**Friday 20th September 09:00-12:45 – Workshop 7**

# API First: Building ABL Clients from Swagger documents

*Hosted by Mike Fechner & Peter Judge, Consultingwerk*

As business applications are required to provide more integration with other systems, the first question many developers ask when integrating REST based API's into their application is, where's the Swagger file. OpenAPI (typically called Swagger) is a standard for describing REST and typically JSON based API's, and a variety of systems – from payments processing to authentication services – document access to their systems using an OpenAPI document.

In this session we'll introduce the basic concepts of OpenAPI and the Swagger file, tooling that can be used with this and what it takes to use OpenAPI as a foundation to build ABL clients to access those API's.

The session will use the OpenEdge HTTP Client and the JSON object model in the ABL to access REST services.

# 1. Workshop requirements

Attendees of the workshop need on OpenEdge Progress Developer Studio installation. A licensed product or the Classroom Edition should work. The labs have been tested with OpenEdge 12.8.4 – but any release from 11.7 on should work equally fine.

Attendees need to be able to create an PASOE instance – or deploy OE Manager to an already existing PASOE instance, when not already done. These labs assume that the instance is using port 8820 for http and 8821 for https.

The workshop uses a reduced deployment of the SmartComponent Library framework.

Workshop labs and generator code are provided on a Github repository: https://github.com/consultingwerk/swagger-workshop

A Git client can be used to download the files – alternatively a browser allows to download the files.

The Generator files are in the Consultingwerk\Studio\SwaggerToAbl folder. Relevant other files are Consultingwerk\JsonSerializable.cls and Consultingwerk\Util\JsonHelper.cls

# 2. Introduction to Swagger and Tools used in the workshop

Based on Slide deck *“API First: building ABL clients from Swagger documents”*

# 3. Install PASOE Instance

Install a simple PASOE instance – no Database Connection or any specific parameters/propath required

Ensure the PASOE instance can be started

Try to access http://localhost:8810/web/ping

# 4. Deploy OE Manager web app to PASOE Instance

Start PROENV and from within the PASOE instance directory execute bin\tcman deploy %dlc%\servers\pasoe\extras\oemanager.war

Ensure OE Manager Swagger page can be accessed: http://localhost:8810/oemanager/

**Note: The trailing slash in the OE Manager URL is important!**

Download the OE Manager Swagger file from:
http://localhost:8810/oemanager/doc/openapi.json

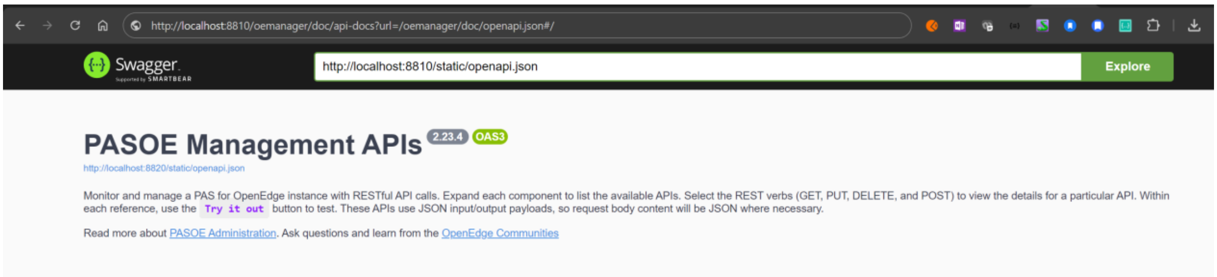
## 5. Copy OE Manager Swagger file to PASOE's static folder

Copy the openapi.json file from Swagger to the PASOE's webapps\ROOT\static folder. This allows you to open that file in PASOE's build in Swagger page.

Navigate to http://localhost:8810/oemanager/

Paste http://localhost:8810/static/openapi.json into the field next to the "Explore" Button and click the Explore Button.

