**Digitaliseringsstyrelsen – eDelivery**

**Installation of eDelivery Solution**

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

.

|  |  |
| --- | --- |
| **Version:** | 1.0 |
| **Status:** | Final |
| **Author:** | Netcompany |



**Document history**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Author | Status | Comments |
| 1.1 | 17.01.2020 | Mateusz Fronczek | Final | Minor cleanup, Pmode – service metadata mapping and procedure after certificates issue is fixed added |
| 1.0 | 20-12-2019 | Netcompany | Final | Output from eDelivery pilot project |

**References**

|  |  |  |  |
| --- | --- | --- | --- |
| Reference | Title | Author | Version |
|  |  |  |  |

Table of contents

[1 Introduction 3](#_Toc29385039)

[2 Prerequisites 3](#_Toc29385040)

[2.1 Gaining accesses 3](#_Toc29385041)

[2.2 Fetching code 3](#_Toc29385042)

[2.2.1 Bitbucket repository: 3](#_Toc29385043)

[2.2.2 Compressed .zip file in Netcompany toolkit: 4](#_Toc29385044)

[2.3 Command terminal 5](#_Toc29385045)

[2.4 Docker 5](#_Toc29385046)

[2.4.1 Useful docker commands 6](#_Toc29385047)

[3 Repository structure 7](#_Toc29385048)

[3.1.1 AP – Access Point 7](#_Toc29385049)

[3.1.2 SMP – Service Metadata Publisher 9](#_Toc29385050)

[4 Service metadata publisher (SMP) 11](#_Toc29385051)

[4.1 Installation of the SMP using The NC docker solution 11](#_Toc29385052)

[4.2 SMP Trust list 14](#_Toc29385053)

[4.2.1 Recommended implementation of trust list in the SMP 15](#_Toc29385054)

[4.2.2 SMP Trust List issues 15](#_Toc29385055)

[4.2.2.1 Admin console issue 15](#_Toc29385056)

[4.2.2.2 Preloading certificates issue 17](#_Toc29385057)

[4.2.3 Updating certificates with the workaround method 18](#_Toc29385058)

[5 Access point (AP) 20](#_Toc29385059)

[5.1 Spinning up AP locally 20](#_Toc29385060)

[6 Application deployment 24](#_Toc29385061)

[6.1 Application deployment on the VM 24](#_Toc29385062)

[6.1.1 Connection to the VM 24](#_Toc29385063)

[6.1.2 Code management 25](#_Toc29385064)

[6.1.3 Applications deployment 25](#_Toc29385065)

[6.1.4 Application stopping 27](#_Toc29385066)

[6.1.5 Viewing Application logs 27](#_Toc29385067)

[7 Testing eDelivery solution 28](#_Toc29385068)

[7.1 Applications Configuration 28](#_Toc29385069)

[7.1.1 Configure Local AP 28](#_Toc29385070)

[7.1.2 Configure VM AP 29](#_Toc29385071)

[7.1.3 Configure VM SMP 29](#_Toc29385072)

[7.1.3.1 PMode and service metadata mapping 33](#_Toc29385073)

[7.2 Test execution 33](#_Toc29385074)

[7.2.1 Configure SOAP UI 34](#_Toc29385075)

[7.2.2 Execute Test 34](#_Toc29385076)

[8 Glossary 37](#_Toc29385077)

# Introduction

As a part of the pilot project by Netcompany in collaboration with Digitaliseringsstyrelsen for an eDelivery solution an installation guide of the proposes solution have been made.

This document describes how to install an eDelivery solution from Netcompany using a containerized solution in Docker.

The Installation guide is aimed to be written in a way that no previous knowledge with eDelivery is required to use it.

Prerequisites

This chapter describes the prerequisite steps that are necessary to install and run either the AP or the SMP. This includes gaining access to the source code and necessary credentials.

## Gaining accesses

In order to install the produced eDelivery solution you need credentials either for:

1. Bitbucket repository. If you do not have one, then either contact the liaison in Digitaliseringsstyrelsen responsible for the interaction with Netcompany or use the zip file from the Netcompany toolkit site.
2. Netcompany toolkit. If you do not have access, contact the liaison in Digitaliseringsstyrelsen responsible for the interaction with Netcompany or refer to the main pilot implementation report: “Pilot implementering af eDelivery – Rapport”.

## Fetching code

The eDelivery code can be fetched in two ways. First is cloning a bitbucket git repository and second is downloading a .zip package of code. Bitbucket enables git features like pulling newest changes but requires bitbucket credentials. For quicker installation use the .zip file.

### Bitbucket repository:

Source code of eDelivery is stored in the **eDelivery** repository: <http://bitbucket.org/nc-dp/edelivery>

The repository is located in the **nc-dp** space which is the **Digital Post** bitbucket space.

Accessing and cloning **eDelivery** repository requires having credentials to the bitbucket.

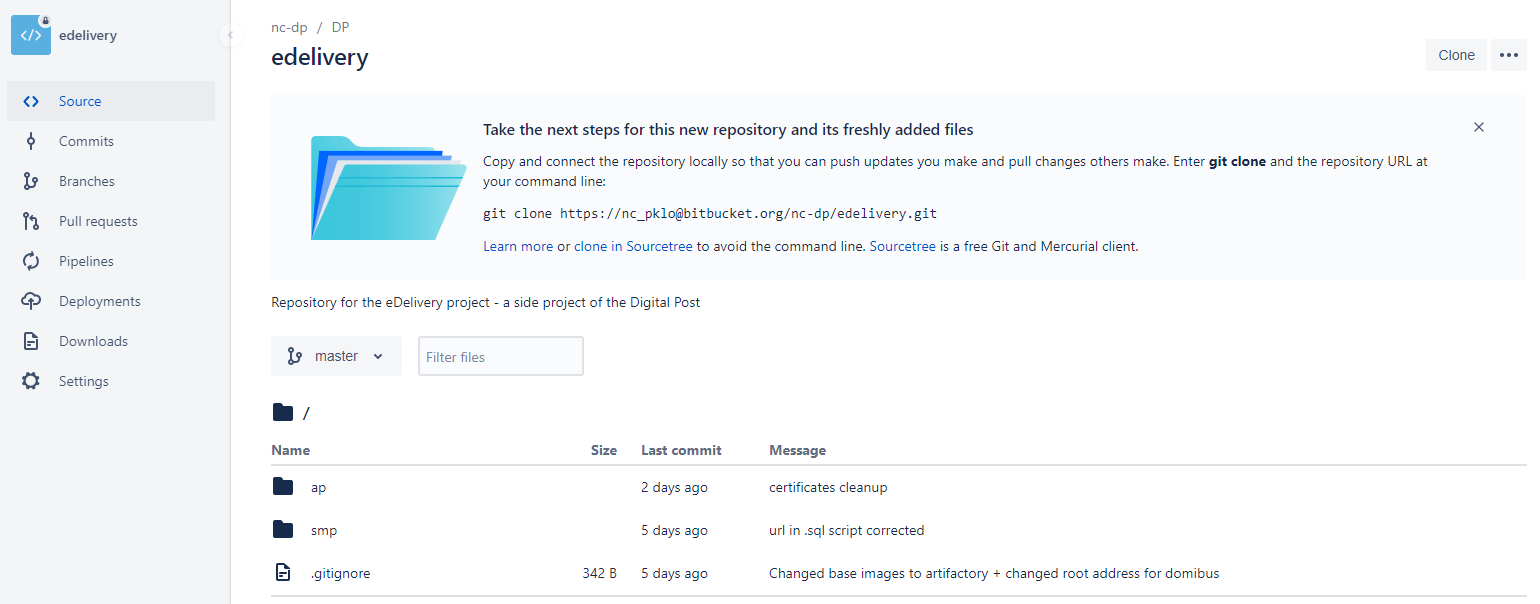
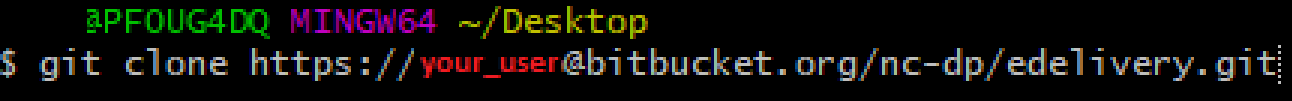


Figure 1. eDelivery repository screenshot

Fetching process:

1. Install git on your local machine. Follow the tutorial: <https://www.atlassian.com/git/tutorials/install-git>
2. Gain credentials to the **eDelivery** repository.
3. Access the repository: <https://bitbucket.org/nc-dp/edelivery/src/master/>
4. Click ‘Clone’ in the upper right corner
5. Copy the address, using HTTPS  
   
6. Create the folder where the repository will be stored.

Open your terminal and paste the command cloned in the step 4.  
  
Figure 2. Git bash screenshot

1. Once cloning is finished, you can access the eDelivery repository.

### Compressed .zip file in Netcompany toolkit:

Source code is also zipped and stored in the Netcompany toolkit: [Link](https://goto.netcompany.com/cases/GTE610/DIGEDEL/Deliverables/edelivery.zip)

In case of issues with the link, then refer to the main project report mentioned in section 2.1.

The .zip file always contains the newest version of a master branch of the **eDelivery** bitbucket repository.

The package does not contain the .git files.

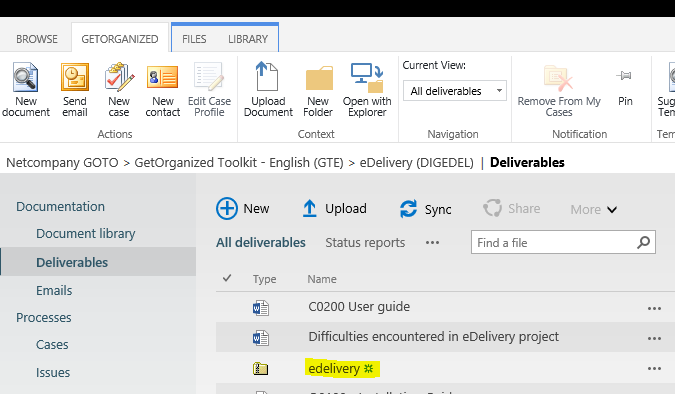


Figure 3. Toolkit screenshot

Downloading process:

1. Gain credentials to the Netcompay toolkit.
2. Click on the link: [https://goto.netcompany.com/cases/GTE610/DIGEDEL/Deliverables/edelivery.zip](https://goto.netcompany.com/cases/GTE610/DIGEDEL/Deliverables/edelivery%20-%20code.zip)
3. Unpack the downloaded file into your repository folder.

## Command terminal

A command terminal needs to be chosen / installed.

In the installation process, used terminal is **git bash**.

Other terminals (Windows CMD, powershell) can be used, however, the commands in the guide are based on the git bash syntax.

Installation of **git bash**: <https://gitforwindows.org/>

## Docker

For running the containerized solution **docker desktop** is required.

It can be downloaded from: [https://www.docker.com/](https://www.docker.com/products/docker-desktop)

For simple testing purposes you can download and run Docker Desktop on your local PC.

The version of docker and docker-compose that has been used for development:

|  |  |
| --- | --- |
| Program | Version |
| Docker | 19.03.5 |
| Docker-compose | 1.24.1 |

### Useful docker commands

Docker and docker-compose commands that are useful in the installation

|  |  |
| --- | --- |
| Command | Description |
| docker-compose build | Docker images for given service are built |
| docker-compose up | Docker images are started |
| docker-compose up --build | Docker images are first built and then started |
| docker-compose up -d | Docker containers are started and run in background |
| docker-compose down | Docker containers stop |
| docker ps | Running containers are listed |
| docker images | Docker images are |
| docker images rm IMAGE\_ID | Remove docker images. Useful while developing images. |

# Repository structure

The repository is structured into two main folders.

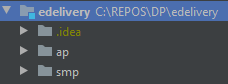


Figure 4. eDelivery folder structure

One for SMP and one for AP.

