

Inhaltsverzeichnis

In case of technical questions.....	2
Server access.....	2
Authentication.....	3
OAuth2.....	3
Api Key.....	4
Endpoint for most-recent API versions.....	5
Errors and Exceptions.....	6
Mediation, Transformation and Orchestration.....	6
Classes.....	6
Invoice class.....	7
Invoice.....	7
CalculatedInvoice.....	11
Operations/Endpoints.....	12
Ping.....	13
Validate.....	13
Phive.....	13
pdf.....	15
parse.....	15
Invoice2xml.....	15
Extract.....	15
detach.....	16
combine.....	16
combineXML.....	16
cii2ubl.....	17
xmltohtml.....	17
xmltopdf.....	17
Example PHP Client.....	17
Mustangserver Hello World.....	17
Preparations.....	17
Allowing Client Credentials.....	19
Get access token.....	20
OAuth2 Authentication.....	21
Validation of electronic invoices.....	27
Example C# client.....	29
Converting PDF.....	30
Interactive testing using Postman/Bruno.....	30
Performance Tests with Jmeter.....	33
Terms of service.....	36
Test terms.....	36
Production terms.....	36
Troubleshooting.....	37
Inhouse variant.....	38
Version history.....	41

In case of technical questions

Please contact

Jochen Stärk

jstaerk@usegroup.de

or

Martin Wachtveitl

mwachtveitl@usegroup.de

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 36.

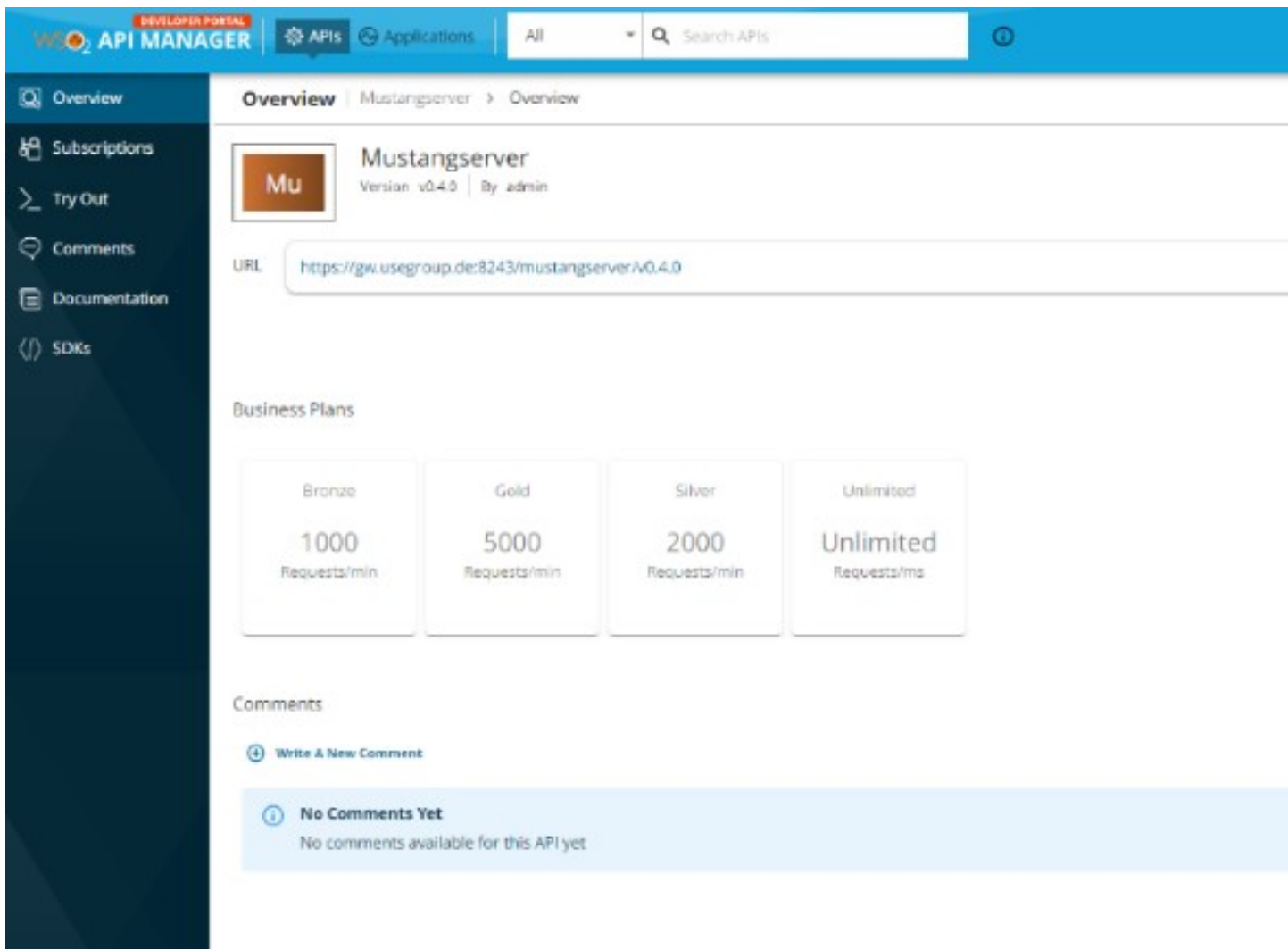
To register please access <https://api.usegroup.de:9443/devportal/services/configs> and click "Create Account" link on the bottom left. Select a username, "Proceed to self register", enter the rest of the data and have your email verified.

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 an 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.

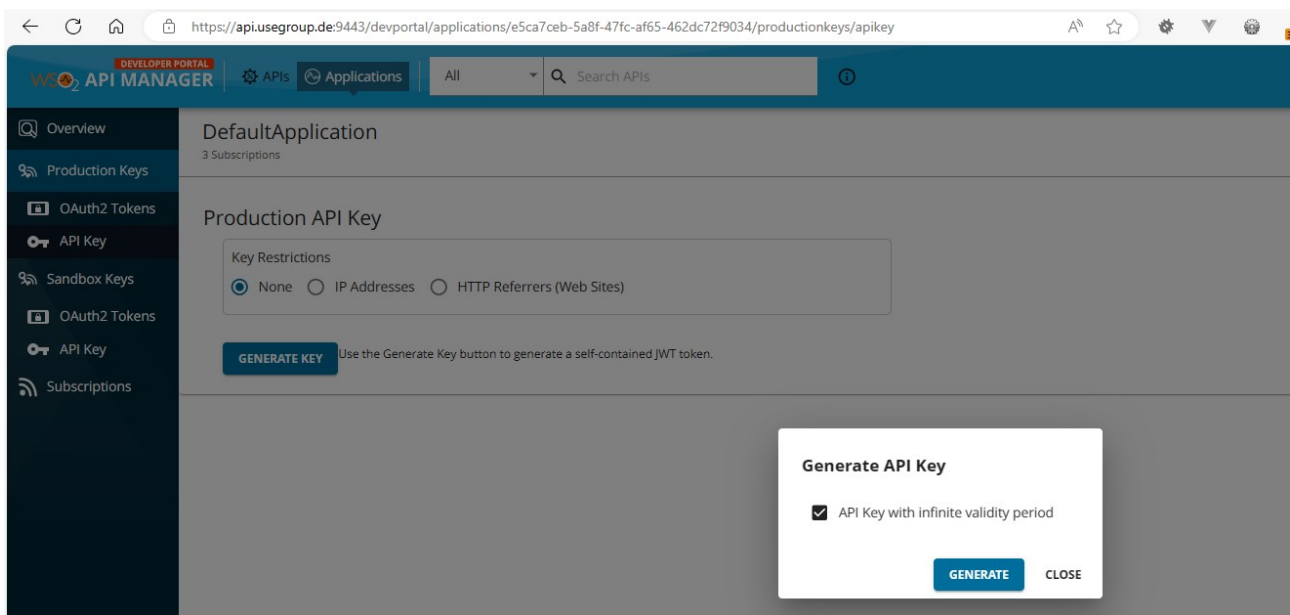
Then you will have to enable client credentials in the applications tab, <https://api.usegroup.de:9443/devportal/applications> Default Application, OAuth2 tokens. The

procedure is described more detailed in the PHP Client chapter „Allowing Client Credentials“ on page 19 but is generic to all examples and does not apply for PHP only.

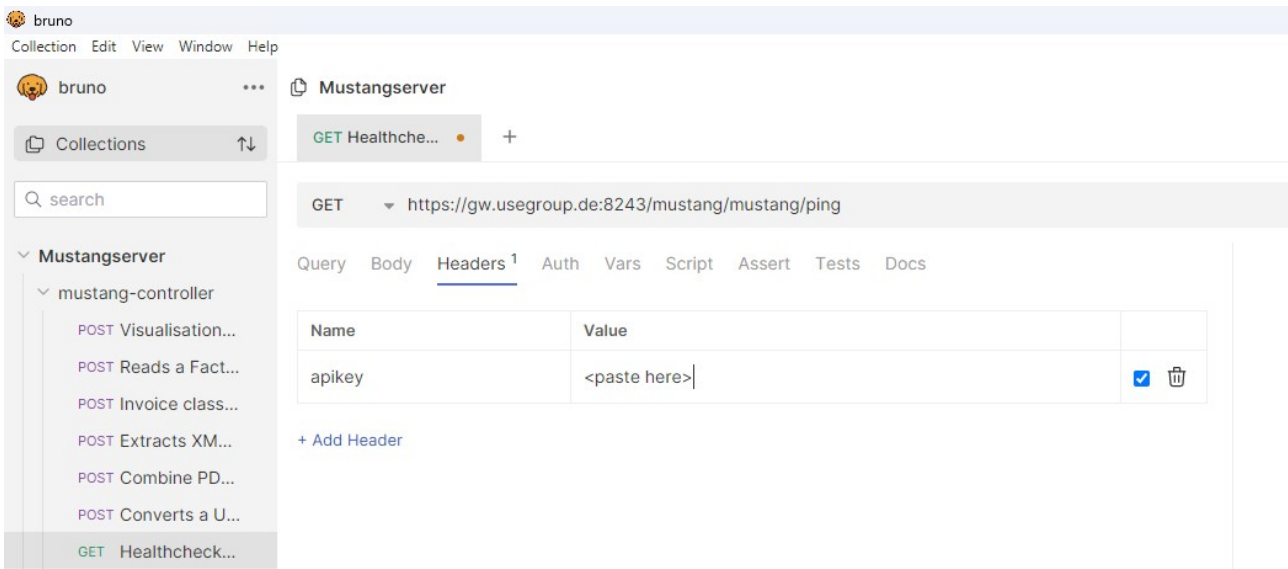
Please note that most clients will use the Mustangserver version which had been *selected* in the backend when downloading the OpenAPI definition. Feel free to replace /mustang/<version>/mustang by as described in Endpoint for most-recent API versions on page 5.

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 30ff)



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();
$gclient=new GuzzleHttp\Client(['headers' => ['apikey' => $apikey]]);

$apiInstance = new Swagger\Client\Api\MustangControllerApi(
    $gclient,
    $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",
  "statusCode": 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

Classes

Mustangserver has two important main classes, invoice and calculatedInvoice. Invoice contains tradeparty classes for recipients and senders and item classes which in turn contain instances of product classes.

The difference between invoice and calculatedInvoice is that the latter contains (redundant, because calculatable) properties like grandTotal. These attributes are provide for courtesy when reading invoices but are *not required* when writing. However, if they are present when writing (e.g. combine or invoice2XML) they will raise an error if the invoice calculation does not match. This can be useful when a PDF file has been generated with another process and the Factur-X-XML is supposed to have the same values, in which case the PDF values can be provided because an error would be more helpful than a silently incorrect Factur-X file because the PDF has been calculated incorrectly, or uses features not yet available in Mustangserver.

