**TIBCO Order Management 5.1 Base Image**

[Installation Details](#_og49hvckn4q)

[AWS Requirements](#_3qftw4ppnzbf)

[Security Group Requirements](#_d4krlja9z1im)

[Connecting to the Instance](#_ojintcx8lhmm)

[FIXTHIS:Configure OM Authorisation Service IP address. Cannot login if wrong IP](#_evkrm3nqdoxi)

[Starting TIBCO Order Management](#_2795iu9f8xtc)

[Order Management URLs](#_2k1l8ttap056)

[Usernames & Passwords](#_ggd41s1m4nuh)

[GITHUB files](#_sy3c5554riod)

[BusinessWorks Sample Process Component](#_pib1i0k5yz04)

[Generic Process Component:](#_53u19qwn2783)

[Batch Process Component:](#_zakjo18z7f1y)

[Sending an Order via the Swagger Interface](#_zew425yj9qsx)

[Authorizing](#_capeiobvyuoo)

[Sending a Sample Order](#_l38dh35wxumt)

[Sending an Order via the TMF Adapter Swagger Interface](#_e8io9krtfjbp)

[Sending an Order through the OCS UI (OCS Depricated in 5.1)](#_m9b8hqd3kv78)

[Checking Order Status](#_i7ltr2393vov)

[Checking TCI BusinessWorks Logs](#_mrerydttplll)

[Publishing Catalog Changes to Order Management & OCS](#_u3jugmyxoit8)

[Updating OCS Catalog](#_qr2351i58egj)

[Configure OM UI](#_9nprmzw9ryz4)

[Cancel](#_8fswdd6t9n0l)

[Amend](#_lx0z8tfkclnd)

# 

| **Version** | **Comment** | **By** |
| --- | --- | --- |
| 1.0 | 16th December 2021: Initial Draft | dwinston@tibco.com |
| 1.1 | 31 October 2022 - image updates after HF7 | Iain |
| 1.2 | November: Use cases |  |

# Installation Details

TIBCO Order Management 5.1

TIBCO Offer and Pricing Engine 5.1 (HF7)

TIBCO EMS 8.5

TIBCO EBX with Product & Service Catalog (PSC-EBX) 1.2.0

Configured with Postgresql

# AWS Requirements

AMI Image:[ami-0505a501ca2404af2](https://eu-west-1.console.aws.amazon.com/ec2/home?region=eu-west-1#ImageDetails:imageId=ami-0505a501ca2404af2)

AWS Region: EU-West-1 (Ireland)

You will need to request access from [iharfiel@tibco.com](mailto:iharfiel@tibco.com) and you will need to provide your AWS Account ID

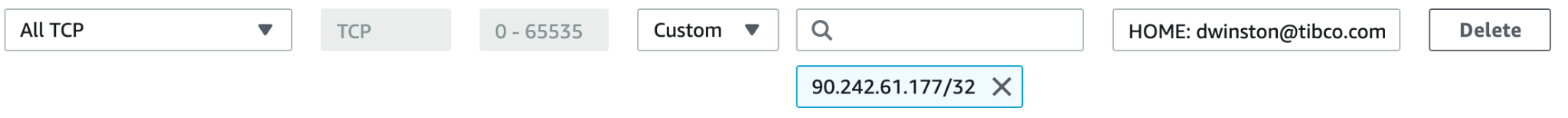
AWS Image type needs to be at least 4 CPU, 16GB RAM for base usage. Anything complicated will require more CPU/RAM.

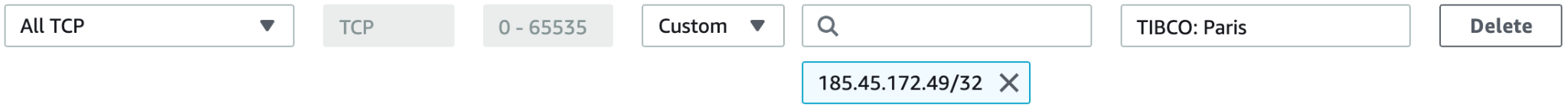
You will benefit from assigning an elastic IP Address during your demonstration/POC.

You will also need a Security Key file in order to ssh into the machine. You will need to generate this yourself if you do not already have one. If you do not have an Access Key, then visit the FAQ on #aws-accounts-org.

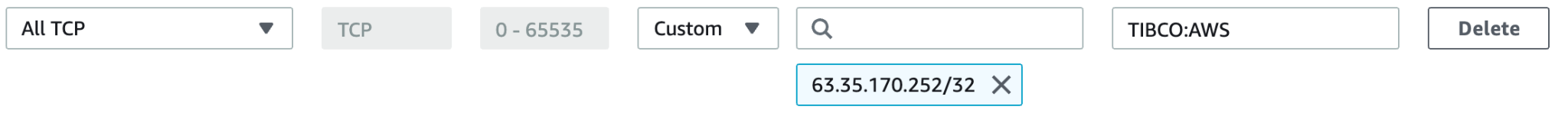
## Security Group Requirements

There are many ports that need opening, so instead of opening individual ports - allow All TCP to your/your office IP address as below:

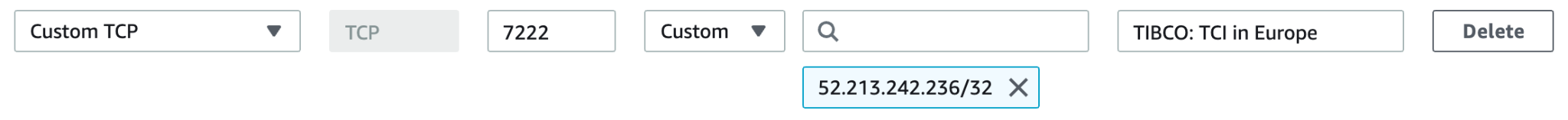


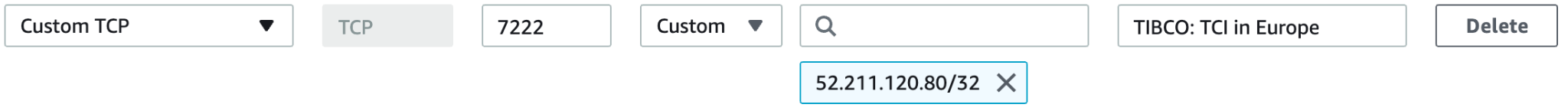


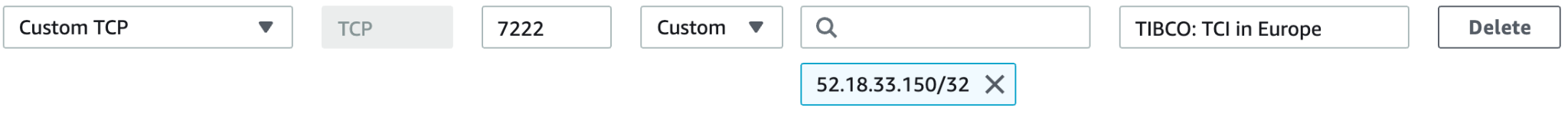
Because of the way that networking is configured as well as EMS, you will also need to open all ports to the AWS instance itself. So in the example below, if the IP address of the AWS Instance is 63.35.170.252, then you need to open all ports to that machine too as the example below.



Also, if you have any components, such as BusinessWorks Process components that are running in the TIBCO Cloud and listening to the JMS server on the machine, then you also need to ensure that ports are also opened to TCI as below:







# Connecting to the Instance

Once your instance has been created, started and the appropriate security group assigned, make a note of the IP address.

You should also have access to the appropriate AWS Access Key to allow you to connect. If you do not have an Access Key, then visit the FAQ on #aws-accounts-org.

The Image is based on Red Hat Linux, and is set up with the user ec2-user as AWS Standard.

To connect via ssh, substituting the access key file and IP\_ADDRESS appropriately:

| ​​ssh -i ~/access\_key.pem ec2-user@<IP\_ADDRESS> |
| --- |

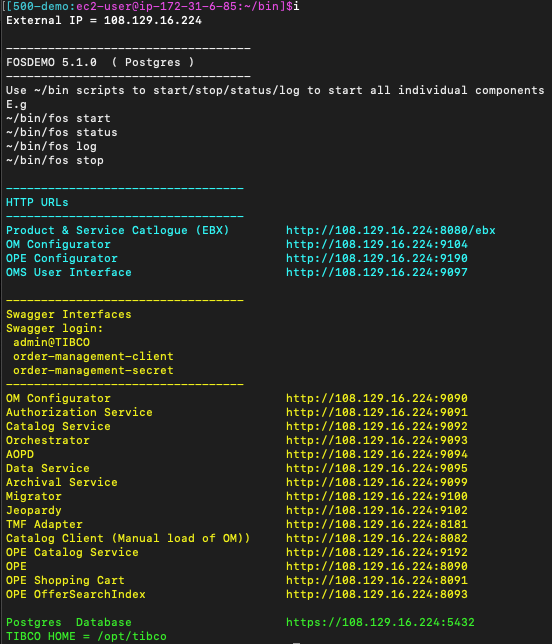
# 

# FIXTHIS:Configure OM Authorisation Service IP address. Cannot login if wrong IP

# 

# Starting TIBCO Order Management

Once logged in, ~/bin/motd.sh will automatically be run and will show the following. **Typing the alias “i” at any point in the shell will re-display this screen**.



You will notice that the script works out the external IP address so that you can easily re-use these IP addresses to get to the Web UI’s and the Swagger documentation.

To start all components within TIBCO Order Management, all you need to do is to run:

| fos start |
| --- |

Other commands that you can run are:

| fos status fos stop fos log |
| --- |

Note: When you run “fos start”, it also runs a script that updates the postgres database tables that hold the configuration details with the external IP address. The reason for this, is to allow the OAuth authentication service to work correctly when using the Swagger interfaces to send data into TIBCO Order Management.

You can also stop/start individual components through the scripts available in ~/bin, for example:

| fos-omsui start fos-ocs stop  Or  fos-bwce log  etc….. |
| --- |

# 

# Order Management URLs

HTTP User Interfaces:

| Product & Service Catalog (EBX) http://xx.xx.xx.xx:8080/ebx OM Configurator UI http://xx.xx.xx.xx:9104 OPE Configurator http://xx.xx.xx.xx:9190 OMS User Interface http://xx.xx.xx.xx:9097 ~~OCS http://xx.xx.xx.xx:9098~~ |
| --- |

Swagger Interfaces:

| Configuration http://xx.xx.xx.xx:9090  Authorization Service http://xx.xx.xx.xx:9091 Catalog Service http://xx.xx.xx.xx:9092 Orchestrator http://xx.xx.xx.xx:9093 AOPD http://xx.xx.xx.xx:9094 Data Service http://xx.xx.xx.xx:9095 OCS http://xx.xx.xx.xx:9098 Archival Service http://xx.xx.xx.xx:9099 Migrator http://xx.xx.xx.xx:9100 Jeopardy <http://xx.xx.xx.xx:9102>  Catalog Client http://xx.xx.xx.xx:8082 TMF Adapter http://xx.xx.xx.xx:8181 OPE Catalog Service http://xx.xx.xx.xx:9192 OPE http://xx.xx.xx.xx:8090 OPE Shopping Cart http://xx.xx.xx.xx:8091 OPE OfferSearchIndex http://xx.xx.xx.xx:8093 |
| --- |

# Usernames & Passwords

Product & Service Catalog (EBX): Username = admin, Password = admin

OM Configurator: Username = admin@TIBCO, Password = admin

OPE Configurator: Username = admin@TIBCO, Password = admin

OMS User Interface: Username = admin, Password = admin

~~OCS: Username = admin, Password = admin~~

Postgres maintenance db Username = postgres, Password = tibco123.

# GITHUB files

Although the AMI icontains everything and should work OOTB key source files have been saved on github should there be a need to change or recover the AMI from scratch.

Files are stored on Git and can be found in the foillowing repository:

<https://github.com/TIBCOUK/FOSDemo>

There are two branches for historic reasons: Main and Master.