### AP – Access Point

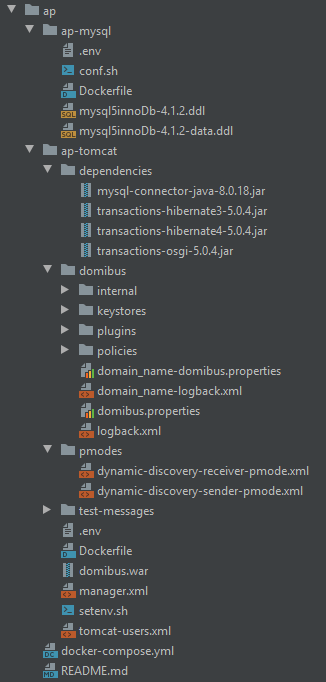


Figure 5. eDelivery folder structure

Files/Folders description:

|  |  |  |  |
| --- | --- | --- | --- |
| Folder | Subfolder | File | Description |
|  |  | docker-compose.yml | File composing deployment for AP |
|  |  | README.md | README file |
| ap-mysql |  | .env | Environment variables for AP DB |
|  |  | conf.sh | Script automating execution of scripts configuring AP DB |
|  |  | Dockerfile | Dockerfile describing layers of the AP DB image |
|  |  | mysql5innoDb-4.1.2.ddl | Script creating tables in the AP DB |
|  |  | mysql5innoDb-4.1.2-data.ddl | Script inserting data into the AP DB |
| ap-tomcat | Dependencies | \* | Folder with .jar dependencies required to make the AP start up |
|  | Domibus | Internal | Folder with configuration .xml files |
|  |  | KeyStores | Folders with a keystore and a truststore, holding certificates |
|  |  | Plugins | Folder containing plugins required for AP to run |
|  |  | Policies | Folder containing AS4 protocol policy .xml files |
|  |  | domain\_name-domibus.properties | AP properties file – important! |
|  |  | domain\_name-logback.xml | Configuration file defining logging in AP |
|  |  | domibus.properties | Second file with AP properties – important! |
|  |  | logback.xml | Configuration file defining logging in AP |
|  | PModes | \* | Folders with PModes for AP – for a sender and receiver AP |
|  | Test-messages | dynamic-discovery-test-message.xml | File with an exemplary SOAP message that needs to be send in AP connection test |
|  |  | .env | Environment variables for AP tomcat |
|  |  | Dockerfile | Dockerfile describing layers of the AP Tomcat image |
|  |  | Domibus.war | Main binary file with AP application |
|  |  | Manager.xml | Configuration file for Tomcat |
|  |  | Setenv.sh | Script with Tomcat configuration after booting up |
|  |  | Tomcat-users.xml | Configuration file with Tomcat users |

### SMP – Service Metadata Publisher

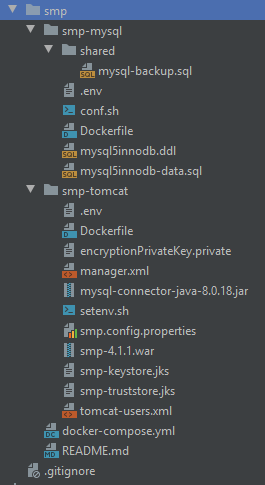


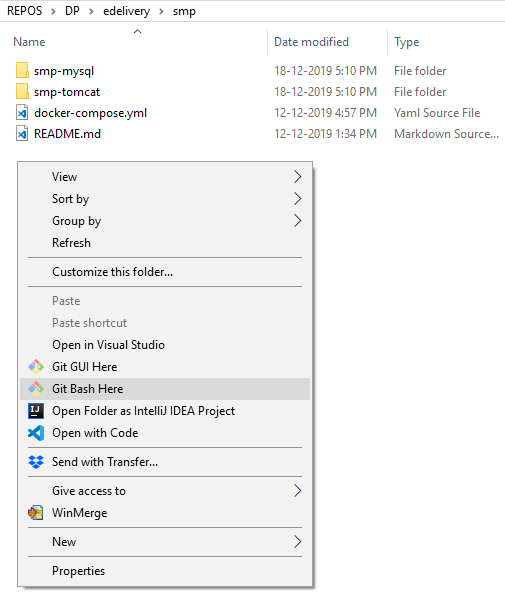
Figure 6. eDelivery folder structure

|  |  |  |  |
| --- | --- | --- | --- |
| Folder | Subfolder | File | Description |
|  |  | docker-compose.yml | File composing deployment for AP |
|  |  | README.md | README file |
| smp-mysql | shared | mysql-backup.sql | File prepared for future SMP DB backups |
|  |  | .env | Environment variables for SMP DB |
|  |  | conf.sh | Script automating execution of scripts configuring SMP DB |
|  |  | Dockerfile | Dockerfile describing layers of the SMP DB image |
|  |  | mysql5innodb.ddl | Script creating tables in the SMP DB |
|  |  | mysql5innodb-data.sql | Script inserting data into the SMP DB |
| smp-tomcat |  | .env | Environment variables for SMP tomcat |
|  |  | Dockerfile | Dockerfile describing layers of the SMP Tomcat image |
|  |  | encryptionPrivateKey.private | Private key used for encrypting keystore and truststore passwords |
|  |  | manager.xml | Configuration file for Tomcat |
|  |  | mysql-connector-java-8.0.18.jar | Plugin for the DB connection |
|  |  | setenv.sh | Script with Tomcat configuration after booting up |
|  |  | smp.config.properties | SMP configuration file – important! |
|  |  | smp-4.1.1.war | Main binary file with SMP application |
|  |  | smp-keystore.jks | Keystore file holding SMP certificates |
|  |  | smp-truststore.jks | Truststore file holding SMP certyficates |
|  |  | tomcat-users.xml | Configuration file with Tomcat users |

# Service metadata publisher (SMP)

## Installation of the SMP using The NC docker solution

The process of spinning up the SMP locally is:

Open your terminal in the eDelivery repository and move to the ‘SMP’ folder. Alternatively, if you use git bash, open the SMP folder and in there right-click mouse and choose ‘Git Bash here’.  
  
Figure 7. Opening Git Bash in a folder

1. Build docker images: type in the terminal **docker-compose build** in order to build docker images for smp\_db – database and ap\_ws – tomcat with SMP application

docker-compose build

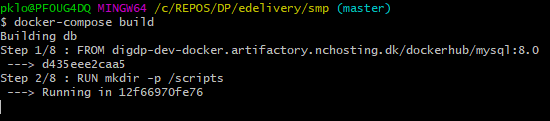


Figure 8. Git Bash screenshot

As a result, after typing **docker images** the list of freshly images can be inspected:

Docker images

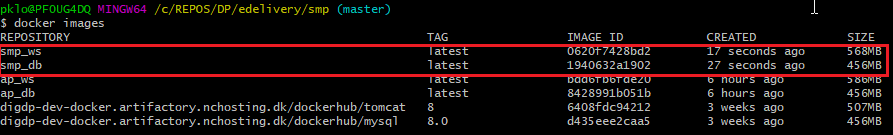
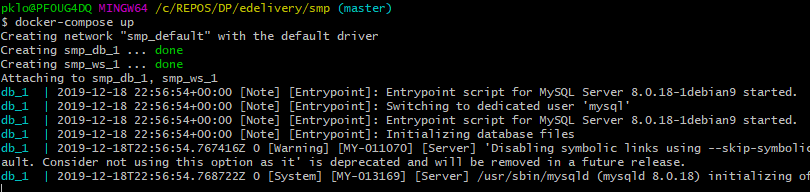


Figure 9. Git Bash screenshot

1. Spin up docker containers: **docker-compose up**

docker-compose up

Docker containers are created out of the images and the application is booting up

  
Figure 10. Git Bash screenshot

**Note**

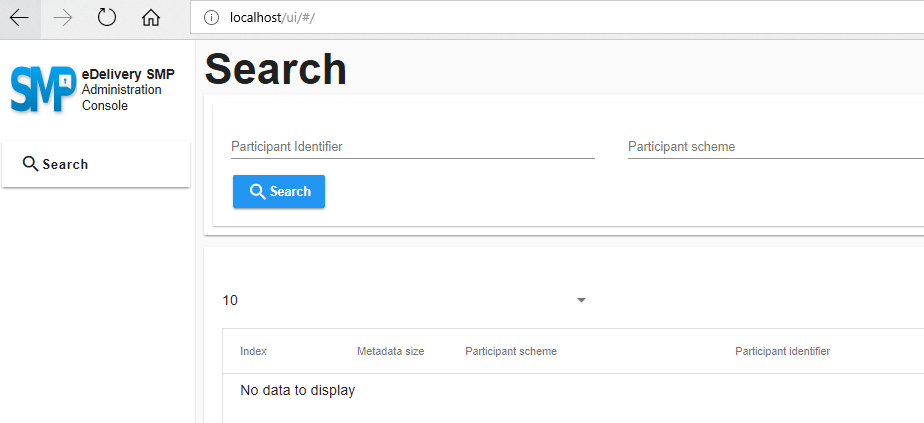
2 commands above can be substituted with

docker-compose up --build

In order to run the application in the detached state, so the application runs in the background and is not affected by closing the terminal, use the command with the **-d** flag.

docker-compose up –build -d

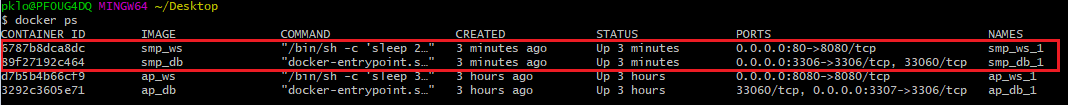
1. Once you enter the browser, **localhost/ui/#/**, you can see that the application has correctly booted up.

  
Figure 11. SMP Admin Console

1. For inspection, you can check running containers. Type ‘docker ps’.

docker ps

You will see a list list of running containers on the Docker deamon.

  
Figure 12. Git Bash screenshot

Note:

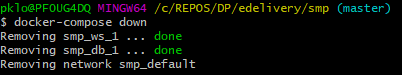
If the AP is running not in the in detached mode (docker-compose up -d), the ‘docker ps’ need to be typed in new command terminal.

1. To stop the application:
   1. If you do not run in the detached mode, in the git bash, type ‘ctrl + c’

ctrl + c

  
Figure 13. Git Bash screenshot

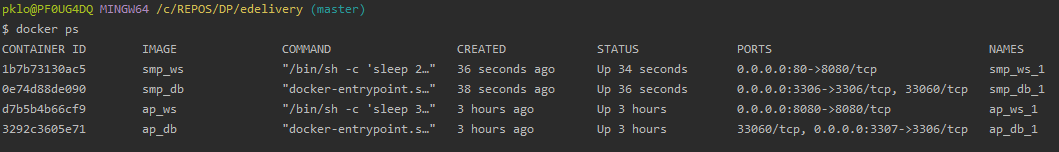
* 1. If you run in the detached mode, code terminal need to be opened in the ap folder so it can access the ap docker-compose.yml file, and type docker-compose down. It will stop the db and the tomcat container.

  
Figure 14. Git Bash screenshot

1. Logs inspection.
   1. Application does not run in a detached mode – logs are printed in the terminal where the application was booted up.
   2. Application runs in a detached mode – logs are saved in the log file. To view it, the docker container with Tomcat and deployed application need to be accessed.
      1. In the terminal type ‘docker ps’

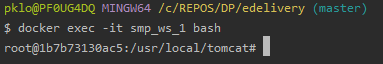
docker ps

A list of running containers is displayed:

  
Figure 15. Git Bash screenshot

* + 1. Get into the Tomcat container shell.

docker exec -it smp\_ws\_1 bash

  
Figure 16. Git Bash screenshot

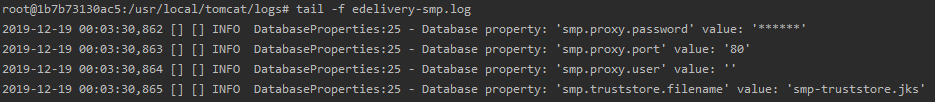
* + 1. Go to folder /logs

cd /logs

  
Figure 17. Git Bash screenshot

* + 1. Open the logs file

tail -f domibus.log

  
Figure 18. Git Bash screenshot

* + 1. Exit the console

exit

  
Figure 19. Git Bash screenshot

## SMP Trust list

Trust List in SMP is implemented in form of a KeyStore and TrustStore. Its purpose is:

KeyStore – Stores SMP certificate with both private and public key.

TrustStore – Stores public certificates of APs for their authentication.

Configuration of Trust List in the SMP can be done in two ways:

1. Using administration console - boot the application with default properties for the KeyStore and TrustStore so they are autogenerated and then add the certificates manually through the administration console
2. Predefining in deployment process - create KeyStore and TrustStore files and add them to the SMP deployment process and define in the SMP configuration file ‘SMP.CONFIG.PROPERTIES’

Neither of these solutions works out of the box, so following chapters describe how it should be implemented, what the issues were and the workarounds.

### Recommended implementation of trust list in the SMP

CEF eDelivery SMP documentation recommends uploading/configuration of certificates in a way:

1. Boot up SMP with default settings – default KeyStore and TrustStore files together with default privateEncryptionKey for encrypting passwords for the stores are generated.
2. Insert Certificates into both KeyStore and TrustStore through the Administration Console.
3. Done

ISSUE: Loading AP certificates into the SMP TrustStore does not work as intended.