Invoice class

Invoice

Minimal example

```
{
  "number": "471102",
  "currency": "EUR",
  "issueDate": "2018-03-04T00:00:00.000+01:00",
  "dueDate": "2018-03-04T00:00:00.000+01:00",
  "deliveryDate": "2018-03-04T00:00:00.000+01:00",
  "sender": {
    "name": "Lieferant GmbH",
    "zip": "80333",
    "street": "Lieferantenstraße 20",
    "location": "München",
    "country": "DE",
    "taxID": "201/113/40209",
    "vatID": "DE123456789",
    "globalID": "4000001123452",
    "globalIDScheme": "0088"
  },
  "recipient": {
    "name": "Kunden AG Mitte",
    "zip": "69876",
    "street": "Kundenstraße 15",
    "location": "Frankfurt",
    "country": "DE"
  },
  "zfitems": [
    {
      "price": 9.9,
      "quantity": 20,
      "product": {
        "unit": "H87",
        "name": "Trennblätter A4",
        "description": "",
        "vatpercent": 19,
        "taxCategoryCode": "S"
      }
    },
    {
      "price": 5.5,
      "quantity": 50,
      "product": {
        "unit": "H87",
        "name": "Joghurt Banane",
        "description": "",
        "vatpercent": 7,
        "taxCategoryCode": "S"
      }
    }
  ]
}
```

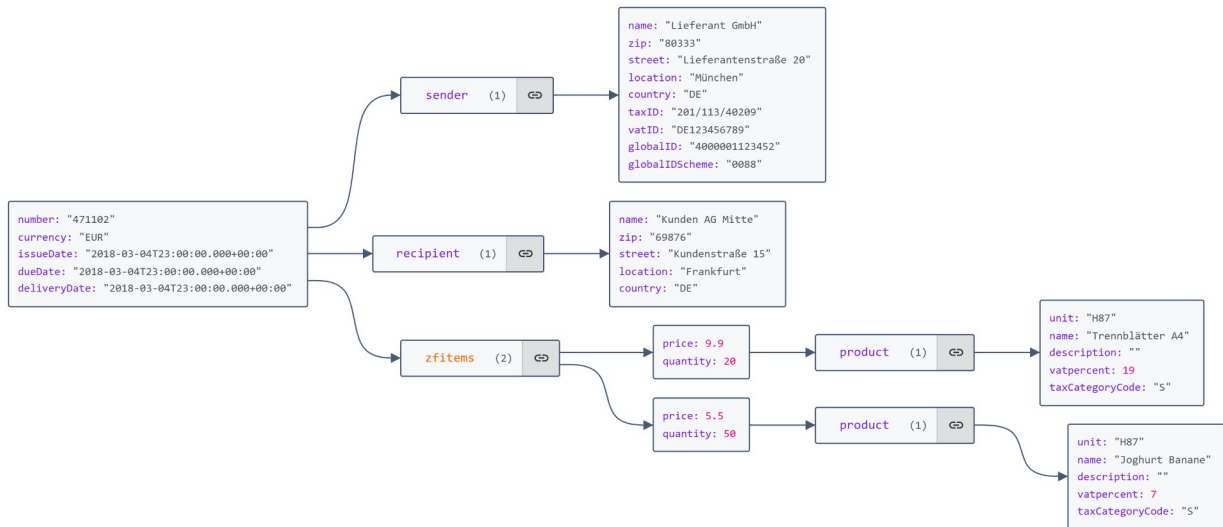


Schaubild 1: Sample JSON structure to write invoices

The following BTs are mapped as follows:

BT ID	JSONPath	CII reading
	recipient	
BT-44	recipient.name	
BT-46	recipient.id	
BT-48	recipient.vatid (also memtioned as recipient.vatID)	
BT-50	recipient.street	
BT-51	recipient.additionalAddress	
BT-163	recipient.additionalAddressExtension	
BT-52	recipient.location	
BT-53	recipient.zip	
BT-55	recipient.country	
ID BT-46	recipient.globalID	
ID Scheme BT-46	recipient.globalIDScheme	
BT-56	recipient.contact.name	
BT-57	recipient.contact.phone	
BT-58	recipient.contact.email	
BT-45	recipient.legalOrganisation.tradingBusinessName	
	sender	
BT-27	sender.name	
BT-28	sender.legalOrganisation.tradingBusinessName	

BT-29	sender.globalID
BT-29-ID	sender.globalIDScheme
BT-31	sender.vatid (also mentioned as sender.vatID)
BT-32	sender.taxID
BT-35	sender.street
BT-36	sender.additionalAddress
BT-162	sender.additionalAddressExtension
BT-38	sender.zip
BT-37	sender.location
BT-40	sender.country
BT-41	sender.contact.name
BT-42	sender.contact.phone
BT-43	sender.contact.email

Invoice

BT-1	number
BT-3	documentCode
BT-5	currency
BT-13	buyerOrderReferencedDocumentID
BT-14	sellerOrderReferencedDocumentID
BT-16	despatchAdviceReferencedDocumentID
BT-10	referenceNumber

Allowance at invoice level

BT-92	zfallowances.totalAmount
BT-95	zfallowances.categoryCode
BT-96	zfallowances.taxPercent
BT-97	zfallowances.reason
BT-98	zfallowances.reasonCode

Charge amount at invoice level

BT-102	zfcharges.categoryCode
BT-99	zfcharges.totalAmount
BT-103	zfcharges.taxPercent
BT-104	zfcharges.reason

BT-105 zfcharges.reasonCode

Payment Details & Invoice Comment

BT-9 dueDate
BT-84 tradeSettlement.iban

Price Details

BT-129 zfitems.quantity
BT-130 zfitems.product.unit
BT-146 zfitems.price
BT-151 zfitems.product.taxCategoryCode

BT-152 zfitems.product.vatpercent

BT-132 zfitems.buyerOrderReferencedDocumentLineID

BT-153 zfitems.product.name
BT-155 zfitems.product.sellerAssignedID

BT-157 zfitems.product.globalID

BT-157-ID zfitems.product.globalIDScheme

BT-128 zfitems.additionalReferences.issuerAssignedID

Allowance at line level

BT-137 Missing – basic amount

BT-136 zfallowances.totalAmount

BT-138 Missing – percentage
BT-139 zfallowances.reason
BT-140 zfallowances.reasonCode

Charges at invoice inline level

BT-142 Missing – basic amount

BT-143 Missing – percentage
BT-141 zfcharges.totalAmount
BT-144 zfcharges.reason
BT-145 zfcharges.reasonCode

CalculatedInvoice

E.g. https://www.mustangproject.org/files/MustangGnuaccountingBeispielRE-20201121_508.pdf

parses as

```
{ "documentName": null, "documentCode": "380", "number": "RE-20201121/508",
"ownOrganisationFullPlaintextInfo": null, "referenceNumber": "AB321", "shipToOrganisationID": null,
"shipToOrganisationName": null, "shipToStreet": null, "shipToZIP": null, "shipToLocation": null,
"shipToCountry": null, "buyerOrderReferencedDocumentID": null, "invoiceReferencedDocumentID": null,
"buyerOrderReferencedDocumentIssueDateTime": null, "ownForeignOrganisationID": null,
"ownOrganisationName": "Bei Spiel GmbH", "currency": "EUR", "paymentTermDescription": null,
"issueDate": "2020-11-20T23:00:00.000+00:00", "dueDate": "2020-12-11T23:00:00.000+00:00",
"deliveryDate": "2020-11-09T23:00:00.000+00:00", "sender": { "name": "Bei Spiel GmbH", "zip":
"12345", "street": "Ecke 12", "location": "Stadthausen", "country": "DE", "taxID": null, "vatID":
"DE136695976", "additionalAddress": null, "additionalAddressExtension": null, "bankDetails":
[ { "accountName": null, "bic": null, "iban": "DE88200800000970375700" } ] }, "contact": null,
"legalOrganisation": null, "uriUniversalCommunicationID": null, "uriUniversalCommunicationIDScheme":
null, "globalID": null, "globalIDScheme": null, "vatic": "DE136695976", "id": null, "email": null,
"asTradeSettlement": [ { "accountName": null, "bic": null, "iban": "DE88200800000970375700" } ] },
"recipient": { "name": "Theodor Est", "zip": "88802", "street": "Bahnstr. 42", "location":
"Spielkreis", "country": "DE", "taxID": null, "vatID": null, "additionalAddress": null,
"additionalAddressExtension": null, "bankDetails": [], "contact": null, "legalOrganisation": null,
"uriUniversalCommunicationID": null, "uriUniversalCommunicationIDScheme": null, "globalID": null,
"globalIDScheme": null, "vatic": null, "id": "2", "email": null, "asTradeSettlement": null },
"deliveryAddress": null, "cashDiscounts": [], "notes": null, "sellerOrderReferencedDocumentID":
null, "contractReferencedDocument": null, "totalPrepaidAmount": null, "paymentTerms": null,
"invoiceReferencedIssueDate": null, "specifiedProcuringProjectID": null,
"specifiedProcuringProjectName": null, "despatchAdviceReferencedDocumentID": null,
"creditorReferenceID": null, "grandTotal": 571.04, "valid": false, "ownStreet": "Ecke 12", "ownZIP":
"12345", "zfallowances": null, "zfitems": [ { "price": 160, "quantity": 1, "tax": null,
"grossPrice": null, "lineTotalAmount": null, "basisQuantity": 1, "detailedDeliveryPeriodFrom": null,
"detailedDeliveryPeriodTo": null, "id": null, "product": { "unit": "HUR", "name": "Design (hours)",
"sellerAssignedID": null, "buyerAssignedID": null, "description": "", "countryOfOrigin": null,
"attributes": null, "intraCommunitySupply": false, "vatpercent": 7, "globalID": null,
"globalIDScheme": null, "reverseCharge": false, "taxCategoryCode": "S", "taxExemptionReason":
null }, "notes": null, "referencedDocuments": null, "additionalReferences": null,
"buyerOrderReferencedDocumentLineID": null, "itemAllowances": null, "itemCharges": null,
"itemTotalAllowances": null, "additionalReferencedDocumentID": null, "value": 160 }, { "price":
0.79, "quantity": 400, "tax": null, "grossPrice": null, "lineTotalAmount": null, "basisQuantity": 1,
"detailedDeliveryPeriodFrom": null, "detailedDeliveryPeriodTo": null, "id": null, "product":
{ "unit": "H87", "name": "Ballons", "sellerAssignedID": null, "buyerAssignedID": null,
"description": "", "countryOfOrigin": null, "attributes": null, "intraCommunitySupply": false,
"vatpercent": 19, "globalID": null, "globalIDScheme": null, "reverseCharge": false,
"taxCategoryCode": "S", "taxExemptionReason": null }, "notes": null, "referencedDocuments": null,
"additionalReferences": null, "buyerOrderReferencedDocumentLineID": null, "itemAllowances": null,
"itemCharges": null, "itemTotalAllowances": null, "additionalReferencedDocumentID": null, "value":
0.79 }, { "price": 0.025, "quantity": 800, "tax": null, "grossPrice": null, "lineTotalAmount": null,
"basisQuantity": 1, "detailedDeliveryPeriodFrom": null, "detailedDeliveryPeriodTo": null, "id":
null, "product": { "unit": "LTR", "name": "Hot air ,heiÙe Luft" (litres)", "sellerAssignedID": null,
"buyerAssignedID": null, "description": "", "countryOfOrigin": null, "attributes": null,
"intraCommunitySupply": false, "vatpercent": 19, "globalID": null, "globalIDScheme": null,
"reverseCharge": false, "taxCategoryCode": "S", "taxExemptionReason": null }, "notes": null,
"referencedDocuments": null, "additionalReferences": null, "buyerOrderReferencedDocumentLineID":
null, "itemAllowances": null, "itemCharges": null, "itemTotalAllowances": null,
"additionalReferencedDocumentID": null, "value": 0.025 } ], "ownTaxID": null, "tradeSettlement": [ {
"accountName": null, "bic": null, "iban": "DE88200800000970375700" } ], "ownVATID": "DE136695976",
"ownLocation": "Stadthausen", "ownCountry": "DE", "zfccharges": null, "detailedDeliveryPeriodTo":
null, "notesWithSubjectCode": null, "detailedDeliveryPeriodFrom": null, "vatdueDateTypeCode": null,
"zflogisticsServiceCharges": null, "additionalReferencedDocuments": null, "subjectNote": null,
"tradeSettlementPayment": null }
```

