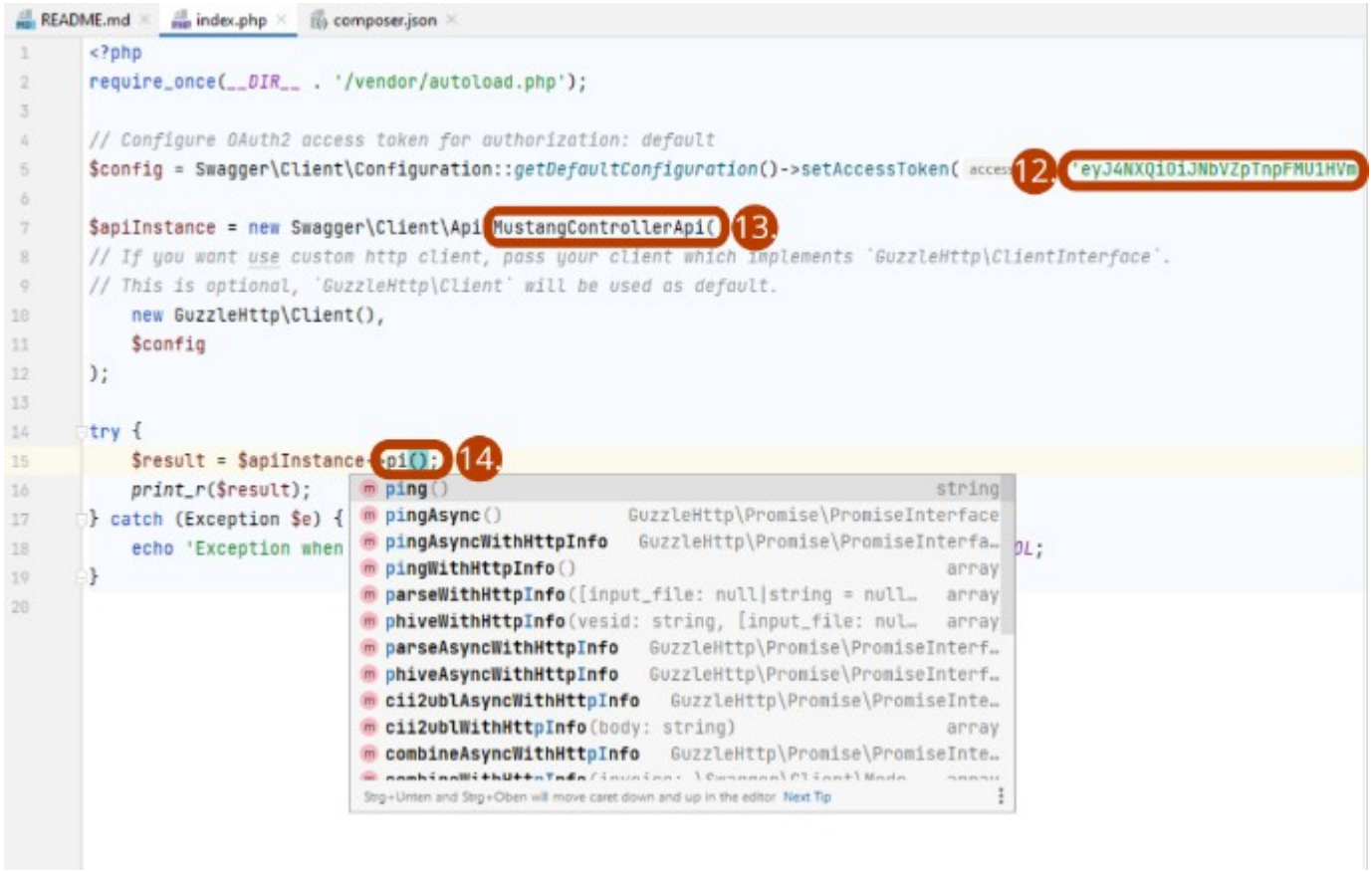
Screenshot 4:



#8. Click on Applications (8.), Default Application,
#9. Production Keys/OAuth2 Token (9.).