Alternative approach, with predefined SMP configuration, so that certificates are automatically inserted when the application is booting up. This approach is correctly implemented in the AP configuration.

1. Export the SMP certificate into the KeyStore and AP certificate into the TrustStore format (.jks).
2. Configure SMP building process so that the .jks files are copied into appropriate place in the smp\_ws\_1 Docker container.
3. Set up smp.config.properties so the application knows about the .jks files while booting up and uses those in spite of creating new ones.

ISSUE: Both KeyStore and TrustStore .jks files are not loaded during booting up application. Application fails to start.

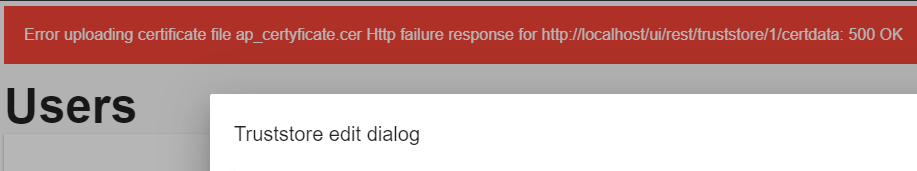
### SMP Trust List issues

Both described approaches have issues while applying them.

#### Admin console issue

First issue is inserting certificates using the administration console. Although inserting certificate into the KeyStore works correctly, it fails when trying to load the AP certificates into the trust list.

After trial of uploading the certificate, an error is displayed on SMP Admin console:

  
Figure 20. SMP uploading certificate to the TrustStore error

And the SMP logs show an Exception:

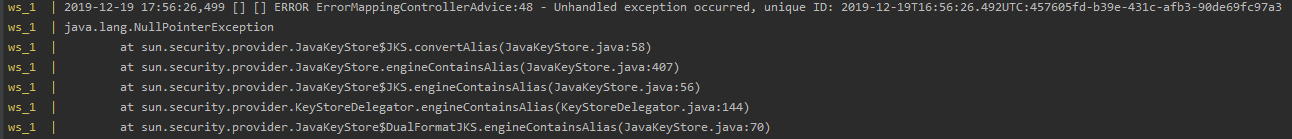


Figure 17. Exception SMP logs

**What is the issue:**

After contacting SMP Support it turned out that current version of SMP 4.1.1 is does not support type of X509 certificate that was issued for DK eDelivery – one with ‘multivalue - RDN’.

The response from support was:

‘We are currently investigating this multivalue – RDN uploading certificate issue in the SMP and will keep you posted about the timeframe for the fix. ‘

It is anticipated to be fixed in February 2020.

**The** workaround**:**

It turned out that certificates are possible to be loaded into the TrustStore but using alternative approach than the SMP guide is recommending. How to load it:

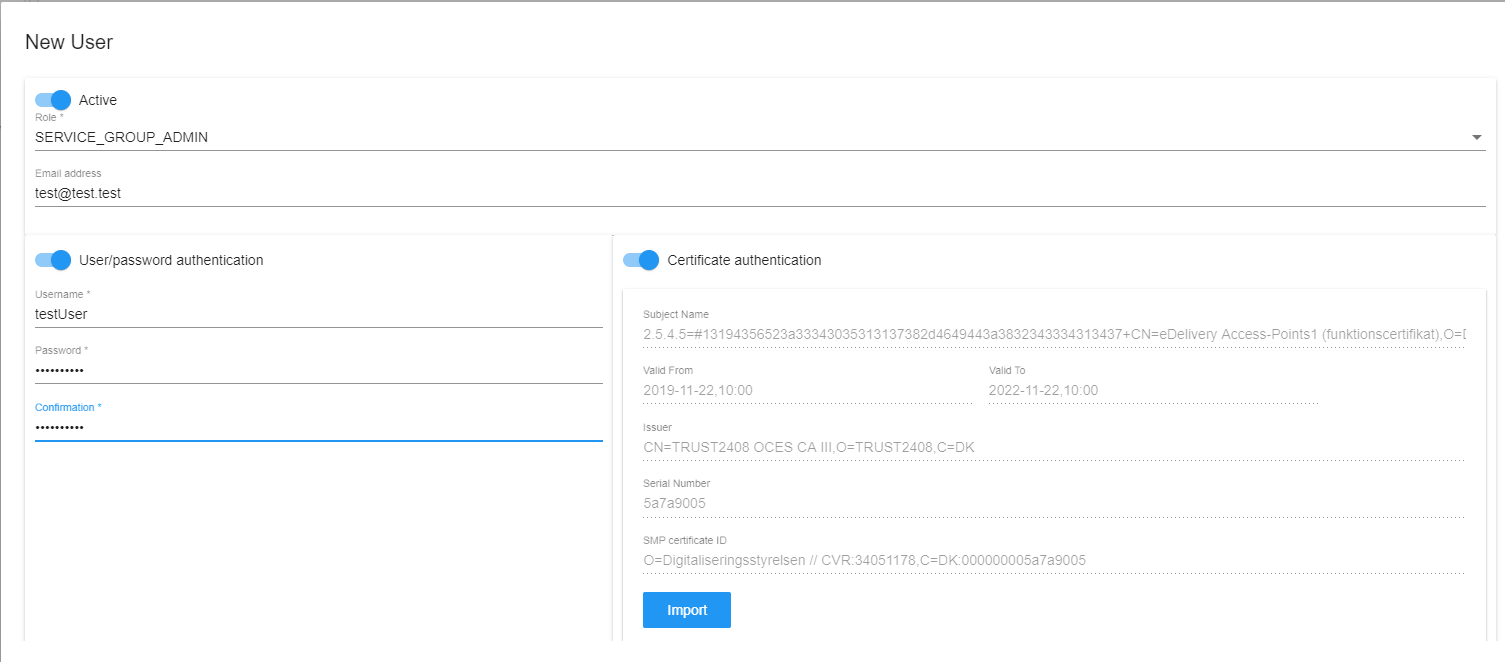
1. Start up the SMP according to instructions in this guide.
2. Access SMP admin console, login as ‘system’ user and click on the ‘Users’ tab.
3. In spite of clicking ‘Edit truststore’, click ‘+New’.
4. Add new user with even artificial data, and additionally import desired AP certificate.  
   

Figure 18. SMP Admin console

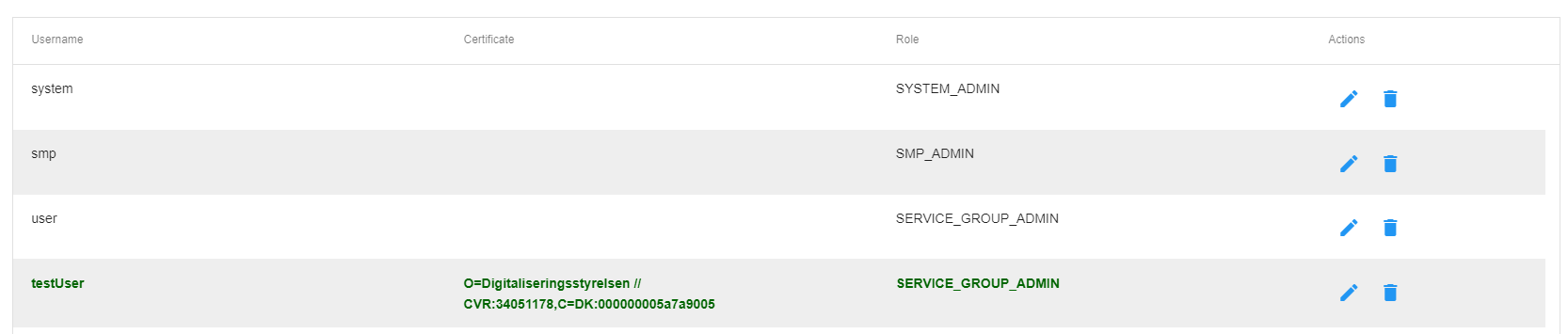
1. Click ‘OK.
2. The certificate was imported into TrustStore.  
   

Figure 19. SMP Admin console

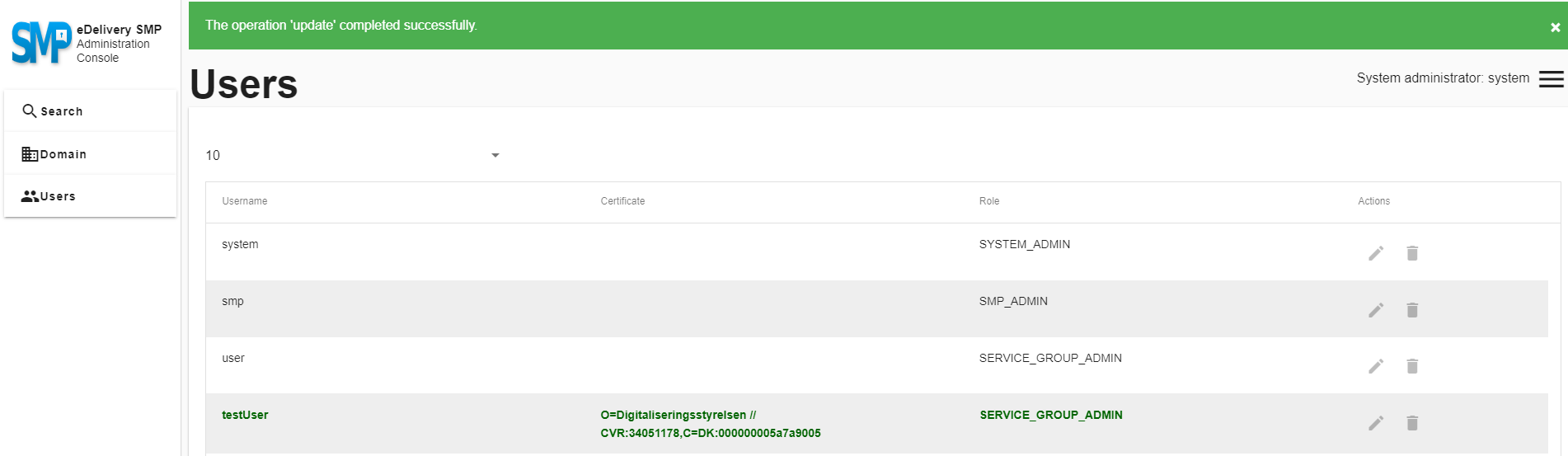
1. Click ‘Save’.
2. The certificate has been updated successfully.  
   

Figure 20. SMP Admin console

This workaround works, certificate is correctly uploaded into the TrustStore, however this is a workaround and certificate should be imported into the TrustStore using recommended way as soon as the problem with ‘Multivalue-RDN’ is solved.

#### Preloading certificates issue

Second approach is to create the KeyStore and the TrustStore .jks files together with the private key for encrypting private keys, copy them into the appropriate folders in the image building process and specify their configuration in the SMP configuration file.

This approach, however, does not work either. It is because of the application does not accept KeyStore and TrustStore .jks files that are created by exporting from the original certificates (SMP and AP). Application does not boot up with this approach

**The** workaround**:**

The workaround summary is: Autogenerate KeyStore and TrustStore, add certificates using the Administration console, make an SQL dump, export KeyStore and TrustStore into the local host, add the edited key stores to the deployment process and preload the database with dumped values, so the files are preloaded in the booting up process.