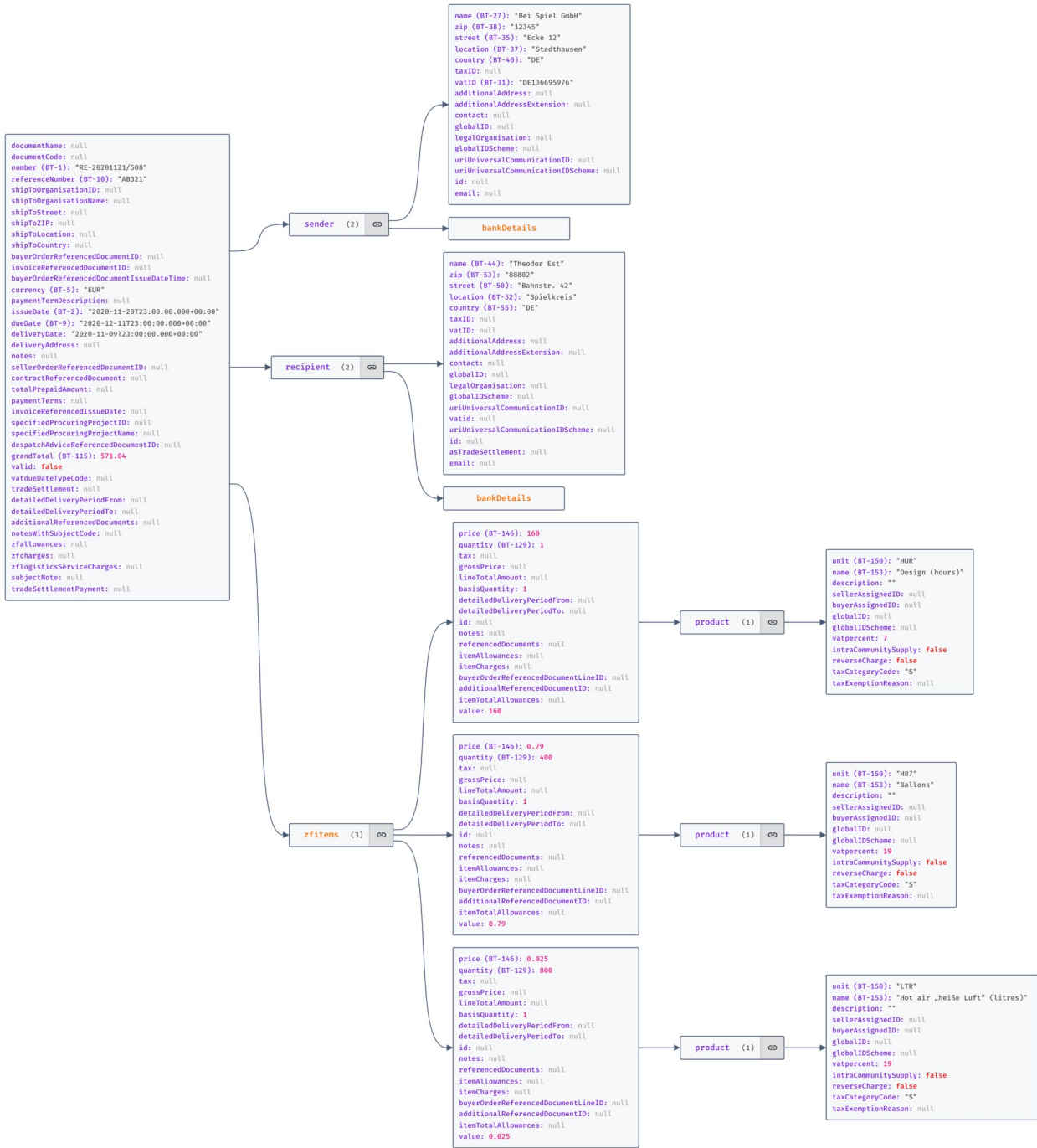


Schaubild 2: Sample read invoice JSON structure

Operations/Endpoints

Mustangserver's available operations are

Ping

Just a test, always just returns „pong“. The only operation accessible 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 its guideline ID.

Phive

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

The available 173 format/standard/version-combinations (VES IDs) are

de.xrechnung:cii:1.2.0	de.xrechnung:ubl-invoice:3.0.2
de.xrechnung:cii:1.2.1	es.gob:facturae:3.0.0
de.xrechnung:cii:1.2.2	es.gob:facturae:3.1.0
de.xrechnung:cii:2.0.0	es.gob:facturae:3.2.0
de.xrechnung:cii:2.0.1	es.gob:facturae:3.2.1
de.xrechnung:cii:2.1.1	es.gob:facturae:3.2.2
de.xrechnung:cii:2.2.0	eu.cen.en16931:cii:1.0.0
de.xrechnung:cii:2.3.1	eu.cen.en16931:cii:1.1.0
de.xrechnung:cii:3.0.0	eu.cen.en16931:cii:1.2.0
de.xrechnung:cii:3.0.1	eu.cen.en16931:cii:1.2.1
de.xrechnung:cii:3.0.2	eu.cen.en16931:cii:1.2.3
de.xrechnung:ubl-creditnote:1.2.0	eu.cen.en16931:cii:1.3.0
de.xrechnung:ubl-creditnote:1.2.1	eu.cen.en16931:cii:1.3.1
de.xrechnung:ubl-creditnote:1.2.2	eu.cen.en16931:cii:1.3.10
de.xrechnung:ubl-creditnote:2.0.0	eu.cen.en16931:cii:1.3.11
de.xrechnung:ubl-creditnote:2.0.1	eu.cen.en16931:cii:1.3.12
de.xrechnung:ubl-creditnote:2.1.1	eu.cen.en16931:cii:1.3.2
de.xrechnung:ubl-creditnote:2.2.0	eu.cen.en16931:cii:1.3.3
de.xrechnung:ubl-creditnote:2.3.1	eu.cen.en16931:cii:1.3.4
de.xrechnung:ubl-creditnote:3.0.0	eu.cen.en16931:cii:1.3.5
de.xrechnung:ubl-creditnote:3.0.1	eu.cen.en16931:cii:1.3.6
de.xrechnung:ubl-creditnote:3.0.2	eu.cen.en16931:cii:1.3.6; qualifier=a
de.xrechnung:ubl-invoice:1.2.0	eu.cen.en16931:cii:1.3.7
de.xrechnung:ubl-invoice:1.2.1	eu.cen.en16931:cii:1.3.8
de.xrechnung:ubl-invoice:1.2.2	eu.cen.en16931:cii:1.3.9
de.xrechnung:ubl-invoice:2.0.0	eu.cen.en16931:ubl-creditnote:1.0.0
de.xrechnung:ubl-invoice:2.0.1	eu.cen.en16931:ubl-creditnote:1.1.0
de.xrechnung:ubl-invoice:2.1.1	eu.cen.en16931:ubl-creditnote:1.2.0
de.xrechnung:ubl-invoice:2.2.0	eu.cen.en16931:ubl-creditnote:1.2.1
de.xrechnung:ubl-invoice:2.3.1	eu.cen.en16931:ubl-creditnote:1.2.3
de.xrechnung:ubl-invoice:3.0.0	eu.cen.en16931:ubl-creditnote:1.3.0
de.xrechnung:ubl-invoice:3.0.1	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:1.3.12
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:1.3.6; qualifier=a
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:1.3.12
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:1.3.6; qualifier=a
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:1.0.11
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:1.0.11
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:1.0.11
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:1.0.11
eu.peppol.bis3.aunz.ubl:invoice:1.0.9
eu.peppol.bis3.sg.ubl:creditnote:2023.12.0
eu.peppol.bis3.sg.ubl:creditnote:2023.7.0
eu.peppol.bis3.sg.ubl:invoice:2023.12.0
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-response:2024.5.0
eu.peppol.bis3:catalogue:2023.11.0
eu.peppol.bis3:catalogue:2023.5.0
eu.peppol.bis3:catalogue:2024.5.0
eu.peppol.bis3:creditnote:2023.11.0
eu.peppol.bis3:creditnote:2023.5.0
eu.peppol.bis3:creditnote:2024.5.0
eu.peppol.bis3:despatch-advice:2023.11.0
eu.peppol.bis3:despatch-advice:2023.5.0
eu.peppol.bis3:despatch-advice:2024.5.0
eu.peppol.bis3:invoice-message-response:2023.11.0
eu.peppol.bis3:invoice-message-response:2023.5.0
eu.peppol.bis3:invoice-message-response:2024.5.0
eu.peppol.bis3:invoice:2023.11.0
eu.peppol.bis3:invoice:2023.5.0
eu.peppol.bis3:invoice:2024.5.0
eu.peppol.bis3:mlr:2023.11.0
eu.peppol.bis3:mlr:2023.5.0
eu.peppol.bis3:mlr:2024.5.0
eu.peppol.bis3:order-agreement:2023.11.0
eu.peppol.bis3:order-agreement:2023.5.0
eu.peppol.bis3:order-agreement:2024.5.0
eu.peppol.bis3:order-cancellation:2023.11.0
eu.peppol.bis3:order-cancellation:2023.5.0
eu.peppol.bis3:order-cancellation:2024.5.0
eu.peppol.bis3:order-change:2023.11.0
eu.peppol.bis3:order-change:2023.5.0
eu.peppol.bis3:order-change:2024.5.0
eu.peppol.bis3:order-response-advanced:2023.11.0
eu.peppol.bis3:order-response-advanced:2023.5.0
eu.peppol.bis3:order-response-advanced:2024.5.0
eu.peppol.bis3:order-response:2023.11.0
eu.peppol.bis3:order-response:2023.5.0
eu.peppol.bis3:order-response:2024.5.0
eu.peppol.bis3:order:2023.11.0
eu.peppol.bis3:order:2023.5.0
eu.peppol.bis3:order:2024.5.0
eu.peppol.bis3:punch-out:2023.11.0
eu.peppol.bis3:punch-out:2023.5.0

eu.peppol.bis3:punch-out:2024.5.0
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:1.1.4
eu.peppol.reporting:eusr:1.1.5
eu.peppol.reporting:tsr:1.0.4
eu.peppol.reporting:tsr:1.0.5
it.fatturapa:invoice:1.2.0
it.fatturapa:invoice:1.2.1
it.fatturapa:invoice:1.2.2

org.peppol.jp.pint:credit-note:0.1.2
org.peppol.jp.pint:invoice:0.1.2
org.peppol.pint.my:creditnote-self-billing:1.0.0
org.peppol.pint.my:creditnote:1.0.0
org.peppol.pint.my:invoice-self-billing:1.0.0
org.peppol.pint.my:invoice:1.0.0
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 PDFVersion. 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. PDFVersion 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 , Order-X or Xrechnung-file.

Will return a `CalculatedInvoice`, i.e. a `grandTotal` will be available. If it is *set* when writing (`invoice2XML` and `combineInvoice`), it will be compared vis á vis the calculated items and an exception will be thrown if the values do not match. In particular when writing a Factur-X PDF this should be used to ensure the machine readable XML-values match the human readable PDF values.

The object will look as described for `CalculatedInvoice` example on page 11.

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".

Please refer to the documentation of the Invoice class on page 7

Extract

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

detach

Parameter: file

Extracts all file attachments from the PDF (including e.g. a factur-x.xml) and the XML, if the invoice has attachments, and returns a JSON structure with base64-encoded contents like this:

```
{
  "pdf": [],
  "xml": [ { "aFileA.png": "iVBORw0KGgoAAAANSU...
xhBQAAACBjSFJNAAB6JgAAGIQAAPoAAACA6AAAdTAAAOpgAAA6mAAAF3CculE8AAAAAmJLR0QA/
4ePzL8AAAAJcEhZcwAACxMAAAAsTAQCanBgAAAAHdElnRQfkaQIXGQ0qsHJfAAAAo01EQVRIx+3MoQrCUBjF8f+9dzBtgh
a72eAYLJgMQ4tvMREfQuw+wGA2mxarxTeYgi9gEwyDNAOfYTrM5u+kw4/
DcQAwi4PO+Q5gK2AsUVW+UEdBQUFBQUFB4W9IQhmRL3oABpKppLaLmEIm2cXMR6897c++cEM3WBPUBw2eVrzfR/
Gtd6Cs4SFH11/
DFn18oDTLbOcgPwWFacpNVpstvAG3bSYVfhBdGAAACV0RVh0ZGF0ZTpjcmVhdGUAMjAyMC0wMS0wM1QyMzoyNToxMysw
MDowMEN9AywAAAAIdEVYdGRhdGU6bW9kaWZ5ADlwMjAtMDRlMDU6MjU6MTMrMDA6MDAyILuQAAAAAE1FTkSuQmCC"
}, { "sameFileB.png": "iVBORw0KGgoAAAANSU...
xhBQAAACBjSFJNAAB6JgAAGIQAAPoAAACA6AAAdTAAAOpgAAA6mAAAF3CculE8AAAAAmJLR0QA/
4ePzL8AAAAJcEhZcwAACxMAAAAsTAQCanBgAAAAHdElnRQfkaQIXGQ0qsHJfAAAAo01EQVRIx+3MoQrCUBjF8f+9dzBtgh
a72eAYLJgMQ4tvMREfQuw+wGA2mxarxTeYgi9gEwyDNAOfYTrM5u+kw4/
DcQAwi4PO+Q5gK2AsUVW+UEdBQUFBQUFB4W9IQhmRL3oABpKppLaLmEIm2cXMR6897c++cEM3WBPUBw2eVrzfR/
Gtd6Cs4SFH11/
DFn18oDTLbOcgPwWFacpNVpstvAG3bSYVfhBdGAAACV0RVh0ZGF0ZTpjcmVhdGUAMjAyMC0wMS0wM1QyMzoyNToxMysw
MDowMEN9AywAAAAIdEVYdGRhdGU6bW9kaWZ5ADlwMjAtMDRlMDU6MjU6MTMrMDA6MDAyILuQAAAAAE1FTkSuQmCC"
} ]
}
```

The input file can be a PDF/A-3, or a XML. Please note a PDF/A-3 may have both pdf and xml attachments at the same time (embedded within the xml which is embedded in the PDF).

combine

Combines a JSON encoded invoice object (as described for „parse“, pg 15) 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").