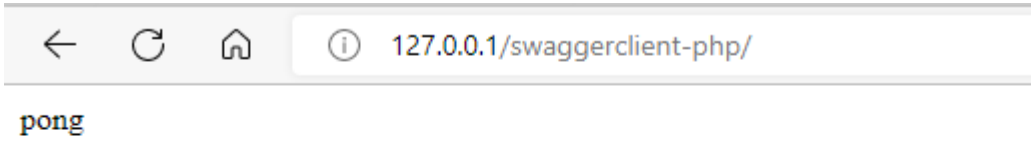
#10. As preparation for Part B: Check Client Credentials and click the Update button on the bottom of the page.
#11. For this part we will use a token which will expire shortly. Click Generate Access Token (11.), Generate and copy the resulting token. Paste it in

Screenshot 5:



#12. index.php (12.), in the same file

#13. change ErrorController to Mustangcontroller (13.) and
#14. handle() to ping() (14.). Please note that usual PHP editors will give you code completion.
Now you can open resulting index.php via your server and PHP processort in your browser, it should now look like Screenshot 6:



Validation of electronic invoices

Concerning Screenshot 4:
In index.php paste the following code

$client = new GuzzleHttp\Client();
$res = $client->request('POST', 'https://gw.usegroup.de:9443/oauth2/token', [

```
    'auth' => ['<15.>', '<16.>'],
    'form_params' => [
        'grant_type' => 'client_credentials',
    ]
]);

$json = json_decode($res->getBody(), true);
```

Screenshot 7:

#15. copy Consumer Key (15., from screen 4) to the beginning of index.php (16.),

#17. reveal and copy Consumer Secret (18.).

#19 replace the static access token which will become invalid by $json["access_token"]

#20 Create or download a invoice to be validated, e.g. https://www.mustangproject.org/files/MustangGnuaccountingBeispielRE-20201121_508.pdf and save it as factur-x.pdf

#21 Change the method to validateFile and

#22 html escape the validation result, so that the result in the browser looks like screenshot 8:

That's it. Instead of displaying the XML you can now parse it.

Feel free to also try the async functions.

Converting PDF

Converting from PDF to PDF/A, fixing faulty PDF/A and removing file attachments (like Factur-X) are all done with the same Endpoint .../mustang-doc/v1.0.0/mustang/pdf. Please note that Mustangserver-docs is a separate API which needs to be separately be subscribed to.

In depth interactive testing using Postman/Bruno

Bruno is an open source alternative to Postman, graphical user interfaces to create/execute requests on REST APIs.

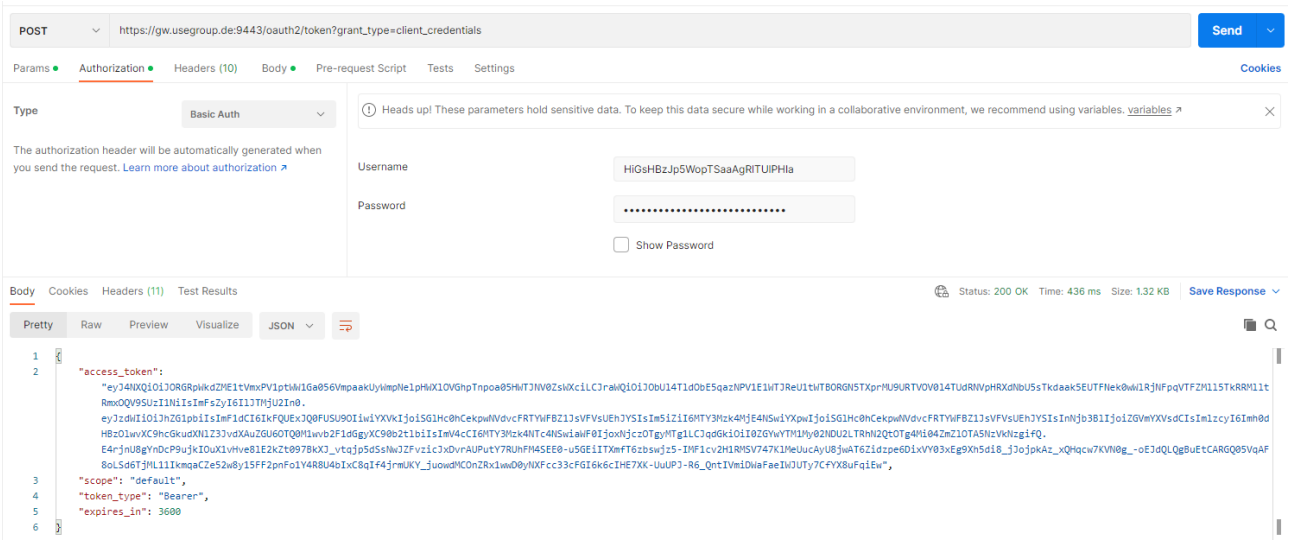
This example is based on Postman, you will need enabled client credentials as described on page 10 in Allowing Client Credentials.