The workaround details are:

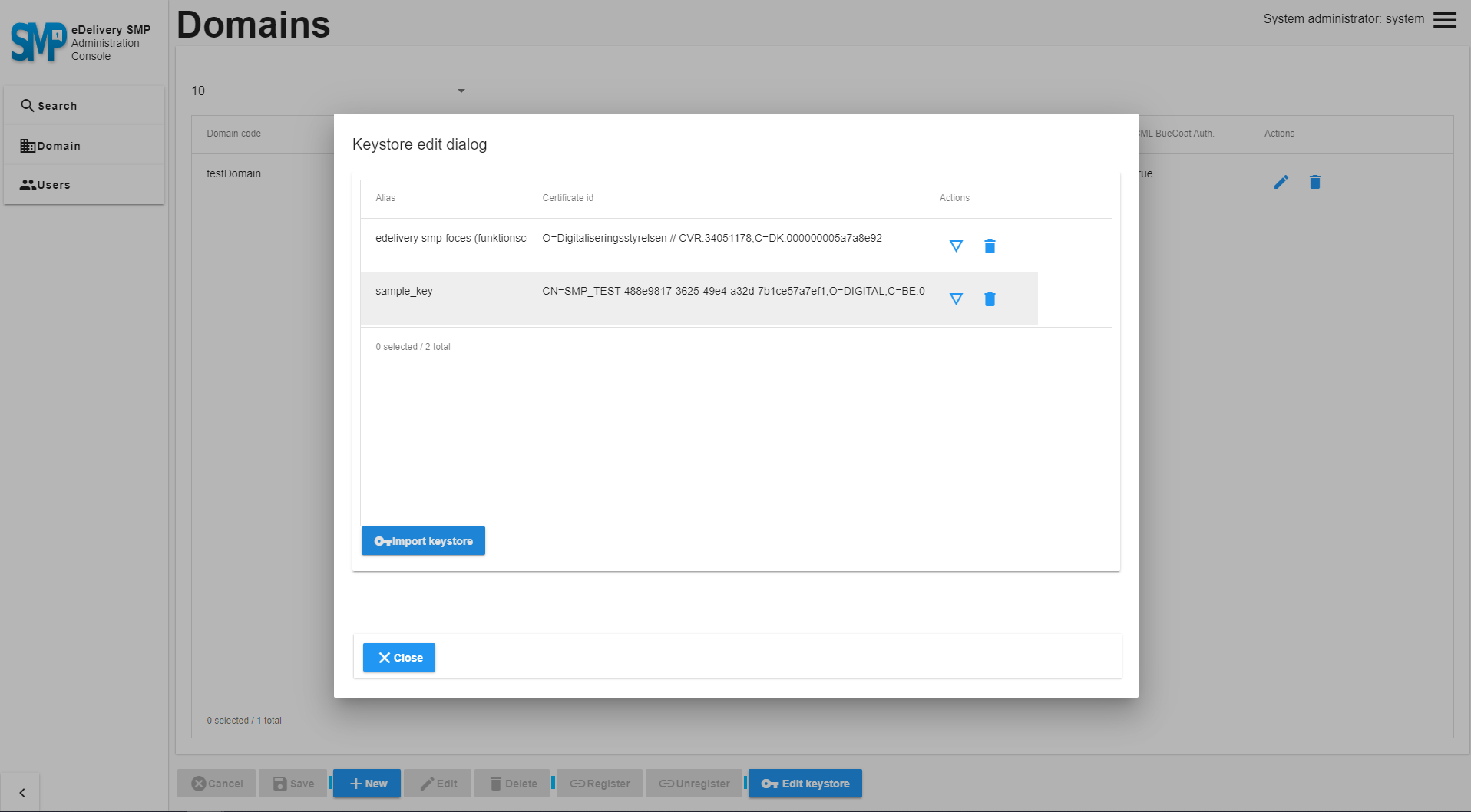
1. SMP has been booted up with default properties – keystore.jks and truststore.jks together and the privateEncryptionKey.private files were automatically generated, together with their passwords.
2. Certificates are added to the SMP using Administration Console according to the workaround described in the previous section.
3. All 3 files were copied out of the smp\_ws\_1 docker container to local computer.
4. A MySQL dump was performed for the table SMP\_CONFIGURATION table. The table holds the names of the trust list files and their passwords.
5. Keystore.jks truststore.jks and privateEncryptionKey.private are added to the smp\_ws\_1 tomcat container configuration – they are added the docker image and located in the proper folder.
6. MySQL dump script with SMP\_CONFIGURATION table values is inserted into the smp\_ws\_1 image and is executed right after the MySQL DB is booted up. Thanks to this step no new trust lists and passwords are generated.
7. When SMP Tomcat container is booting up, it checks the SMP\_CONFIGURATION table and if it is empty, new trust list and password files are generated. The table is already defined with KeyStore and TrustStore with preloaded appropriate certificates, and matching encoded by the privateEncriptionKey password.

This solution is working fine and both KeyStore and TrustStore in SMP has properly configured certificates, however, it is cumbersome and as soon as the problem on the SMP side is fixed, it should be abandoned and official recommended way of implementing certificates into the TrustStore and KeyStore should be implemented (as is in the AP).

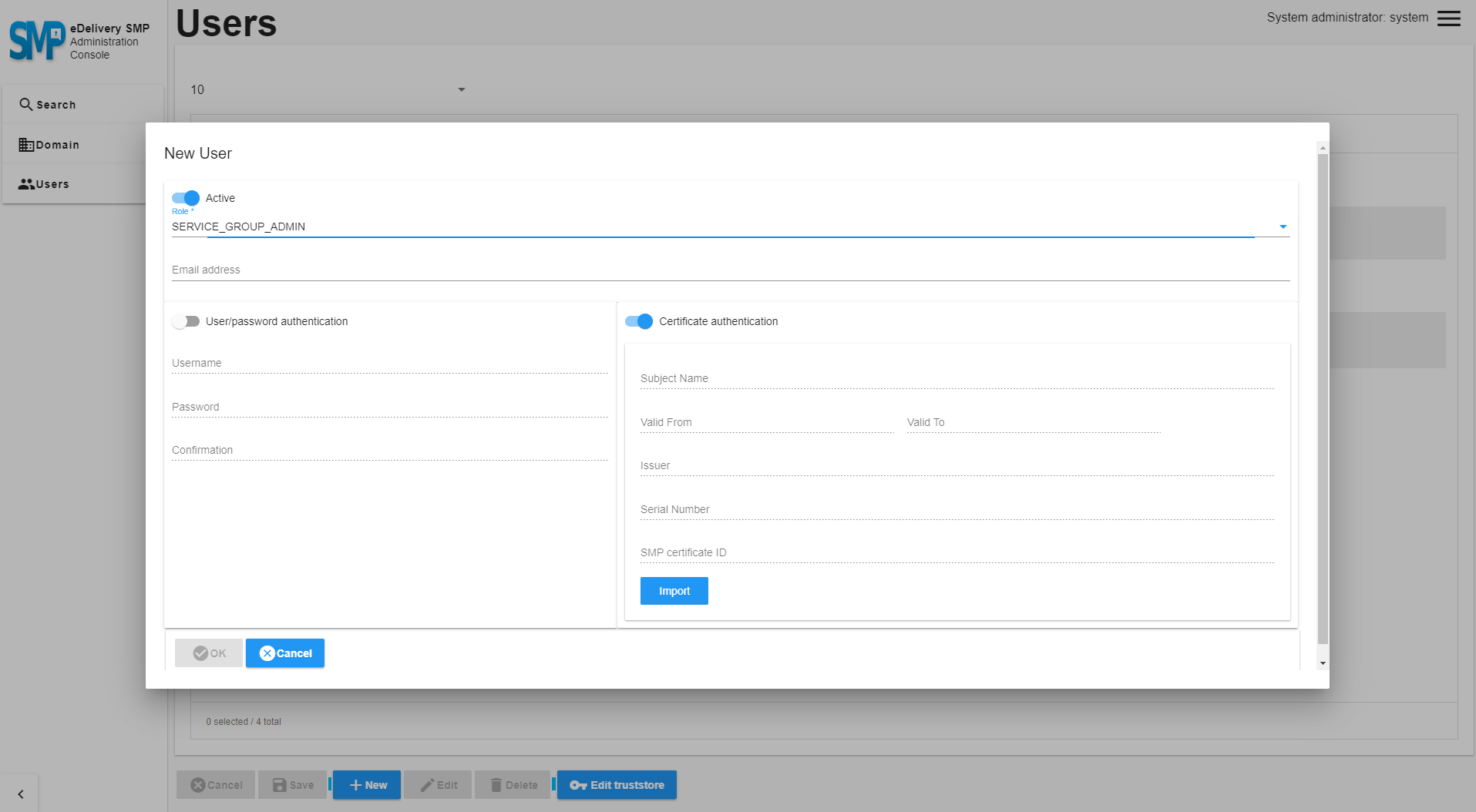
### Updating certificates with the workaround method

The configuration of the trust list on the SMP is dependent on its MySQL backend, its KeyStore, its TrustStore and its property file. The current solution is based on a semi-automatic approach as described by the instructions below.

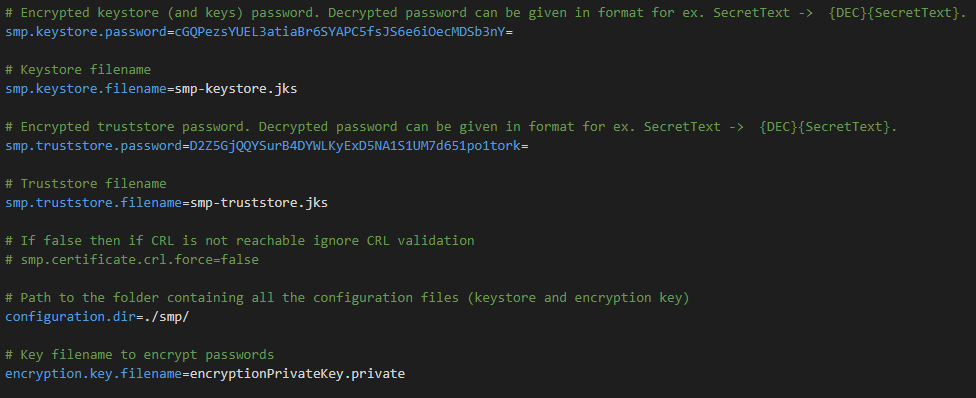
1. Let the SMP be built, composed and run while containing no KeyStore, TrustStore, private key nor anything in the database (i.e. remove the import of the ‘smp\_schema\_SMP\_CONFIGURATION.sql’ file)
   1. This will auto-generate everything necessary (such as passwords for the KeyStore/TrustStore) and store related information in the database
2. Go to the SMP Admin Console, log in as ‘system’ and open the ‘Domain’ tab on the left. Press the button ‘Edit KeyStore’ as seen below and import the PKCS file for the SMP:

  
Figure 21. SMP Admin console

1. Add each AP as a new user as type ‘SERVICE\_GROUP\_ADMIN’ and use certificate authentication:

  
Figure 21. SMP Admin console

1. Copy the KeyStore, TrustStore and private key from the SMP tomcat container onto the host.
2. Perform a ‘mysqldump’ of the SMP\_configuration table from the MySQL container and transfer the file (such as ‘smp\_schema\_SMP\_CONFIGURATION.sql’) back to the host.
3. Now you should have a complete KeyStore, TrustStore and private key on the host machine which can be used by docker to load the application automatically. Please make sure that…
   1. … the ‘smp.config.properties’ file use the correct files and passwords (from the database) as seen below:

  
Figure 22. SMP properties file

* 1. … the ‘Dockerfile’ copies the correct files onto the container
  2. … the SMP MySQL container loads the exported configuration data whenever the container starts (by modifying conf.sh)

# Access point (AP)

Access Point application installation is based on docker containers.

Access Point is comprised of two connected containers:

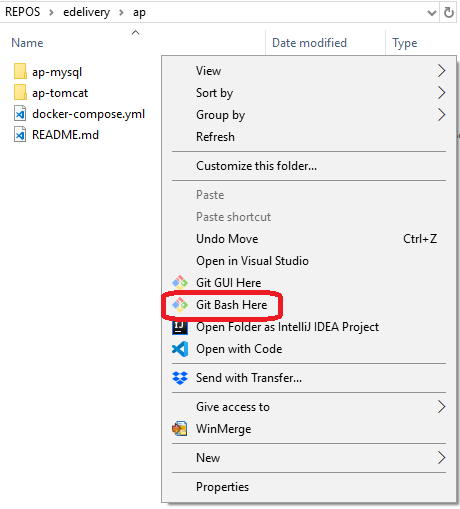
1. AP\_WS – container with Tomcat web application container and deployed Access Point application.
2. AP\_DB – container with MySql database server.

Both containers are connected with each other with Docker Compose. Docker-compose file is connecting both containers dockerfiles and defines their inner relations.

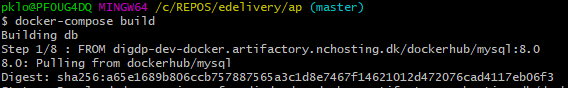
Installing / spinning up the AP locally is in general as simple as using docker-compose up command, and it is one of the biggest advantages of using docker and docker-compose.

## Spinning up AP locally

The process of spinning up the AP locally is:

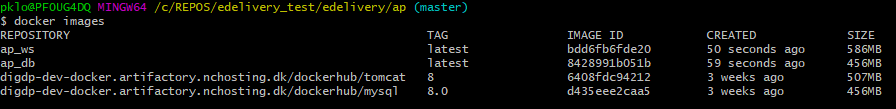
1. Open your terminal in the eDelivery repository and move to the ‘ap’ folder. Alternatively, if you use git bash, open the ap folder and in there right-click mouse and choose ‘Git Bash here’.  
     