If returnJSON (optional) is true (default false) the return value will be a JSON whose key „pdf“ is base64 encoded. If ignorePDFAEerrors (optional) is true (default false) the PDFbox pre-validation will raise no exceptions if the input PDF/A file is invalid.

The attribute grandTotal will be calculated by multiplying item quantities with their prices, adding the lines, adding charges and removing allowances, and adding the calculated VAT amounts.

Sample for writing, e.g. Invoice2XML: Please refer to the documentation of the Invoice class on page 7

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").

If returnJSON (optional) is true (default false) the return value will be a JSON whose key „pdf“ is base64 encoded. If ignorePDFAEerrors (optional) is true (default false) the PDFbox pre-validation will raise no exceptions if the input PDF/A file is invalid.

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 specified 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](#)

xmltopdf

converts a UBL or CII XML file (parameter file) into a human readable PDF/A-3 in german language. The PDF will not have file attachments, i.e. you still have to combineXML if you want to get a Factur-X/ZUGFeRD file from a XML.

Example PHP Client

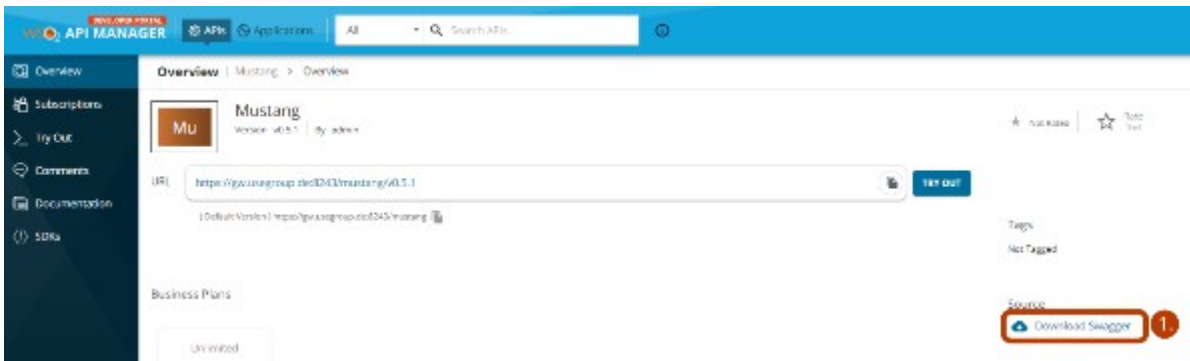
This example operates in a PHP context but <https://editor.swagger.io/> also allows

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.

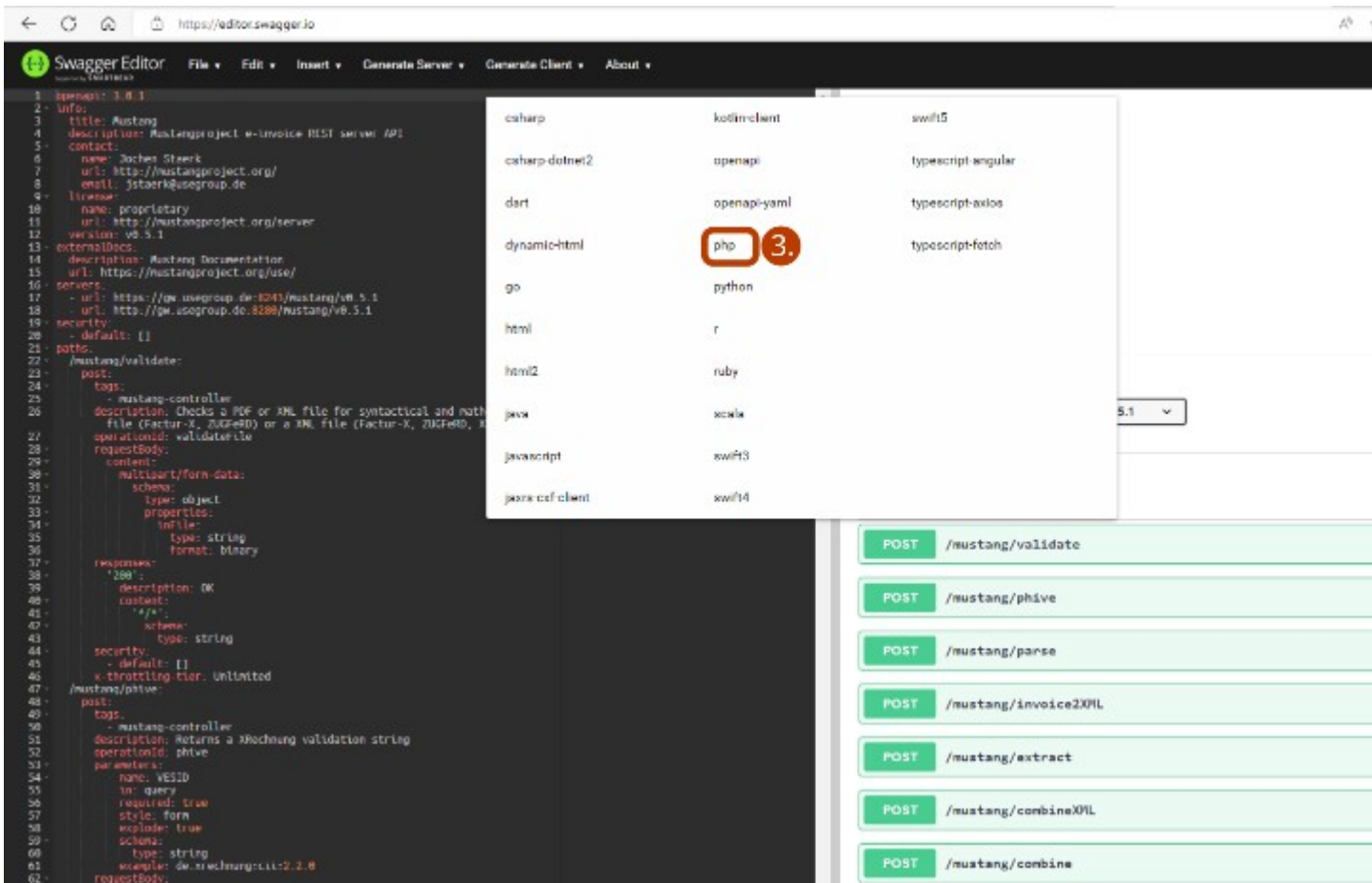
However, it is possible: Swagger editor is open source under the APL license

(<https://github.com/swagger-api/swagger-editor>) and e.g. a Docker Image can be 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.

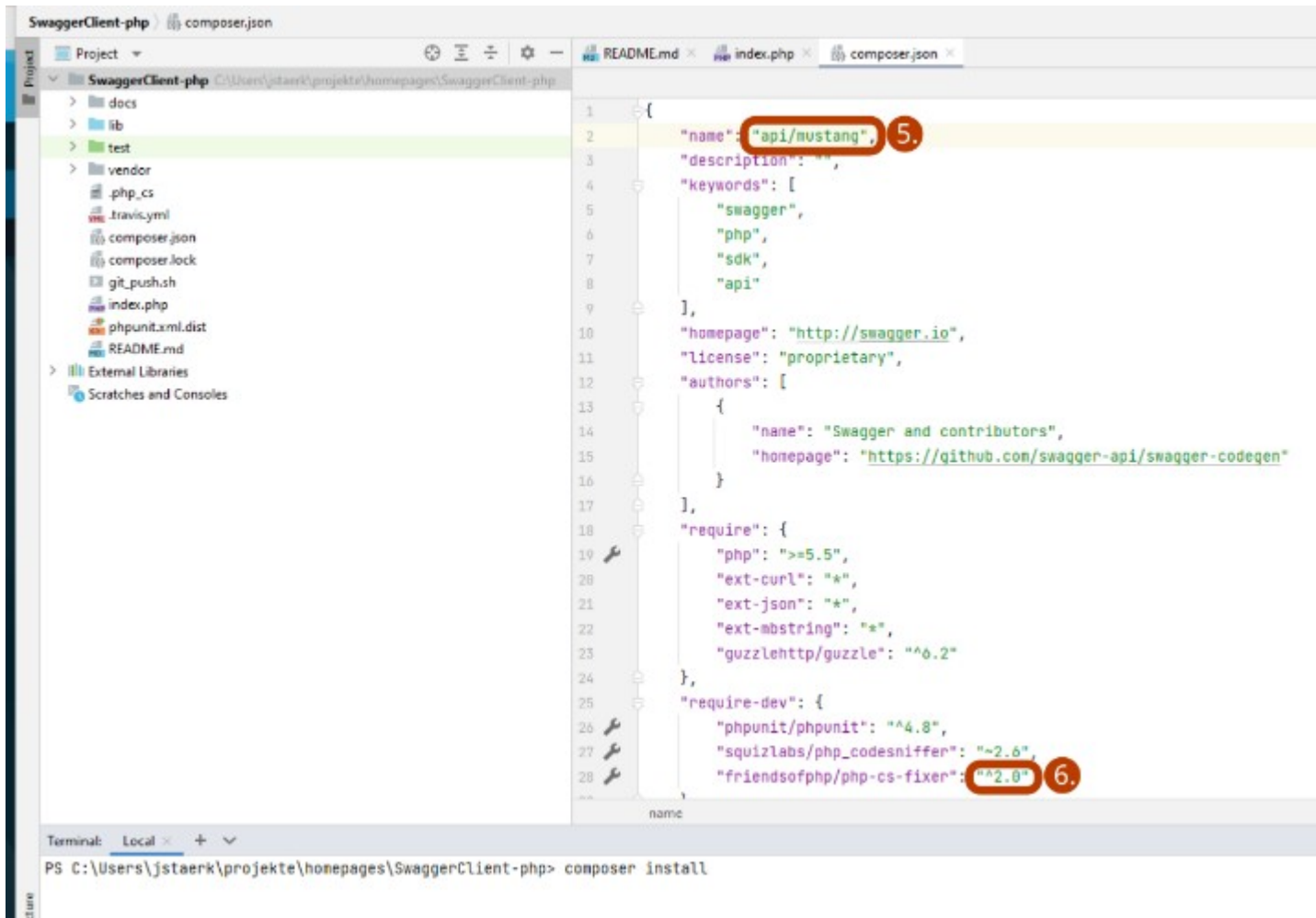
Screenshot 2:



#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:

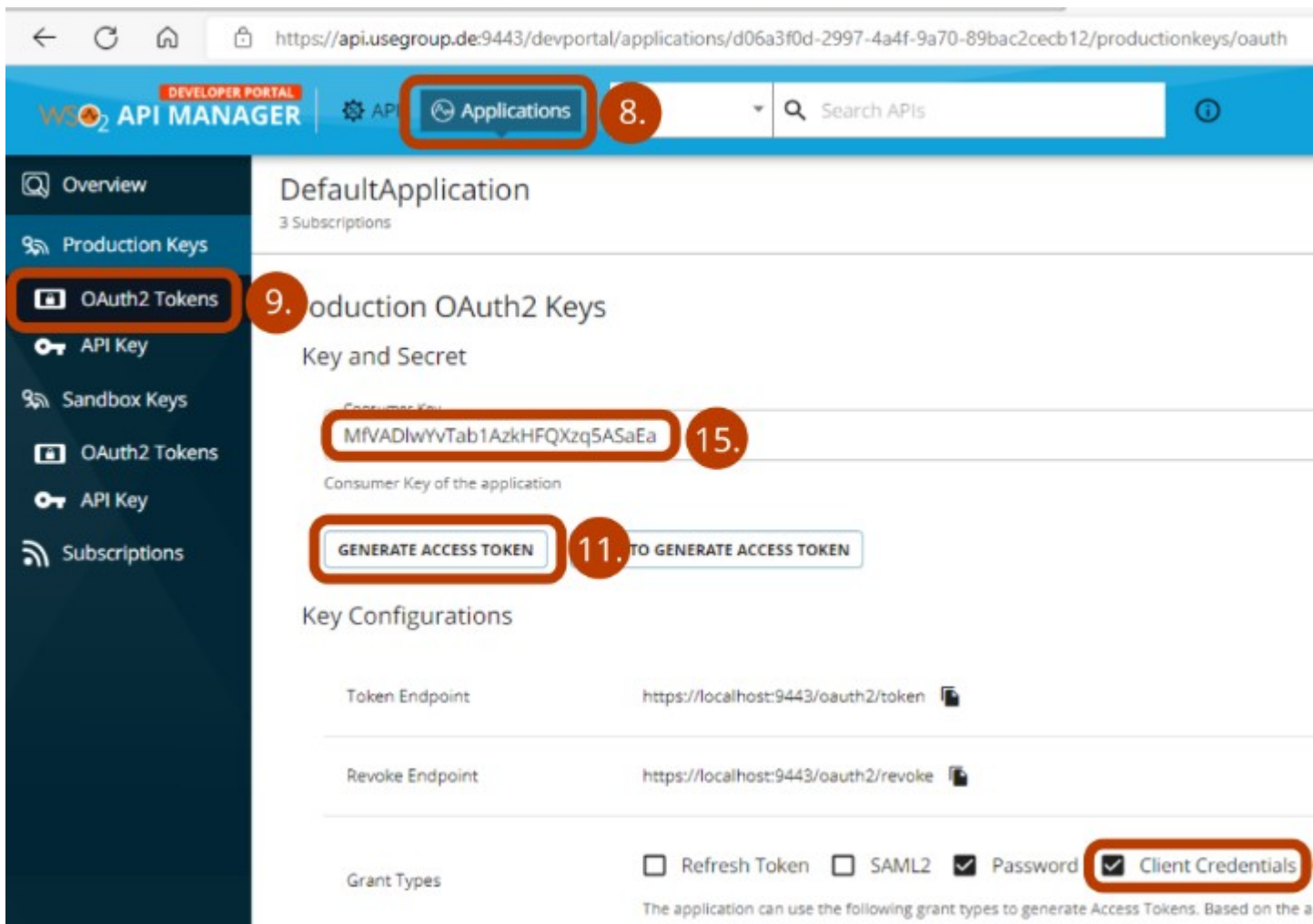


#6. If you want to use PHP8+ upgrade the version number of php-cs-fixer to ^2.0. Then run "composer install" in that directory.