This will load openapi.json from the location where we can maintain it.



http://localhost:8810/oemanager/doc/api-docs?url=/oemanager/doc/openapi.json#/

http://localhost:8810/static/openapi.json

PASOE Management APIs 2.23.4 OAS3

http://localhost:8820/static/openapi.json

Monitor and manage a PAS for OpenEdge instance with RESTful API calls. Expand each component to list the available APIs. Select the REST verbs (GET, PUT, DELETE, and POST) to view the details for a particular API. Within each reference, use the `Try it out` button to test. These APIs use JSON input/output payloads, so request body content will be JSON where necessary.

Read more about PASOE Administration. Ask questions and learn from the OpenEdge Communities

## 6. Use Swagger V3 online editor to add response type for ApsvTransport Properties update request (getProperties_1)

Alternatively use VS Code or any other editor that supports JSON.

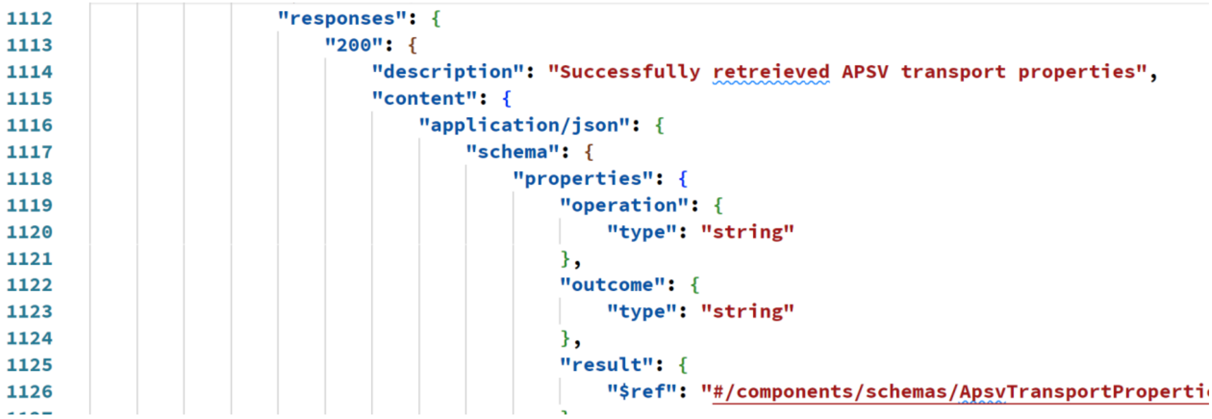
https://editor-next.swagger.io/

Locate the section "paths" ->

paths["/applications/{AppName}/webapps/{WebAppName}/transports/apsv/properties"].get.responses["200"]

Insert the response schema within the response definition for 200/OK:

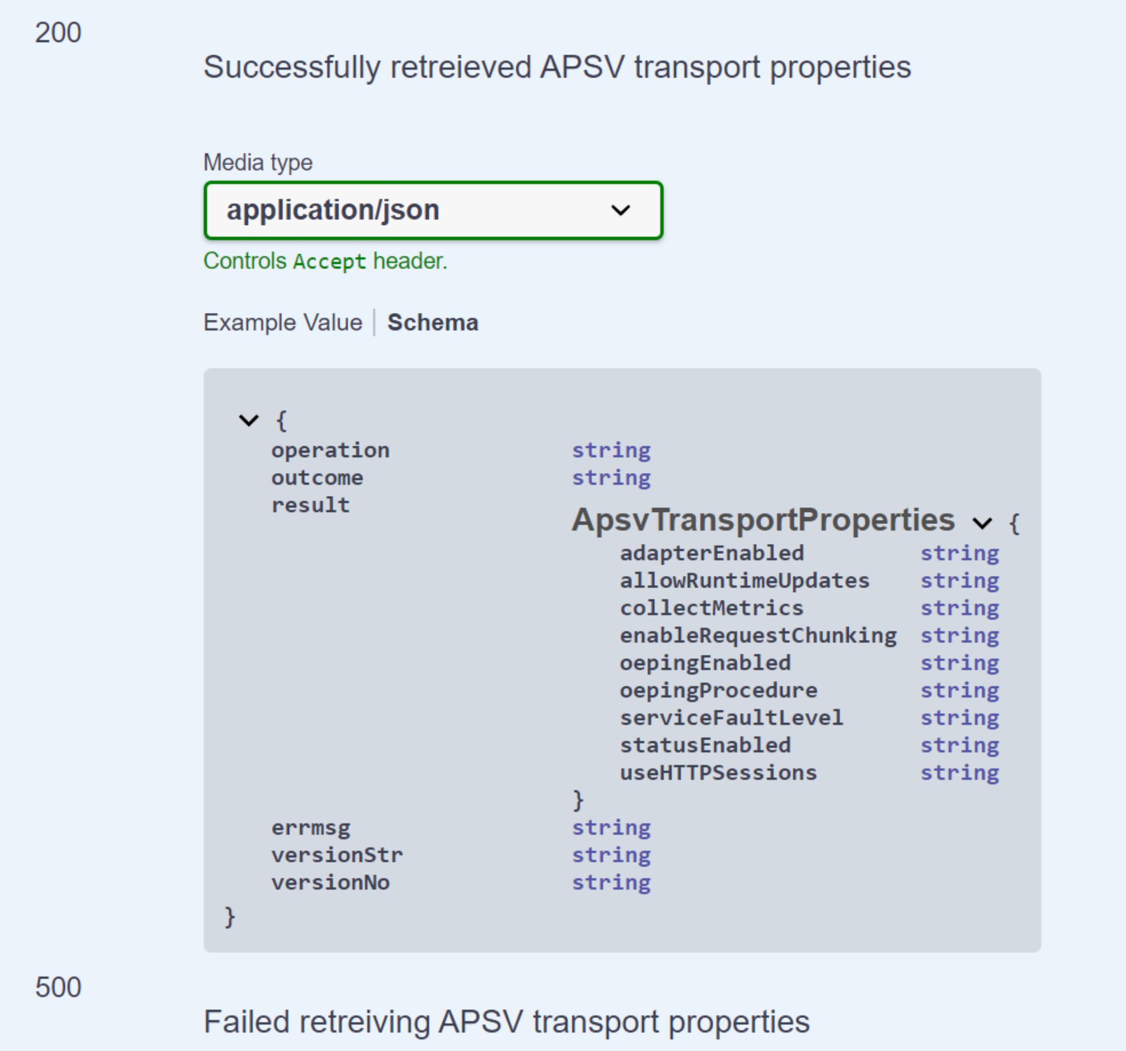
```
"content": {
  "application/json": {
    "schema": {
      "properties": {
        "operation": {
          "type": "string"
        },
        "outcome": {
          "type": "string"
        },
        "result": {
          "$ref": "#/components/schemas/ApsvTransportProperties"
        },
        "errmsg": {
          "type": "string"
        },
        "versionStr": {
          "type": "string"
        },
        "versionNo": {
          "type": "string"
        }
      }
    }
  }
}
```



```
                "responses": {
                    "200": {
                        "description": "Successfully retreieved APSV transport properties",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "properties": {
                                        "operation": {
                                            "type": "string"
                                        },
                                        "outcome": {
                                            "type": "string"
                                        },
                                        "result": {
                                            "$ref": "#/components/schemas/ApsvTransportProperti
```

Store the modified file in PASOE static folder, access OE Manager Swagger page as described in section 5.

Note, the schema now visible for the 200/OK response.



200

Successfully retreieved APSV transport properties

Media type

application/json

Controls `Accept` header.

Example Value | **Schema**

```
{
    operation            string
    outcome              string
    result               ApsvTransportProperties {
                             adapterEnabled          string
                             allowRuntimeUpdates     string
                             collectMetrics          string
                             enableRequestChunking   string
                             oepingEnabled           string
                             oepingProcedure         string
                             serviceFaultLevel       string
                             statusEnabled           string
                             useHTTPSessions         string
                         }
    errmsg               string
    versionStr           string
    versionNo            string
}
```

500

Failed retreiving APSV transport properties

Update the PUT request for the APSV transport properties in the same way!