   Figure 22. Git Bash opening in a window.
2. Build docker images: type in the terminal **docker-compose build** in order to build docker images for ap\_db – database and ap\_ws – tomcat with AP application

docker-compose build

  
Figure 23. Git Bash screenshot

As a result, after typing **docker images** the list of freshly images can be inspected:

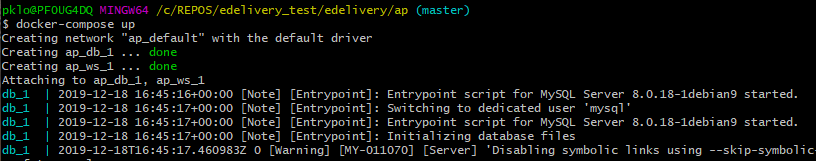
Docker images

  
Figure 24. Git Bash screenshot

1. Spin up docker containers: **docker-compose up**

docker-compose up

Docker containers are created out of the images and the application is booting up

  
Figure 25. Git Bash screenshot

**Note**

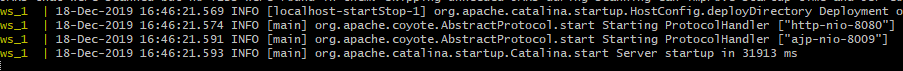
2 commands above can be substituted with

docker-compose up --build

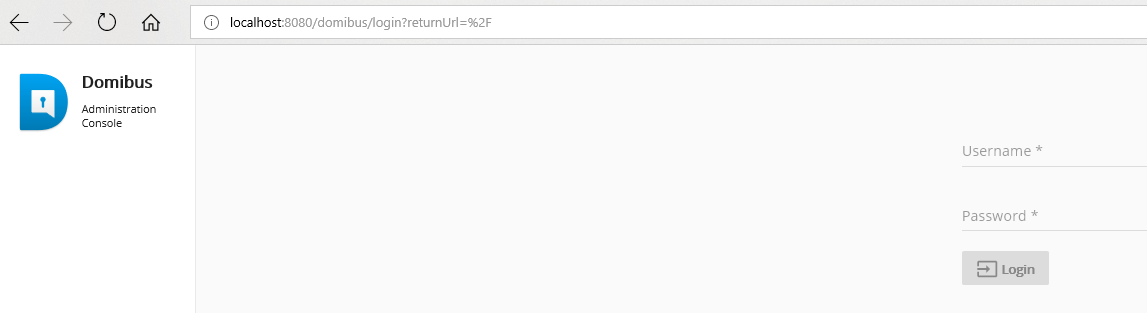
In order to run the application in the detached state, so the application runs in the background and is not affected by closing the terminal, use the command with the **-d** flag.

docker-compose up –build -d

The logs present that the application has been correctly booted up:

  
Figure 26. Git Bash screenshot

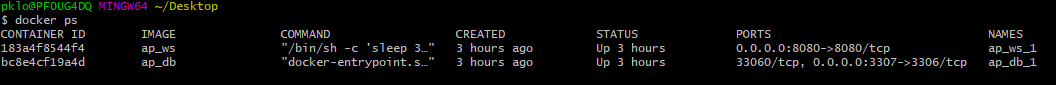
1. Once you enter the browser, **localhost:8080/domibus**, you can see that the application has correctly booted up.

  
Figure 27. Access Point administration console

1. For inspection, you can check running containers. Type ‘docker ps’.

docker ps

You will see a list list of running containers on the Docker deamon.

  
Figure 28. Git Bash screenshot

Note:

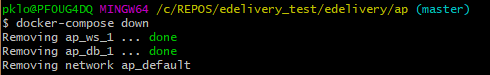
If the AP is running not in the in detached mode (docker-compose up -d), the ‘docker ps’ need to be typed in new command terminal.

1. To stop the application:
   1. If you do not run in the detached mode, in the git bash, type ‘ctrl + c’

Ctrl + c

  
Figure 29. Git Bash screenshot

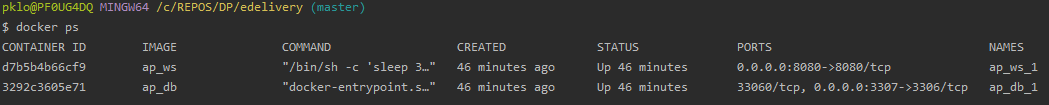
* 1. If you run in the detached mode, code terminal need to be opened in the ap folder so it can access the ap docker-compose.yml file, and type docker-compose down. It will stop the db and the tomcat container.

  
Figure 30. Git Bash screenshot

1. Logs inspection.
   1. Application does not run in a detached mode – logs are printed in the terminal where the application was booted up.
   2. Application runs in a detached mode – logs are saved in the log file. To view it, the docker container with Tomcat and deployed application need to be accessed.
      1. In the terminal type ‘docker ps’

docker ps

A list of running containers is displayed

  
Figure 31. Git Bash screenshot

* + 1. Get into the Tomcat container shell.

docker exec -it ap\_ws\_1 bash

  
Figure 32. Git Bash screenshot

* + 1. Go to folder /logs

cd /logs

  
Figure 33. Git Bash screenshot

* + 1. Open the logs file

tail -f domibus.log

  
Figure 34. Git Bash screenshot

* + 1. Exit the console

exit

  
Figure 35. Git Bash screenshot

## AP keystore

It’s not obvious that parties’ names have to correspond with aliases in keystores. If there is no key in keystore with alias same as partyId in PMode and message triggering the dispatch, Domibus fails to send message.

Please make sure, that key in keystore alias is the same as partyId in PMode, so effectively that it is the same as ‘Common Name’ property from the certificate subject.

# Application deployment

Both applications Access Point (AP) and Service Metadata Publisher (SMP) need to be deployed on the server to be publicly accessible. eDelivery prototype is deployed on a Virtual Machine hosted by Netcompany.

Deployment on the Virtual Machine consists of almost identical steps that local deployment described in previous section. It is thanks to the advantages of application containerization which provide deployment automation. Deployment of the eDelivery will look very similar on any machine with installed prerequisites.

VM prerequisites:

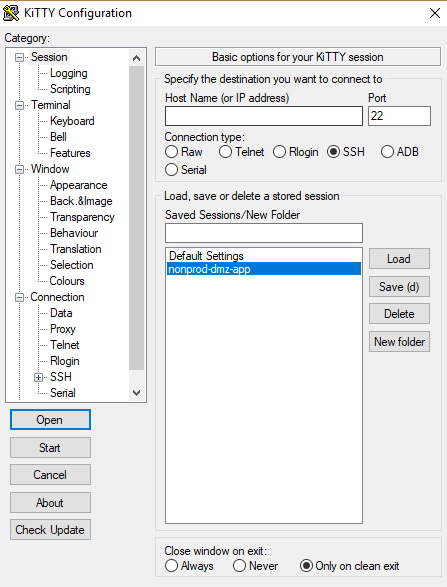
* Installed OS (Oracle Linux 7 in the VM)
* installed Git
* installed Docker/Docker-compose
* provided internal/external accesses
* opened required ports

For VM details and maintenance of the eDelivery deployed on the server, please acquaint with Infrastructure guide and the Maintenance guide documents.

## Application deployment on the VM

Deployment process for the Virtual Machine. Both Access Point (AP) and Service Metadata Publisher (SMP) has almost identical deployment process.

### Connection to the VM

1. Log in to the Virtual Machine.   
   Using Kitty and providing required Credentials.  
     
   Figure 36. Kitty screenshot
2. Once connected, a terminal should be opened  
     
     
   Figure 37. VM terminal screenshot

### Code management

In order to facilitate code management on the Virtual Machine, the ‘eDelivery’ repository has been cloned into the Virtual Machine.

The repository is located in the common folder under the path: **/data/projectdata/eDelivery**

**Managing code**

When any changes have been committed to the repository, they should be pulled to the VM repository before building/rebuilding the application.

1. Change path after logging into the VM:

cd /data/projectdata/edelivery

  
Figure 38. VM terminal screenshot

1. Pull code

git pull

  
Figure 39. VM terminal screenshot

In order to pull any changes, password to the bitbucket need to be typed into the console.

  
Figure 39. VM terminal screenshot

### Applications deployment

The procedure of deployment is identical for AP and SMP and described in the Local installation section in the current guide.

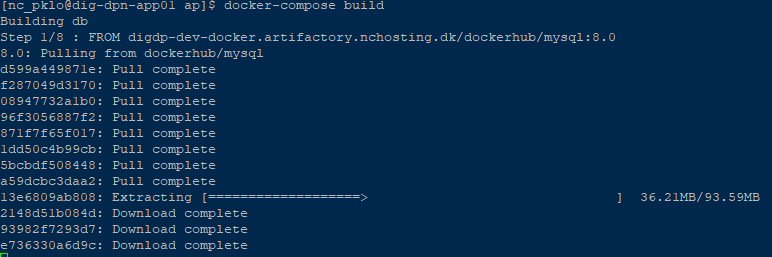
Exemplary procedure for the Access Point application:

1. Change directory to the Access Point folder.

/data/projectdata/edelivery/ap

1. Build images

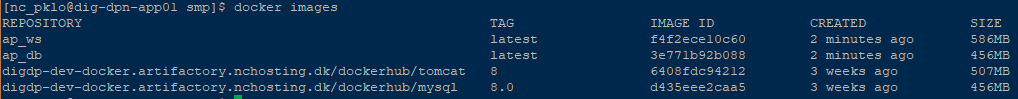
docker-compose build

Figure 40. VM terminal screenshot

1. Inspect build images

Inspect whether the images were correctly built.

docker-compose build

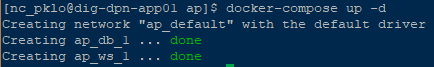
Figure 41. VM terminal screenshot

1. Start the application

**IMPORTANT NOTE**

**Need to start the application in the detached mode (-d). Otherwise when the VM terminal will be closed, application will be stopped.**

docker-compose up -d

Figure 42. VM terminal screenshot

### Application stopping

Stopping application on the VM is exactly the same as locally.

In order to do so, terminal directory must be changed to one locating docker-compose.yml file for given application (SMP or AP).

Container (Application) stopping is done using command:

docker-compose down

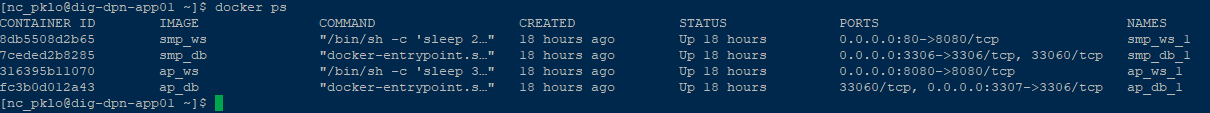
### Viewing Application logs

Inspecting logs is of crucial importance to evaluate proper application work.

Viewing application logs is done exactly the same was as inspecting logs for application installed locally in the detached mode.

1. Inspect running containers:

docker ps

Figure 43. VM terminal screenshot

1. Connect into the tomcat containers shell

docker exec -it ap\_ws\_1 bash

  
Figure 44. VM terminal screenshot

1. Change directory to logs directory

cd logs

  
Figure 45. VM terminal screenshot

1. Print logs into the console

tail -f domibus.log

  
Figure 46. VM terminal screenshot

# Testing eDelivery solution

In order to verify if the installation of eDelivery is correct, a connection test between two Access Points need to be performed.

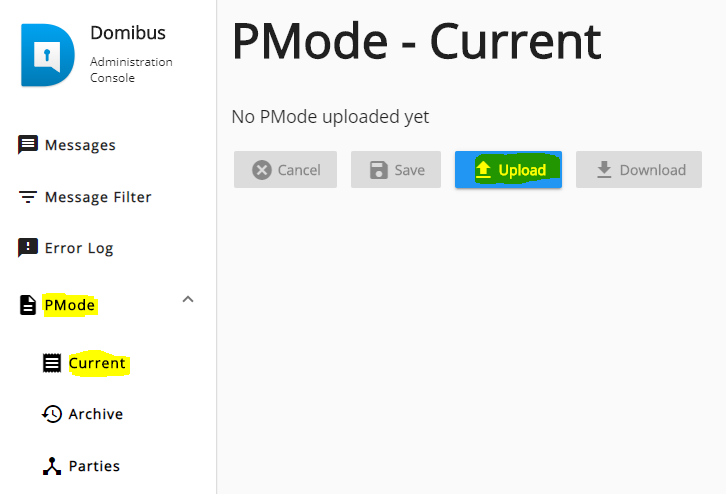
The test includes sending a test message between two Access Points. One installed on a local machine and second deployed on the Virtual Machine.