#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 authentication: Check Client Credentials and click the Update button on the bottom of the page.

Get access token

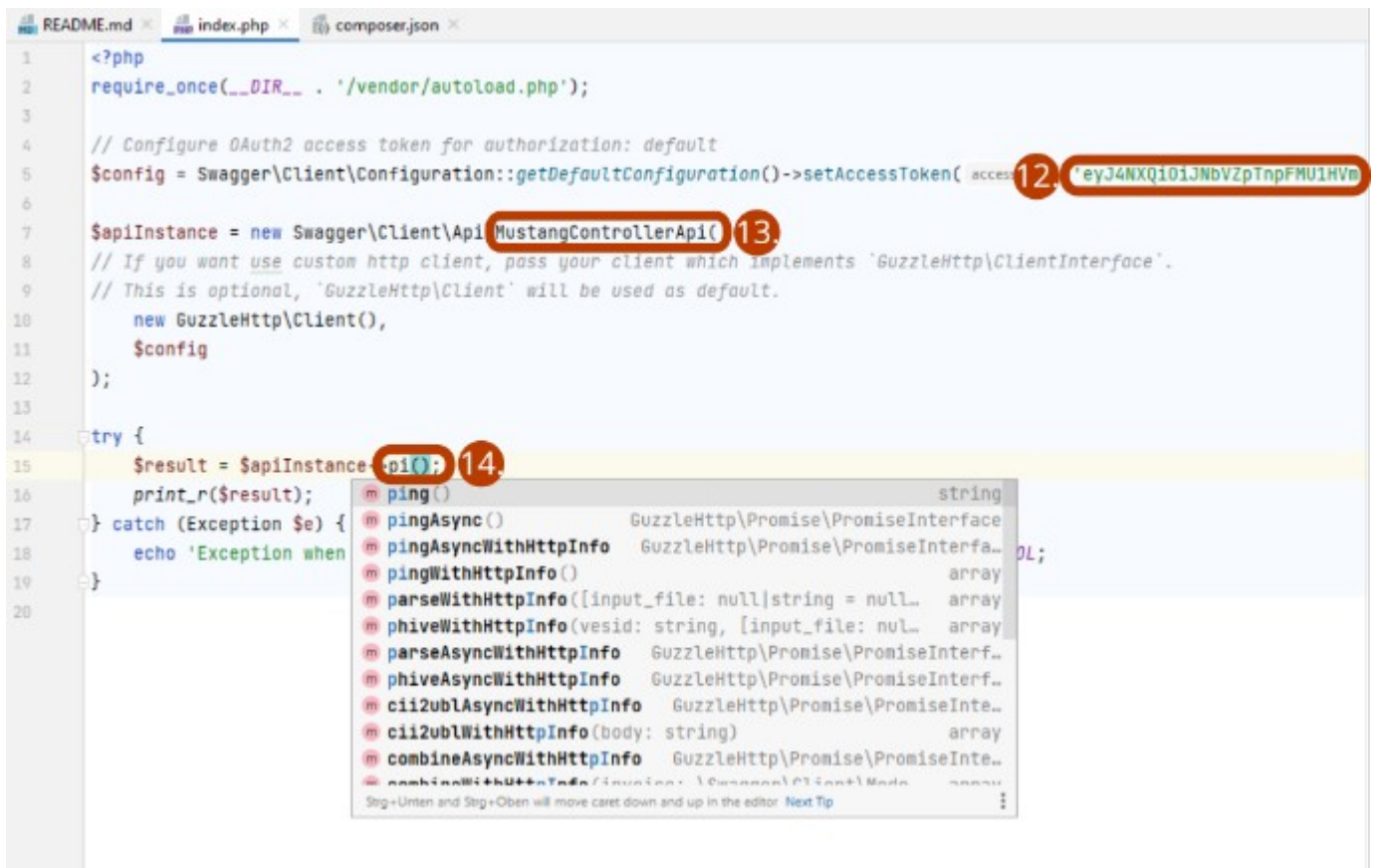
#11. For this part we will use a token which will expire shortly. Click Generate Access Token (11.), Generate and copy the resulting token. Paste it in

Screenshot 5:

```

1 <?php
2 require_once(__DIR__ . '/vendor/autoload.php');
3
4 // Configure OAuth2 access token for authorization: default
5 $config = Swagger\Client\Configuration::getDefaultConfiguration()->setAccessToken( 'access' 12. 'eyJ4NXQ1O1JNbVZpTnpFMU1HVm
6
7 $apiInstance = new Swagger\Client\Api MustangControllerApi( 13.
8 // If you want use custom http client, pass your client which implements 'GuzzleHttp\ClientInterface'.
9 // This is optional, 'GuzzleHttp\Client' will be used as default.
10 new GuzzleHttp\Client(),
11 $config
12 );
13
14 try {
15 $result = $apiInstance->ping(); 14.
16 print_r($result);
17 } catch (Exception $e) {
18 echo 'Exception when
19 }
20

```

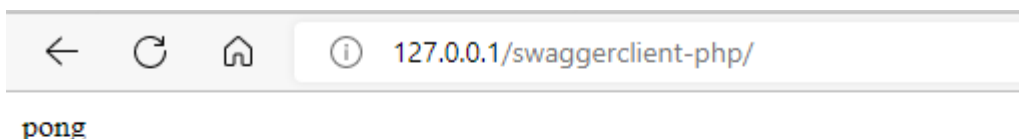


#12. index.php (12.), in the same file

#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 processor in your browser, it should now look like Screenshot 6:



pong

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:

```
<?php

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

$client = new GuzzleHttp\Client();
$res = $client->request( method: 'POST', uri: 'https://gw.usegroup.de:9443/oauth2/token', [
    'auth' => [16. 'nfVADLwYvfab1AzkHFQXzq5ASaEa', 'client secret' 18.],
    'form_params' => [
        'grant_type' => 'client_credentials',
    ]
]);

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

// Configure OAuth2 access token for authorization: default
$config = Swagger\Client\Configuration::getDefaultConfiguration()->setAccessToken($json["access_token"] 19.);

$apiInstance = new Swagger\Client\Api\MustangControllerApi(
    // If you want use custom http client, pass your client which implements 'GuzzleHttp\ClientInterface'.
    // This is optional, 'GuzzleHttp\Client' will be used as default.
    new GuzzleHttp\Client(),
    $config
);

try {
    $result = $apiInstance->validateFile( in file: "factur-x.pdf"); 21.
    print_r(htmlentities($result)) 22.;
```