# 7. Generate ABL Types for ApsvTransport endpoints using Swagger to ABL Generator

The procedure labs\generate-oemanager-types.p is a sample for generating the required ABL classes to either represent the request payload or the response. The JSON Path determines which classes to generate.

```
assign cPath =
"paths/~~/applications~~/~{AppName}~~/webapps~~/~{WebAppName}~~/transports~
~/apsv~~/properties/put/requestBody/content/application~~/vnd.progress+json
/schema".
```

The path looks a bit weird, this is our representation of JSON path’s. The JSON Object structure is like this:

*“paths”: {*

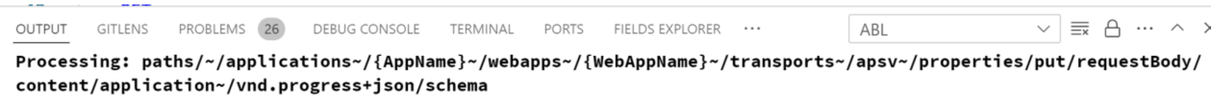
*“/applications/{AppName}/webapps/{WebAppName}/transports/apsv/properties”: {*

*“put”: { “requestbody” : { “content”: { “application/vnd.progress+json”: { “schema” …*

The curly braces in the path parameters need to be escaped with the tilde.

The slashes in the path need to be escaped with double tilde.

When you run the procedure using CTRL-Shift-P and “ABL: Run with _progres -b” the procedure will produce the following output:



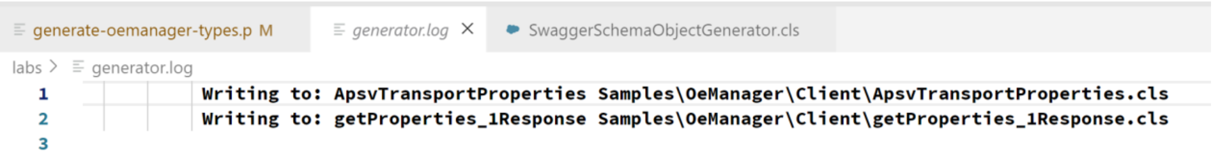
```
Processing: paths/~/applications~/{AppName}~/webapps~/{WebAppName}~/transports~/apsv~/properties/put/requestBody/
content/application~/vnd.progress+json/schema
```

And the file Samples\OeManager\Client\ApsvTransportProperties.cls will be generating representing the request body to the update properties request.

Repeat the steps for the JSON path for the response of the GET and PUT request.

Check the generated files in the Samples\OeManager\Client folder.

Check the output in the generator.log file in the labs project.


generate-oemanager-types.p M
generator.log
SwaggerSchemaObjectGenerator.cls
labs > generator.log

```
Writing to: ApsvTransportProperties Samples\OeManager\Client\ApsvTransportProperties.cls
Writing to: getProperties_1Response Samples\OeManager\Client\getProperties_1Response.cls
```

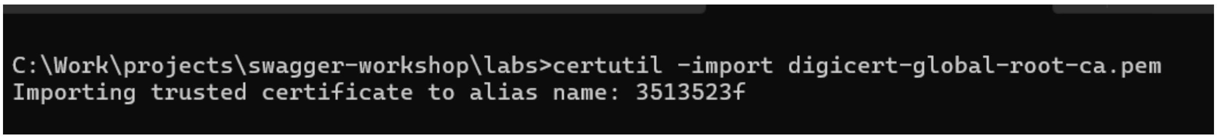
## 8. Implement OE Manager Client using the OpenEdge HTTP Client

Copy labs/OeManagerTransportsApsvClient.cls into the Demo/Swagger/oemanager/Client folder. This class has an implementation of calling the endpoint using the OpenEdge HTTP client.

## 9. Import SSL Certificate of California Tax Web Service

Open PROENV and change directory to the labs/ directory. Import the root certificate using the *certutil -import digicert-global-root-ca.pem* command.

You should see a response similar to the below



```
C:\Work\projects\swagger-workshop\labs>certutil -import digicert-global-root-ca.pem
Importing trusted certificate to alias name: 3513523f
```

## 10. Generate ABL Types for California Tax Web Service using Swagger to ABL Generator

Similar to section 7.

## 11. Implement Client using the OpenEdge HTTP Client

Copy labs/TaxInfoClient.cls into the Demo/Swagger/CaliforniaTax /Client folder. This class has an implementation of calling the endpoint using the OpenEdge HTTP client.