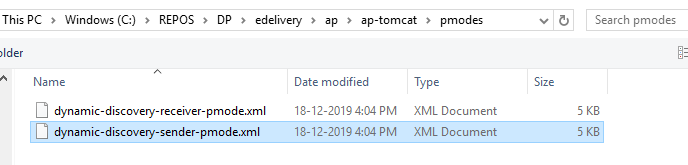
## Applications Configuration

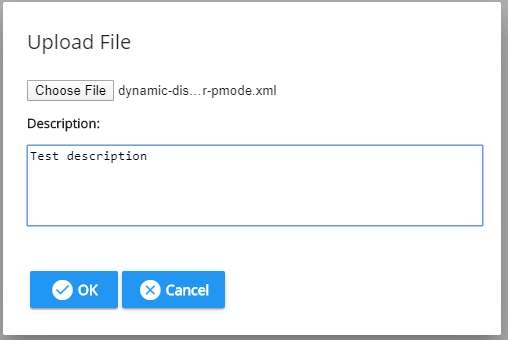
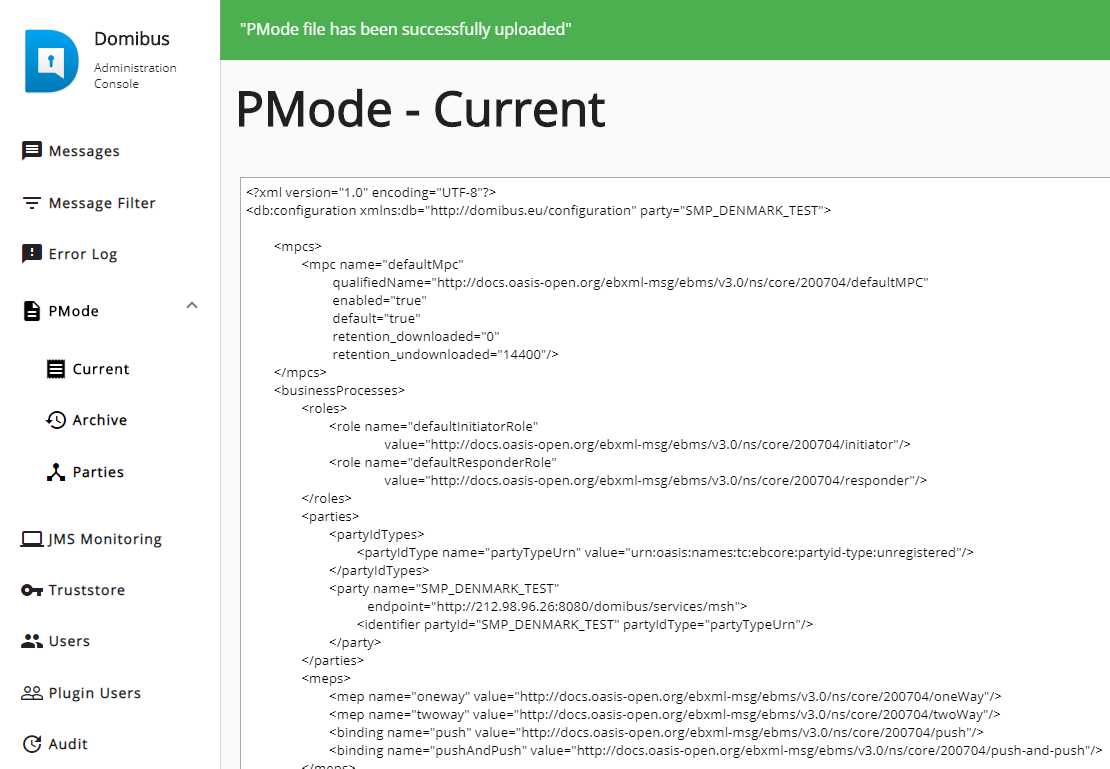
In order to perform connection test, prerequisites for the test must be fulfilled:

1. Access Point need to be installed on a local machine.
2. Access Point and Service Metadata Publisher need to be correctly deployed on the VM, with exposed all connections and ports.
3. SOAP UI need to be installed and configured locally.

### Configure Local AP

1. Start up your local Access Point according to instruction of Local Installing AP.
2. Open AP console: <http://localhost:8080/domibus>
3. Type in default password:  
   User: ‘admin’  
   Password: ‘123456’
4. Type in new password, write it down, Log in.
5. Click on PMode -> Current -> Upload  
     
   Figure 47. AP Admin console
6. Upload the PMode file dynamic-discovery-sender-pmode.xml from the eDelivery repository in path edelivery\ap\ap-tomcat\pmodes.

  
Figure 48. Repository folder

1. Write any description, click OK.  
     
   Figure 49. AP Admin console
2. PMode should be properly uploaded  
     
   Figure 50. AP Admin console

### Configure VM AP

Instructions how to configure AP on the VM, if the AP is freshly redeployed.

If the steps below have been already taken, there is no need to do them again.

1. Deploy the Access Point on the VM according to instructions in previous chapters.
2. Go to the AP console on the VM: <http://212.98.96.26:8080/domibus/>
3. Log in using credentials stored in the Excel file.
4. Upload the Sender PMode - follow the same steps as in configuring local AP, but in step 6 choose dynamic-discovery-receiver-pmode.xml

Once the issue with certificates is solved and SMP and AP use the target certificates, please edit the Pmode to use the proper partyID – replace all occurrences of “SMP\_DENMARK\_TEST” with the Common Name property from the target certificate subject.

### Configure VM SMP

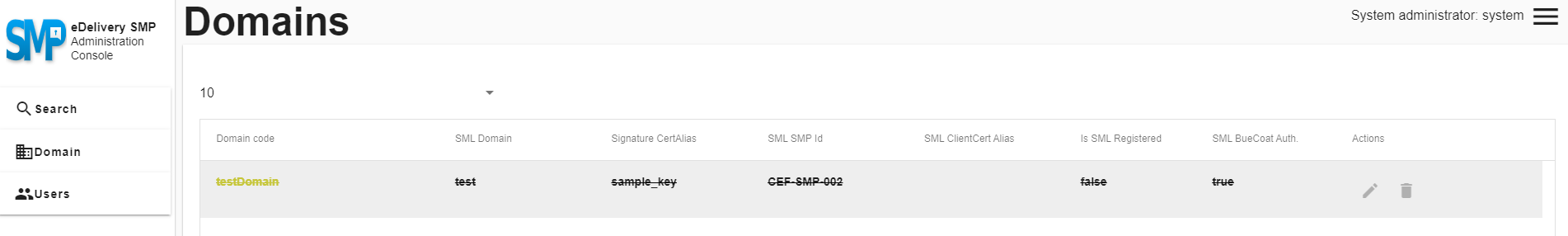
Instructions how to configure AMP on the VM, if the AMP is freshly redeployed.

If the steps below have been already taken, there is no need to do them again.

1. Deploy the Service Metadata Publisher on the VM according to the instruction in previous chapters.
2. Go the SMP console on the VM: <http://212.98.96.26>

Set up the domain:

1. Log in using credentials stored in the Excel file, as the ‘system’ user.
2. Click on the ‘Domain’ tab.
3. Delete existing sample domain.

  
Figure 51. SMP Admin console

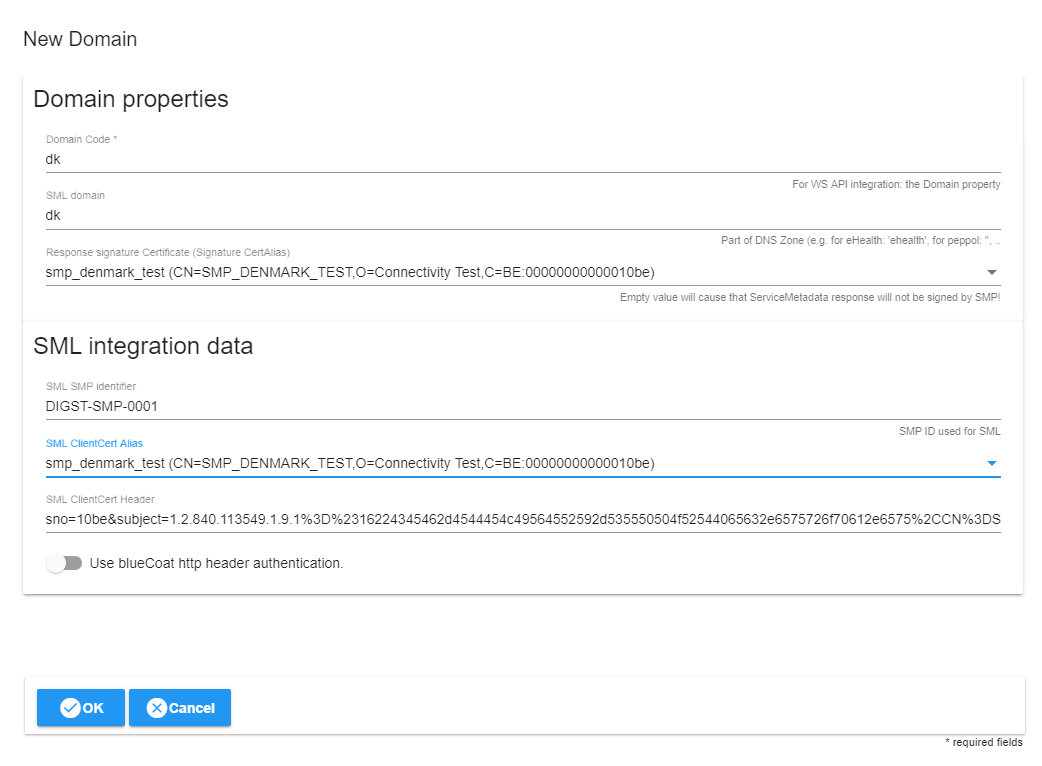
1. Click ‘save’ button.
2. Click on ‘+New’ tab.
3. Fill the form according to the screenshot:  
   

Figure 52. SMP Admin console

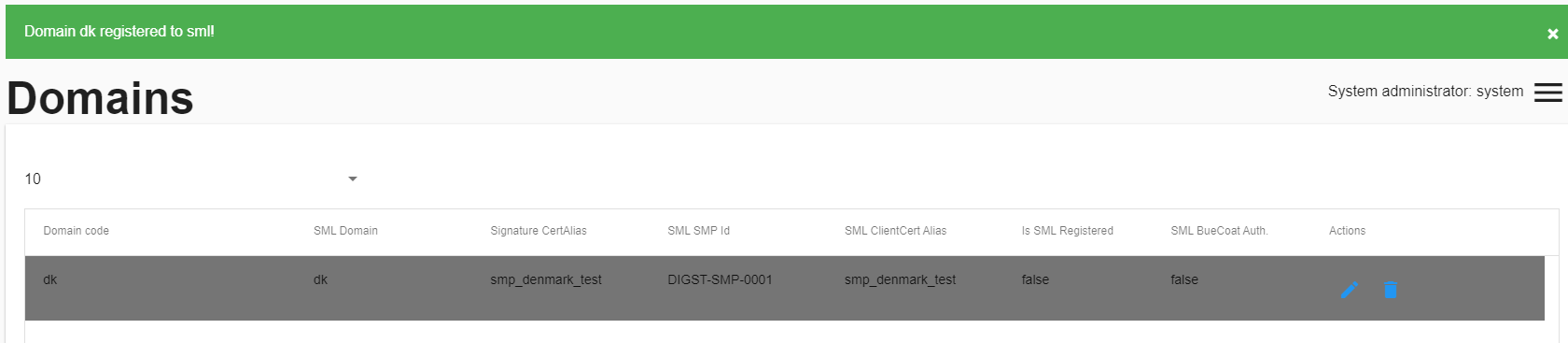
1. Click OK.
2. Select the new domain and click ‘Save’.
3. Click on the domain and click ‘Register’.
4. Click ‘Yes’.
5. Successful green bar should appear.  
   

Figure 53. SMP Admin console

1. You can check if the SMP was correctly registered by entering [https://acc.edelivery.tech.ec.europa.eu/edelivery-sml/listDNS and searching ‘DIGST-SMP-0001](https://acc.edelivery.tech.ec.europa.eu/edelivery-sml/listDNS%20and%20searching%20‘DIGST-SMP-0001)’.  
   

Create Service Group in order to register metadata.