* Main branch contains:
  + JSON seed configurations,
  + Configuration and component management scripts,
  + This document
  + To Be Added : roles/standalone/application.properties file for each FOS component
* Master contains the BWCE process component sources:
  + [TIBCO\_OM\_PlanFragments.module](https://github.com/TIBCOUK/FOSDemo/tree/master/TIBCO_OM_PlanFragments.module)
  + [TIBCO\_OM\_PlanFragments](https://github.com/TIBCOUK/FOSDemo/tree/master/TIBCO_OM_PlanFragments)

# 

# BusinessWorks Sample Process Component

The image contains simple processes to handle, happy day, respond with error to support cancel/ammend scenario and batch. Top level BW processes are:

* generic\_demo\_planfragment\_component
* FOSBatchOrderProcess.bwp

We created a sample BW Process that is configured to listen to the EMS Queue to receive notifications of Actions that BW needs to fulfill as part of the order. This was just a simple process for this demonstration.

The source code can be found at : <https://github.com/TIBCOUK/FOSDemo> (Master Branch)

## Generic Process Component:

There is a generic BW process for handling happy day orders and via UDF on the order can respond with Business or Technical error depending on the order’s UDF value:. The UDF id called “simulate”.

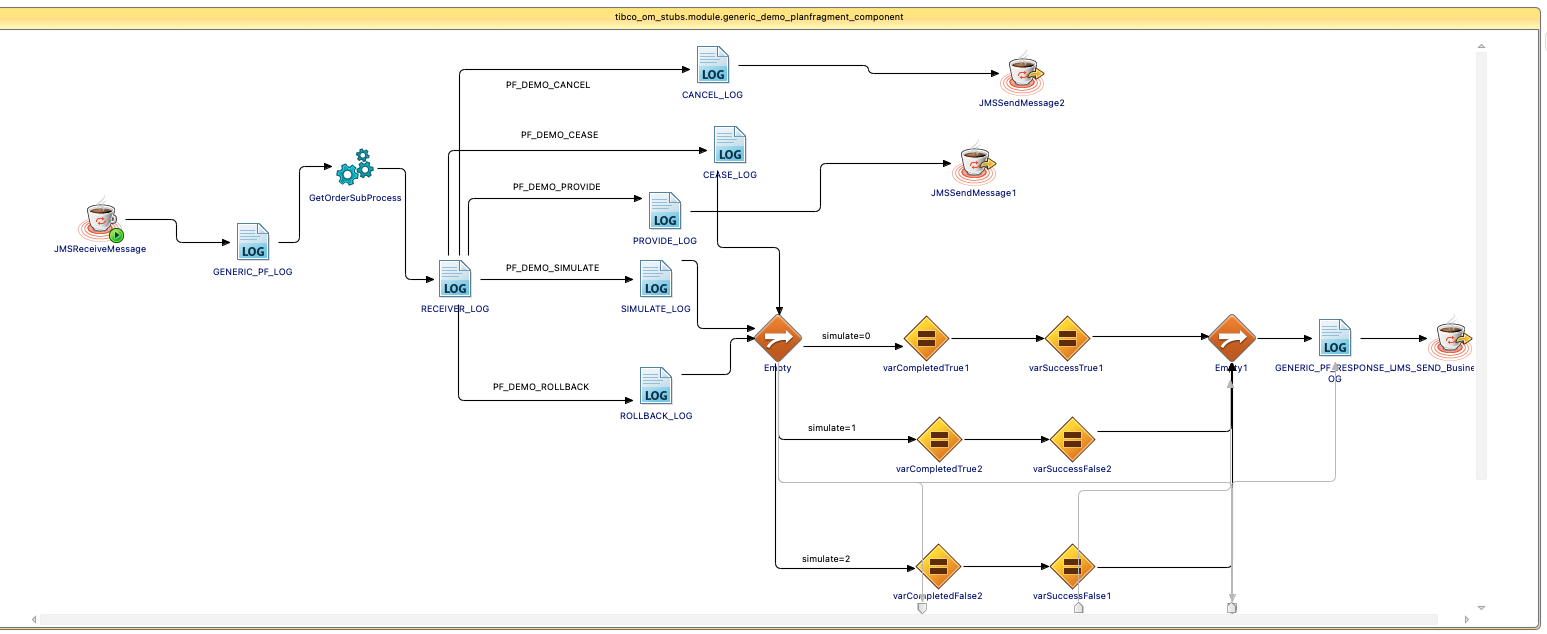
Simulate = 0 : Respond with true,true : Success

Simulate = 1 : Respond with true,false : Business Error

Simulate = 2 : Respond with false,false : Technical Error

The diagram below shows the supported Plan Franment supported.

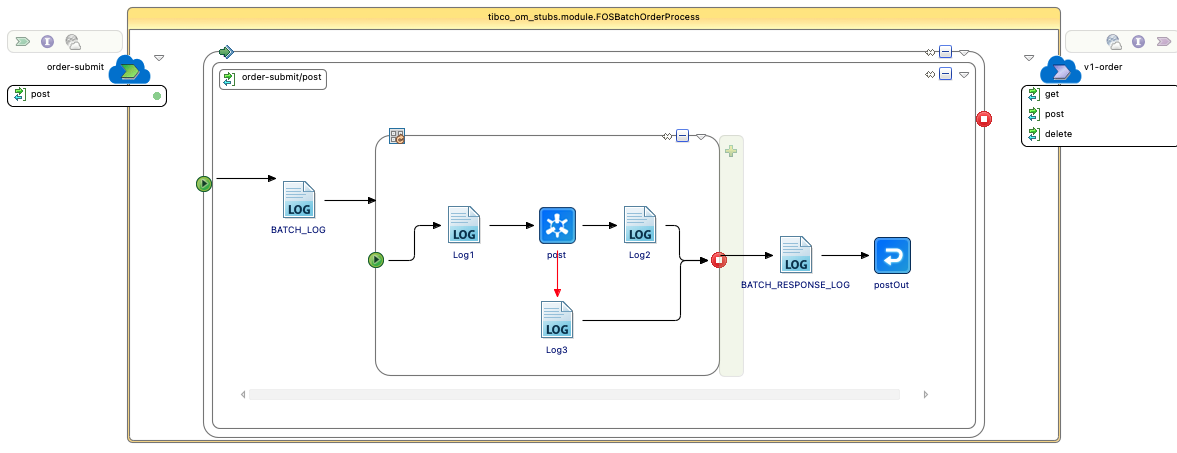
FIX THIS: Process flow is looking at PF NAME not PF ID. Be careful.



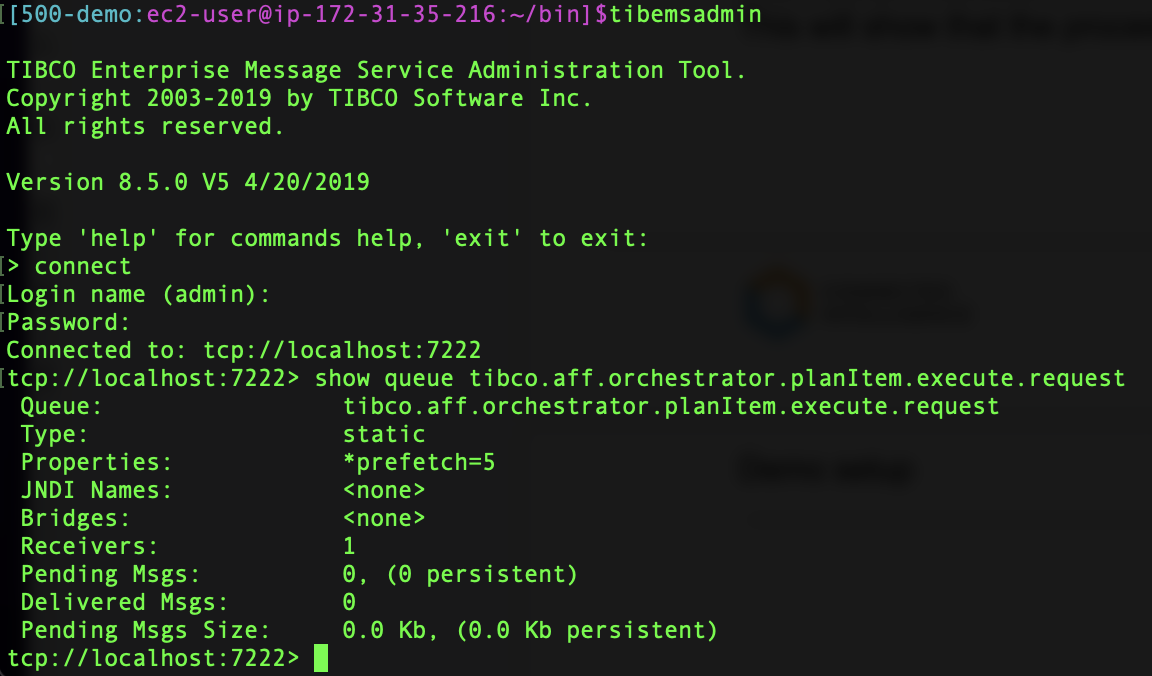
See example order in following section.

## Batch Process Component:

FIXTHIS : Worked but to be documented.



When the App is deployed and started, you can use tibemsadmin to view the queue tibco.aff.orchestrator.planItem.execute.request to check that there are “Receivers” on the queue. This will show that the process is ready and waiting for JMS messages to be sent.



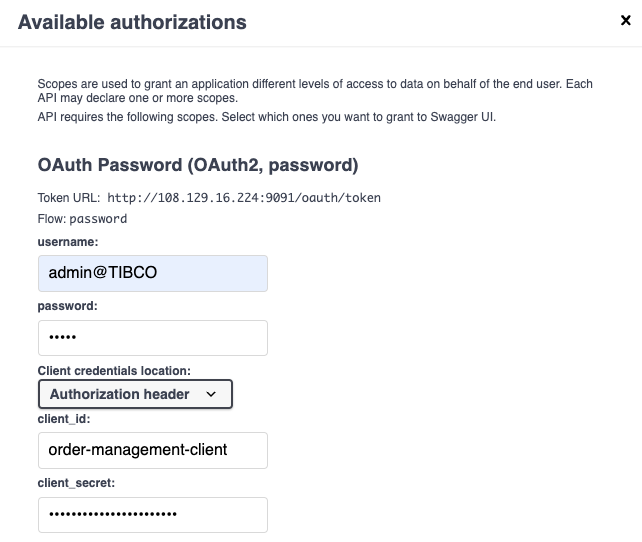
# Sending an Order via the Swagger Interface

Open the Swagger interface for the Orchestrator running on port 9093:



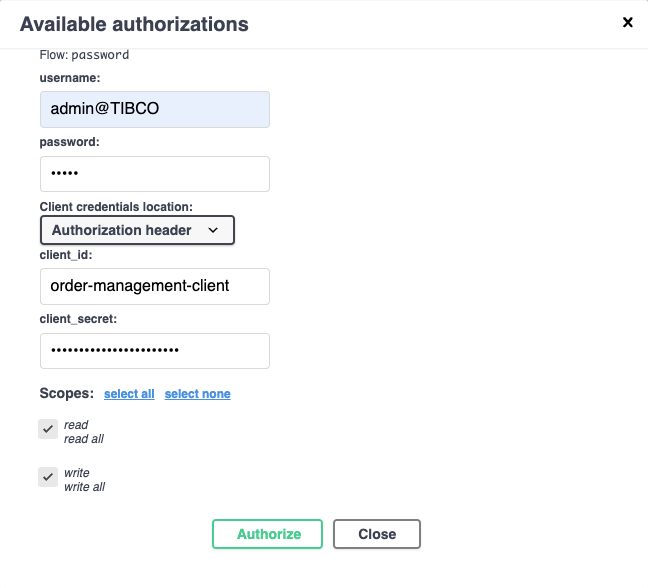
## Authorizing

Before using any of the APIs you need to “Authorize” yourself. Click the “Authorize” button:



The username/password is admin@TIBCO/admin. The client\_secret is order-management-secret

Scroll down the dialog:



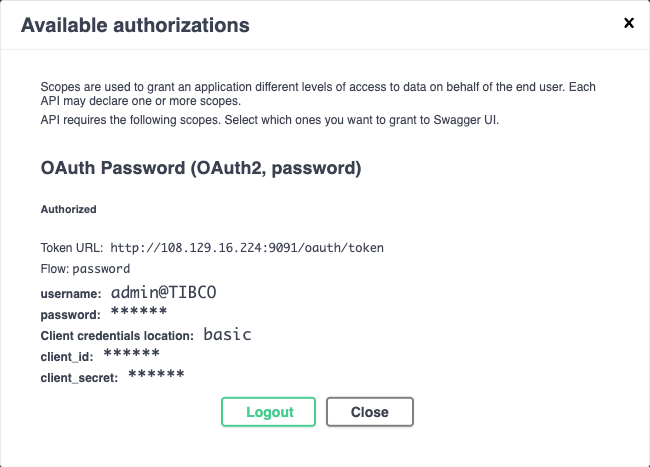
Click on both Read all and write all, and click Authorize.