#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

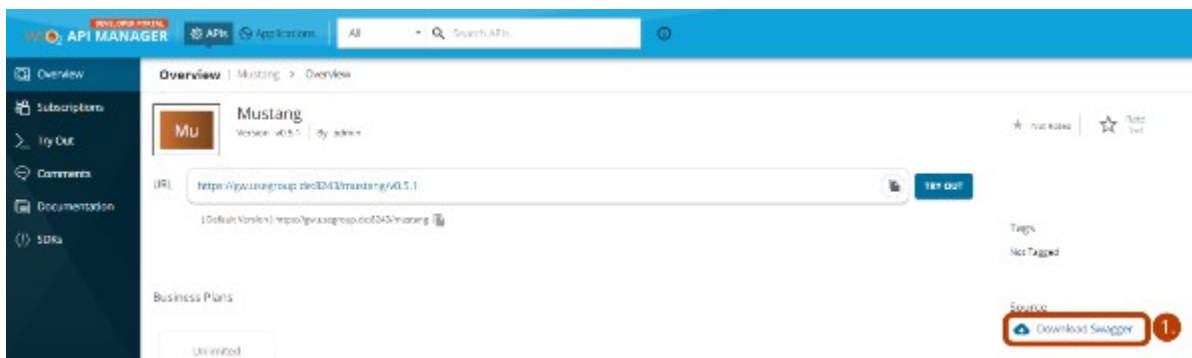
#22 html escape the validation result, so that the result in the browser looks like screenshot 8:

```
← ↻ 🏠 ⓘ 127.0.0.1/swaggerclient-php/
<?xml version="1.0" encoding="UTF-8"?> <validation filename="tovalidate14506017597035795196mustang
<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 :-)

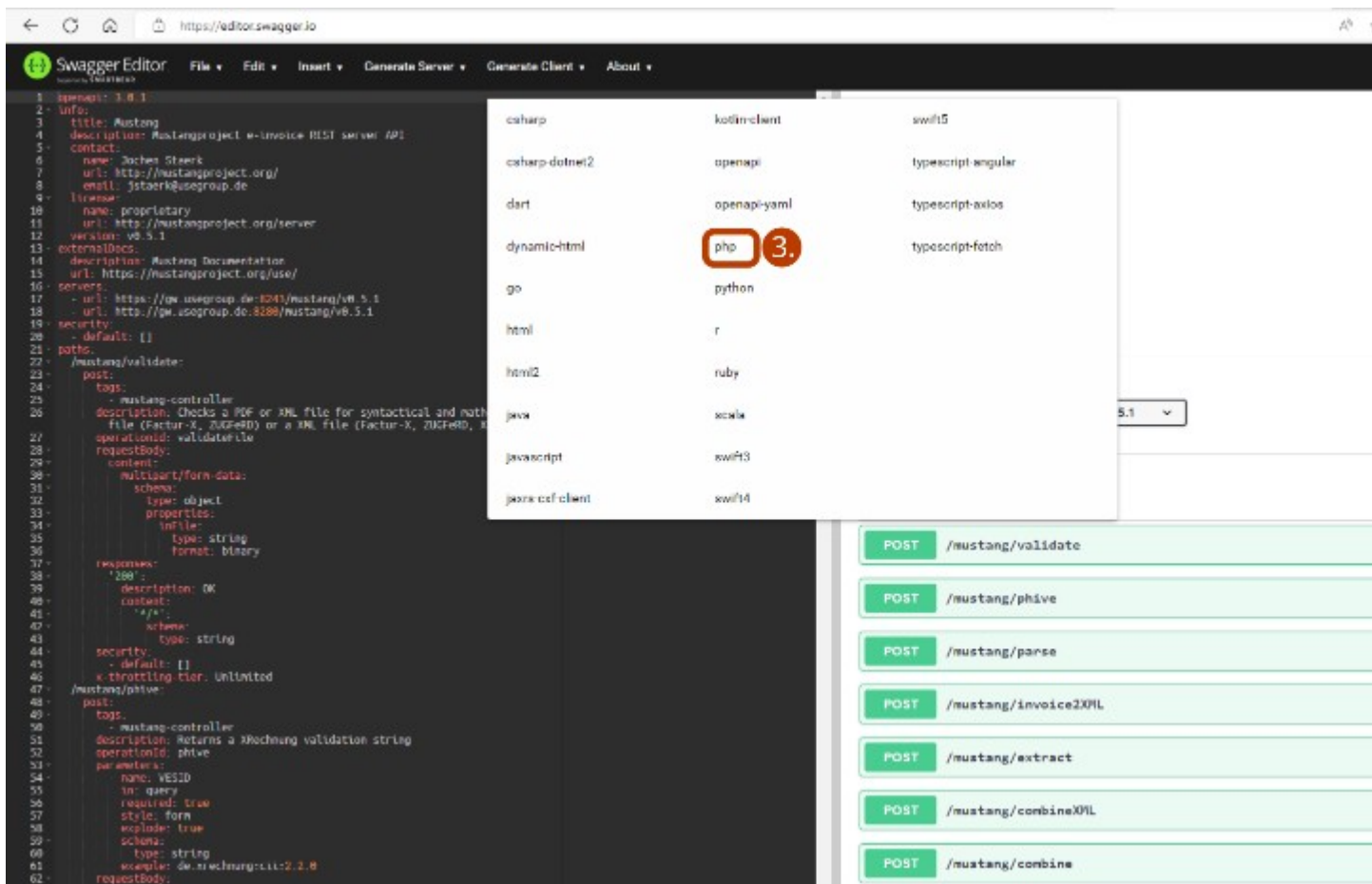
Feel free to also try the async functions.

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.)

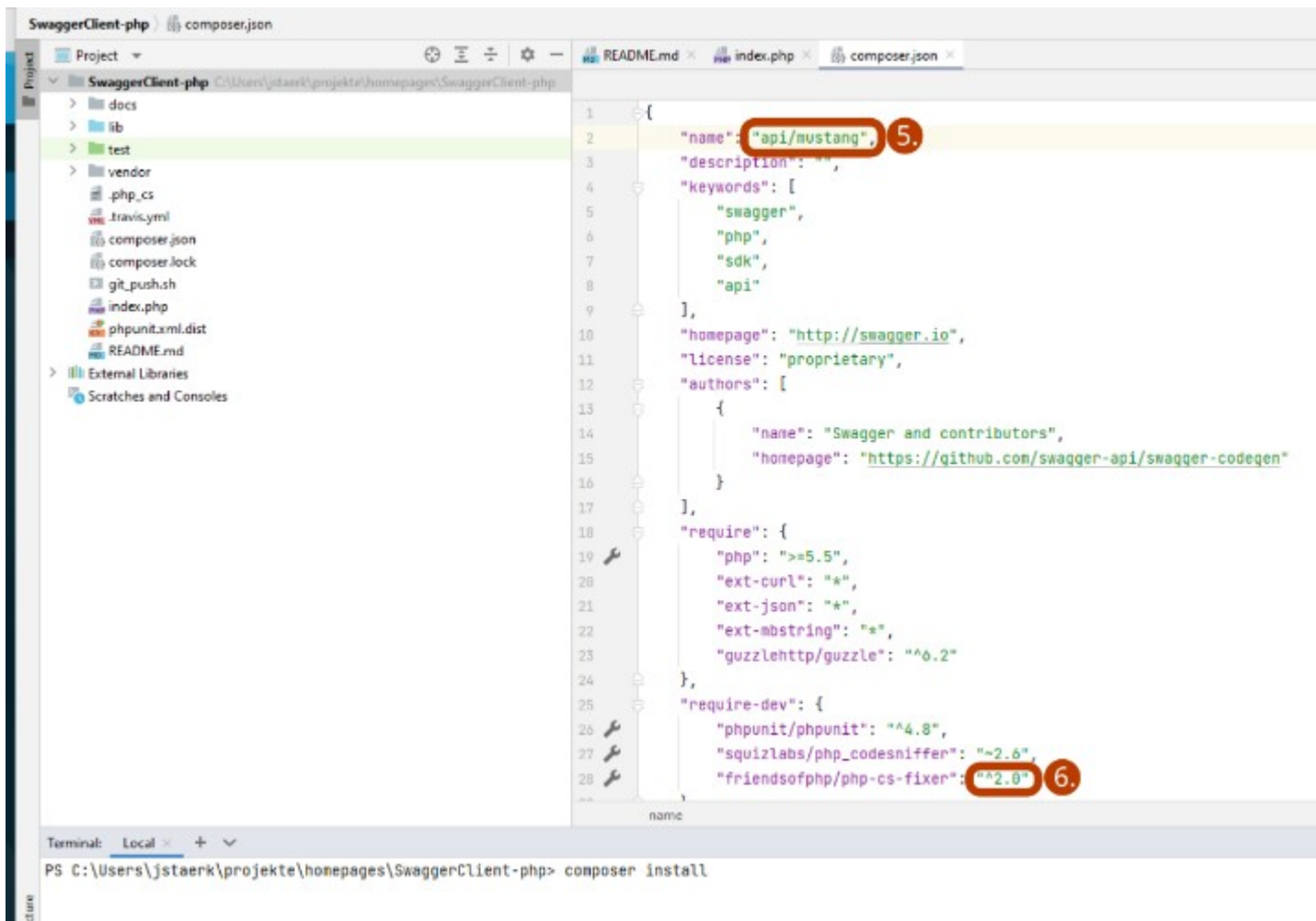
Screenshot 2:

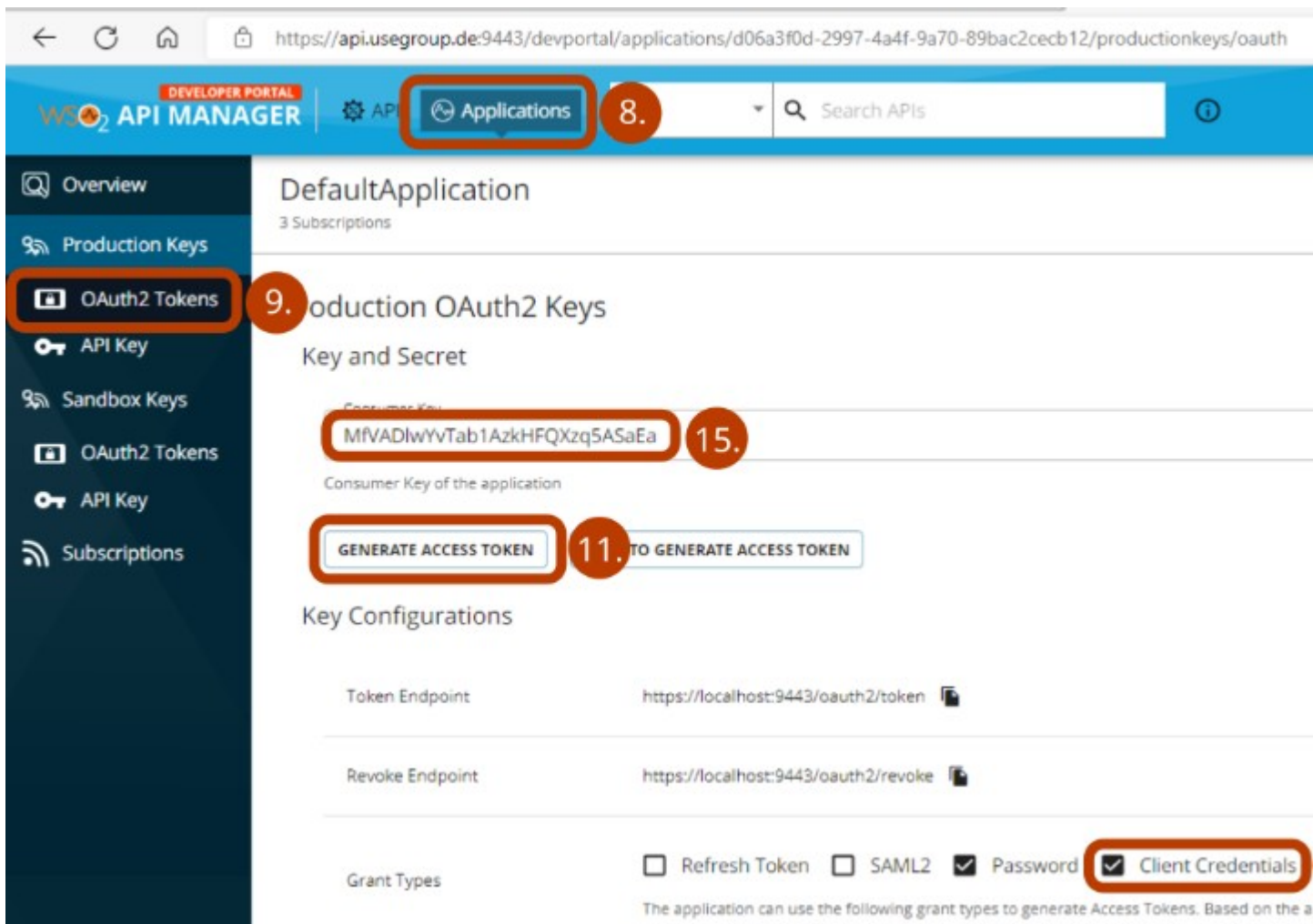


#4. Extract the downloaded file, edit composer.json.

#5. change the name of the project in the composer.json file to lowercaps/lowercaps (5.)

Screenshot 3:





#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:

```

1 <?php
2 require_once(__DIR__ . '/vendor/autoload.php');
3
4 // Configure OAuth2 access token for authorization: default
5 $config = Swagger\Client\Configuration::getDefaultConfiguration()->setAccessToken( 'access' 12. 'eyJ4NXQ1OD1JNnVZpTnpFMU1HVm
6
7 $apiInstance = new Swagger\Client\Api\ MustangControllerApi( 13.
8 // If you want use custom http client, pass your client which implements 'GuzzleHttp\ClientInterface'.
9 // This is optional, 'GuzzleHttp\Client' will be used as default.
10 new GuzzleHttp\Client(),
11 $config
12 );
13
14 try {
15 $result = $apiInstance->ping(); 14.
16 print_r($result);
17 } catch (Exception $e) {
18 echo 'Exception when
19 }
20

```

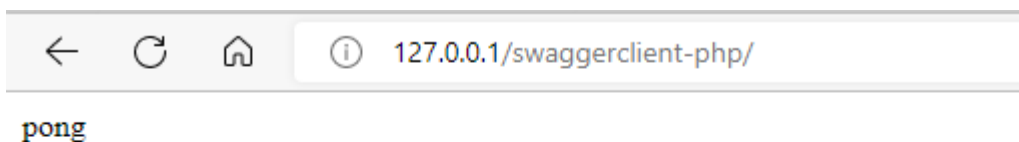


#12. index.php (12.), in the same file

#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 processor in your browser, it should now look like Screenshot 6:



pong

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:

```

<?php

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

$client = new GuzzleHttp\Client();
$res = $client->request( method: 'POST', uri: 'https://gw.usegroup.de:9443/oauth2/token', [
    'auth' => ['16.', 'MfVADlwYvTabIAzkHFQXzq5ASaEa', '18.', 'client_secret'],
    'form_params' => [
        'grant_type' => 'client_credentials',
    ]
]);

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

// Configure OAuth2 access token for authorization: default
$config = Swagger\Client\Configuration::getDefaultConfiguration()->setAccessToken('$json["access_token"]' 19.);

$apiInstance = new Swagger\Client\Api\MustangControllerApi(
    // If you want use custom http client, pass your client which implements 'GuzzleHttp\ClientInterface'.
    // This is optional, 'GuzzleHttp\Client' will be used as default.
    new GuzzleHttp\Client(),
    $config
);

try {
    $result = $apiInstance->validateFile( in file: "factor-x.pdf"); 21.
    print_r(htmlentities($result)); 22.
} catch (Exception $e) {
    echo $e->getMessage();
}

```

#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 factor-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:

```
← ↻ 🏠 ⓘ 127.0.0.1/swaggerclient-php/
<?xml version="1.0" encoding="UTF-8"?> <validation filename="tovalidate14506017597035795196mustang
<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.

Example C# client

With node.js installed use

```
npm install @openapitools/openapi-generator-cli -g
```

then

```
openapi-generator-cli generate -i "swagger.json" -g csharp -o "csharpproject"
```

This will create a Library which uses RestSharp to access Mustang. Then use Microsoft Visual Studio Community or higher (not Visual Studio Code) to open Org.OpenAPITools.sln .

Get yourself an Api Key(see page 4)

Uncomment e.g. the ping test in src\Org.OpenAPITools.Test\Api\MustangControllerApiTests.cs

And change the constructor to

```
public MustangControllerApiTests()
{
    Configuration c = new Configuration();
    c.DefaultHeader.Add("apikey", "<your api key>");
    instance = new MustangControllerApi(c);
}
```

Alternatively, to use oauth,

see „Allowing Client Credentials“ on page 19

use

```

public MustangControllerApiTests()
{
    Configuration c = new Configuration();

    c.OAuthFlow = OAuthFlow.APPLICATION;
    c.OAuthTokenUrl = "https://api.usegroup.de:9443/oauth2/token";
    c.OAuthClientId = "<your client id>";
    c.OAuthClientSecret = "<your client secret>";

    instance = new MustangControllerApi(c);
}

```

If you want to use an API Key. With CTRL+E, T you can see the test explorer and with CTRL+R, A you can run all tests (of which only pingTest will be enabled).

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.