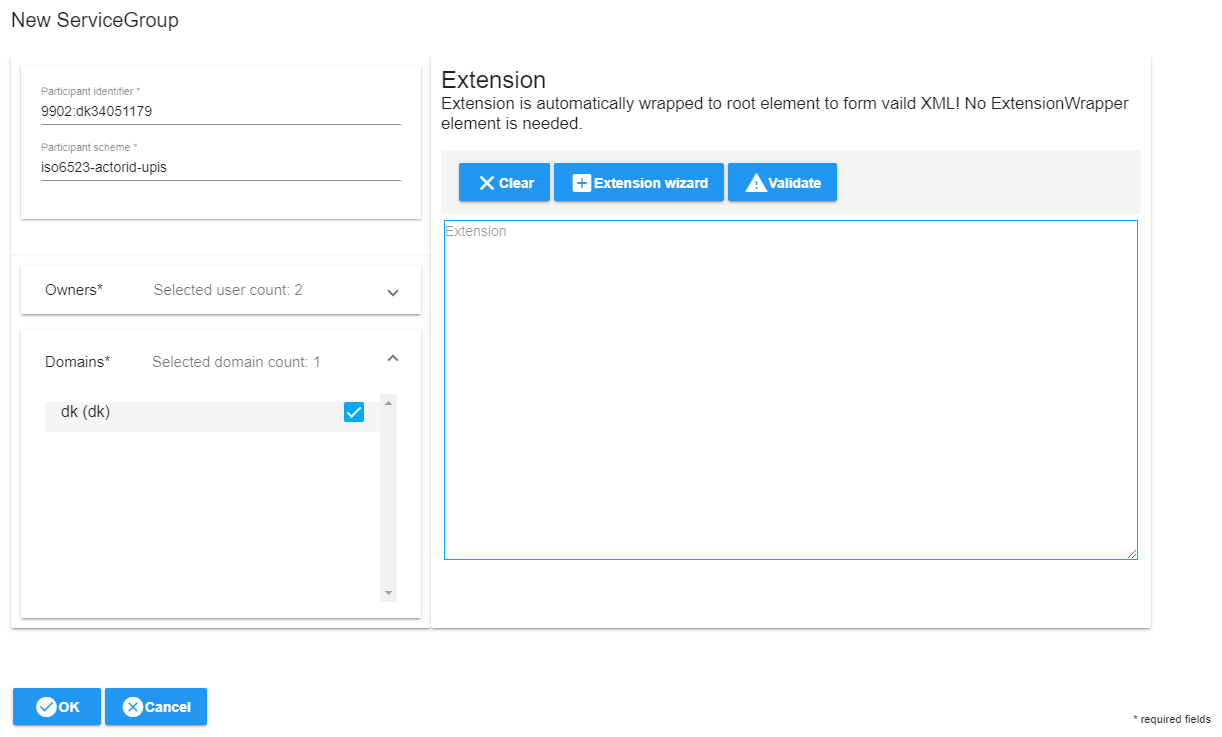
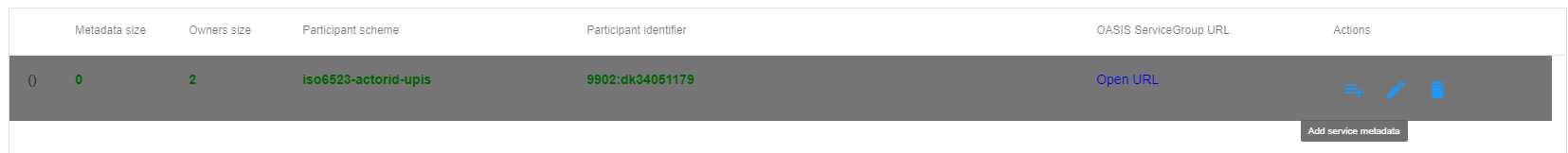
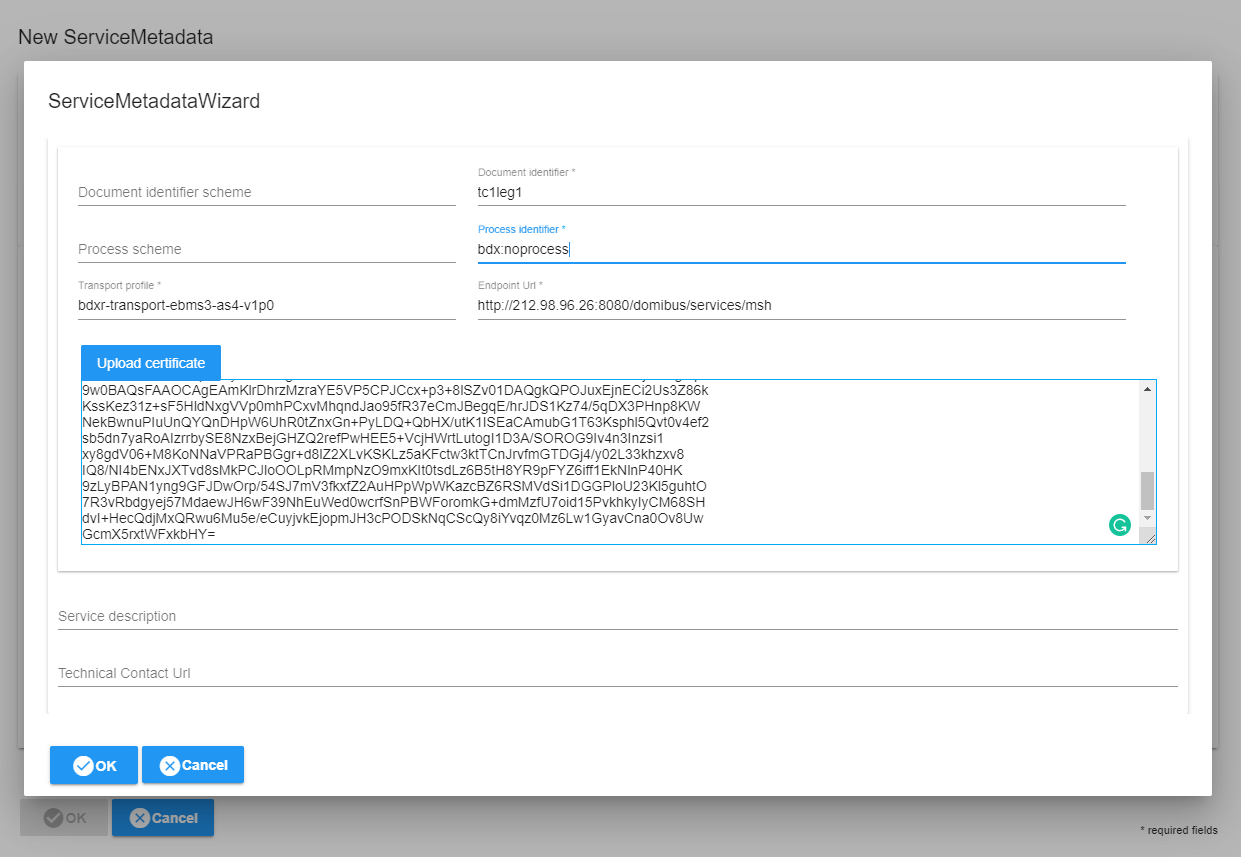
1. Log out.
2. Log in as ‘smp’ user.
3. Click on ‘Edit’ tab.
4. Click ‘+New’ button
5. Open the file dynamic-discovery-test-message.xml in eDelivery repository under path: edelivery\ap\ap-tomcat\test-messages.
6. Copy values from the file dynamic-discovery-test-message.xml into the wizard.
7. Fill values according to the screenshot:  
   

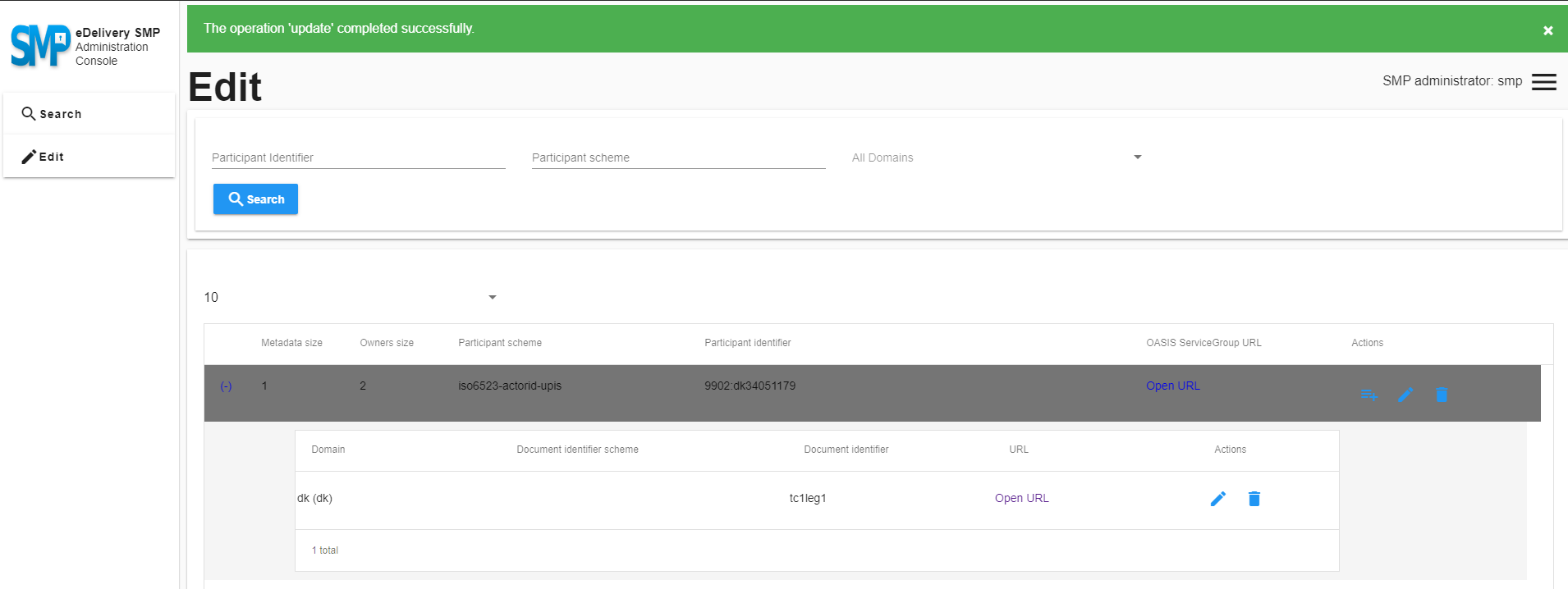
Figure 54. SMP Admin console

1. Click Extension wizard.
2. Click ‘Ok’.
3. Click on the icon ‘add Service Metadata’  
   
4. Type in Document identifier: ‘tc1leg1’
5. Click ‘Metadata Wizard’ and fill accordingly:

  
Figure 55. SMP Admin console

Load the test certificate. (proper AP certificate once the issue with certificates is solved).

1. Click on the newly created Service group.
2. Click ‘Save button’.
3. A Green ribbon should appear.

  
Figure 55. SMP Admin console

#### PMode and service metadata mapping

In order to add more processes in the future one shall edit PMode in AP and add new service metadata record in SMP. Mapping between these two is as follows:

|  |  |
| --- | --- |
| PMode | Service metadata |
| Process identifier | Service tag value:  <ns:Service type="PROCESS\_SCHEME">  PROCESS\_IDENTIFIER  </ns:Service> |
| Process scheme | Service tag type:  <ns:Service type="PROCESS\_SCHEME">  PROCESS\_IDENTIFIER  </ns:Service> |
| Document identifier | Action tag value:  <ns:Action>  DOCUMENT\_IDENTIFIER  </ns:Action> |
| Document identifier scheme | Action tag value prefix:  <ns:Action>  DOCUMENT\_IDENTIFIER\_SCHEME::DOCUMENT\_IDENTIFIER  </ns:Action> |

## Test execution

After the APs and SMP are configured, a test can be performed.

It is based on sending SOAP request to local AP, which with proper configuration will try to send an exemplary message to the AP deployed on the VM.

The local AP is using Dynamic Discovery to find the AP address, with usage of SMK (Test SML), which redirects to the DNS which maps logical SMP address to physical SMP.

**NOTE**

Because of the issue with NAPTR – Cname, Dynamic Discovery using SMK/SML is currently not possible.

There is a workaround made by routing logical address to physical address directly in the AP docker-compose file:

  
Figure 56. SMP docker-compose file screenshot

Please do note that this will only work for participant ID 9902:DK34051179.