You may find that you have a “Failed to Fetch” error message. This is due to CORS security settings. If you are using Chrome - you can restart chrome from the command line using the following switch to disable CORS:

| open -a Google\ Chrome --args --disable-web-security --user-data-dir |
| --- |

A good guide to follow for other browsers and situations can be found here: <https://medium.com/swlh/avoiding-cors-errors-on-localhost-in-2020-5a656ed8cefa>

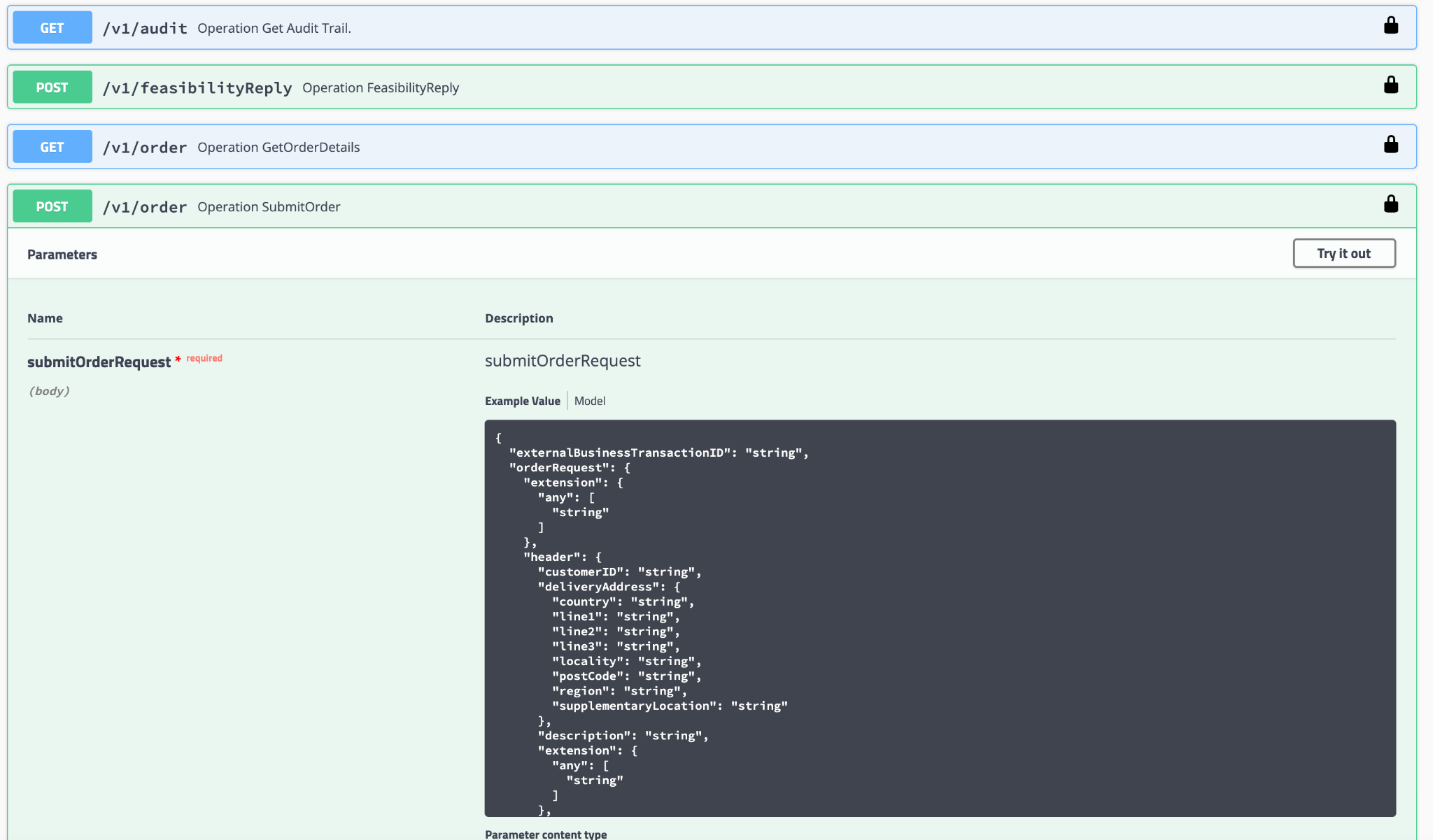
When authorized - you will see the following dialog - just press Close at this point:



This process is the same across all Swagger Interfaces.

## Sending a Sample Order

Scroll down and expand the “POST” method for operation SubmitOrder and click on “Try it out”:

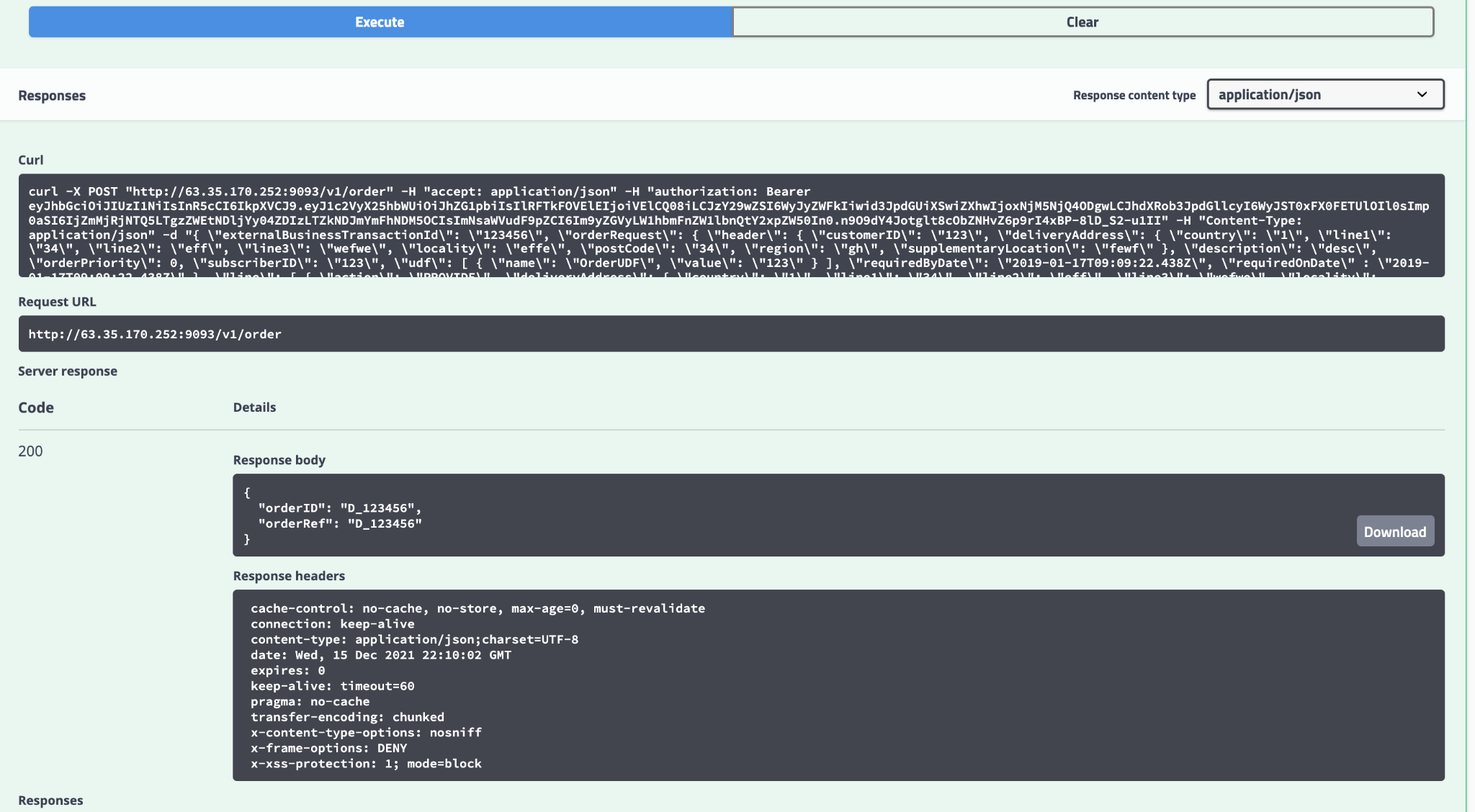


Replace the JSON in the box with the following sample Order. Replace the OrderID and any other details as you see fit.

| {  "externalBusinessTransactionId": "Iain",  "orderRequest": {  "header": {  "customerID": "07776257046",  "deliveryAddress": {  "country": "44",  "line1": "34",  "line2": "eff",  "line3": "wefwe",  "locality": "effe",  "postCode": "34",  "region": "gh",  "supplementaryLocation": "Portsmouth"  },  "description": "desc",  "orderPriority": 0,  "subscriberID": "123",  "udf": [  {  "name": "OrderUDF",  "value": "123"  }  ],  "requiredByDate": "2019-01-17T09:09:22.438Z",  "requiredOnDate" : "2019-01-17T09:09:22.438Z"  },  "line": [  {  "action": "PROVIDE",  "deliveryAddress": {  "country": "44",  "line1": "34",  "line2": "eff",  "line3": "wefwe",  "locality": "effe",  "postCode": "34",  "region": "gh",  "supplementaryLocation": "Portsmouth"  },  "lineNumber": "1",  "productID": "MS\_01",  "productVersion": "1",  "quantity": 1,  "subscriberID": "123",  "requiredByDate": "2019-01-17T09:09:22.438Z",  "requiredOnDate" : "2019-01-17T09:09:22.438Z",  "udf": [  {  "name": "simulate",  "value": "1"  }  ],  "uom": "uom"  }  ],  "orderRef": "990082"  }  } |
| --- |

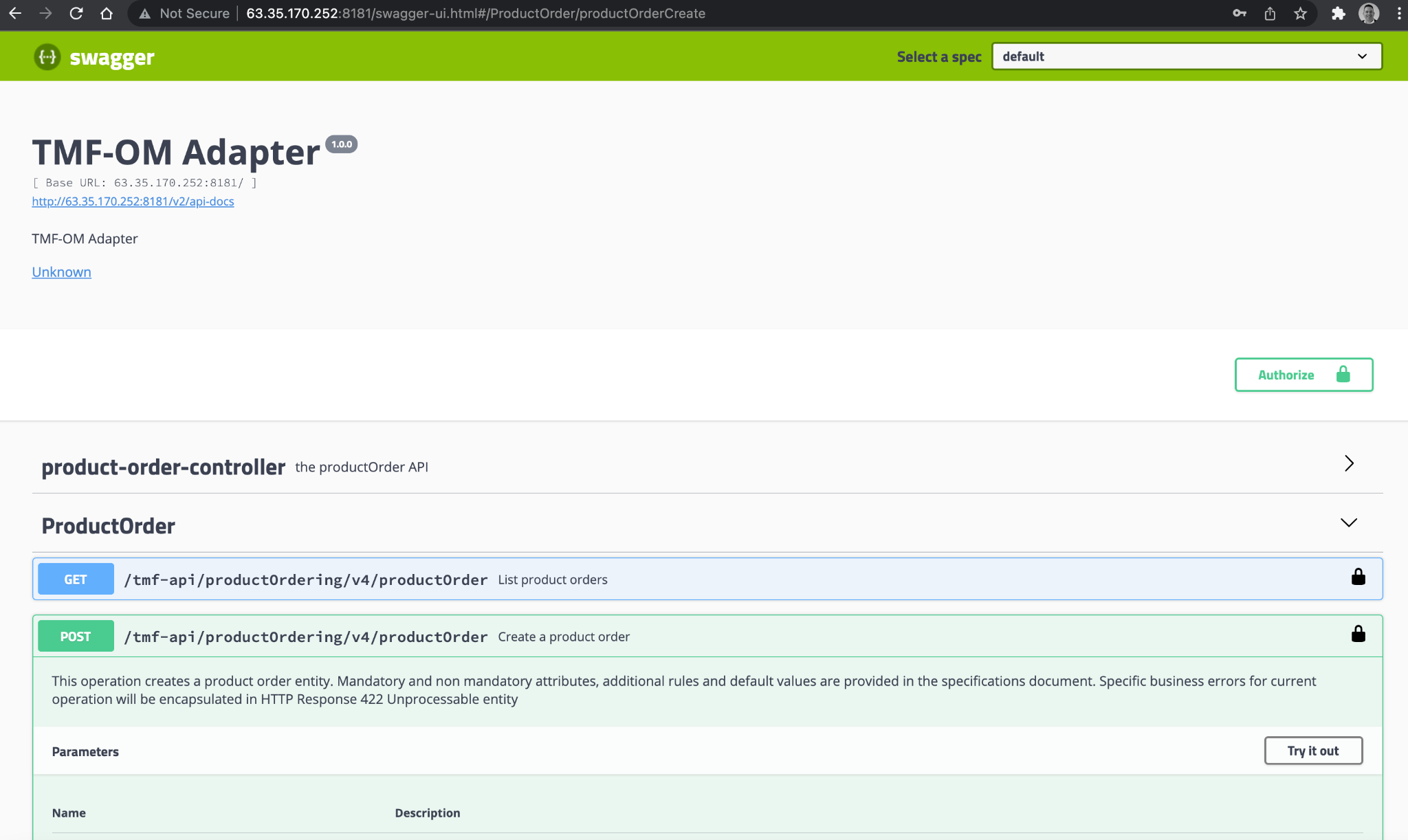
PO\_PAYG, MS\_01, PO\_01 have been tested in the base platform at this stage. As you expand the product catalog (See later sections) you will be able to order further products.