Use Import|File|Upload Files to upload you Openapi.yaml file into a Mustangserver collection.

Add a request to https://gw.usegroup.de:9443/oauth2/token?grant_type=client_credentials and call it Token Request. Postman will auto-detect the Parameter in the URL. Change the type to POST.

In Headers, add a new field Content-Type with application/json as its value.

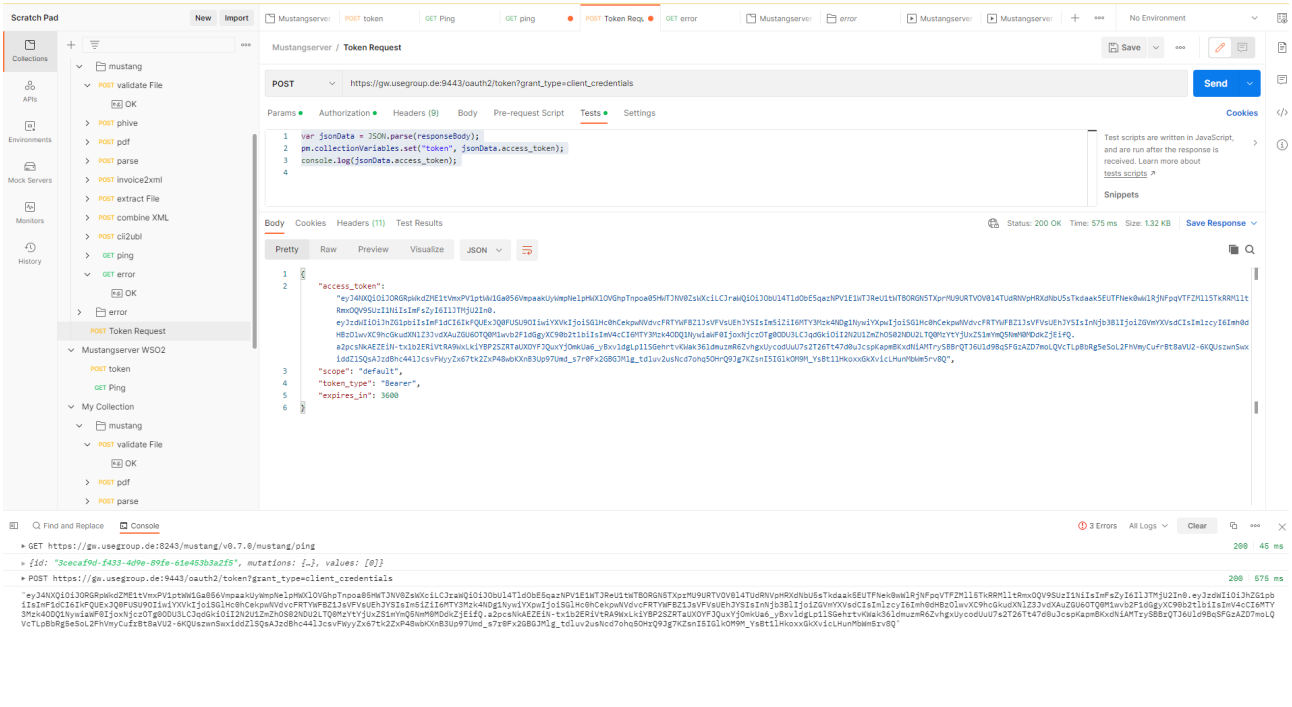
The tab Authorization should be Basic auth with your client id and secret as username and password. Once you click Send, an according access token should be submitted:



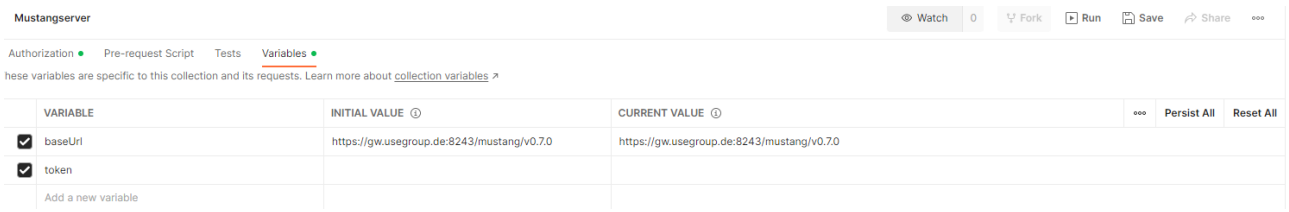
Add

```
var jsonData = JSON.parse(responseBody);
pm.collectionVariables.set("token", jsonData.access_token);
console.log(jsonData.access_token);
```