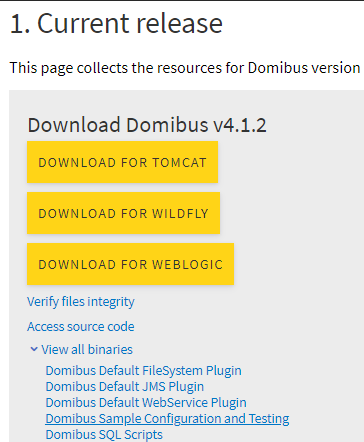
The issue is being discussed with SMP Support team, and should be fixed once the issue is solved on the SMP side.

### Configure SOAP UI

In order to send the message through the SOAP UI, it needs to be first downloaded and configured.

1. Download SOAP UI: <https://www.soapui.org/downloads/soapui.html>
2. Install it locally.

Go into the AP website <https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Domibus> and download package: ‘Domibus Sample Configuration and Testing’

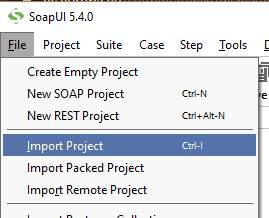
  
Figure 57. AP website screenshot

1. Unzip the package in specified folder.

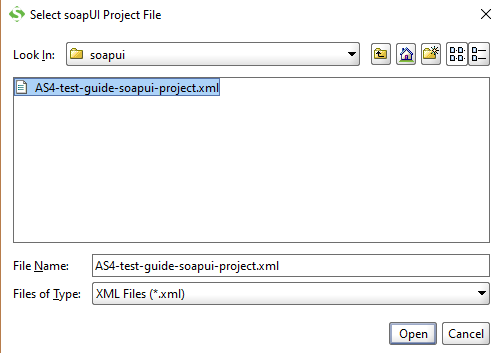
### Execute Test

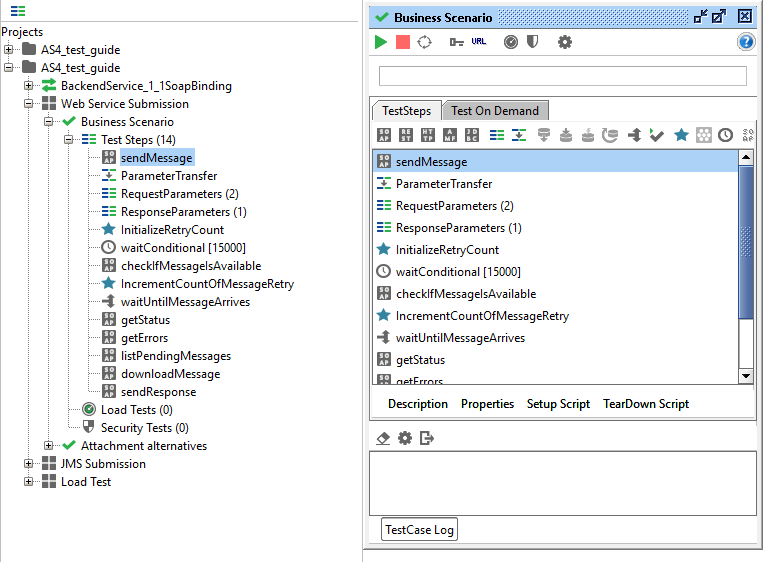
To execute the connection test:

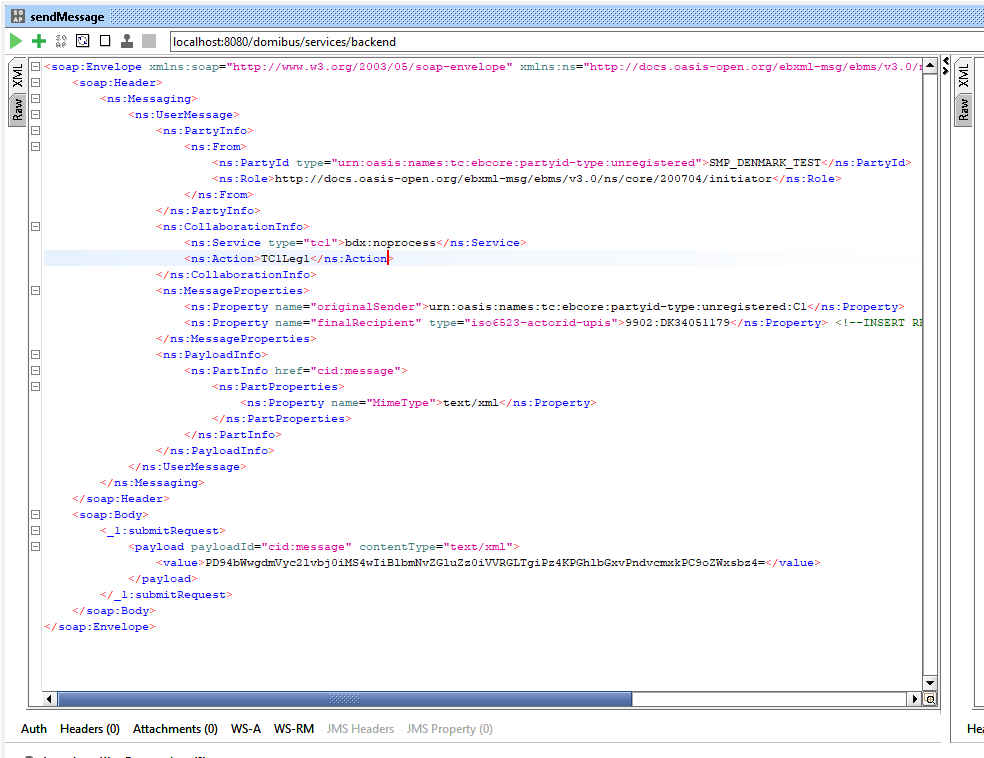
1. Open Soap UI.
2. Click on ‘Import Project’

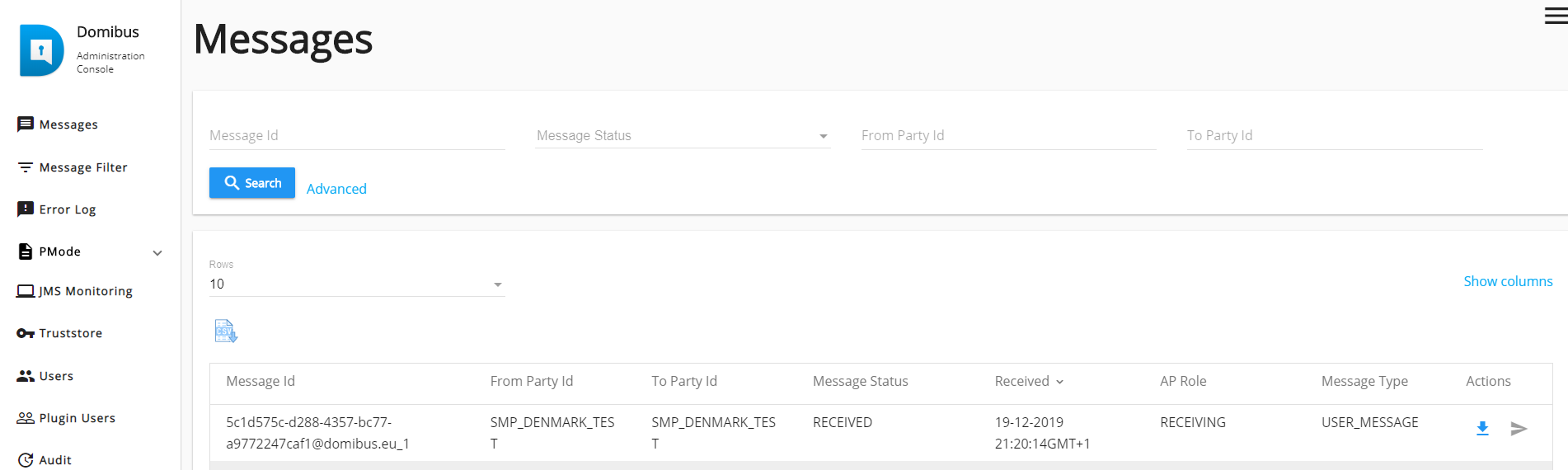
  
Figure 58. SOAP UI screenshot

1. Open the folder with unpacked ‘Domibus Sample Configuration and Testing’ and select AS4-test-guide-soapui-project.xml.

  
Figure 59. SOAP UI screenshot

1. When the project loads, click AS$\_test\_guide -> Web Service Submission -> Business Scenario -> sendMessage (double-click)  
     
   Figure 60. SOAP UI screenshot
2. In the new window change the address to ‘http://localhost:8080/domibus/services/backend’ and in the message body open the file ‘dynamic-discovery-test-message.xml’ in the eDelivery repository in path edelivery\ap\ap-tomcat\test-messages.

  
Figure 61. SOAP UI screenshot

1. Click Send message (Green triangle)
2. In the AP on the VM a new message should appear.
3.   
   Figure 62. AP console log screenshot

It means that the message was sent correctly!

Glossary

|  |  |
| --- | --- |
| Term | Description |
| Access Point (AP) | The Access Point (AP) serve as the primary participants in the CEF eDelivery network. These AP-participants use the network to transfer documents between each other and learn about other participants using the SMP and its provided metadata. The AP of CEF eDelivery implements the AS4 message exchange protocol according to the CEF eDelivery AS4 profile. This ensures standardized, interoperable, secure and reliable data exchange. |
| AS4 | AS4 (Applicability Statement 4) is an open standard messaging protocol based on web services to securely exchange messages between participants. The AS4 profile of CEF eDelivery is the AS4 Usage Profile/ implementation guidelines initially defined by e-SENS based on the AS4 specification of OASIS, itself a profile of OASIS ebXML Messaging Services Version 3.0, which in turn is based on various Web Services specifications of OASIS. The eDelivery AS4 profile is now maintained by CEF. |
| Capability Lookup | Capability Lookup is a technical service to accommodate a dynamic and flexible interoperability community. A capability lookup can provide metadata about the communication partner’s interoperability capabilities on all levels defined in the European Interoperability Framework (Legal, Organizational, Process, Semantic and Technical interoperability levels). The metadata can be used to dynamically set interoperability parameters between the Sending and Receiving Parties. |
| CEF eDelivery | CEF eDelivery is a building block helping public administrations businesses and citizens to exchange electronic data and documents with each other in an interoperable, secure, reliable and trusted way. |
| Domain Name System (DNS) | The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. |
| Domibus | Domibus is the open source project of the AP maintained by the European Commision. |
| Dynamic Discovery | To send messages using the CEF eDelivery messaging infrastructure, the AP of the sending party needs to obtain the communication information of the receiving party, e.g. lookup address and its communication capabilities. Such information can be obtained via a Static or a Dynamic discovery process.  The static discovery process uses a static list of the receiving parties stored on the sending AP and their configuration which is programmatically selected and added to the message. Dynamic discovery allows the sending AP to query an external service storing up-to-date information about every receiving party in the network. The dynamic discovery in CEF eDelivery is implemented with three components, namely the SMP, the SML and the DNS. The benefits of the dynamic model are more automation, higher scalability and more flexibility – however the dynamic model also have a higher lookup overhead compared to the static model. |
| Keystore | A .jks file containing **public/private key certificates** for encrypting network traffic between participants. In other words, the keystore contains the private certificate that is used locally by the SMP/AP. |
| NGDP | New generation digital post |
| Participant | Every individual instance of a component of the eDelivery network – for instance the SMP, SML and various AP’s. |
| Service Metadata | Information necessary for invoking a service using CEF eDelivery components. It is a combination of information on the end entity recipient (such as its identifier, certificate, supported business documents and processes in which it accepts those documents) and its associated endpoints (such as the transport protocol and its address). |
| Service Metadata Locator (SML) | Service Metadata Locator (SML) is a component of CEF eDelivery that is responsible for Dynamic Service Location: in order to send a message, the AP of a sending party needs to discover where the information about a receiving party is stored. The SML serves this purpose and guides the AP of the sending party towards this location, which is called the SMP.  In other words, the SML is used to retrieve/add/update/delete information about the receiving parties and SMPs location on a Domain Name System (DNS). The SML is a centralized component and is essential for Dynamic Discovery. |
| Service Metadata Publisher (SMP) | Service Metadata Publisher (SMP) is a component of CEF eDelivery that is responsible for Capability Lookup: once the AP of the sending party discovered the address of the receiving party's SMP (Service Metadata Publisher), it is able to retrieve the required information to interoperate with the receiving party (i.e. metadata). SMP are registers of the message exchange capabilities and location of parties (i.e. metadata). SMP's are usually used in a distributed way and are essential for Dynamic Discovery. |
| Truststore | A .jks file containing **authentication certificates** for determining whether participants are trustworthy. In other words, the truststore contains all the certificates that are used publicly by the SMP/AP. |