Interactive testing using Postman/Bruno

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

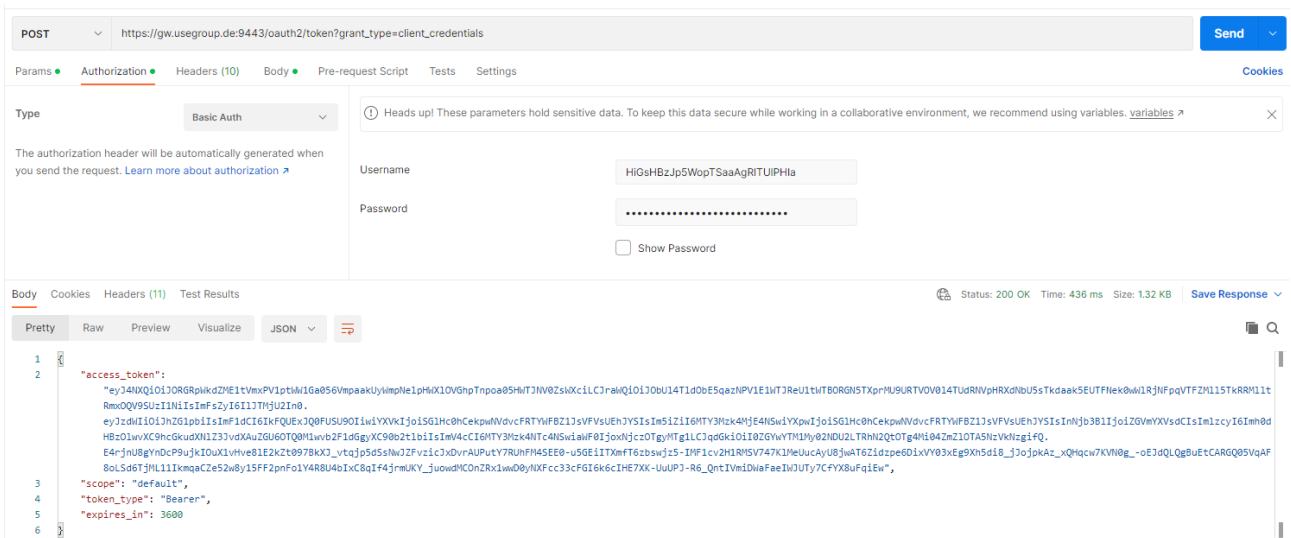
This example is based on [Postman](#), you will need enabled client credentials as described on page 19 in Allowing Client Credentials.

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:



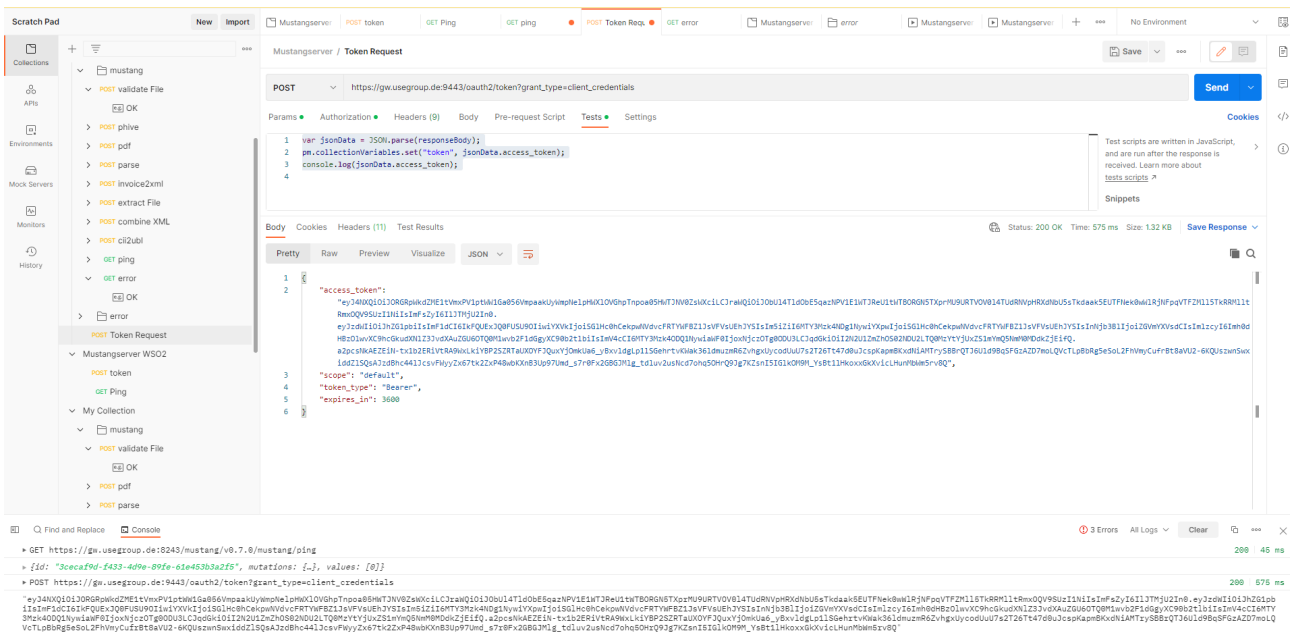
Add

```

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.

VARIABLE	INITIAL VALUE	CURRENT VALUE	Persist All	Reset All
<input checked="" type="checkbox"/> baseUrl	https://gw.usegroup.de:8243/mustang/v0.7.0	https://gw.usegroup.de:8243/mustang/v0.7.0		
<input checked="" type="checkbox"/> token				
Add a new 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.

Mustangserver / mustang / ping

GET `{{baseUrl}}/mustang/ping` [Send]

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Type: Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [variables](#)

Token: `{{token}}`

Body: pong

Status: 200 OK Time: 45 ms Size: 704 B Save Response

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 factor-x:

Mustangserver / mustang / validate File

POST `{{baseUrl}}/mustang/validate` [Send]

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

form-data

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> @File	factur-x (6).pdf X	
Key	Value	Description

Body: [JSON]

```

18   <duration start="1673985270293" finish="1673985273311">00:00:03.018</duration>
19   </job>
20   </jobs>
21   <batchSummary totalJobs="1" failedToParse="0" encrypted="0">
22     <validationReports compliant="1" nonCompliant="0" failedJobs="0">1</validationReports>
23     <featureReports failedJobs="0">0</featureReports>
24     <repairReports failedJobs="0">0</repairReports>
25     <duration start="1673985260818" finish="1673985273315">00:00:11.497</duration>
26   </batchSummary>
27   </report>
28   <info>
29     <signature>Mustang</signature>
30     <duration unit="ms">12748</duration>
31   </info>
32   <summary status="valid"/>
33 </pdf>
34 </com>

```

3 Errors All Logs Clear

OST `https://gw.usegroup.de:9443/oauth2/token?grant_type=client_credentials`

OST `https://gw.usegroup.de:8243/mustang/v0.7.0/mustang/validate`

Performance Tests with Jmeter

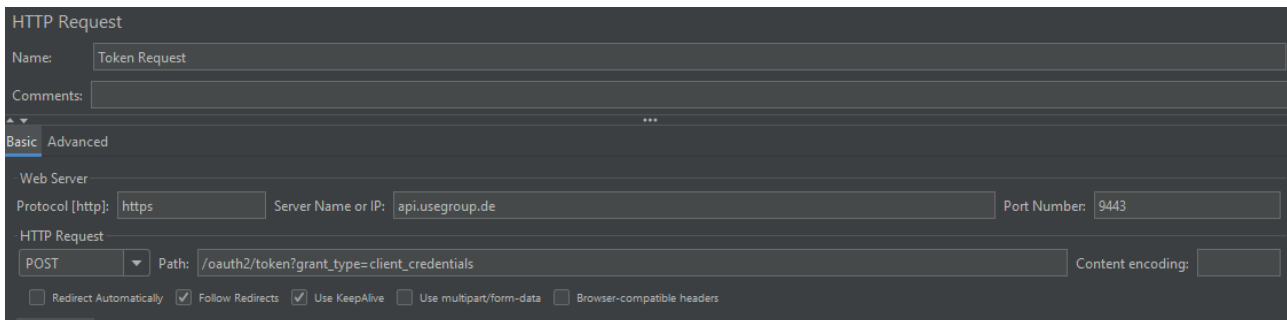
[Jmeter](#) is a generic load testing tool and load generator.

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.

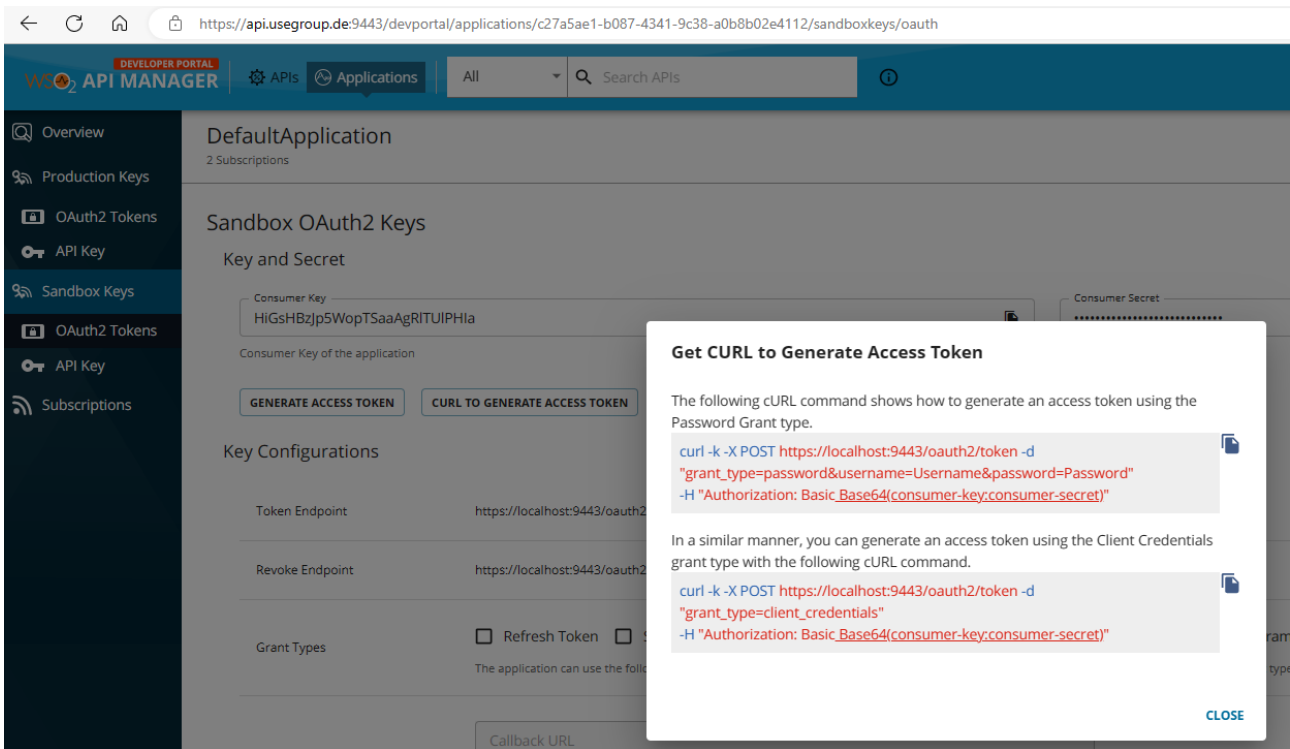
<https://openapi-generator.tech/> supports a Jmeter export but that does not handle authentication so here we describe how to set up some Jmeter performance test manually.

For this example, you will need enabled client credentials as described on page 19 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:

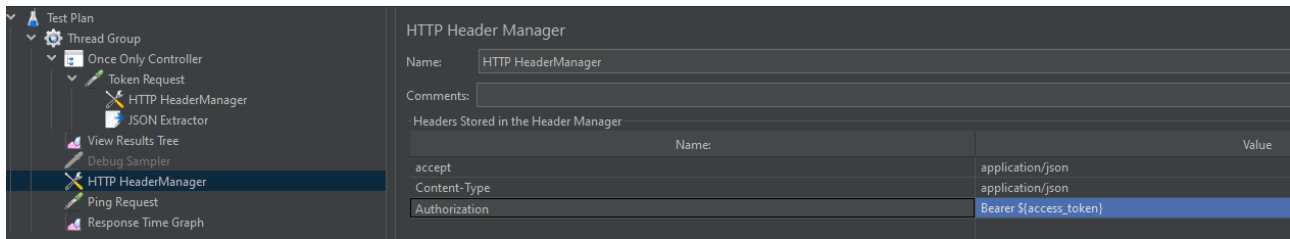
And from the response, also below Token Request, extract the JSON value access token into a Jmeter Variable access token using a JSON Extractor:



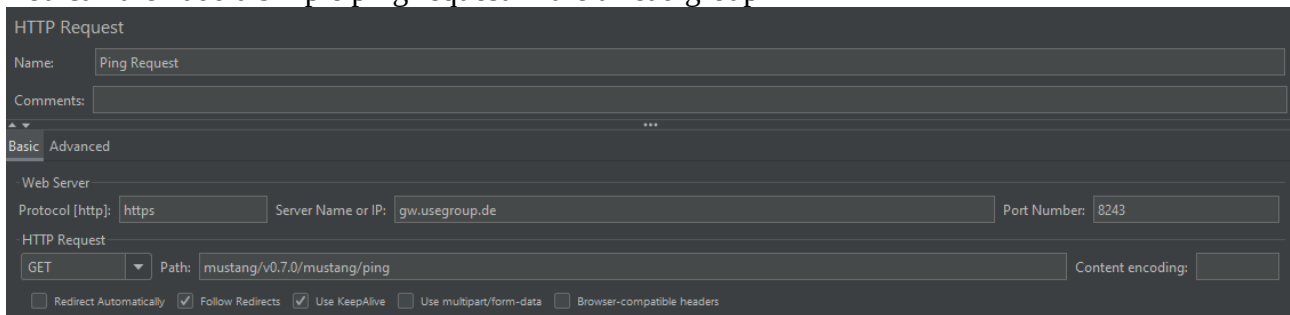
Add a sampler View Result Tree to confirm the results and a debug sampler if you like (in the results tree you will then be able to e.g. see the current variables when you click on the results of the debug sampler).

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

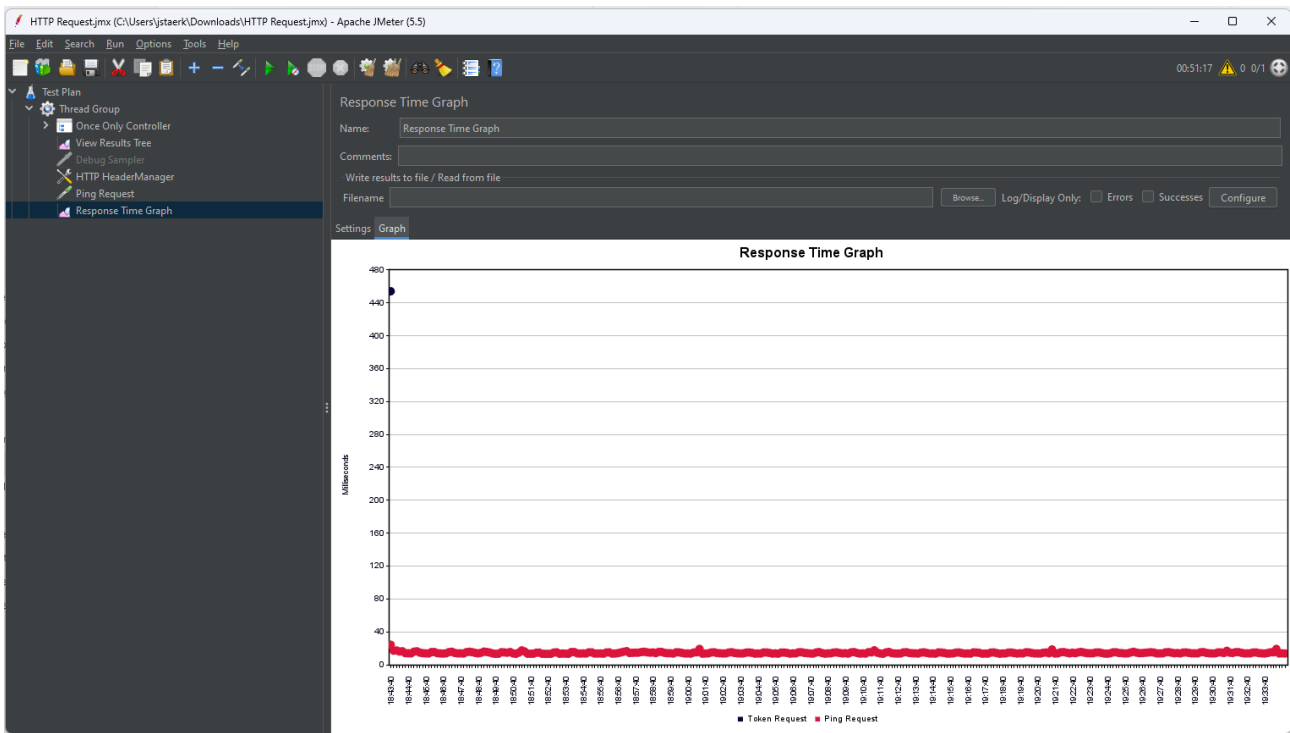
Now we will set the ordinary authentication as header: add another Header Manager outside of the Token request and add the variable as token, i.e. Authorization being Bearer `${access_token}`



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/>

Troubleshooting

- The ping endpoint is intentionally simple and can be used to check basic functionality
- A HTTP response code 400 Post method not allowed on methods which do allow post may (temporarily) indicate a throttled user or subscription

Inhouse variant

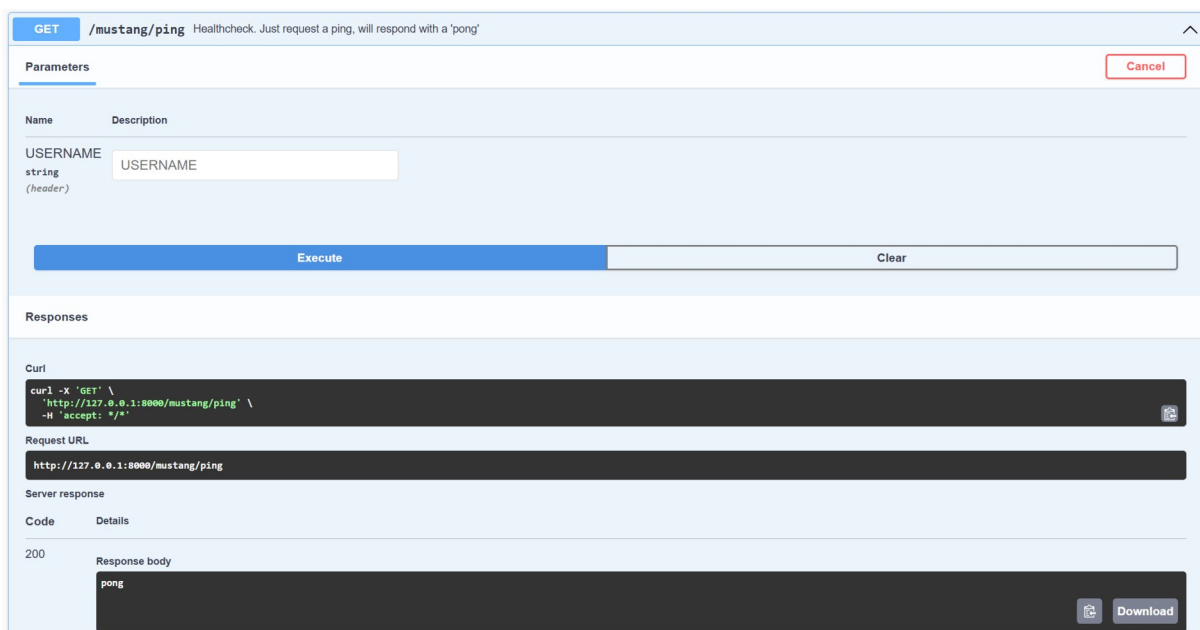
If you bought the inhouse version: the container registry is dev.usegroup.de:5050 and the default port the server starts on is 8000