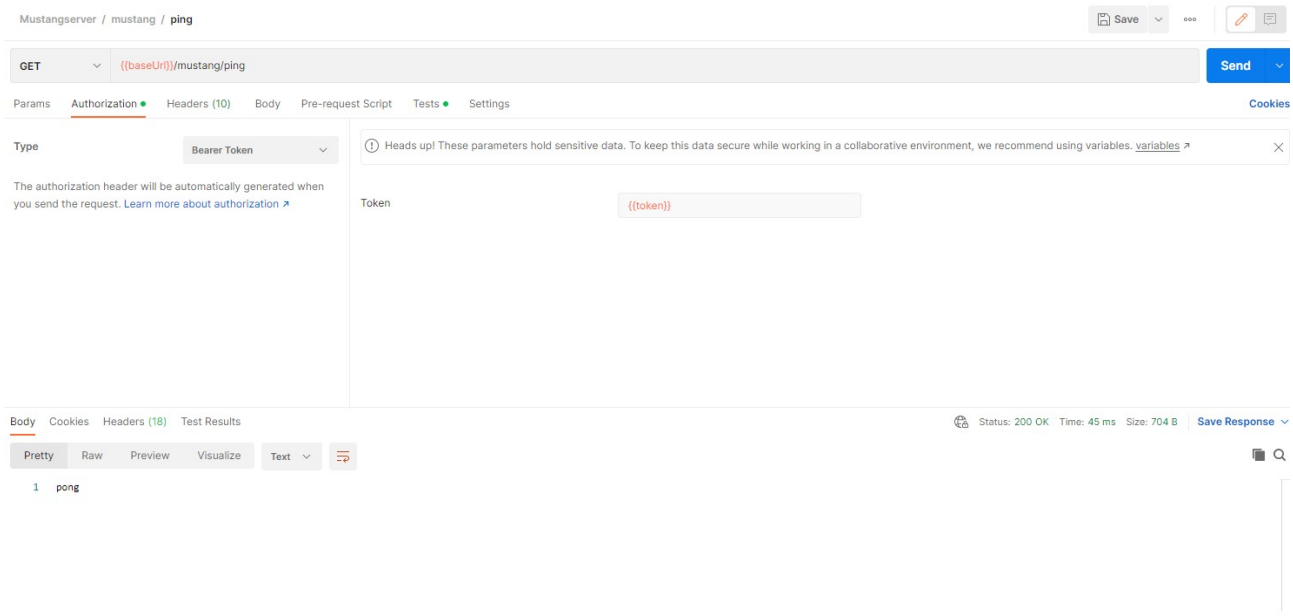
In the „Tests“ tab and View|Show Postman Console (Alt+CTRL+C) . Once you click Send again you should be able to see the access token also in the Console:



In the collection, click on the variables tab and add „token“ as a collection variable.