When ready, click “Execute” and you will see a response similar to the following:



# Sending an Order via the TMF Adapter Swagger Interface

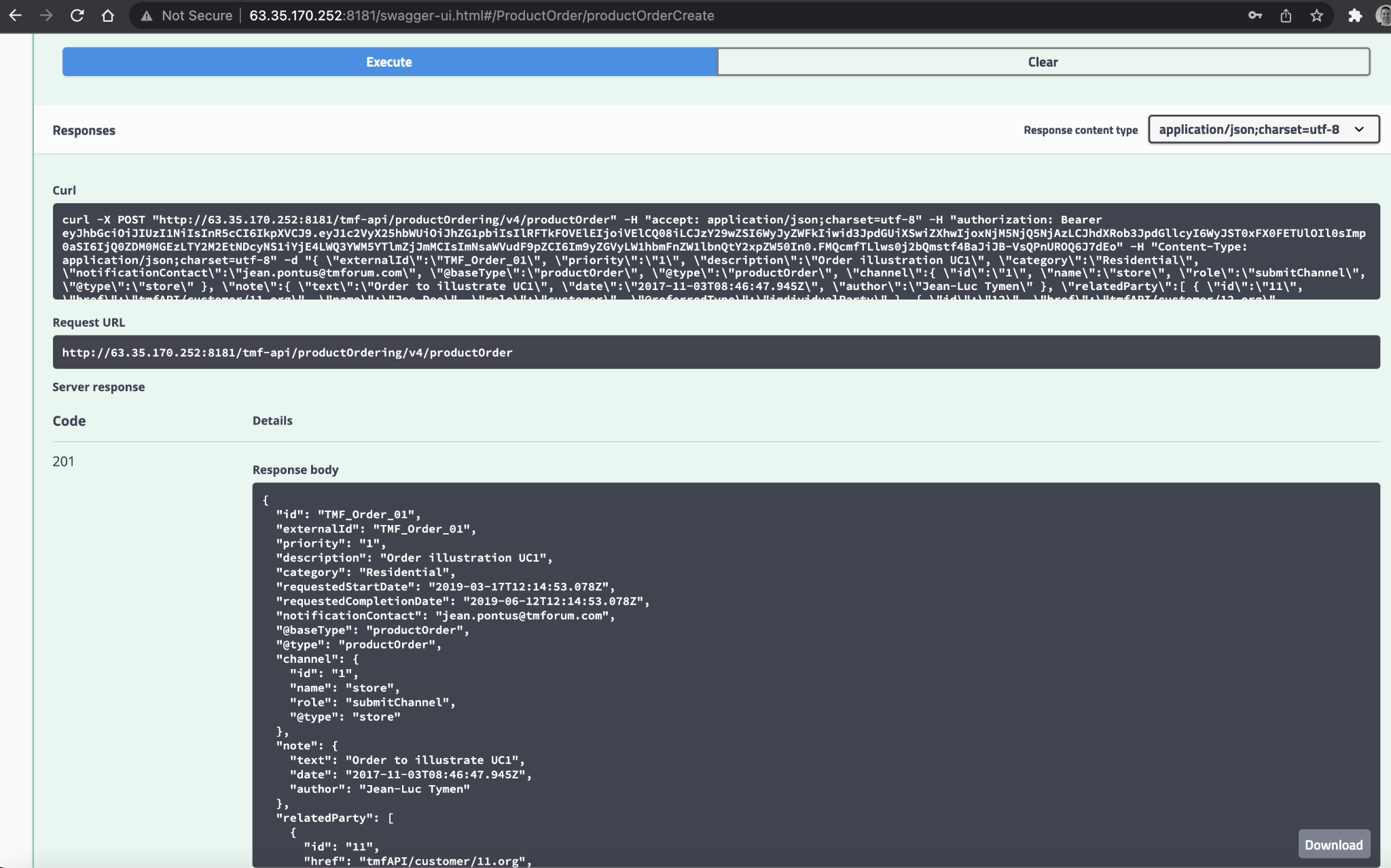
Open the Swagger interface for the TMF Adapter running on Port 8181, and Authorize through the OAuth dialog as completed previously.



Expand the POST method for the Product Order operation and click “Try it out” button, then replace the JSON with the following sample order:

| {  "externalId":"Telco01",  "priority":"1",  "description":"Order illustration UC1",  "category":"Residential",  "notificationContact":"jean.pontus@tmforum.com",  "@baseType":"productOrder",  "@type":"productOrder",  "channel":{  "id":"1",  "name":"store",  "role":"submitChannel",  "@type":"store"  },  "note":{  "text":"Order to illustrate UC1",  "date":"2017-11-03T08:46:47.945Z",  "author":"Jean-Luc Tymen"  },  "relatedParty":[  {  "id":"11",  "href":"tmfAPI/customer/11.org",  "name":"Joe Doe",  "role":"customer",  "@referredType":"individualParty"  },  {  "id":"12",  "href":"tmfAPI/customer/12.org",  "name":"Arthur Pence",  "role":"subscriber",  "@referredType":"individualParty"  }  ],  "billingAccount":{  "id":"BA1513",  "href":"tmfAPI/billingAccount/BA1513.org",  "name":"BA01",  "@referredType":"billingAccount"  },  "orderItem":[  {  "id":"100",  "action":"add",  "quantity":1,  "productOffering":{  "id":"14277",  "name":"PO\_PAYG",  "href":"href"  }  }  ],  "requestedCompletionDate": "2019-06-12T17:44:53.078+05:30",  "requestedStartDate": "2019-03-17T17:44:53.078+05:30" } |
| --- |

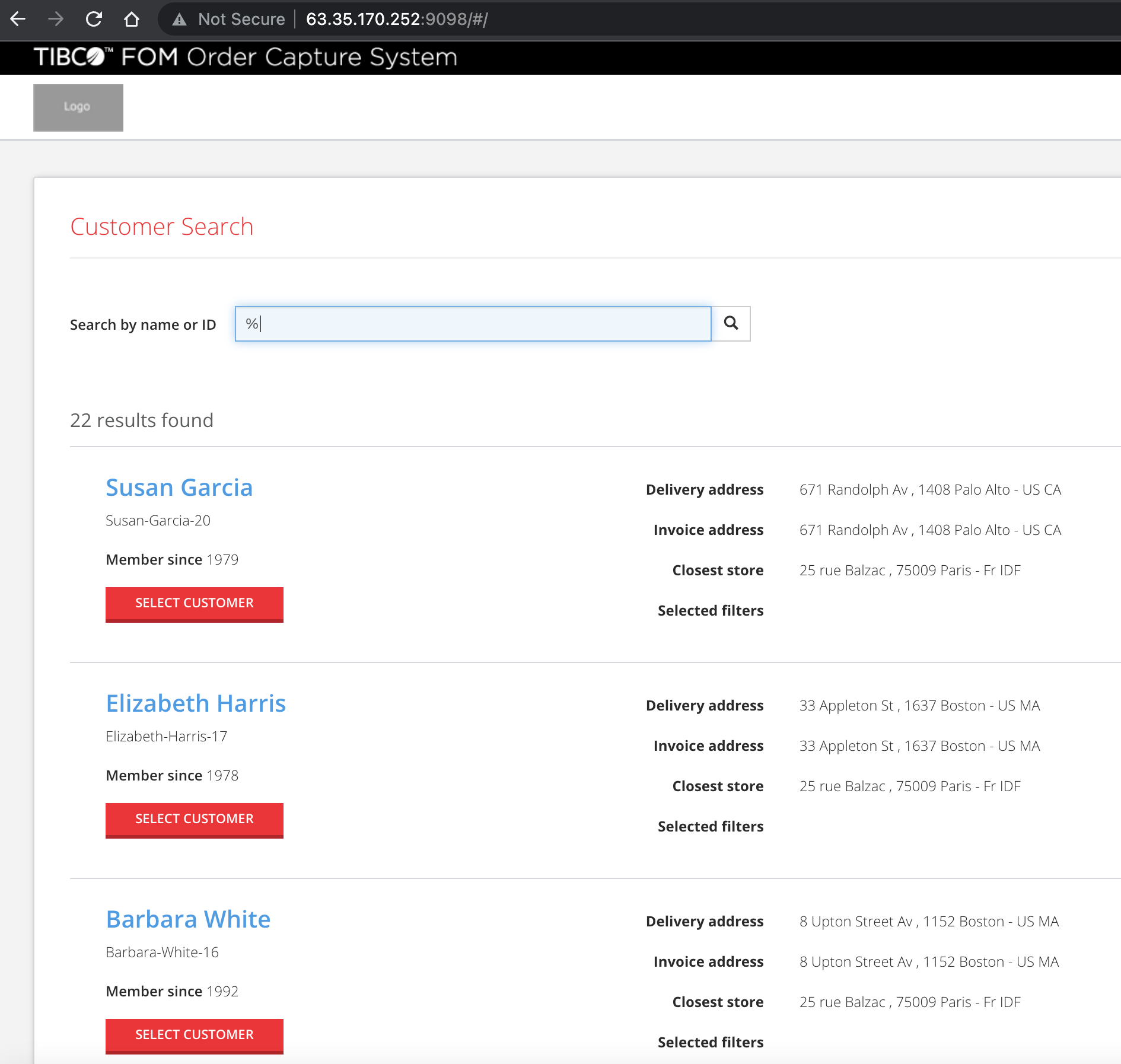
Replace the “externalID” value currently “Telco01” with your own Order Number. After pressing “Execute”, you will see a response similar to the following:



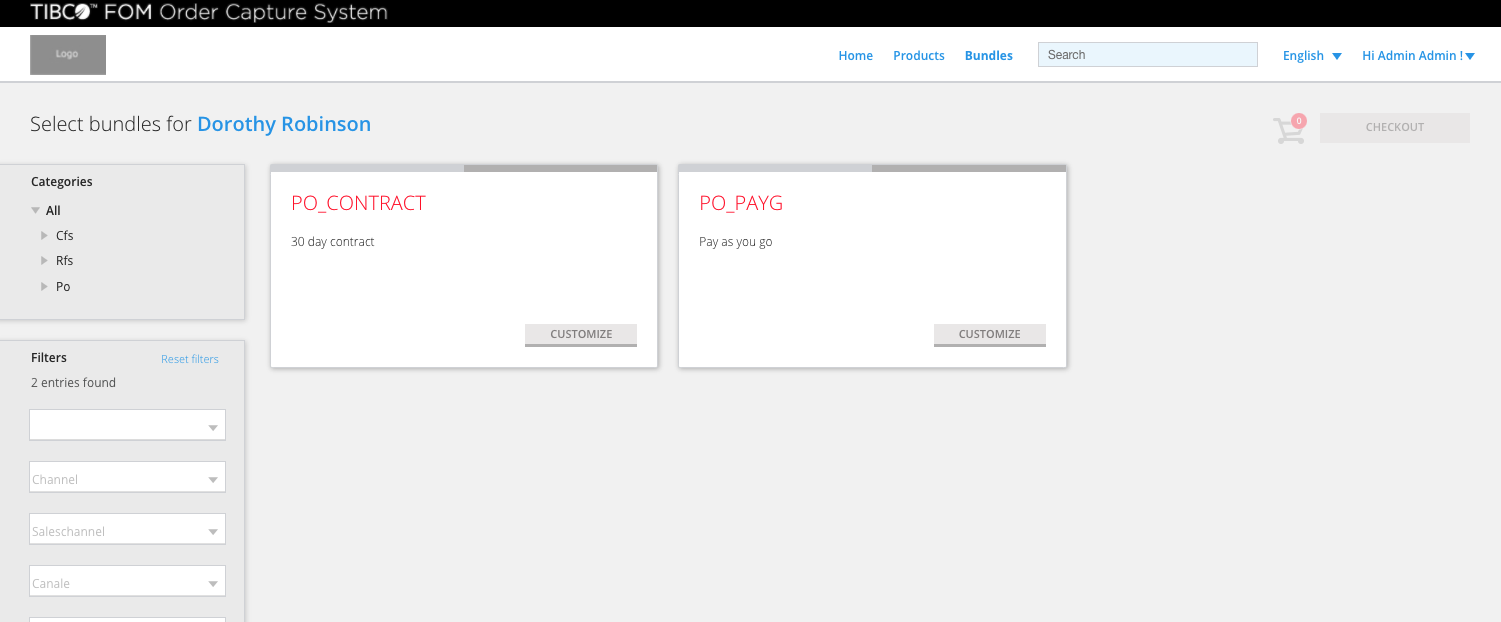
# Sending an Order through the OCS UI (OCS Depricated in 5.1)

Open the Order Capture System user interface which is running on Port 9098. Username/Password is admin/admin

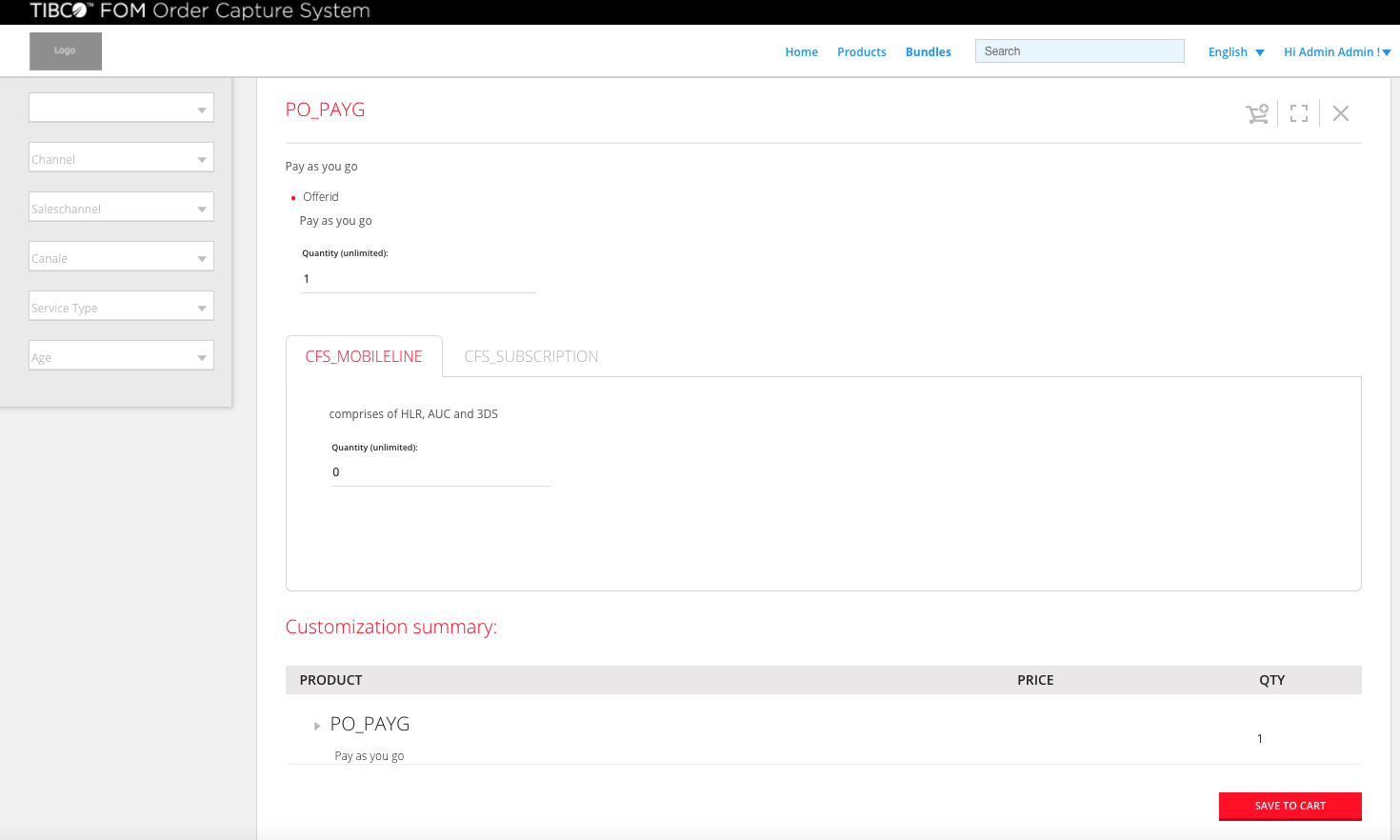
Once logged in, click on “Home” and use the % wildcard to search for customers and choose a customer:



Then you will be shown a list of products that the customer can place an Order For:

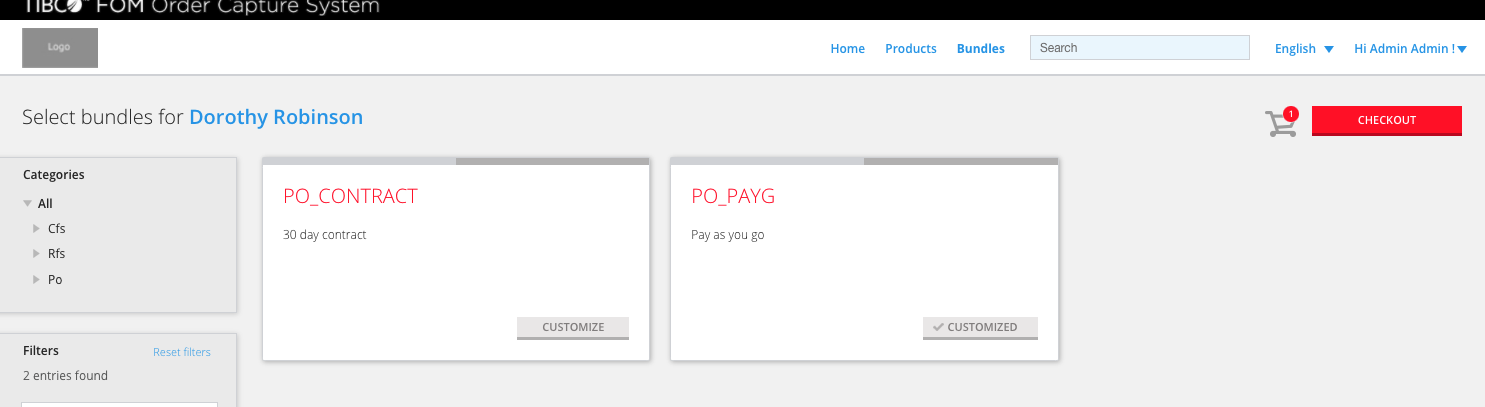


Click on “Customize” for the PO\_PAYG product:

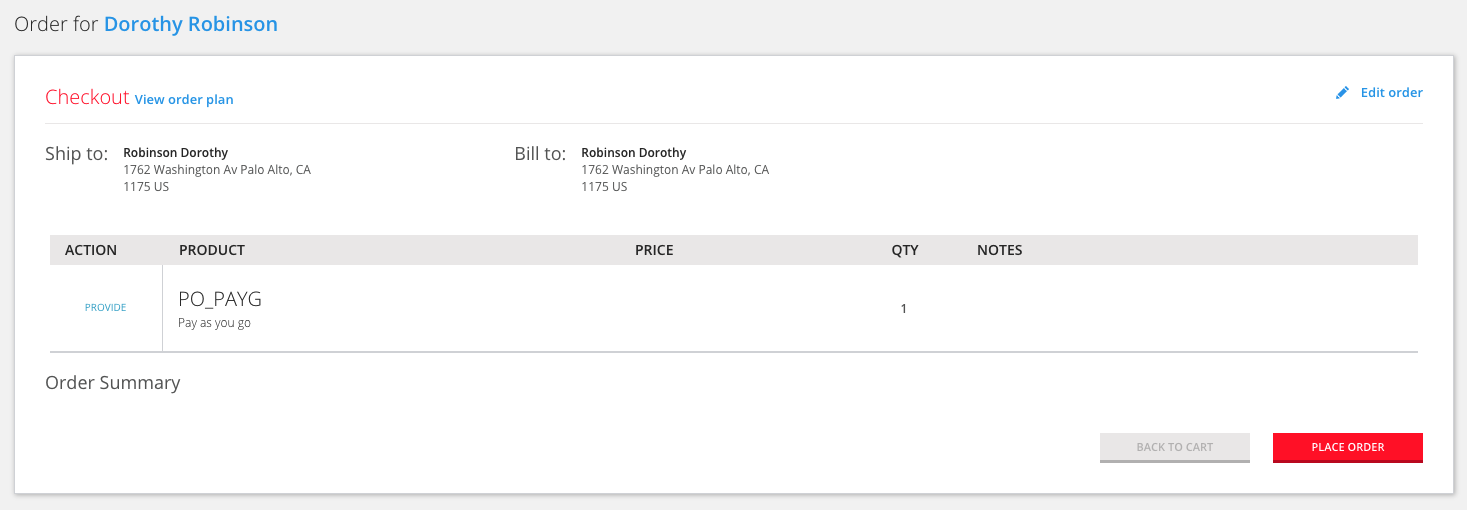


You can see the details of the Product, change the qty etc. and click on “Save to Cart” or click on the shopping cart icon in the top right corner.

Click on the Checkout button:



Followed by “Place Order”:



And that’s it - your order is placed!

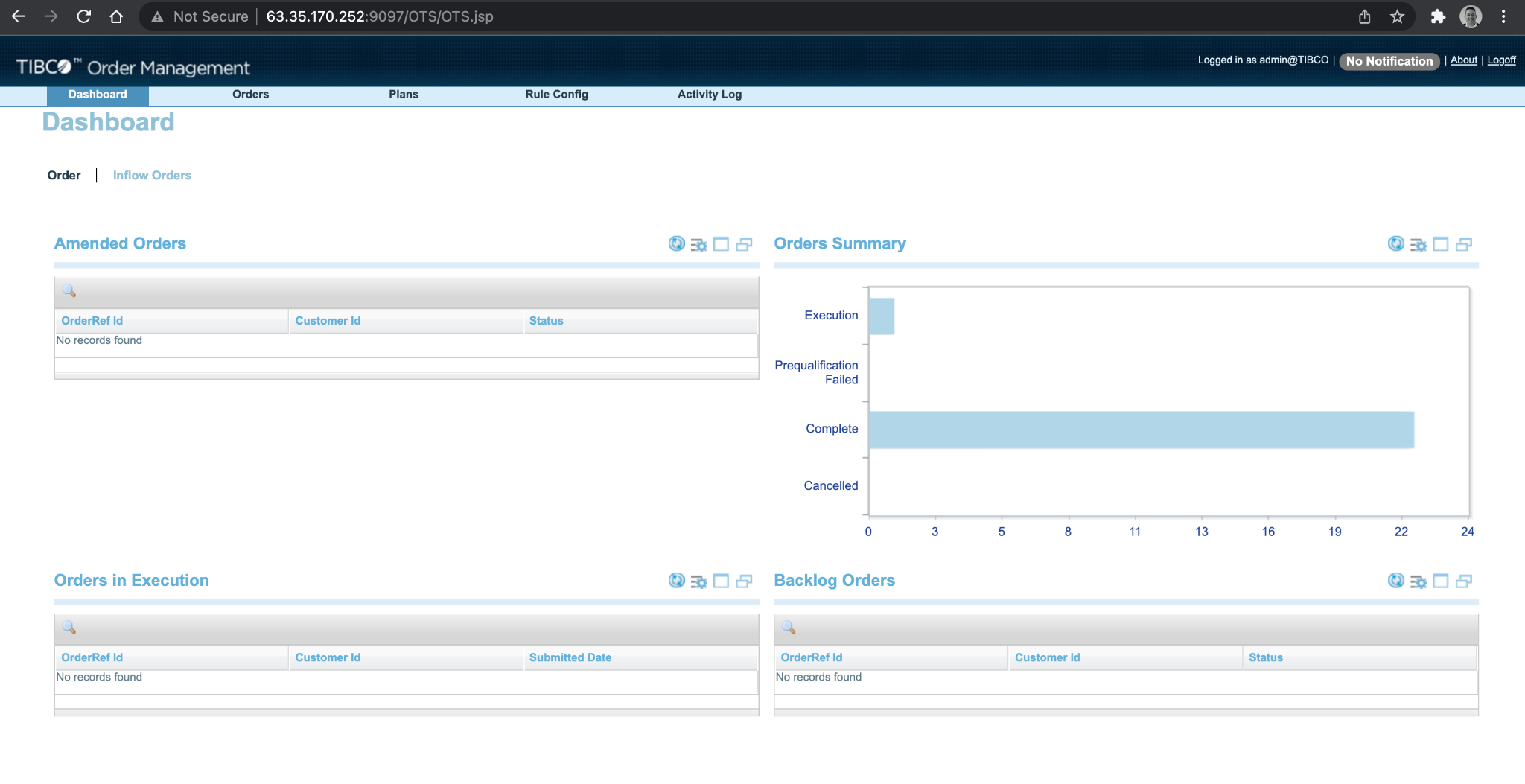
You will see in the TIBCO Order Management UI (see next section) that your Order has an Order ID that starts with OCS\* to identify where the order came from:



# 

# Checking Order Status

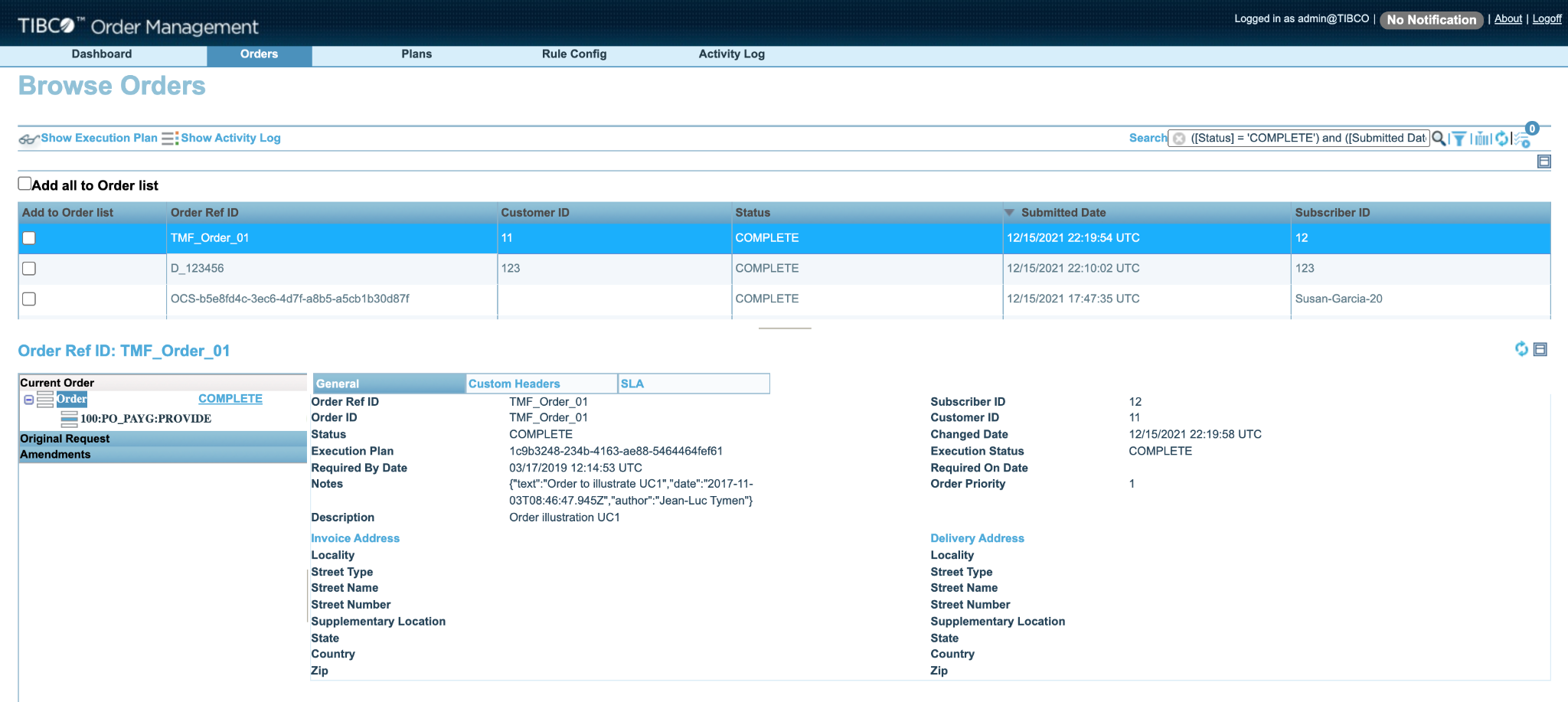
Open the Order Management UI which is running on Port 9097 (username/password = admin/admin)



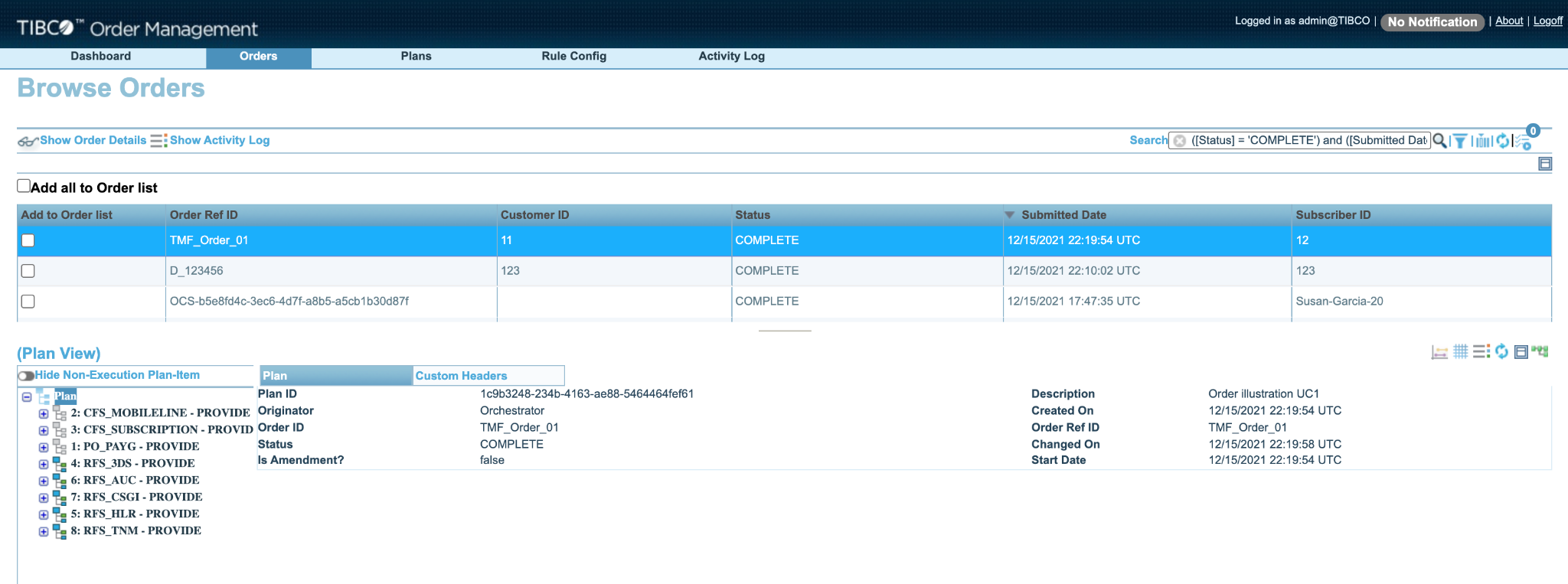
Click on the “Complete” bar of the Order Summary Chart to take you to all completed orders.



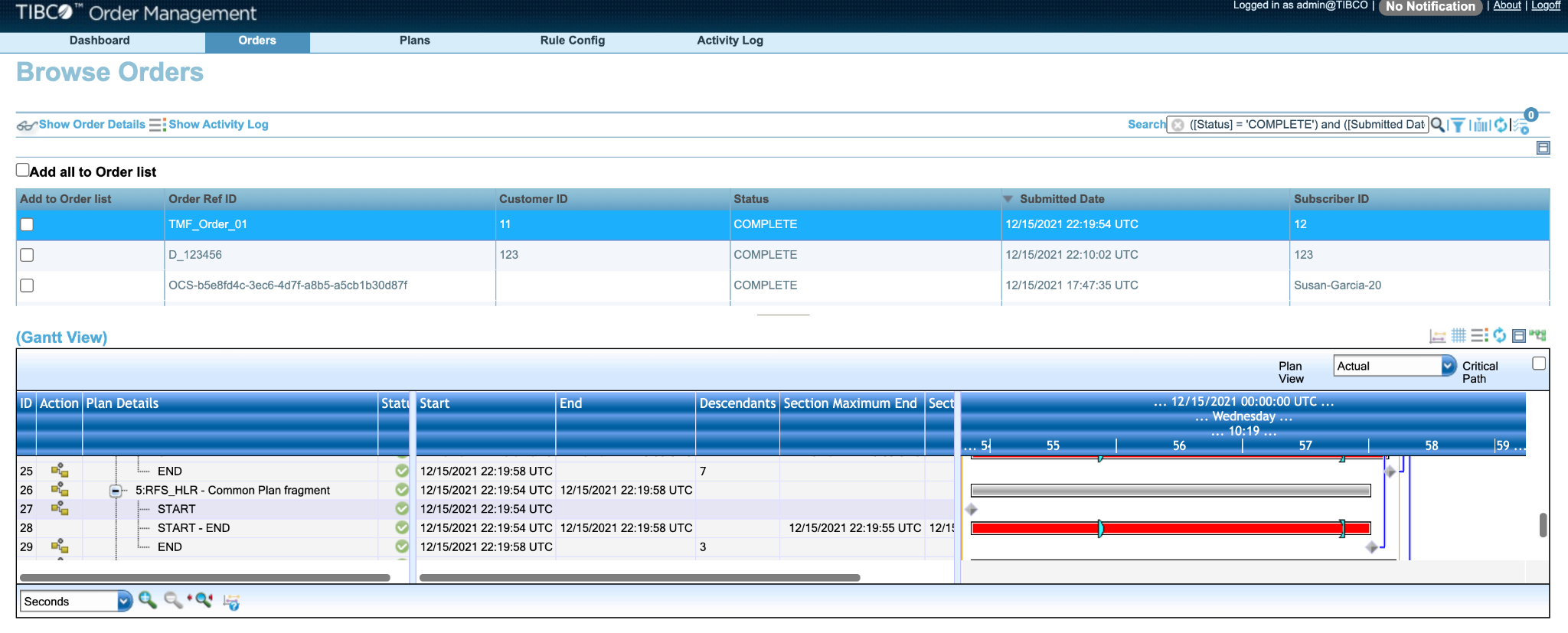
In my screenshot - the top two orders are the ones we sent via the Swagger interfaces. Click on one of the orders to inspect:



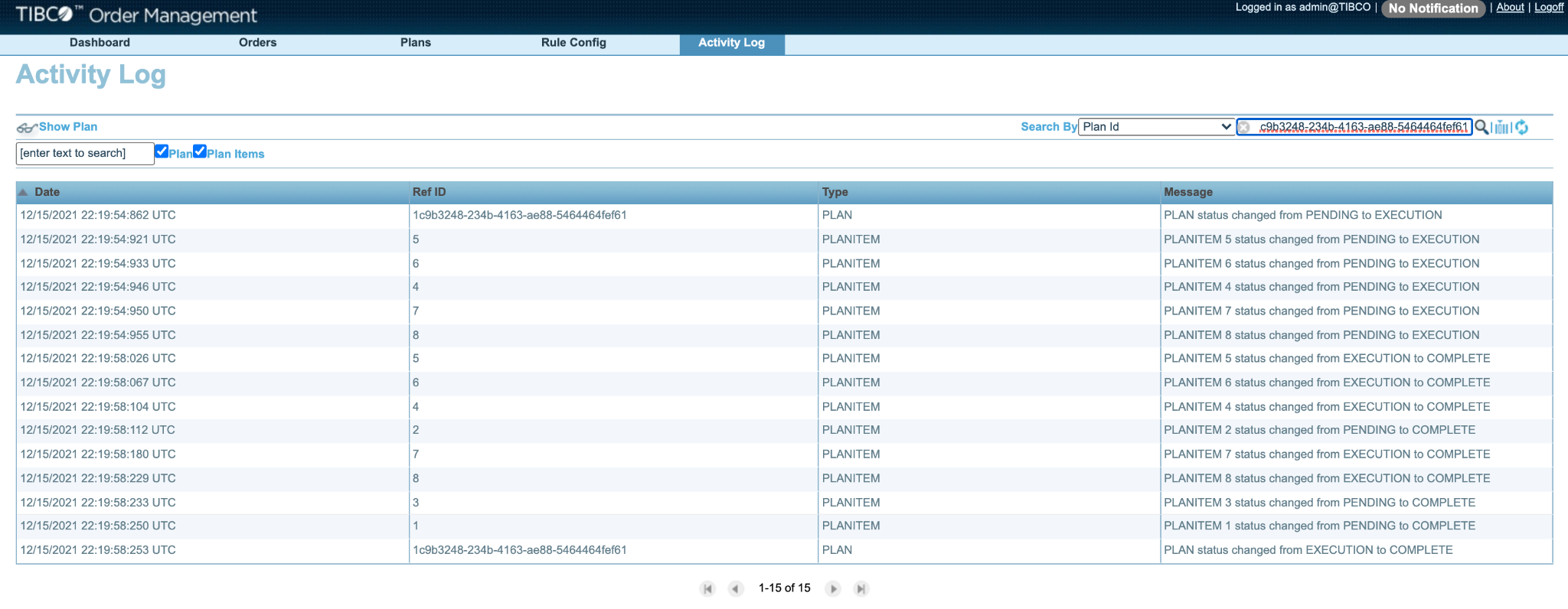
Click on “Show Execution Plan”:



On the left hand side - you can see the order breakdown and you can use the icons on the right hand side to view a Gantt chart or Audit trail.

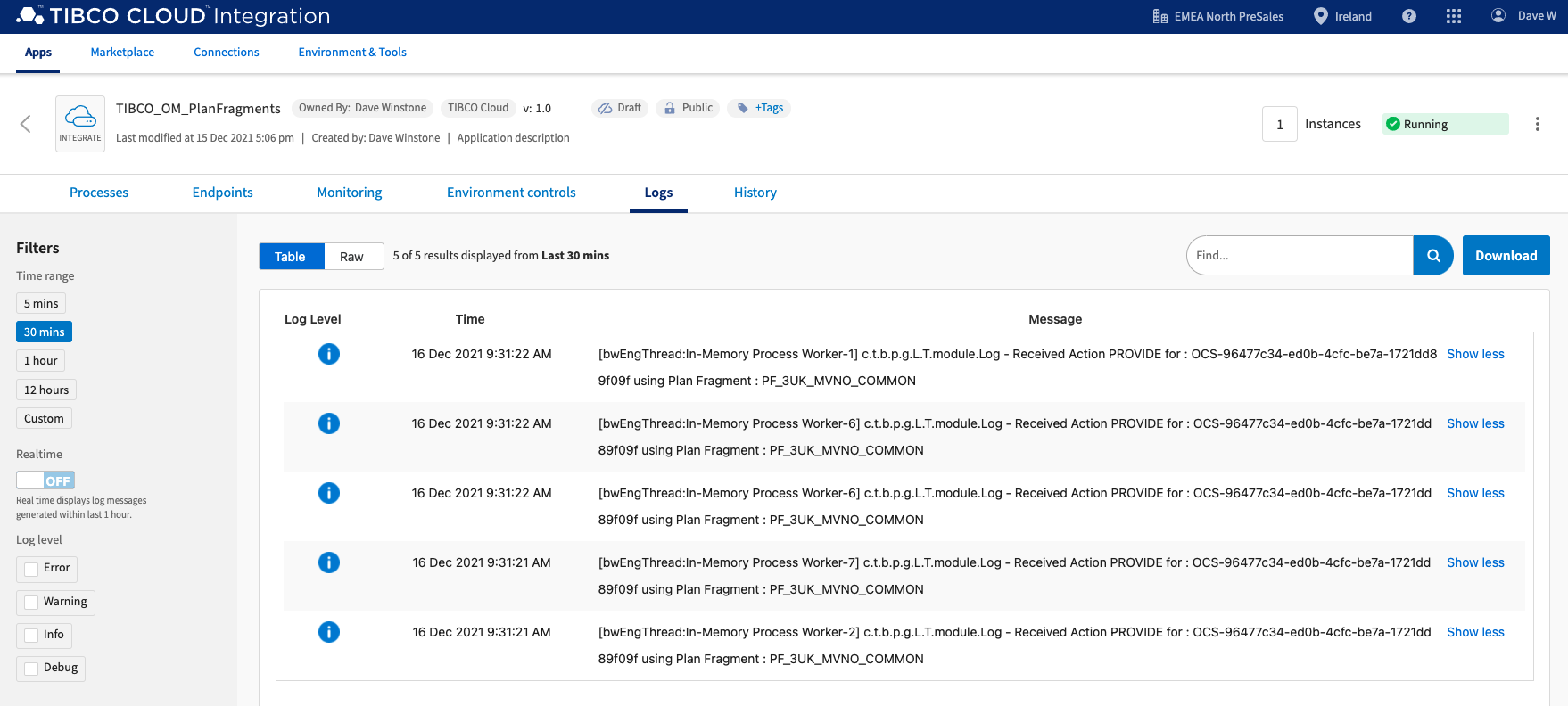


NOTE: The GANTT Chart View is best displayed in ***Firefox***. Please ensure you use this browser for most of your demo.



# Checking TCI BusinessWorks Logs

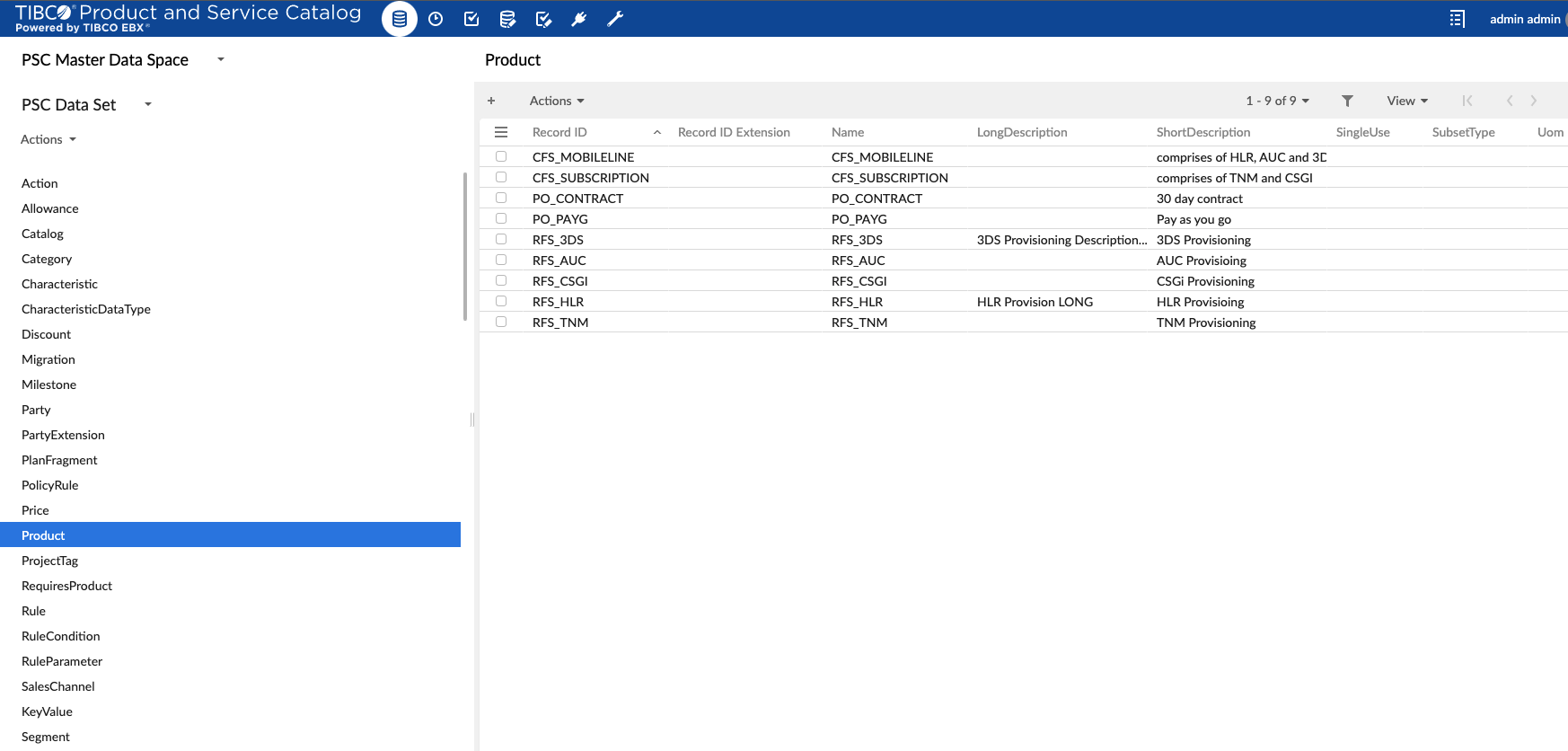
Log back into your TIBCO Cloud instance and navigate to the BusinessWorks Process Component App that we deployed earlier, and view the Logs:



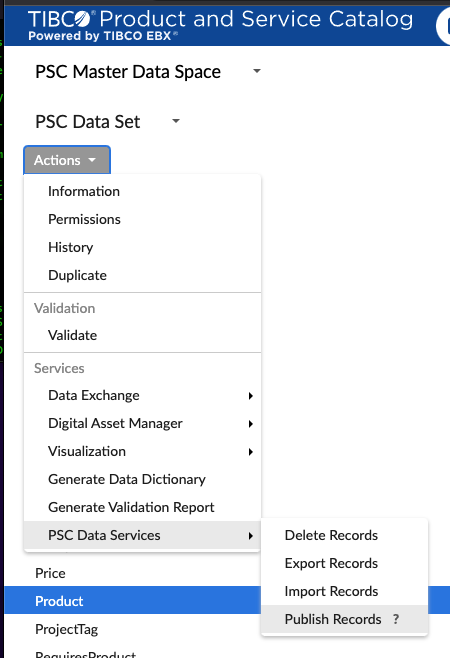
You can see from the log entries that we received the PROVIDE action for the different components of the Order.