```
docker login dev.usegroup.de:5050 -u <username> -p <token>
docker run -e MUSTANG_SERVER_VERSION=1.4.0 -dp 8000:8000 dev.usegroup.de:5050/internal/mustangserver
```

Afterwards you should be able to access

`http://<ip>:8000/swagger-ui/index.html`

and e.g. perform a ping like described above. You can leave username empty. The correct response to ping is „pong“.

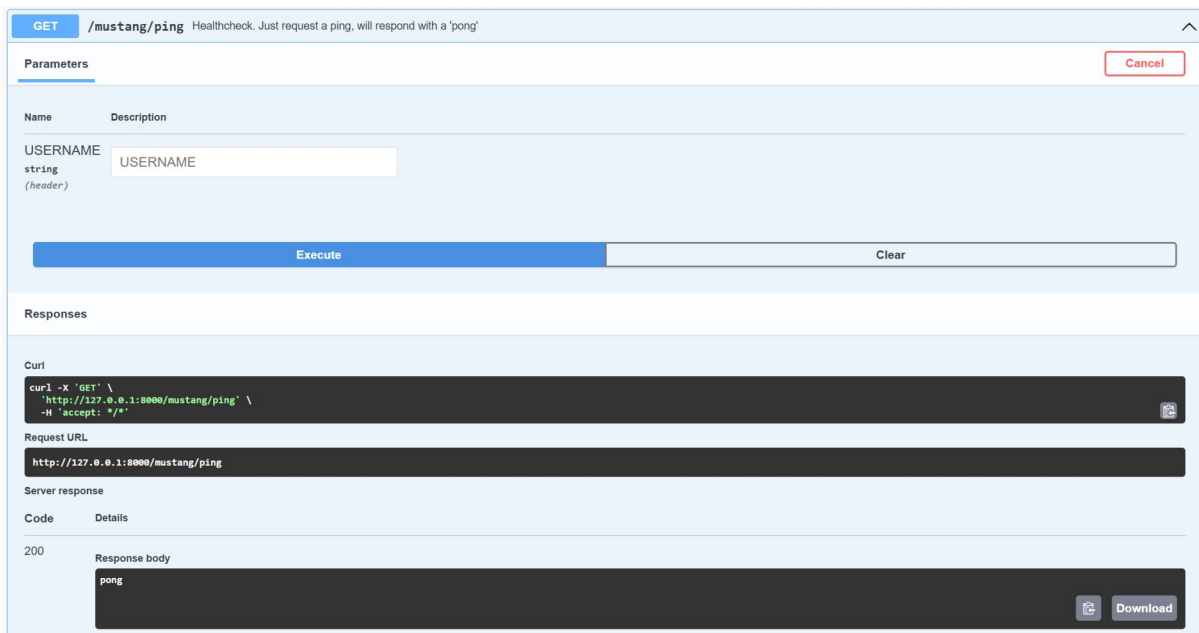


The screenshot shows a REST client interface for a GET request to `/mustang/ping`. The interface includes a 'Parameters' section with a 'USERNAME' field set to 'USERNAME'. Below this is an 'Execute' button and a 'Clear' button. The 'Responses' section shows the following details:

- Curl:**

```
curl -X 'GET' \
  'http://127.0.0.1:8000/mustang/ping' \
  -H 'accept: */*'
```
- Request URL:** `http://127.0.0.1:8000/mustang/ping`
- Server response:**

Code	Details
200	<p>Response body</p> <pre>pong</pre>



By specifying additional `-e key=value` pairs (e.g. `docker run -e MUSTANG_SERVER_VERSION=1.4.0 -e mustang.allowedIPs=123.45.67.89 -p 8888:8000 dev.usegroup.de:5050/internal/mustangserver:latest`) you can set config variables.

Available vars are

Config	Default	Description
<code>server.port</code>	8000	The port the OpenAPI HTML Client and backend operate on
<code>mustang.allowedIPs</code>		If specified, a comma-separated list of IPs who will be allowed to connect
<code>mustang.oAuth</code>	false	Off by default
<code>mustang.additionalLog</code>		Additional entries for the log line of the request, can e.g. be set to the instance's ID to allow a consolidated. Logs are stored in <code>/opt/mustangserver/log</code> and in a future version there may be a filebeat to be configured
<code>keycloak.enabled</code>	false	Almost never to be set to true, even if you connect to a keycloak server, if true it will start an keycloak server on its own
<code>spring.security.oauth2.client.registration.login-app</code>		Oauth2 specific setting (if enabled)
<code>keycloak.client-id</code>		
<code>spring.security.oauth2.client.registration.authorization</code>		Oauth2 specific setting (if enabled)
<code>keycloak.authorization-grant-type</code>	<code>n_code</code>	

spring.security.oauth2.client.registration.keycloak.scope	openid	Oauth2 specific setting (if enabled)
spring.security.oauth2.client.provider.keycloak.issuer-uri	http://localhost:8080/realms/SpringBootKeycloak	Oauth2 specific setting (if enabled)
spring.security.oauth2.client.provider.keycloak.user-name-attribute	preferred_username	Oauth2 specific setting (if enabled)
spring.security.oauth2.resourceserver.jwt.issuer-uri	http://localhost:8080/realms/SpringBootKeycloak	Oauth2 specific setting (if enabled)
message-from-application-properties	Die Anwendung wird in der Default Environment gestartet!	Will just be shown on the console
server.servlet.context-path		Usually not set at all but could be set to e.g. /api/v1
springdoc.api-docs.path	/api-docs	Where the openapi spec file will be found
springdoc.swagger-ui.path	/swagger-ui	Path of the HTML GUI
logging.config	classpath:logback-spring.xml	

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, necessity 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, 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.

1.3.1 on 2024-07-29 JSON structure documented, added Troubleshooting

1.4.0 added description of detach and xmltopdf endpoint and of C# sample client. Added section classes.

1.4.1 added chapter on Docker config for in-house server. Corrected timezone in example. Added BT Mapping and invoice class description.