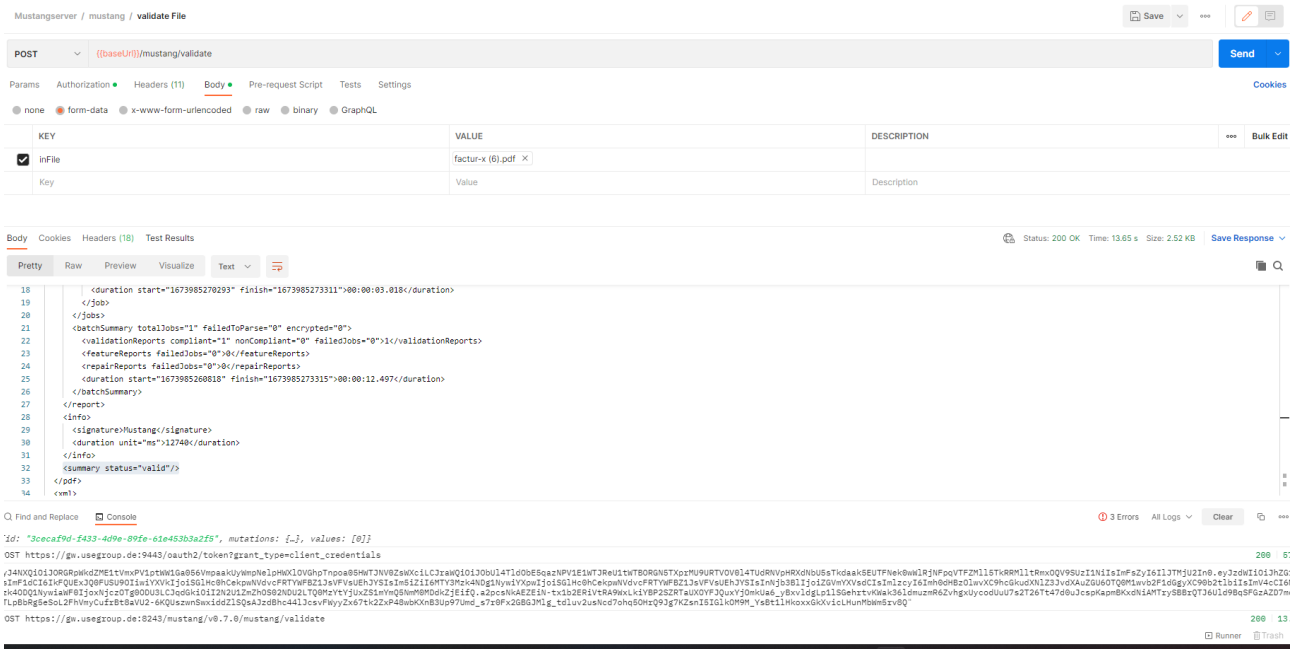


You can now add the variable {{token}} as authorization to any request. The variable will be available and the requests work after you click on „Send“ of the Token Request for the first time.



Please note that ping was answered by pong.

Where appropriate, e.g. in the validation endpoint, Postman will allow you to select files, this being a valid factur-x:



Performance Tests with Jmeter

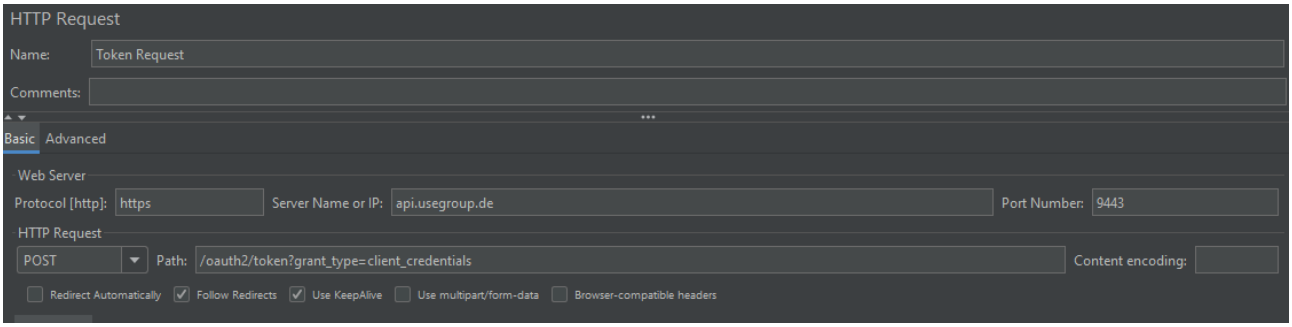
Jmeter is a generic load testing tool and load generator.

If you want to perform load tests, in order not to affect our production servers, we will happily grant you access to our mirror infrastructure, i.e. we will guarantee that the hardware, software and settings are identical.

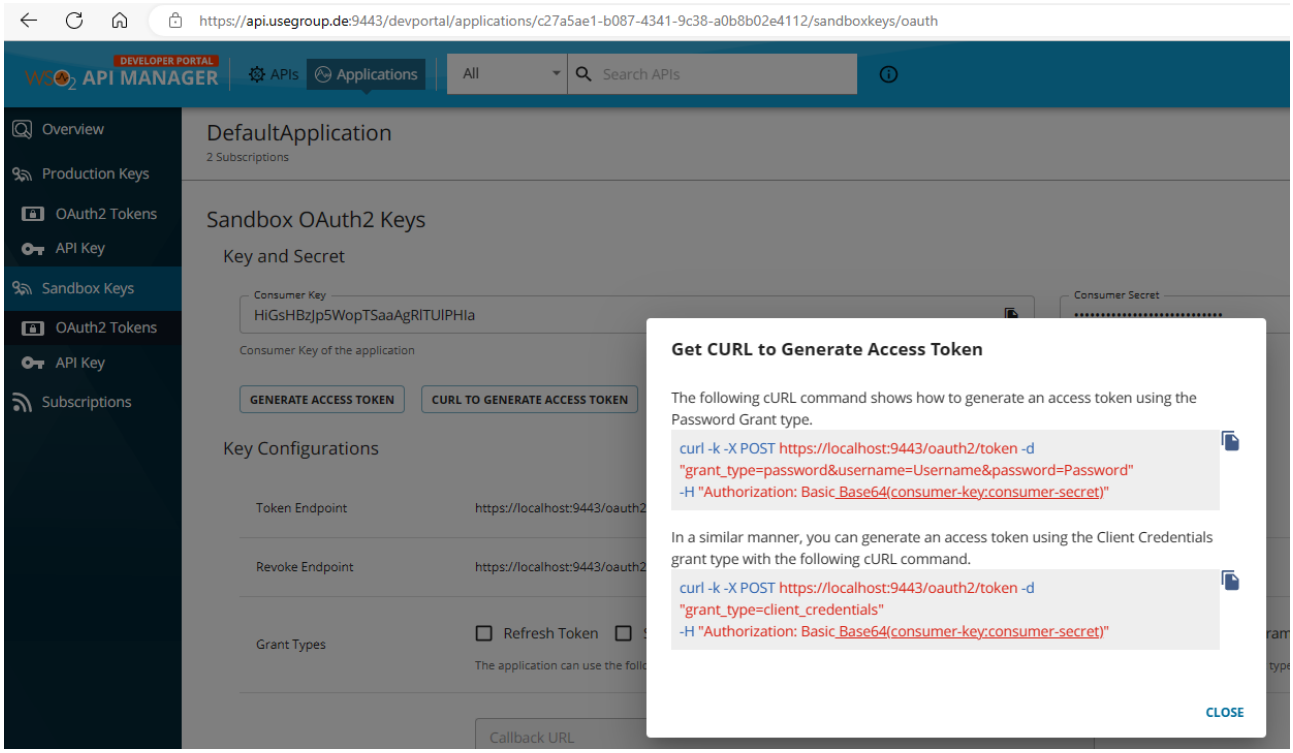
https://openapi-generator.tech/ supports a Jmeter export but that does not handle authentication so here we describe how to set up some Jmeter performance test manually.

For this example, you will need enabled client credentials as described on page 10 in Allowing Client Credentials.

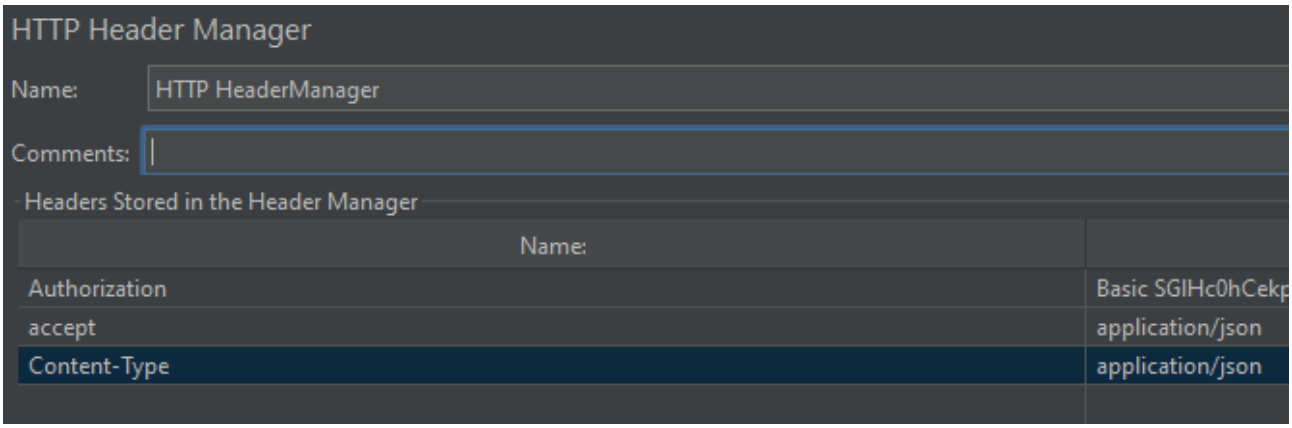
Right click your Test plan, add a Thread Group with a Once Only controller. Below that, add a HTTP request sampler, we'll call it Token Request. This is how it is defined: Change protocol to https, method to POST, add server name and port number, and add the path:



Base64encode your <consumer key>:<consumer secret> as described on the applications page of the API management:



In Jmeter, below the Token request, add a HeaderManager with Basic Authorization as described:

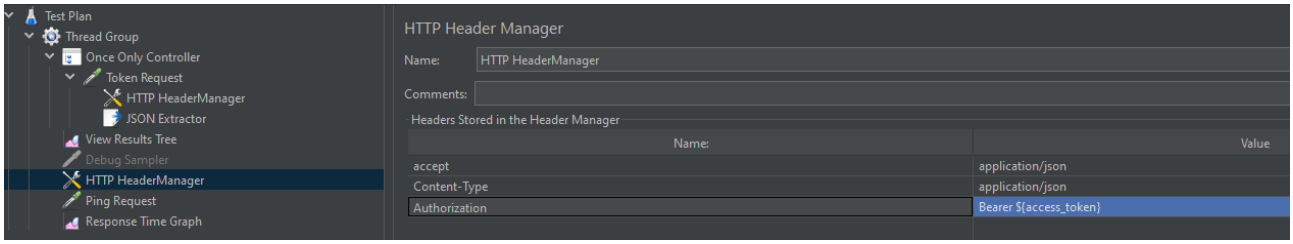


And from the response, also below Token Request, extract the JSON value access token into a Jmeter Variable access token using a JSON Extractor:

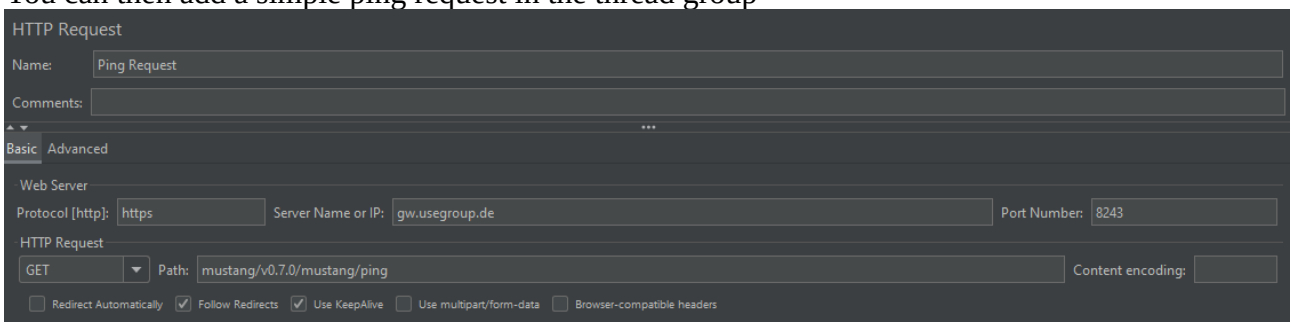
Add a sampler View Result Tree to confirm the results and a debug sampler if you like (in the results tree you will then be able to e.g. see the current variables when you click on the results of the debug sampler).

Run|Start should give you green entries in the results tree.

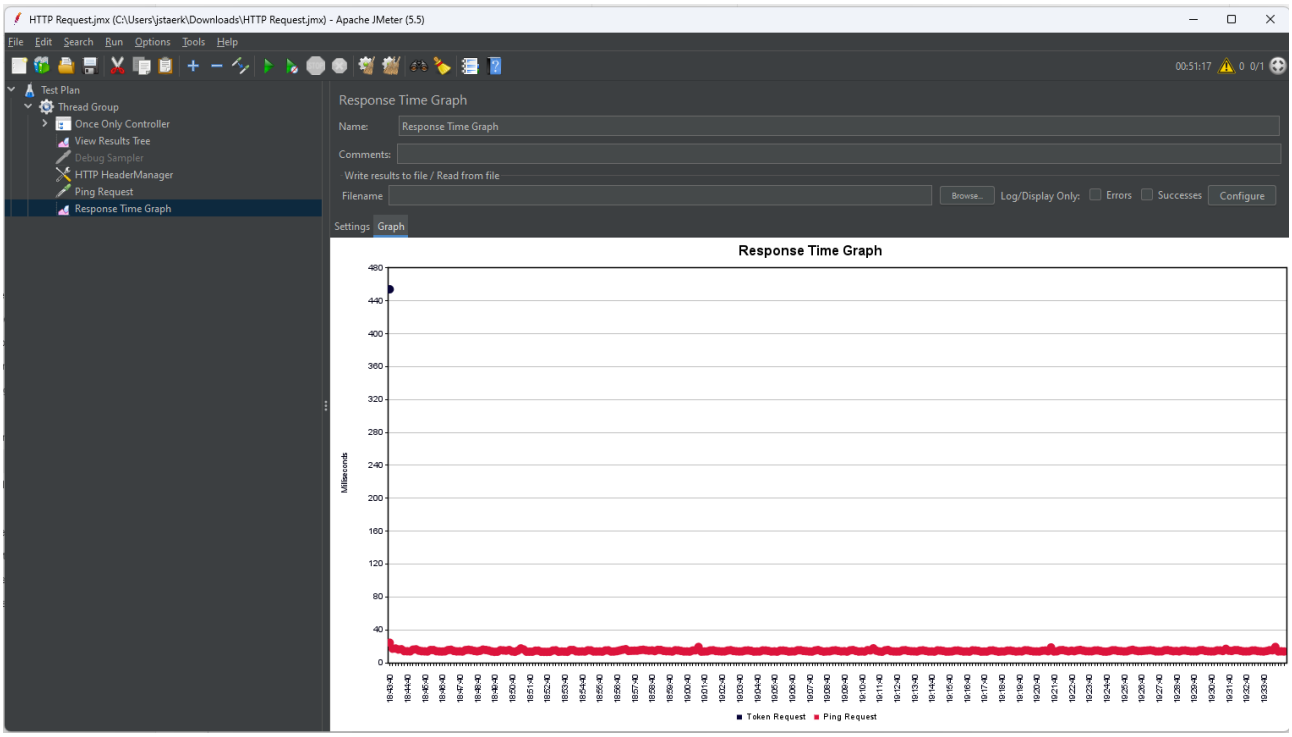
Now we will set the ordinary authentication as header: add another Header Manager outside of the Token request and add the variable as token, i.e. Authorization being Bearer ${access_token}



You can then add a simple ping request in the thread group



and e.g. a Response time graph.You can then set the Thread group loop count to infinite, start the sampling and check the results tree. After a while the response time graph will look like this, indicating the initial login took ~440ms and the usual response time to our „ping“ is ~20ms.



Terms of service

Test terms

To test and evaluate the service a valid email address has to be provided. Unless otherwise agreed (info@usegroup.de) test access is restricted to one account per legal entity, i.e. usually company. This email address will also be used to send availability, information about the roadmap, development and status with an expected maximum volume of one per week. You can terminate your test phase by unsubscribing from the announcements newsletter list. After the signup, access can then happen free of charge, with a limit of 1,000 operations/month, unless access is revoked by usegroup. You are not allowed to share personal data (e.g. real invoice recipient's names, addresses, email addresses, bank credentials or real invoice contents). Access may be revoked because the general test phase has ended, the test phase is over for a certain customer, or due to other terms which do not need to be disclosed. Under this test terms we also do not guarantee the availability nor the correctness of the service.

https://api.usegroup.de:9443/authenticationendpoint/privacy_policy.do

Production terms

To access Mustangserver productively including a data processing agreement a Mustang Pro license is required. Further info can be obtained at https://www.mustangproject.org/pro/

Version history

Of this document:

0.7.0 on 2023-01-19 by Jochen.

0.8.0 on 2023-02-25 by Jochen: Added Mustangserver 0.8.0 (=Order-X)

0.8.1 on 2023-02-26 by Jochen: added C++-Client, PDF/A-param to PDF endpoint

1.0.0 on 2023-09-26 by Jochen: most recent endpoint, neccessity to subscribe, split between mustangserver and mustangserver-docs, updated list of Ves-IDs, added change password url

1.1.0 added username field, exception handling

Of mustang server:

1.1.0 invoice2XML parameter standard was renamed to format. Better Exception handling. CombineXML now also allows PDF/A-3 input.

1.2.0 on 2024-01-31 /combineXML /parse and /invoice2XML now support XRechnung 3.0.1

1.3.0 on 2024-02-29 Endpoint /combine is now available again (combine JSON and PDF to fx), /phive has been updated, now auto-detects Ves-IDs if none specified and now also supports e.g. XR 3.0.1 (from 135 to 143 VesIDs). /xmltohtml and API keys documented. Mentioned Bruno.