# Publishing Catalog Changes to Order Management & OCS

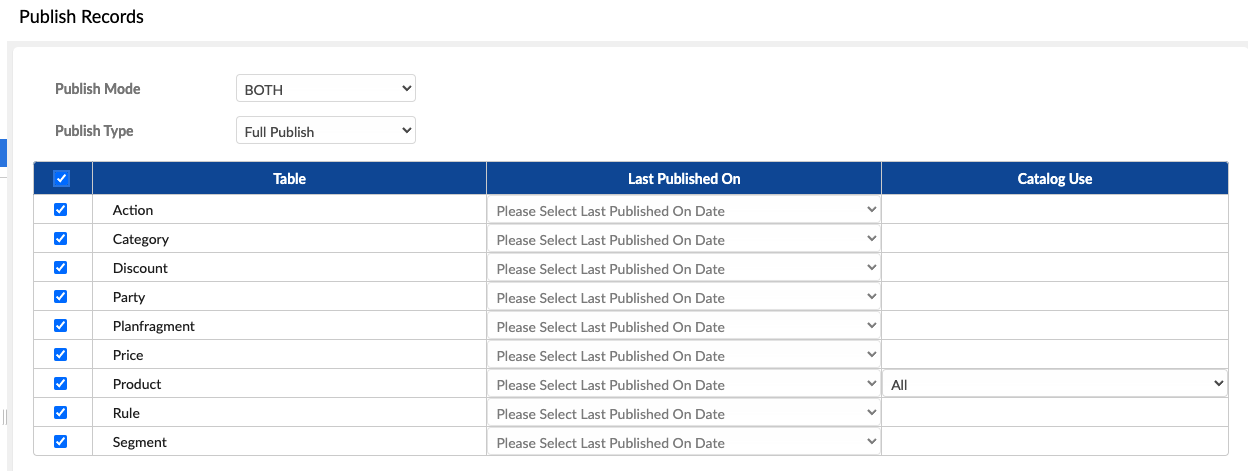
Log into the PSC-EBX page running on port 8080. Username/Password = admin/admin



Click on “Actions” in the top left of the page, scroll down to PSC Data Services and select “Publish Records”:



Change the “Publish Mode” to “BOTH” (options are JMS or File), choose “Full Publish” as the Publish Type and select which tables you want to Publish. Click on Publish at the bottom of the page when ready.



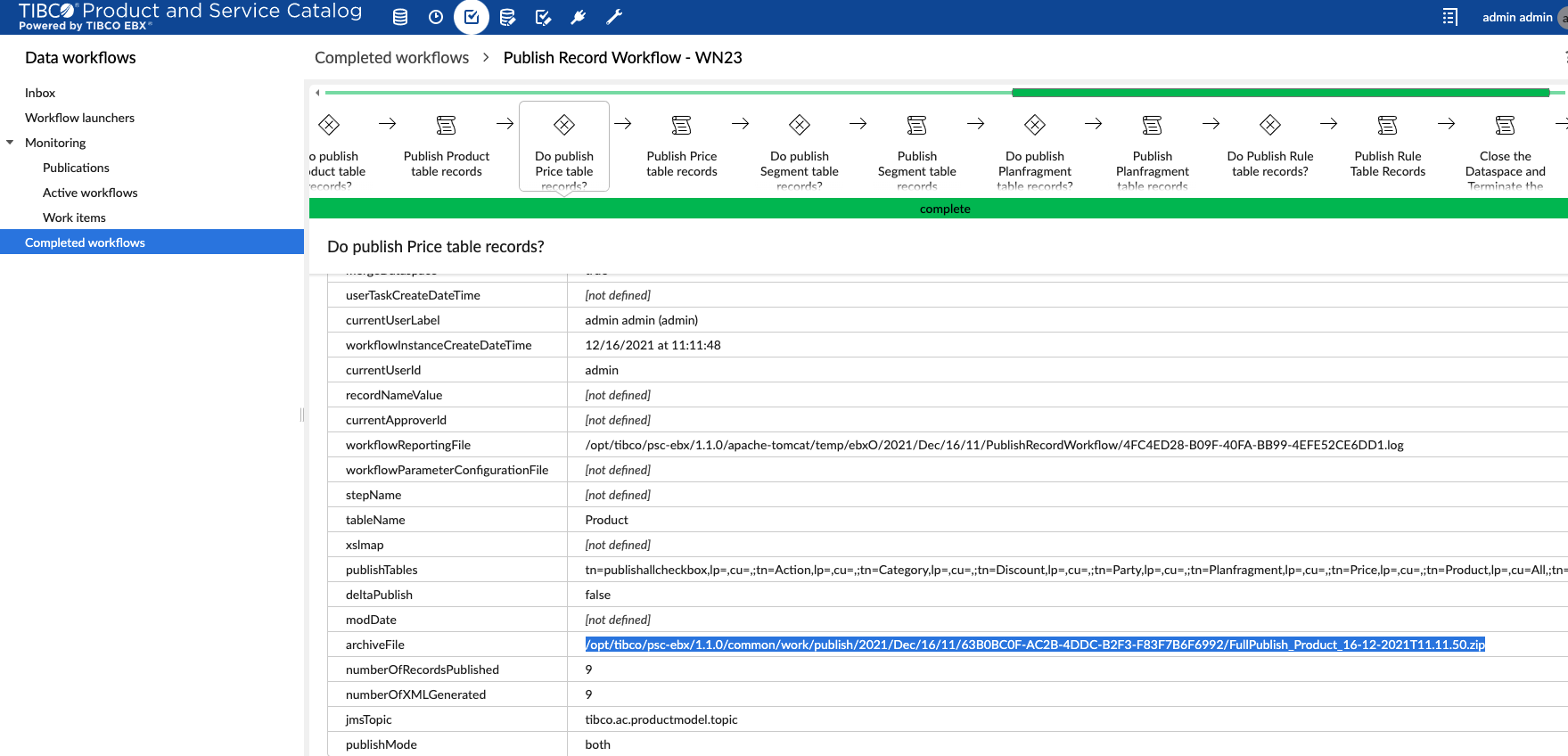
You get an option to view the workflow in progress - but in reality it’s so fast it will already be completed!

TIBCO Order Management receives the new data over JMS and automatically updates the catalog, and the products are ready to be ordered.

The Order Capture System (OCS) does not support the JMS method and only uses the File method (hence selecting “BOTH” earlier). Read the next section to update the Order Capture System.

# Updating OCS Catalog

When Publishing data to a file from the PSC-EBX Platform, you’ll have to view the workflow in order to determine the location of the exported file:



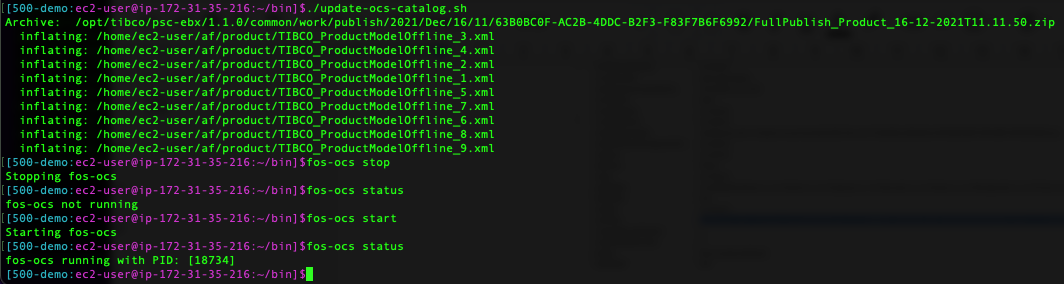
The highlighted archive file in the screenshot above is the latest generated file. This would need to be located in the file system and unzipped into the OCS catalog file location (~/af/product) and OCS restarted to read the new product details etc.

We have created a script that locates and unzips the latest export for you, and this can be found in ~/bin/update-ocs-catalog.sh

| #!/bin/sh  #Find the latest full publish of the product catalog in the last 10 minutes (-mmin 10) and copy to the OCS catalog directory, then unzip find /opt/tibco/psc-ebx/1.1.0/common/work/publish -name FullPublish\_Product\* -mmin -10 -exec unzip -o {} -d ~/af/product \; |
| --- |

This find command looks for all FullPublish\_Product\* files that have been changed in the last 10 minutes, in the appropriate publish directory and then unzips it in the correct OCS directory.

All the remains is that you stop and start the OCS component to ensure the latest product catalog is being used:



At this point, OCS is now using the latest Product Catalog and the new products can be ordered (assuming they are configured correctly!)

# Configure OM UI

For more readable OMUI in respect of Plan Item name set the following 3 configurations in the OMUI section of the configurator. This will place the actual Product, PlanFragment and PlanIten names on the UI other wise you see names like 1,2,3 etc.

| com.tibco.af.omsui.planItemExpression.dependencyView | {ProductID} | {PlanFragmentID} | {PlanItemID} |
| --- | --- |
| com.tibco.af.omsui.planItemExpression.ganttView | {ProductID} | {PlanFragmentID} | {PlanItemID} |
| com.tibco.af.omsui.planItemExpression.gridView | {ProductID} | {PlanFragmentID} | {PlanItemID} |

Do this via the Configurator UI -> OMS UI section or via Swagger Configurator API using the following : [/v1/configuration/{applicationId}](http://108.129.16.224:9090/swagger-ui/index.html#/Application%20Properties%20Apis/updateConfiguration): PUT and OMSUI applicationid.

Do for each property:

[

{

"propName": "com.tibco.af.omsui.planItemExpression.dependencyView",

"propValue": "{ProductID} | {PlanFragmentID} | {PlanItemID}",

"valueType": "String"

}

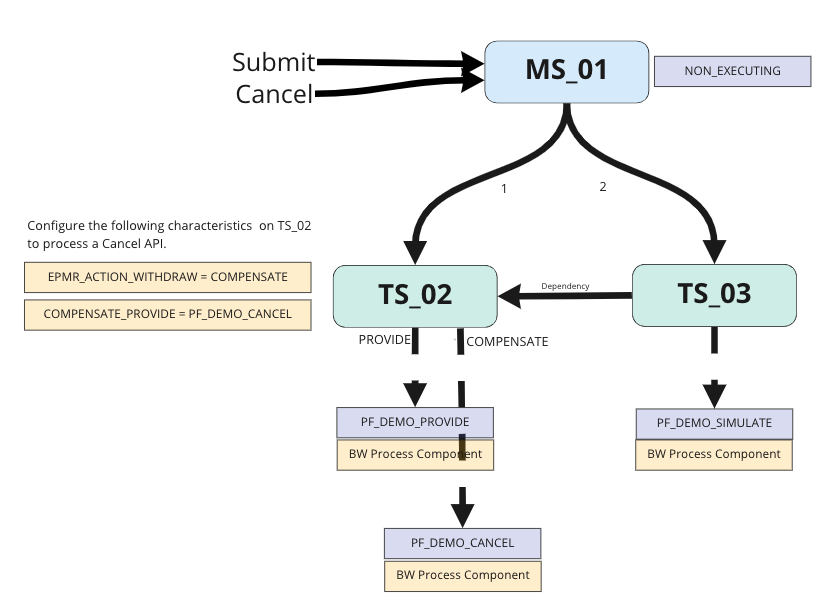
]

Uscases

Miro board: <https://miro.com/app/board/uXjVOyfA7cM=/>

## Cancel

Catalog configuration for cancel demo:



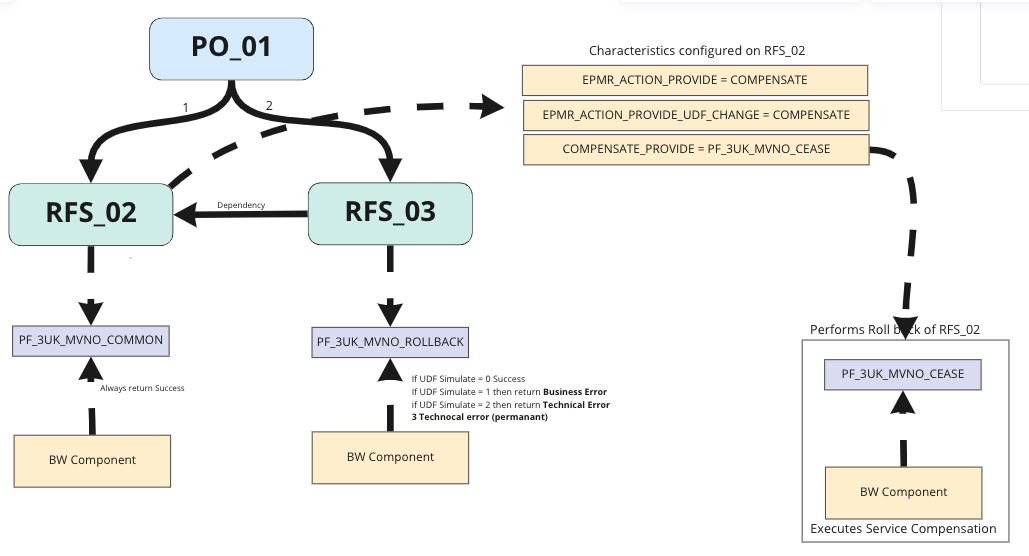
API execution Order:



## Amend

Catalog configuration for Amend demo.

At the moment the Amend demo actually performs a Cancel. This is because it was the first attempt at a cancel it was the only way - at the time - we knew how to do it. Use this config as a basis for a better Amend demo.



Amend API execution Order:

